Technical Report:
DATASIM

Data and Training Analytics Simulated Input Modeler

31 March 2020

This work was supported by the U.S. Advanced Distributed Learning (ADL) Initiative (HQ0034-19-C-0061). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ADL Initiative or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes.

Technical Report: DATASIM

Data and Training Analytics Simulated Input Modeler

Prepared by Shelly Blake-Plock, PI, Yet Analytics, Inc.
Submitted to ADL on March 31, 2020

1. Purpose of this Document

The purpose of this document is to present a report and related documentation regarding the Alpha TRL4 prototype of the Data and Training Analytics Simulated Input Modeler (DATASIM). The report explores the intended use cases of the DATASIM platform, the progress that has been made on the prototype reference software, and the aspects of the application which would benefit from further development in the next iteration of the platform.

Table of Contents

---

2. Project Description

DATASIM is a software platform capable of generating simulations resulting in realistic xAPI data. The platform uses xAPI Profile Patterns, Templates, and Concepts to model behavior for a cohort of simulated actors. These datasets may be used to benchmark and stress-test components of the Total Learning Architecture (TLA) and other distributed learning projects. The datasets that DATASIM is capable of generating can also offer insight into the available Patterns and paths available in an xAPI Profile.

Because DATASIM requires valid xAPI Profiles in order to generate datasets, it also has the capability to validate xAPI Profiles and other inputs.

In total, the Alpha version of DATASIM can help learning scientists, engineers, ISDs, IT staff, and decision-making stakeholders to determine the effectiveness of xAPI data design and xAPI implementation. A future iteration of DATASIM may be able to provide key conformance testing functions throughout the TLA.

DATASIM is open source under the Apache 2.0 License and is funded by the Advanced Distributed Learning Initiative at the United States Department of Defense.

3. Technical Summary

This section of the report explores in detail the main functional areas of DATASIM, how they work together, and what has been implemented so far in this stage of the prototype platform.

3.1 Inputs and Validation

DATASIM uses four primary input specifications to guide a simulation.

- The xAPI Profile — or multiple xAPI Profiles — provide the Patterns and Templates from which the Statements are generated.
- Personae provide the basic attributes for simulated Actors.
- Alignments detail relationships between Actors and components of the Profile(s).
- Parameters define various aspects of how the simulation runs.

3.1.1 Profiles and Profile Validation

The DATASIM prototype platform is capable of accepting one or more valid xAPI Profiles. All Statements are generated by the Patterns, Templates, and Concepts contained within the Profile(s). Because of this, DATASIM has in-built xAPI Profile validation capability. Some of the aspects of the xAPI Profile that DATASIM validates include:

- Structural validation for types of properties and basic syntax
- ID validation for objects and versions
- Validation of all related IRIs and all relations between xAPI Profile objects
- Extension validation
- Context validation

Errors are returned to the user so that remediation is possible. This capability, in addition to the resulting dataset analysis, makes DATASIM useful as an all around xAPI Profile design tool.

3.1.2 Personae

The Personae input to DATASIM is a list of xAPI Actors and any attributes about them needed for the simulation. Actors are required to run a DATASIM simulation, because without them there would not be any simulated learner(s) to engage in an activity.

This input takes the form of a JSON object containing an array of conformant Actors, an example of which is below:

```
{"name": "trainees",
 "objectType": "Group",
 "member": [{"name": "Bob Fakename",
             "mbox": "mailto:bob@example.org"},
            {"name": "Alice Faux",
             "mbox": "mailto:alice@example.org"},
            {"name": "Fred Ersatz",
             "mbox": "mailto:fred@example.org"}]}
```

The actors can be defined in groups, and may have a *name* and *objectType* and must have one of the four Inverse Functional Identifiers (*mbox, mbox_sha1sum, openid, account, see https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-Data.md#inversefunctional*).

Once defined, these will be the only Actors to engage in actions within the simulation.

