Measuring the Power of Learning.™

Research Report
ETS RR–18-28

The Evidence Trace File: A Data Structure for Virtual Performance Assessments Informed by Data Analytics and Evidence-Centered Design

Jiangang Hao

Robert J. Mislevy

December 2018

ETS Research Report Series

Since its 1947 founding, ETS has conducted and disseminated scientific research to support its products and services, and to advance the measurement and education fields. In keeping with these goals, ETS is committed to making its research freely available to the professional community and to the general public. Published accounts of ETS research, including papers in the ETS Research Report series, undergo a formal peer-review process by ETS staff to ensure that they meet established scientific and professional standards. All such ETS-conducted peer reviews are in addition to any reviews that outside organizations may provide as part of their own publication processes. Peer review notwithstanding, the positions expressed in the ETS Research Report series and other published accounts of ETS research are those of the authors and not necessarily those of the Officers and Trustees of Educational Testing Service.

The Daniel Eignor Editorship is named in honor of Dr. Daniel R. Eignor, who from 2001 until 2011 served the Research and Development division as Editor for the ETS Research Report series. The Eignor Editorship has been created to recognize the pivotal leadership role that Dr. Eignor played in the research publication process at ETS.

ETS Research Report Series ISSN 2330-8516

RESEARCH REPORT

The Evidence Trace File: A Data Structure for Virtual Performance Assessments Informed by Data Analytics and Evidence-Centered Design

Jiangang Hao & Robert J. Mislevy

Educational Testing Service, Princeton, NJ

Virtual performance assessments (VPAs), such as game- and simulation-based assessments, provide promising ways for assessing complex and integrated skills. However, the high cost, long development cycle, and complex scoring process significantly hinder the adoption of VPAs, particularly in large-scale assessments with tight deadlines and limited budgets. One significant impediment is the lack of consensus about how to capture and store evidence-bearing process data from each VPA session into a format that facilitates quality checks, scoring, and analysis. In current practice, data from VPAs are generally stored in logs, either as standalone files or as a set of entries in a database. Programmers have long used logs to save process information about software systems for monitoring and debugging the software system itself. These purposes are fundamentally different from the purpose of assessment, which aims to capture evidence to support the measurement of a test taker's proficiency with respect to targeted constructs. Collapsing information that might be needed for the two purposes into a single log file can lead to confusion, even confrontation, among VPA designers, software developers, and scoring experts. To mitigate the problem, in this paper we introduce the notion of a structured *evidence trace file* (ETF) as distinguished from the familiar but idiosyncratic term, *log file*. The ETF is intended to express captured telemetry data in a way that meets the needs from the assessment perspective. Its specification as a standalone deliverable allows VPA designers and scoring experts to cleanly define and assure the quality of the information they need from VPA sessions in forms they find most useful for assessment purposes without adding additional complexity or structures for the information that software developers need for debugging purposes. In turn, software developers will be clearly informed about what information is expected from the assessment perspective and can manage the data as they choose as long as they can deliver the specified ETF. In addition to its application in VPAs, the ETF structure can also be employed more generally with digitally based assessments (DBAs) in which response-process data are captured and used for assessment purposes, such as timing information, response changes, and tool usage in multiple-choice tests. As a known data structure, it can sometimes also be employed for additional purposes in VPA analysis, such as transmitting information across phases of multiple-stage analysis.

Keywords Evidence trace file; log file; game-based assessment; virtual performance assessment; EPCAL

doi:10.1002/ets2.12215

Developing a virtual performance assessment (VPA), such as game or simulation-based assessment (O'Neil, Baker, & Perez, 2016), is a complex process involving people with diverse expertise and training. As such, establishing common terminology for communicating key ideas is important for an efficient development process. Evidence-centered design (ECD; Almond, Steinberg, & Mislevy, 2002; Almond, Mislevy, Steinberg, Williamson, & Yan, 2015; Mislevy, Steinberg, & Almond, 2003) provides a general framework for designing and communicating about assessments, including VPAs. At the heart of ECD is an evidentiary argument: Particular kinds of things test takers do in situations with particular features provides evidence about targeted aspects of their capabilities.

With familiar large-scale multiple-choice tests, the test developer designs the necessary stimulus features into the items then predetermines the features of the test taker's work to evaluate and the procedures for evaluating them: The test taker selects an option, and the option is compared to the response key. The process becomes more complicated with VPAs (Behrens, Mislevy, DiCerbo, & Levy, 2012; Mislevy et al., 2014).

