Mining Artificially Generated Data to Estimate Competency

Robby Robson[1], Fritz Ray[2], Mike Hernandez[3], Shelly Blake-Plock[4], Cliff Casey[5], Will Hoyt[6], Kevin Owens[7], Michael Hoffman[8], Benjamin Goldberg[9]

ABSTRACT

The context for this paper is the *Synthetic Training Environment Experiential Learning – Readiness* (STEEL-R) project [1], which aims to estimate individual and team competence using data collected from synthetic, semi-synthetic, and live scenario-based training exercises. In STEEL-R, the *Generalized Intelligent Framework for Tutoring* (GIFT) orchestrates scenario sessions and reports data as experience API (xAPI) statements. These statements are translated into assertions about individual and team competencies by the *Competency and Skills System* (CaSS). Mathematical models use these assertions to estimate the competency states of trainees. This information is displayed in a dashboard that enables users to explore progression over time and informs decisions concerning advancement to the next training phase and which skills to address.

To test, tune, and demo STEEL-R, more data was needed than was available from real-world training exercises. Since the raw data used to estimate competencies are captured in xAPI statements, a component called DATASIM was added. DATASIM simulated training sessions by generating xAPI statements that conformed to a STEEL-R *xAPI Profile*. This facilitated testing of STEEL-R and was used to create a demo that highlighted the ability to map data from multiple training systems to a single competency framework and to generate a display that team leaders can use to personalize and optimize training across multiple training modalities.

This paper gives an overview of STEEL-R, its architecture, and the features that enabled the use of artificial data. The paper explains how xAPI statements are converted to assertions and how these are used to estimate trainee competency. This is followed by a section on xAPI Profiles and on the xAPI Profile used in STEEL-R. The paper then discusses how artificial data were generated and the challenges of modeling longitudinal development and team in these data. The paper ends with a section on future research.

1. INTRODUCTION

The research reported in this paper relates to the US Army *Synthetic Training Environment* (STE) initiative that "brings together live and virtual training environments, aiming to deliver accessible exercises that mimic the full complexity of the physical world" [2]. To support this initiative, the initiative is developing infrastructure and a suite of *Training Management Tools* (TMT) that permit diverse training systems – including desktop game-based, mixed reality, virtual reality, augmented reality, and sensor-instrumented live training – to be rendered and integrated in a single training environment and training to be optimized within this environment.

R. Robson, B. Goldberg, S. Blake-Plock, C. Casey, W. Hoyt, M. Hernandez, and F. Ray. Mining artificially generated data to estimate competency. In A. Mitrovic and N. Bosch, editors, *Proceedings of the 15th International Conference on Educational Data Mining*, pages 828–833, Durham, United Kingdom, July 2022. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.6852926

The goal of the *STE Experiential Learning – Readiness* (STEEL-R) project is to support the STE TMT with software that collects evidence from training scenarios and uses this to estimate team and individual competency and performance probabilities, recommend training interventions, and inform the design of training scenarios. Research to date has focused on US Army battle drills [3], i.e., short tactical team scenarios intended to train individuals and teams to an automated response level. These include cognitive, psychomotor, and affective skills and behaviors that can be trained in a series of training systems that progress from first-person shooter game-like environments such as Virtual Battle Space 3 [4] (*synthetic* in this paper), to mixed reality and augmented reality environments (*semi-synthetic*), and field exercises in which trainees are instrumented with sensors (*live*). Traditionally, observer controllers / trainers (OC/Ts) are present and can alter conditions on the fly to change difficulty or add stressors. A goal of the STE TMT is to accelerate development and skill retention by using data-driven automation and the capabilities of intelligent tutoring systems to support assessment and facilitate personalized coaching.

The training addressed by STEEL-R is *experiential*, meaning that learning and mastery require repeated deliberate practice under varied conditions. To support experiential skill acquisition, the underlying competency and predictive models in STEEL-R must take longitudinal data and progression into account. This type of training also heavily involves team tasks and *team dimensions* such as cohesion, communication, and backup behaviors [5]. This adds further complexity to the underlying models and places further requirements on the data that must be collected to generate and test these models. Since the demand for such data is too large to be met by small trials, and since it is important that STEEL-R demonstrate good results prior to deployment in high-stakes real-world training, we saw *artificial data* as the best way to proceed in the early and middle stages of our research. We use the term *artificial* rather than *synthetic* in this paper to avoid confusion with synthetic training.