3.1.3 Alignments

An Alignment represents a way to influence the simulation by explicitly weighting an Actor's relationship to a part of the xAPI Profile. Each actor can have alignments to multiple parts of the Profile (including Patterns, Statement Templates, and Concepts), and the weight system ranges from -1 to 1 (with 1 being an extremely high propensity for interaction in the simulation and -1 being the opposite). During the simulation these weights factor in but do not completely predict the outcome as there is still entropy in Actor behavior (discussed more in **Section 3.1**). The records are a combination of Actor, IRI to align to, and weight and follow this format:

```
{"mbox::mailto:bob@example.org":
  {"https://example.org/activity/a": 0.5,
   "https://example.org/activity/c": -0.2},
 "mbox::mailto:alice@example.org":
  {"https://example.org/activity/c": 0.7,
   "https://example.org/activity/d": -0.02}}
```

In this example, the Actor identified by mbox “mailto:bob@example.com” has a positive alignment (0.5) to the activity identified by IRI “https://example.org/activity/a” and a negative alignment (-0.2) to the activity identified by IRI “https://example.org/activity/c”.

In the resulting simulation, “bob” will be much more likely to engage with activity “a” if it is present in the xAPI Profile.

3.1.4 Simulation Parameters

The simulation parameters input dictates aspects of how the simulation runs which are not covered by the other inputs. This includes Start Time, End Time, Timezone, Max Statement, and Seed.

When run, the simulation will create a time sequence from the Start Time to the End Time. Generated xAPI statements will have corresponding dates and times throughout that simulated time period. The simulation’s timeline will move forward from Start to End, in chronological order.

As an alternative to End Time, Max Statements can be input, which causes the simulation to run from the Start Time until it has produced that exact number of xAPI Statements. When the simulation reaches the point of Max Statements, it will stop. Therefore, it is not a good determiner of the end time of the simulation. However, it is useful for obtaining a specific number of records.

The Seed is important as it controls the inputs to all random value generation and enables the repeatability of the simulation. A simulation run with the same inputs and the same seed will deterministically create the same xAPI Statements. Changing the seed value will create an entirely different simulation. The Seed is expected to be an integer.

An example of simulation parameters is below:

```
{"start": "2019-11-18T11:38:39.219768Z",
 "end": "2019-11-19T11:38:39.219768Z",
 "timezone": "America/New_York",
 "seed": 42}
```

This particular simulation will generate activities occurring over one day, from 11/18/2019 at 11:38am to 11/19/2019 at 11:38am with an input seed of “42”.

3.1.5 Simulation Specification

The Simulation Specification is a more portable syntax which contains all of the inputs listed above and may be used as an alternative, single, input to a simulation. It was designed so that users could easily share simulations with each other, and save them for later use in one file. The format is just a JSON object with all of the previous four inputs as keys with corresponding values.

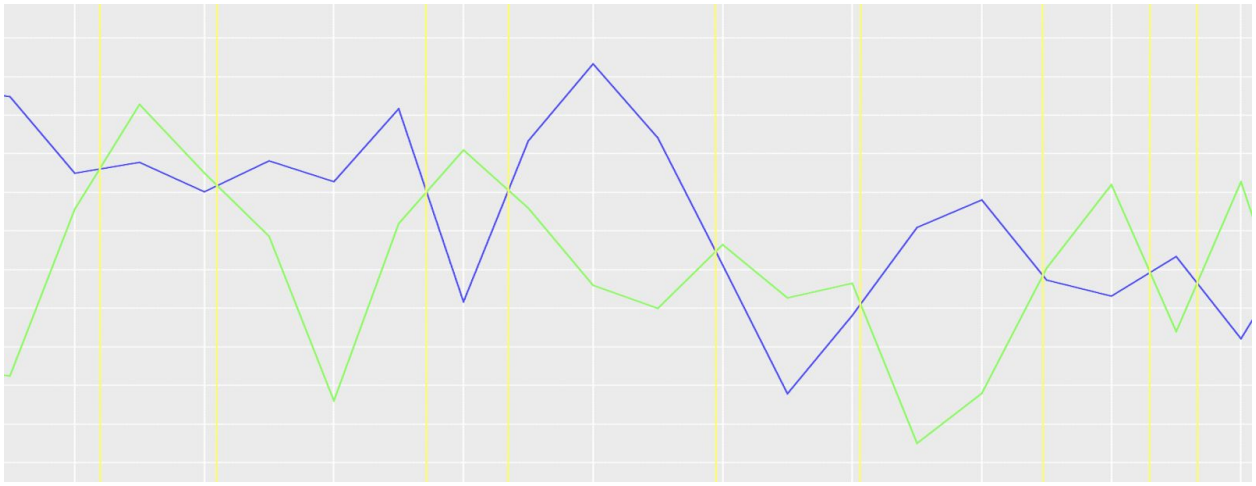
```
{"profiles":[ ... ],
 "parameters": ...,
 "personae": ...,
 "alignments": ... }
```

