# ProgSnap 2: Towards a Standard Representation for Programming Process Data

David Hovemeyer
dhovemey@ycp.edu
York College of Pennsylvania

Kelly Rivers
krivers@andrew.cmu.edu
Carnegie Mellon University

## ABSTRACT

Systems that collect data on student programming activity provide a valuable source of information on how students learn to program. The diversity of information collected by such systems presents a challenge for analyzing data sets, especially for studies involving data originating from multiple systems. For this reason, having a standardized representation for student programming activity would be useful. ProgSnap 2 is a proposed standardized representation for programming snapshot data. It is intended to be sufficiently flexible to accommodate data from a wide variety of sources, and its tabular format is designed to permit direct analysis using statistics software.

## KEYWORDS

computer science education, snapshot representation, student programming data

**ACM Reference Format:**
David Hovemeyer and Kelly Rivers. 2018. ProgSnap 2: Towards a Standard Representation for Programming Process Data. In *SPLICE '19: SPLICE Workshop 2019, February 27th, 2019, Minneapolis, MN*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Recent years have seen increased interest in automated systems to help students learn to program[2, 4]. The data collected by these systems is a valuable source of insight on how students learn to program. Due to the diversity of data collected, analyzing such data sets can be challenging, especially when data from multiple sources are involved. For this reason, a standardized format for programming activity data would be useful.

ProgSnap 2 is a proposed specification for representing programming activity data. It has two main design goals. The first is to be flexible enough to represent data from a variety of sources. The second is to permit direct analysis using statistics software. ProgSnap 2 incorporates ideas from previous proposed standards, including DATASTAND and ProgSnap[3].

This paper briefly summarizes ProgSnap 2. See Price et al.[5] and the draft specification[1] for a more complete description.

## 2 OVERVIEW OF PROGSNAP 2

In the ProgSnap 2 data format, we represent student data using a main event table, which stores information about student actions, and a collection of payload files, which store the actual student work. Each row in the event table refers to an individual payload file, so that researchers can do analysis at a high level but still refer to more detailed data at need. In this paper, we will primarily discuss the main event table and the code representation; for more detail, refer to the official specification.

The main event table contains a series of rows, where each row represents a student action. The columns then represent different properties of these actions, where some columns are mandatory and many are optional. We have designed the event table this way to allow data collectors the freedom to store as much or as little data as they wish to, while still keeping certain necessary data points consistent between data sets. The mandatory columns include:

- **Event Type**: the type of event which took place. Examples include Session.Start, Compile.Error, File.Edit, and Run.Test
- **EventID**: a unique ID for the event
- **Order**: a number that provides an ordering between this event and others. This ordering is not guaranteed to be perfect, but it provides the best guess at ordering by the data provider
- **SubjectID**: an ID referring to the student/user who performed the action
- **ToolInstances**: a string containing all of the tools used in the action. This includes the programming language, IDE, and any external tools
- **CodeStateID**: the ID of the Code State connected to this action

There are too many optional columns to include in this paper, so we will instead highlight a few properties of interest to data providers and researchers:

- **ServerTimestamp/ClientTimestamp**: many data collectors reported having difficulty synchronizing events collected on the client-side versus the server-side. We provide two optional columns to allow the most accurate data representation possible, while the Order column still provides an overall ordering for simple analysis.
- **CourseID/AssignmentID/ProblemID**: there are several columns that can be used to identify the specific context of a problem-solving event, if that context is known.
- **ExperimentalCondition**: the string in this column can be used to clearly identify whether the data was collected as part of a study, and which condition the event should be associated with.
- **EditType/EditTrigger**: these columns allow for further information to be provided on edit events (via a set of enumerated values), while still allowing analysis of general file edits without extra data processing.
- **CompileMessageType/CompileMessageData**: these columns allow for more generalized study of compiler errors, by capturing both the general error type and specific data.

*SPLICE '19, February 27th, 2019, Minneapolis, MN*
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- **InterventionType/InterventionMessage**: the protocol also tracks the results of interventions commonly used in programming studies, such as hint messages.

As is shown in the mandatory CodeStateID column, each event is associated with a code state, which is stored outside of the main event table. As code states vary widely across different contexts, these states can be represented using one of three separate recommended formats. This allows for consistency in analysis while not requiring data providers to force their data into a format that does not match their needs.

- **Table Format**: this format is used to represent small data sets where each snapshot is a single file of small size, and where all code is text-based. All code states are stored in a single CSV file which contains two columns: CodeStateID and Code. The Code column then stores all the code present in the state.
- **Directory Format**: this format is used to represent data sets that are medium-sized, with potentially more than one file per snapshot. Each code state is stored in a different directory inside the CodeStates directory, where the name of the directory is the CodeStateID.
- **Git Format**: this format is used to represent data sets that are particularly large or contain many files per snapshot. It uses a single Git repository to store all of the separate snapshots. Each CodeStateID is then the ID of the commit which, when accessed, represents the state of the code at that action.

## 3 CURRENT STATUS AND FUTURE WORK

As of the time of writing, the contributors to the ProgSnap 2 standard are working on implementing tools to export data from several automated assessment systems to ProgSnap 2 format. Once we have a reasonable variety of ProgSnap 2 data sets, we intend to work on analysis tools. Performing analysis on data sets from multiple systems will be an important step in validating the usefulness of the ProgSnap 2 standard.

We are interested in working with providers and consumers of student programming data in order to help ensure that ProgSnap 2 is broadly useful. The authors welcome contact from anyone interested in participating in this effort.

## REFERENCES

[1] Progsnap 2. https://cssplice.github.io/progsnap2/, 2019. Accessed: 2019-01-18.
[2] Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005.
[3] David Hovemeyer, Arto Hellas, Andrew Petersen, and Jaime Spacco. Progsnap: Sharing programming snapshots for research (abstract only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 709–709, New York, NY, USA, 2017. ACM.
[4] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA, 2010. ACM.
[5] Thomas W. Price, David Hovemeyer, Kelly Rivers, Austin Cory Bart, Andrew Petersen, Brett A. Becker, and Jason Lefever. Progsnap 2: A flexible format for programming process data. In *Proceedings of the 2nd Educational Data Mining in Computer Science Education (CSEDM) Workshop*, Tempe, AZ, 2019.