This paper focuses on the use of artificial data, on the use of xAPI Profiles and DATASIM to produce artificial data, on the challenges encountered, and on the results obtained. We start with an overview of the STEEL-R architecture and it critical features, which is next.

2. STEEL-R ARCHITECTURE

Three systems play a central role in the STEEL-R architecture, shown in green in Figure 1. The first is the *Generalized Intelligent Framework for Tutoring* (GIFT) [6], which orchestrates scenario sessions. It connects to and collects data on trainee actions from training systems via connectors. GIFT examines these actions and assesses whether specified tasks, activities, and expected behaviors are performed or demonstrated at, below, or above expectations.

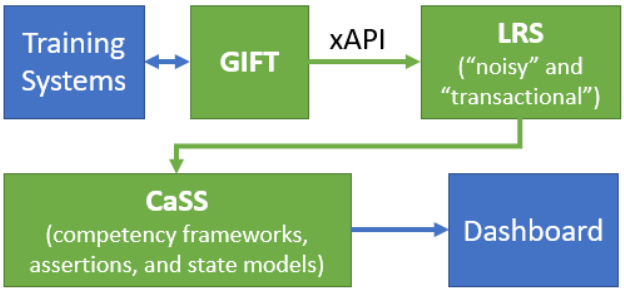

Figure 1: STEEL-R Component Architecture

The second core component is a set of two *Learning Record Stores* (LRSs). Every time an assessed performance state changes, GIFT reports the state and associated session data to a "noisy" LRS via experience API (xAPI) [7] statements. This noisy LRS captures everything that is happening in a session. A second "transactional" LRS filters noisy LRS data, retaining only the overall assessment of each trainee in a session. Data from the transactional LRS is polled by an instance of the *Competency and Skills System* (CaSS) [8], the third core component.

CaSS stores *competency objects* that represent the individual and team tasks, skills, knowledge, attitudes, and behaviors that a given instance of STEEL-R is intended to train and track. These are stored in *competency frameworks* that include relations among the skills, competencies, and behaviors these objects represent. As STEEL-R runs, CaSS collects xAPI statements and formulates *assertions.* An assertion is a statement to the effect that a trainee or team has or has not demonstrated competency based on identified evidence under specified conditions [9].

A GIFT performance assessment generated can generate multiple assertions about multiple competency objects. As explained in Section 3, CaSS computes competency states from these assertions using mathematic models. Competency states and other data are sent by CaSS to a Dashboard that can be used use to view states, track progress, and make informed training decisions. Throughout this process, the chain of evidence is preserved so an OC/T can review and audit it. Dashboard data can be traced back through CaSS, the LRS, and GIFT to trainee actions that can be replayed in a GIFT "gamemaster" interface.

A crucial feature of this architecture is that the data from all training systems is filtered through GIFT, where it is referenced against a common set of competency objects. Data from desktop games, mixed reality simulations, and live exercises can thus be combined to estimate and track the state of each single skill and competency.

2.1 STEEL-R Implementation

STEEL-R uses a Multiple Open System Architecture (MOSA) that integrates GIFT and the US Advanced Distributed Learning (ADL) initiative's Total Learning Architecture (TLA) [10] [11]. Most components (including GIFT and CaSS) are customized versions of open-source software. The LRSs are instances of the Yet Analytics SQL LRS [12], and all components are hosted on a hybrid container-based platform (Docker™). Since STEEL-R is intended to support field operation, it is designed for offline use and the entire system can be deployed on a mid-range rugged laptop. In field deployments, STEEL-R collects data in offline mode and forwards it to a cloud hosted instance when it comes back online. The mechanisms that permit offline operation enable STEEL-R to assemble data received from multiple sources at different times into a coherent sequence of events along a single timeline. This feature turned out to be crucial, as will be discussed in Section 5.2.