3.2 Actor “Motivation”

Having Actors, Profiles, and Alignments arranged in a simulated environment for a fixed period of time does not accomplish the goal if the Actors are not incited to perform actions.

Furthermore, though all of the Actors share the same timeline in the simulated universe, simply having all Actors coordinate and complete actions at preset time intervals along a simulated clock fails to produce realistic behavior data. Each Actor in real life is different and performs actions at their own pace.

In order to accomplish this within the simulation, ARMA (autoregressive moving average) time series generation was leveraged.



Because all of the Actors share the same timeline in the simulated universe, for each simulation there is one “Common Time Series”. Additionally, for each actor there is one “Personal Time Series” generated just for them. All time series are generated from deterministically random values which are seeded by the “seed” input parameter, which makes them repeatable on subsequent identical simulations.

The “heartbeat” of the Actor in a DATASIM simulation is the intersection points between the Personal Time Series and Common Time Series. This allows for the independent action time samples for each Actor, while also allowing the Actors to potentially be influenced by movements in the Common Time Series in their simulated universe. An example is illustrated below:

Fig. 1. Note intersection points between the Personal Time Series (blue) and Common Time Series (green).

3.3 Statement Generation

Given the newly motivated and aligned Actors, Profile, and parameters to define the bounds of the simulation, the next step is to have the Actors engage with the defined Patterns, Templates, and Activities of the xAPI Profile(s) to generate appropriate Statements.

When an Actor is ready to perform an action, the simulation finds the appropriate action by traversing the xAPI Profile. Traversal of the profile is influenced by a number of things, the most important of which is the design of the xAPI Profile itself.

DATASIM initially looks for a primary Pattern within the xAPI Profile. If it has multiple to choose from, it uses a deterministically generated random value in combination with the Alignments of the appropriate Actor to determine which one to select. In this way the Actor's alignment toward or against a particular Pattern has an impact but not full control.

Note that an Actor, due to the entropy inherent in the determination, may engage in an action they are aligned against, and similarly may not engage in an action they are aligned towards.

Once a Pattern is selected, the *alternates, optional, oneOrMore, sequence*, or *zeroOrMore* attributes (see Patterns) are used to select either a Template or another Pattern in much the same way as the initial Pattern was selected, meaning both alignment and entropy are considered. If another Pattern is selected, this traversal and selection repeats until a Template is selected. Once a Template is selected, an xAPI Statement is produced based on the Template definition.

At the point of Statement generation from a Template, the simulation follows the rules of the Template. The granularity of these rules (use of *any*, *included*, etc.) will dictate the richness of the data in individual output statements.

3.4 LRS Integration

In order to make the DATASIM Platform and its resulting simulated datasets usable for analysis, broad LRS Integration is required.

DATASIM's LRS API was built around the xAPI Communications protocol (see here) and as such should be interoperable with every ADL LRS Test Suite conformant LRS.

In the Testing and Evaluation portion of the project, DATASIM demonstrated integration capability with all of the LRS instances it was tested with. This LRS interoperability feature is fully functional using the prototype in CLI or UI mode.

