Research Tools for MIT App Inventor

Evan W. Patton
MIT CSAIL
Massachusetts Institute of Technology
Cambridge, MA, USA
ewpatton@mit.edu

Mark Sherman
Integrated Digital & Data Sciences
Emmanuel College
Boston, MA, USA
shermanm@emmanuel.edu

Michael Tissenbaum
College of Education
U. of Illinois at Urbana-Champaign
Champaign, IL, USA
miketissenbaum@gmail.com

Abstract

Understanding how ideas move through a classroom as students explore, learn, and share their findings with other students is important for computational thinking education. We present a set of enhancements to MIT App Inventor to enable fine-grained analysis of both qualitative and quantitative data. Our framework combines real-time event streams, project snapshots, and screen captures with audio recording to allow for targeted queries around when components or concepts not previously seen appear in a project. This automated approach reduces the amount of time spent sifting through data by providing researchers new tools for performing analysis of App Inventor projects.

Keywords App Inventor, research tools, learning analytics, instrumentation, computational thinking education

1 Introduction

Students learning a new topic can come by information in many different ways. One such way in the classroom is through word of mouth from a teacher or peers. Tracking this information can be difficult for conducting learning studies. In this paper, we present a multimodal tracking mechanism for MIT App Inventor that combines audio, video, observational, and fine-grained edit data streams to provide a new research platform. We also introduce a new Python library for performing queries over App Inventor projects to assist with identifying when new concepts appear in a student's project.

2 Toolkit Overview

The suite of tools we have developed were designed to allow for targeted questions about the development of an MIT App Inventor project and to understand when and how concepts are introduced to students, or when they discover such mechanisms through free exploration. The tools are multimodal, combining different forms of logging with audio and video streams, and a query mechanism for querying the structure of the project code.

BLOCKS+, November 04, 2018, Boston, MA, USA
2018.

2.1 Project Snapshots

Fine-grained analysis of the student projects was afforded by a Snapshot system that we integrated into MIT App Inventor. This system, based on [3], provided a capture of the project's full code state, recorded every time the student made a change to the project- including component and block additions/deletions, parameter changes, renaming, block moving, and more. This system recorded the Designer, Components, and all Blocks. This gave us the ability to scan through the history of a project to detect interesting moments, such as when a relevant component was first added, or a particular block was first recruited. The time stamps of these moments could be used to cross-reference the audio/video data, the in-classroom observations, or teacher notes to determine what the context was like outside of the screen. This allowed for a clearer picture of what the student was experiencing in the classroom, and should allow us to build models of what in-class experiences result in new critical moments in code development.

2.2 Audio/Video Capture

At the beginning of each App Inventor session, a video plugin called Screencastify was used to start a screen and audio recording session, which were uploaded and stored on a secure server. With the video recording, we could see where the student focuses attention and with the audio we could determine what information was being passed verbally off-screen.

2.3 Observation Protocol

Observations were made by in-person researchers following a protocol based on research around computational thinking [2] and collaborative open-ended tinkering [4]. The protocol aimed to capture students seeking and providing help, asking questions, taking control of their or others' computers, moving around the room, and other in-class interactions. These observations were time-coded by the observer and can be used in time-correlated analysis with the programmatic data collected.

2.4 Real-time Event Server

Fine-grained analysis is also offered by building our research platform on top of a version of MIT App Inventor with an event-based edit model, originally built to support real-time

Listing 1. Detect procedures that do not have any callers.

```
AIAFile('sample.aia').blocks(
  (type == procedures_callnoreturn) |
  (type == procedures_callreturn) &
  ~is_called).select(fields.NAME)
```

Listing 2. Count the number of event blocks without a body.

```
AIAFile('sample.aia')
  .blocks(type == component_event)
  .filter(~hasDescendant(lambda x:True))
  .count()
```

editing of projects by students [1]. Every creation, movement, and deletion of components and blocks are sent to a centralized server where they are logged for further analysis. This allows researchers to look at fine grained activities that occur between server-side project save snapshots. This data stream will replace the Project Snapshots in future works, as it will be simpler to collect, while offering the same powers of data capture. This event stream will also be available for future real-time instrumentation applications.

2.5 Project Analysis Tools

We have developed a Python library, called AIATools, for constructing declarative queries over App Inventor projects.[1] AIATools parses Blockly XML files into objects over which it can execute queries. An example query for detecting whether a procedure call block exists for an undefined procedure is shown in Listing 1. Additional support has been developed to combine AIATools with the snapshot service to enable querying changes to projects over time.

3 Discussion

The suite of tools we have presented allow for targeted research queries about how people are using the App Inventor platform. For example, a recent classroom evaluation of the platform involved students working with Internet of Things technology over Bluetooth low energy. This requires the use of the BluetoothLE extension. Using AIATools applied to the snapshots, we can detect when a student incorporates the BluetoothLE extension into their project. With the timestamp of the event, we can cross-reference with the audio/video stream to determine what the student was hearing in the classroom prior to introducing the extension.

4 Conclusion

We presented four extensions to the MIT App Inventor platform to aid in computer science education research. These extensions allow for fine grained pinpointing of when ideas are introduced into the space and how they move between students, and they bridge across multiple modalities to give a more accurate picture of the available means by which students receive and act on new information. We believe that further development and refinement of such tools will be beneficial to the learning sciences community, especially as it pertains to computer science education research.

5 Future Work

While the AIATools library knows about App Inventor's semantics specifically, it can be extended to other languages built on Blockly. Video recording was a surprisingly difficult technical hurdle to overcome, and we plan to implement it directly into App Inventor, as to not rely on the third-party plugin. This will allow greater control and granularity over the recordings, especially the ability to progressively send to the server, which would improve reliability and performance. We also plan to integrate analysis over the real-time data so that queries can be answered over a stream, which would allow for the development of sophisticated tools for real-time analysis of student progress. All of these changes were intended to make the data more robust and the platform easier to deploy for research groups.

References

[1] X. Deng. 2017. *Group collaboration with App Inventor.* Master's thesis. Massachusetts Institute of Technology.

[2] S. Grover and R. Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42, 1 (2013), 38–43.

[3] M. Sherman. 2017. *Detecting Student Progress during Programming Activities by Analyzing Edit Operations on their Blocks-Based Programs.* Ph.D. Dissertation. University of Massachusetts, Lowell.

[4] M. Tissenbaum, M. Berland, and L. Lyons. 2017. DCLM framework: understanding collaboration in open-ended tabletop learning environments. *International Journal of Computer-Supported Collaborative Learning* 12, 1 (2017), 35–64.

[1] https://github.com/mit-cml/aiatools