3. ESTIMATING COMPETENCY

The current version of STEEL-R considers three competency states – *untrained, practiced, and trained* – and three training phases – *crawl, walk, and run*. The three states are derived from US Army doctrine, while the training phases roughly correspond to synthetic, semi-synthetic, and live training The Dashboard component of STEEL-R informs OC/Ts how an individual or team is progressing from untrained to practiced to trained within each training phase and how ready they are for the next training phase, i.e., to move from crawl to walk and walk to run. In future Army versions, an *expert* state may be added, and different states and phases may be used for different application domains.

3.1 The Math Model

CaSS populates the STEEL-R Dashboard with longitudinal data about the competency state of each individual or team. These states are estimated using a mathematical model (the "math model") that involves a *repetition function,* an *evidence function*, and *rollup rules.* The repetition function represents the number of times a skill or competency has been trained, weighted by a forgetting function and a function that accounts for the value of spaced repetition. Similar methods are used in ACT-R [13] and the work of Jastrzembski and others on predicting future training performance [14]. The evidence function assigns a score between -1 and 1 that is derived from the history of performance assessments, taking skill decay, the trustworthiness of the evidence, and performance on related skills and competencies into account. Rollup rules allow performance on related skills and competencies to contribute to an evidence function and allow dependencies on performance under varied conditions and on the states of sub-skills to be added. More details model can be found in [15].

3.2 The Role of CaSS Assertions

The raw data used to evaluate the repetition and evidence functions comes from *assertions*. Assertions are a fundamental data type in CaSS that expresses conclusions drawn from evidence in a uniform way. An assertion can identify its source, the on which the source relied, the source of the evidence, the person or team and the CaSS competency object about which the assertion is made, a timestamp, a decay function, and a parameter that indicates the confidence its source has in the assertion [15]. Assertions can assert that a skill or competency was or was not demonstrated or is or is not possessed. Assertions can also identify the conditions under which the evidence was gathered e.g., difficulty factors and stressors.

CaSS computes the repetition and evidence functions from data in assertions. For example, if a competency object represents the ability of a team or person to perform a task, CaSS examines all assertions about their performance on that task and uses these to determine when and how often and with what results the task was attempted under varied conditions. This information is used to compute the repetition and evidence functions, which are in turn used to estimate whether they are untrained, practiced, or trained within the crawl, walk, and run phases.

3.3 Generating Assertions from xAPI

GIFT does not directly make CaSS assertions. Instead, it assesses actions, activities, and expected behaviors based on data from a training system and emits xAPI statements that require translation into CaSS assertions. CaSS does this with a *decoder*. The decoder has a lookup table that maps activities to competency objects and specifies how the three states reported by GIFT (at, above, and

below expectations) translate into positive or negative assertions about these objects. At present, this lookup table is hard-coded based on subject matter expert (SME) input.

3.4 The Need for Data

The math model and decoder have weights and parameters that can be set in a STEEL-R instance and control competency estimates. As STEEL-R develops, these will be used to compute performance probabilities and to recommend interventions and scenario designs. The system is designed so that its weights and parameters can be machine-learned, but at this stage they are manually set based on experimentation guided by theory. This requires significant data, and machine learning will require even more.

The best data would be data from real-world training exercises. Unfortunately, there are limited opportunities to deploy STEEL-R in such exercises, and since many involve high-stakes training, STEEL-R must be thoroughly stress-tested and shown to produce reasonable results before deployment can be considered. For these reasons, we took the approach of generating artificial data.

Referring to Figure 1, there is a choice as to where artificial data is inserted. One choice is to inject it into CaSS in the form of assertions. This can be used to test the math model, and early on we developed a small web app to do this, see Section 5. This allowed us to check formulas and code and to demonstrate how evidence affected competency estimates, but it was not sufficient to test the entire architecture. As a result, we decided to generate artificial xAPI statements that mimicked those generated by GIFT. This involves xAPI profiles, which are explained next.

4. XAPI PROFILES

The experience API (xAPI) is a mechanism for reporting and retrieving learner activities in an *actor – verb – object – context - results* format [7], [16]. xAPI *statements* in this format are sent to an LRS where they can be retrieved by other systems with appropriate permissions. xAPI statements are usually generated by an education or training system such as an LMS, simulation, or intelligent tutoring system, but statements can come from another LRS, as is done in the STEEL-R handoff between the noisy and transactional LRS. This ability enables multiple LRSs at the edge of a network to feed a central LRS, which improves scalability. The xAPI specification, which includes specifications for LRSs, was first developed by the ADL and is now undergoing more formal standardization in IEEE [16].