3.5 User Interface

DATASIM has a fully functional user interface, written in ClojureScript. It runs in a browser which can be used to interact with the total of DATASIM's current featureset, including:

- Creating and editing all inputs

- Syntax Highlighting, code linting, and validation
- Importing all inputs from files and URL
- Exporting all inputs separately
- Exporting all inputs as a Simulation Specification
- Running a simulation and downloading the resulting xAPI Statements
- Inputting LRS connection details
- Running a simulation and sending the resulting xAPI Statements to the LRS
- Basic Security preventing server usage without credentials

The user interface was designed with a focus on simplicity and consists of only one page and a few functional dialogs.

3.6 Security

The current DATASIM prototype implements Basic Authentication between the UI and the server side to prevent use of the server side Statement generation by an unauthorized party.

The project is open source and is incapable of persisting data, so by definition it is incapable of storing or giving access to sensitive information. On account of that, the CLI does not presently have security features.

4. Using DATASIM

The DATASIM Project is open source and it is split into two projects. The Backend simply provides an API. It is possible to build a novel frontend or scripts against the Backend API instead of using the provided Frontend. Both modules are publicly available on github here:

Backend <https://github.com/yetanalytics/datasim>
Frontend <https://github.com/yetanalytics/datasim-ui>

In each repository, the individual setup and use documentation is available at those links. There is also a demonstration of the frontend available here:

https://yetanalytics.github.io/datasim-ui/

The Backend of DATASIM can also be run alone as a CLI application on any modern JVM. Documentation is available for that usage type on the repo.

Additionally, the project can be used as a library within another project. This can be done directly in Clojure using the git repository as a dependency. Alternatively, when compiled the project produces Java class files which can be leveraged as a library from within any Java or JVM-compatible application.

5. Profiles and Data-Aided Design

Throughout the development and testing of the DATASIM prototype, a number of new and existing xAPI Profiles were used to generate data and assess the capabilities of the system.

Because DATASIM is intended as a design aid for xAPI Profiles, it was used both to repair existing Profiles and as a tool during the creation of new ones.

5.1 cmi5 xAPI Profile

The first xAPI Profile that was used during the development of DATASIM was cmi5.

Because of its use in the xAPI community and its broad set of Patterns, Templates, and Concepts, it was a good xAPI Profile to build with and revealed edge cases right from the beginning. It was also helpful in generating large diverse datasets and helped with performance testing and efficient design.

Note that the current version of the cmi5 xAPI Profile had a small syntax issue and did not pass validation. It was fixed and the updated version can be found in the Appendix below.

5.2 Calibration xAPI Profile

In order to analyze DATASIM's behavior from the perspective of resulting datasets, it was necessary to create a new xAPI Profile. It is called the Calibration Profile. The reason that the Calibration Profile was created has to do with how statements are generated. If a Pattern or

sequence of Patterns share some of the same Statement Templates, it becomes non-trivial to assess exactly what path in the Profile the simulation followed given any resulting set of Statements because many of the same Statements may be generated for more than one chosen Pattern.

It is sometimes, but not always, possible to look at groups of Statements and infer the Pattern sequence based on ordering, however even this can be labor intensive and unintuitive.

The Calibration Profile is simple by design. It has a single primary Pattern which references a sequence of non-primary Patterns. Each non-primary pattern itself has an optional reference to an exclusive Statement Template which is not shared by any other Pattern in the Profile.

Because of this simple hierarchy, simulation behavior in regard to Pattern traversal is trivial to infer from resulting Statement datasets of any size. This kind of xAPI Profile may be expanded in the future as a baseline to test various aspects of DATASIM and other xAPI Profile tools including conformance. Links to the current Calibration Profile are included in the Appendix of this report.

5.3 DoD MOM xAPI Profile

During the Testing and Evaluation phase of the DATASIM prototype, the DoD MOM Profile was engaged in order to evaluate performance and xAPI Profile compatibility.