The xAPI specification is intended to be usable in any education or training ecosystem. To maintain flexibility, it does not specify the context or semantics of xAPI statements. In implementations it is necessary to add definitions and place restrictions on the format and elements in statements to ensure that data is properly reported and interpreted. This is done via *xAPI Profiles* [17].

xAPI Profiles define concepts, templates, patterns, and extensions for use in forming xAPI statements. *Concepts* define the vocabulary and attributes that may appear in xAPI statements, including verbs and activity types, and specify rules for how and when they can be used. *Templates* provide rules for constructing statements. *Patterns* are collections and sequences of templates that describe the actions associated with a task, performance, or learning path. *Extensions* enable new (externally defined) attributes to be used in statements. Together, these rules and definitions enable xAPI statements to be properly formed and interpreted. The xAPI profiles specification [18] establishes rules for serializing profiles in JavaScript Object Notation (JSON) and in JSON for Linked Data (JSON-LD). Using JSON-LD, vocabulary can link to the same or similar terms in other profiles, creating a semantic web of xAPI statements.

4.1 Designing the STEEL-R xAPI Profile

A critical factor in designing any xAPI Profile is creating concepts, statement templates, and patterns that are flexible enough to be used in many different scenarios but restrictive enough to enable data to be reported and understood in use cases of interest. The challenge for STEEL-R is that STEEL-R is meant to support many types of experiential learning. Even in relatively narrow domains, it may be necessary to track and capture data on hundreds (or more) tasks, activities, and behaviors. Profiles could be created that specify the names of tasks and performance levels for each domain, but a more flexible approach is enabled by exploiting the capabilities of GIFT.

4.2 The STEEL-R xAPI Profile

As a scenario session unfolds, GIFT determines if performance on tasks, activities, and behaviors stored in a Domain Knowledge File (DKF) exceeds, meets, or is below expectations [19]. At present, CaSS only uses summative assessments at the session level, but GIFT generates a formative assessment each time a performance state changes and can record information about the exercise, such as the conditions under which performance was assessed.

The xAPI profile designed for STEEL-R uses statement templates that enable tasks, activities, and behaviors to be referenced from the DKF and that report results on the GIFT three-step scale. This simplifies the form of statements, leaving the list of specific activities to GIFT. The STEEL-R templates also allow scenario conditions to be reported and include extensions for linking a GIFT assessment to a recording of the session segment that produced it.

To form a complete chain of evidence, STEEL-R xAPI statements can capture current performance states, changes in performance state, the factors that changed, and the conditions present when the state changed. Of particular interest to our future research is the ability to identify stressors and difficulty factors, both of which can be dynamically altered mid-session. Stress and difficulty are now being included in xAPI statements as discrete variables that are evaluated by GIFT and that CaSS can use in its math models.

Patterns represent the lifecycle of trainee participation in a training session. The templates in these patterns are populated by system events such as starting or joining a session, interactions within the session that could result in changes to a trainee's psychomotor, cognitive, or affective state, and GIFT's conclusions about a learner's overall performance with respect to specified tasks. Event data reported by GIFT is used to select the appropriate template and to filled in the template based template rules.

5. GENERATING ARTIFICIAL DATA

For testing purposes, artificial data was generated in two ways. The first was through a small app that was purpose-built to test and demonstrate the math model. This app allows users to apply hard-coded assertions about competency objects in a framework and displays how the repetition and evidence functions, competency state, and performance probabilities change with each statement.

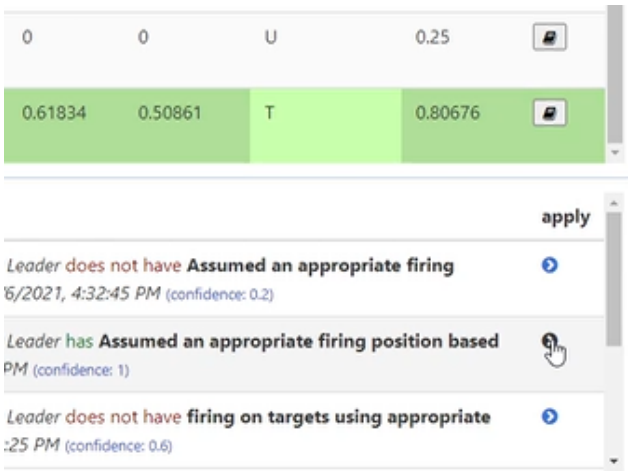

Figure 2: User applying a competency assertion

A second screen shows how the estimated probability of successful performance varies over time as both positive and negative assertions are activated. This visualization proved useful for both demonstrating and validating the math model.

Figure 3: Display showing how applying assertions changed the estimate of performance probability as evidence accumulated.

5.1 DATASIM

The second method used to generate artificial data used an open-source component of the ADL's TLA reference implementation [11] called the *Data and Training Analytics Simulated Input Modeler* (DATASIM) [20]. DATASIM can produce xAPI datasets that conform to one or more xAPI Profiles at small scale (tens to thousands of statements) and large scale (over a billion statements). DATASIM is controlled by a *simulation specification* that a user defines via a user interface. Each simulation specification includes an xAPI Profile, the actors in the simulation, and parameters that specify the involvement of each actor in each type of activity as well as start and stop times, a seed value, and the maximum number of xAPI statements to be generated.

Within a given simulation, DATASIM generates a pseudo-random autoregressive moving average (ARMA) time series (called the *common* time series) and a pseudo-random ARMA time series for each actor, each determined by the seed value. An actor generates an xAPI statement whenever their time series graph crosses the common time series graph [21]. When that happens, a Gaussian that is weighted by parameters in the simulation profile is sampled for each possible pattern and the pattern with the highest value is used. The same is then done to select templates in the pattern, statements in each template, and concepts in each statement.

5.2 Applications to STEEL-R

We used DATASIM to benchmark and stress-test STEEL-R, which was the original purpose of DATASIM [21]. By using the same actors, we could simulate multiple successive training sessions across each training phase and by referencing the same competency objects in the CaSS decoder, we could generate assertions about same competencies, skills, and behaviors at each stage. This let us validate system operation and benchmark performance at every point and for every component downstream of GIFT. It did not, however, give us the desired level of realism.

Although xAPI profiles enable DATASIM to generate data that statistically reflects the right mix of training events and outcomes at the macro level, DATASIM has no mechanism that allows a given actor to develop competency as they engage in successive activities and no mechanism to realistically correlate individual and team behavior. Thus, if DATASIM is used to simulate people performing a series of tasks, it will produce about the right number of successful and unsuccessful task completions but cannot alter its parameters during a simulation so that a person who successfully completes the early tasks will be more likely to successfully complete later ones. Similarly, if one of the actors is a team, DATASIM, the probability of team events cannot be changed based on the activities of team members during a simulation. Since longitudinal data and developmental progression are fundamental to experiential learning, we needed a way to reflect individual development and team dynamics.

We did this by running series of micro-simulations instead of one large one and by manually set parameters between runs. Each micro-simulation produced data for the same actors in a small time slice. This gave us greater control over the progression of outputs and implied team dynamics. STEEL-R treated these as offline data, automatically stringing them together to create a sequence of activities along a single timeline. This produced enough data to test and tune STEEL-R and to implement the use case described next.

5.3 A Use Case and Implementation

In November of 2021 we used DATASIM to implement and demo a use case in which a small team underwent three days of training. Our goal was to highlight how team competency improved and progressed from crawl to walk as interactions with multiple training systems activated cognitive, psychomotor, and affective skills.

On day one, the team trained on Army battle drill 6 (BD6) [3] in a synthetic game-based environment. This was done in multiple sessions with under varied conditions and with varied difficulty levels. DATASIM micro-simulations were manually configured to show performance improvement over the course of day one. On day two, BD6 training continued in a mixed reality environment that activated psychomotor skills. The data from day one showed that the team knew what to do, so day two provided opportunities to apply that knowledge in a safe controlled environment with more realistic interactions. The data generated by DATASIM represented exposure to numerous scenarios and showed further performance improvement. At the start of day three the team leader looked at the

Dashboard (Figure 4) and noticed that the team was progressing well on BD6 but there were potential skill decay issues with a related task trained in a previous battle drill. As a result, the team leader initiated training of this previous battle drill. The third day of training activated some of the same skills as the first two days and resulted in improvements in skills that seemed to have decayed.