As with the cmi5 xAPI Profile, DATASIM's built-in xAPI Profile validator found issues and would not initially allow statement generation from the DoD MOM Profile. Unlike the cmi5 xAPI Profile, which contained a single minor syntax issue, the MOM Profile contained numerous conformance issues, some of which may require interpretation of the original intention to remediate. Issues included "scopeNotes" not being set as language maps, errors on several Timestamps, incorrectly defined inlineSchemas, and required keys missing throughout — including for prefLabel, definition, and type. Because of these issues and the limitations of time, the DoD MOM Profile was not used in testing.

5.4 TCCC xAPI Profile

As part of the DATASIM project, a subset of the Tactical Combat Casualty Care Course (TCCC or TC3) was built out as an xAPI Profile. This subset xAPI Profile was used to test the prototype. The applicable subset of the TC3 xAPI Profile can be found in the Appendix of this report.

The TCCC CUF HC xAPI Profile is

- scoped to TCCC CUF HC and TCCC ASM content available from the deployed medicine website
- partially derived from the xAPI Video CoP profile
- designed for the programmatic generation of diverse xAPI datasets; datasets which include a multitude of emergent and repeatable patterns of behavior; and behavior modeled within this profile as statement templates and patterns.

Because this xAPI Profile was the first to be developed both with the aid of and the intention of generated data, this effort illuminated some of the advantages and nuances of generation-aided xAPI Profile design, namely:

1) Model based data generation prevents poorly designed models from hiding behind the curtain of JSON-LD. This detection is contingent upon sufficient analysis of the simulation datasets as analysis precedes understanding of a model's properties and nuances. In some cases, that analysis consists of stepping through activity logs, while in other cases it involves filtering, transformation, aggregation, and/or visualization of xAPI data in order to describe some aspect of the model; and thus hopefully the real world. If a complex, domain specific, data generating processes can be replicated within simulation results via xAPI profile definitions, the pattern matching construct responsible can be utilized in a number of ways including:

    - continuous refinement and testing on novel datasets, i.e. tuning
    - data mining tasks once the corresponding model is tuned
    - KPI tracking or another live, event based form of tracking; can be implemented within streaming architectures or other information digital highways

2) The power of the generative technique lies in a positive feedback loop established by, and implicit to, iterative xAPI Profile design when equipped with a tool that provides a means of easy experimentation and *rapid prototyping*.

3) The ability to incorporate elements from real world datasets into this feedback loop strengthens the semantic domain modeling processes. The process of incorporating new sources of information into, and deriving data driven insights from these data sources can be automated in ways which promote and/or support autonomous refinement and background services such as anomalous pattern identification.

In general, xAPI profiles written for use within generation should contain:

1) Patterns complex enough to enable programmatic generation of sensible xAPI learner pathways
    - For example, in the case of video: simple played, paused, resumed, and completed logic to prevent obvious logical fallacies in generated datasets
    - Diverse patterns which model a sufficient number of domain specific behavioral patterns and/or permutations within said patterns
2) High and low level statement templates
    - On the one hand: definition of patterns common to a classification of things, i.e. statement templates with a low to medium amount of granularity or specificity
    - On the other: definition of patterns specific to a thing, the classification of which may not be a factor, i.e. statement templates with a medium to high amount of granularity or specificity

The current effort resulted in the derivation of non-trivial behavioral pathways, within the domain of video engagements, from the set of possible video interactions established by the CoP xAPI Video profile.

The current effort also produced behavioral pathways for engagement with a student feedback survey consisting of likert questions. These pathways are written in the abstract language of xAPI patterns and statement templates but are made more concrete through the incorporation of actual TCCC CUF HC video content and TCCC ASM student feedback survey questions.

The videos themselves are defined as Concepts whereas the possible video interactions are defined as Patterns that reference statement templates specific to said videos. This profile builds upon the Video CoP xAPI Profile by providing an expansion of the video interaction behavior permutations, expressed as xAPI Profile patterns. The expansion enables the generation of datasets which will eventually pass muster provided enough tuning and refinement of the definitions within this xAPI Profile.