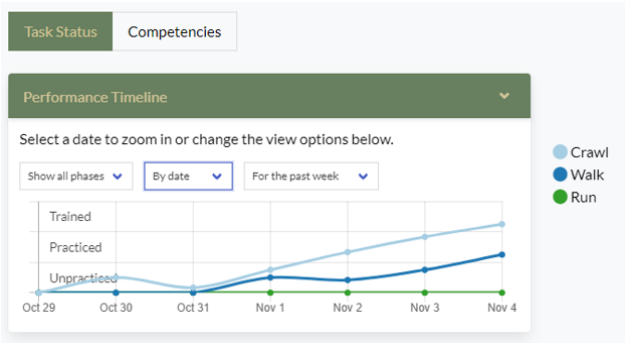

Figure 4: Part of the Dashboard, showing progress on reported by CaSS and derived from artificial data.

6. CONLUSION

The use case we implemented and demonstrated with artificial data showed the art of the possible. It showed a team leader using competency estimates derived from multiple and varied training environments to personalize a training plan and the potential to optimize training time by leveraging multiple training modalities within a given training cycle. The ability to mine artificial data enabled us to stress-test and benchmark STEEL-R and permitted us to visualize the effects of parameters in the formulas and models used to estimate competency. This served as an excellent tool for debugging, tuning, and demonstrating the models, and we continue to take this approach as we make changes and add features.

We note that as of November of 2021, neither difficulty or stress levels were reported by GIFT or used by CaSS. These are critical factors that can be manipulated during training sessions and that should be used when determining whether an individual or team is trained and ready to advance. GIFT is now reporting difficulty and stress levels in xAPI statements, and we are incorporating difficulty and stress into the decoder, math model, and Dashboard.

Finally, our work with DATASIM exposed the need to model the progress of simulated individuals and to correlate individual and team behaviors. We did this with manually manipulated micro-simulations, which is labor intensive and will not scale. In this regard, we are exploring two further research directions. The first is to implement ways to dynamically alter the parameters used to generate artificial data during a single simulation. The second is to enable the parameters that control these alterations to be machine-learned. These will create a virtuous cycle wherein artificial data are used to test and tune new features and models, these features and models are used to improve real-world training, and real-world training data are used to improve the generation of artificial data.

7. ACKNOWLEDGEMENTS

This work was supported by U.S. Army Research Laboratory contract #W912CG20C0020. It is dedicated to Tom Buskirk who contributed to this project as a software developer and who died suddenly in 2021 at the age of 47. He was a joy to work with, was a talented developer, and is dearly missed by the STEEL-R team.

[1] Eduworks Corporation. robby.robson@eduworks.com
[2] Eduworks Corporation. fritz.ray@eduworks.com
[3] Eduworks Corporation. mike.hernandez@eduworks.com
[4] Yet Analytics. shelly@yetanalytics.com
[5] Yet Analytics. cliff@yetanalytics.com
[6] Yet Analytics. will@yetanalytics.com
[7] University of Texas at Arlington. kowens@arlut.utexas.edu
[8] Dignitas Technologies. mhoffman@dignitastechnologies.com
[9] U.S. Army DEVCOM Soldier Center
benjamin.s.goldberg.civ@army.mil

8. REFERENCES

[1] B. Goldberg *et al.*, "Forging Competency and Proficiency through the Synthetic Training Environment with an Experiential Learning for Readiness Strategy," presented at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando, FL, 2021.

[2] A. Stone, "US Army makes headway on Synthetic Training Environment," *Defense News*, 30-Sep-2021. [Online]. Available: https://www.defensenews.com/training-sim/2021/09/30/us-army-makes-headway-on-synthetic-training-environment/. [Accessed: 13-Feb-2022].

[3] U. S. Army, "Appendix J - selected battle drills," *Army Training Publication (ATP) 3-21.8*. 2022.