7. Conclusion and Next Steps

Through this analysis, testing and evaluation it has been demonstrated that this Alpha version of the DATASIM platform has met or exceeded the requirements for a TRL 4 prototype in every category.

Aside from its success at the goal of efficiently generating realistic simulated xAPI datasets, it is also already demonstrating value as a tool to aid xAPI Profile design — as seen in the analysis of the creation of the TCCC Profile. In order to mature the DATASIM platform to the next level of technical readiness, and to address any perceived needs the system has currently, the following areas of potential platform improvement provide a map of potential feature development.

7.1 Throughput, Scale, and Clustering

In order to fully realize the potential of DATASIM as a high-volume load testing platform for the enterprise xAPI infrastructure, it is necessary to be able to scale statement generation nearly linearly.

At this time DATASIM can be scaled in two ways:

- Duplicated horizontally to deliver higher total throughput to a single location
- Scaled vertically with more powerful hardware.

If a single user interface is used by multiple users to run different simulations, it is possible to load balance multiple Backend instances so that many concurrent simulations can be run reliably.

While vertical scaling has a natural upper limit, the current duplication model can scale linearly in terms of pure throughput. The limitation is that the duplicated instances are not coordinated with each other in any way. Instead, they represent n distinct independent simulations which makes the aggregated data less useful for analysis.

Additionally, in future use cases it will be important to run large coordinated simulations to enable timely reproduction of high throughput simulated scenarios (e.g. high resolution

instrumented simulators and real-world exercises) and without a single simulation driving the data generation, the dataset will be less meaningful.

This improvement involves adding clustered communication architecture to the DATASIM backend so that individual nodes collaborate to create a single cohesive simulation with near-linear scaling characteristics. It will be possible to do this and guarantee that the same Statements are generated with the same timestamps by a coordinated cluster for every simulation run. One technical hurdle which may or may not be surmountable is guaranteeing repeatable order of Statement delivery to an LRS from multiple cluster instances engaged in a single simulation.

7.2 Security and Persistence

DATASIM currently does not have proper user accounts or persistence. It will be important as the product matures and features are added to implement a standards-based Authentication, Authorization and Audit framework.

Aside from curbing potential misuse of the system, and creating accountability for all system actions, this will enable features like user preferences, the ability to save simulations by version, and the storing of LRS details to reduce the labor to an Analyst or Data Scientist using DATASIM in the real world.

7.3 Profile and Input Validation

One of the most important aspects of the DATASIM Project is its capability to validate inputs. Chiefly among these is its capacity as a workbench for xAPI Profile validation. This capability should be expanded and validation issues should be communicated clearly and with remediation suggestions.

This will help DATASIM become an even more invaluable tool to the DoD and to the xAPI community as a whole.

8. Appendix

8.1 Subset of the TC3 xAPI Profile

The applicable subset of the TC3 xAPI Profile can be found here:
https://github.com/yetanalytics/datasim/blob/master/dev-resources/profiles/tccc/cuf_hc_video_and_asm_student_survey_profile.jsonld

8.2 TC3 Sample Data

The applicable generated data from the TC3 xAPI Profile can be found here:
https://github.com/yetanalytics/datasim/blob/master/dev-resources/xapi/statements/tccc_dev.json

8.3 Yet Calibration Profile

The Yet Calibration Profile, used for simplified simulation behavior analysis, can be found here:
https://github.com/yetanalytics/datasim/blob/master/dev-resources/bench/calibration.jsonld

8.4 cmi5 (Fixed)

The conformant version of the cmi5 Profile used for generation in this effort can be found here:
https://github.com/yetanalytics/datasim/blob/master/dev-resources/profiles/cmi5/fixed.json

8.5 DATASIM Code Repositories and Documentation

Backend <https://github.com/yetanalytics/datasim>
Frontend <https://github.com/yetanalytics/datasim-ui>