[4] BISIMS, "VBS3," *Bohemia Interactive Simulations*, 2022. [Online]. Available: https://bisimulations.com/products/vbs3. [Accessed: 04-Mar-2022].

[5] R. A. Sottilare, C. Shawn Burke, E. Salas, A. M. Sinatra, J. H. Johnston, and S. B. Gilbert, "Designing Adaptive Instruction for Teams: a Meta-Analysis," *International Journal of Artificial Intelligence in Education*, vol. 28, no. 2, pp. 225–264, Jun. 2018.

[6] R. A. Sottilare, K. W. Brawner, A. M. Sinatra, and J. H. Johnston, "An updated concept for a Generalized Intelligent Framework for Tutoring (GIFT)." 2017.

[7] ADL Initiative, "Experience API (xAPI) standard," *Experience API (xAPI) Standard*. [Online]. Available: https://adlnet.gov/projects/xapi/. [Accessed: 20-Feb-2022].

[8] ADL, "Competency & Skills System (CaSS)," *Advanced Distributed Learning Initiative*. [Online]. Available: https://adlnet.gov/projects/cass/. [Accessed: 04-Apr-2020].

[9] R. Robson and J. Poltrack, "Using competencies to map performance across multiple activities," in *Proceedings of the I/ITSEC*, 2017.

[10] P. S. Gallagher, J. T. Folsom-Kovarik, S. Schatz, A. Barr, and S. Turkaly, "Total Learning Architecture development: A design-based research approach," in *Proceedings of the I/ITSEC*, 2017.

[11] ADL, "Understanding the TLA reference implementation," *Advanced Distributed Learning Initiative*, 2022. [Online]. Available: https://adlnet.gov/guides/tla/service-definitions/TLA-Reference-Implementation.html. [Accessed: 28-Feb-2022].

[12] Yet Analytics, "SQL LRS," *SQL LRS*. [Online]. Available: https://www.sqllrs.com/. [Accessed: 27-Feb-2022].

[13] J. R. Anderson, M. Matessa, and C. Lebiere, "ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention," *Human–Computer Interaction*, vol. 12, no. 4, pp. 439–462, Dec. 1997.

[14] T. S. Jastrzembski, K. A. Gluck, and G. Gunzelmann, "Knowledge tracing and prediction of future trainee performance," Florida State University, 2006.

[15] R. Robson, X. Hu, E. Robson, and A. C. Graesser, "Mathematical Models to Determine Competencies," in *Design Recommendations for Intelligent Tutoring Systems - Competency-Based Scenario Design*, vol. 9, A. M. Sinatra, A. C. Graesser, X. Hu, B. Goldberg, A. J. Hampton, and J. H. Johnston, Eds. Orlando, FL: US Army Research Lab, 2022, pp. 107–112.

[16] IEEE LTSC, "P9274.1.1," *JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access*. [Online]. Available: https://standards.ieee.org/ieee/9274.1.1/7321/. [Accessed: 20-Feb-2022].

[17] ADL, "xAPI-profiles: A set of documents addressing the structure of and supporting services for xAPI Profiles," 2018. [Online]. Available: https://github.com/adlnet/xapi-profiles. [Accessed: 27-Feb-2022].

[18] IEEE LTSC, "P9274.2.1," *Standard for JavaScript Object Notation for Linked Data (JSON-LD) for Application Profiles of Learner Experience Data*, 2022. [Online]. Available: https://standards.ieee.org/ieee/9274.2.1/10570/. [Accessed: 27-Feb-2022].

[19] GIFT, "Domain knowledge file 2021-2," *GIFT Tutoring Portal*, 2021. [Online]. Available: https://www.gifttutoring.org/projects/gift/wiki/Domain_Knowledge_File_2021-2. [Accessed: 27-Feb-2022].

[20] ADL, "Data Simulator for TLA (DATASIM)," *ADL Initiative*, 2020. [Online]. Available: https://adlnet.gov/projects/datasim/. [Accessed: 24-May-2020].

[21] S. Blake-Plock, "DATASIM: Data and Training Analytics Simulated Input Modeler," Yet Analytics, Mar. 2020.