A simulation environment is designed so that the test taker encounters situations with features that will evoke evidence as that test taker acts in less constrained ways, often with different particulars as test takers follow their own paths through the task. For the same reasons, the particular actions that test takers take generally differ as well. From these actions, each

Corresponding author: J. Hao, E-mail: jhao@ets.org

taken in situations that individuals have worked themselves into, evidence-bearing features and patterns of actions will need to be identified and characterized as evidence about competencies. The information needed from the encounter for assessment purposes is the trace of relevant test-taker actions and situation features captured and coded in the ways they are needed in evidence-identification processing.

An evidence trace file (ETF), as introduced in this paper, is a data structure for expressing the information emerging from test takers' interactions with a VPA that is needed for assessment inferences. The basic structure accommodates a wide range of forms of VPAs, which facilitates data management and streamlines communication among designers, researchers, data analysts, and psychometricians. In this paper, we explain the role of the basic form of the ETF using ECD terms and representations. Extensions and variations are noted in the presentation (e.g., using the ETF structure for transmitting information across subsequent phases of evidence identification and measurement modeling) and in the closing discussion. We note that although the discussion of ETF focuses on VPAs, the same notion can be applied to more general digitally based assessments (DBAs) in which additional response-process data are captured along with responses and are used to help improve the assessment (Ercikan & Pellegrino, 2017).

In the following section, we describe the motivation for the ETF and its basic structure. In the next section, we locate the ETF with respect to the ECD proficiency, evidence, and task models. In the remaining sections, we take a closer look at how an ETF can be implemented, using examples from selected VPAs and a digital platform for delivering collaborative tasks (ETS Platform for Collaborative Assessment and Learning [EPCAL], Hao et al., 2017).The final section provides closing comments.

Evidence Trace File

Motivations

The original motivation for defining the ETF is to provide a common data structure to mediate between VPA designers and VPA software programmers. In current practice, assessment designers are usually supplied with very complex logs[1] from the VPA programmers in a format driven by the programmer's convenience. These logs usually include both the information the programmers believe is needed for assessment and the information they need for debugging the software system. Keeping a large amount of assessment-irrelevant information in the logs introduces possibly complicated steps to properly extract the assessment-relevant information that could hinder the evaluation of the completeness and usefulness of the logs. On the other hand, software programmers often need to spend extra efforts to accommodate the additional requirements from the assessment designers into their timeline, delivery priorities, and available data-capturing platforms if the requirements for assessment-relevant information are not clearly specified. These factors increase costs and add development time to the projects. An ETF clearly stipulates the evidence-bearing information that must be captured from the VPA and saved in a standard, organized way that facilitates subsequent analysis for assessment purposes. Assessment designers just need to specify an ETF as a standalone deliverable. The programmers will have the flexibility to capture logs for their purposes as long as they can deliver the needed ETF. In the following, we discuss this usage first and later note how, in some cases, the ETF can also be useful elsewhere in assessment processes.

The ETF differs from the log files that programmers use for debugging purposes in ways driven by programmers' and assessment experts' different uses of VPA activity information. Programmers produce log files, or files of so-called telemetry data, that contain the information they need in forms that serve their needs of monitoring and troubleshooting to ensure that the software behaves in the intended ways under specified conditions (e.g., Barik, DeLine, Drucker, & Fisher, 2016; Russo, Succi, & Pedrycz, 2015).They need to be able to trace activity and information to identify the causes of malfunctions in the system, but other than this, they do not necessarily need to identify specific information in specific ways.

As a result, the way to record the telemetry data optimized for the efficient work of programmers is not necessarily suitable for the assessment purpose. Because simulation and game environments and implementations can vary greatly, not to mention local preferences of software providers, these log files can be extensive and their formatting idiosyncratic. They may contain highly detailed information about program activity, much of which can be irrelevant to the assessment argument per se. Conversely, information that is necessary to the assessment may be not present. This omission can happen in several ways. A first example is when a computational sequence takes different evidentiary meanings under different conditions, but the information to distinguish the conditions is not captured. A second is the case of "the dog

that did not bark in the night"[2]: An important element of evidence is, say, an action a test taker does not take under certain conditions, but instances of those conditions are not identified and captured. In addition to the possible missing of information needed for assessments, the presence of sometimes vast amounts of assessment-irrelevant information in logs increases the effort that is needed to winnow the information necessary for the assessment.

Assessment designers, in contrast, want to systematically identify the information that will in some form constitute evidence about the constructs a VPA is meant to assess. Using the ETF structure, assessment designers can specify both the needed information and a specific data model for transmitting it. The interface between the software programmers and the assessment developers does not depend on the particular scope and format of the programmers' system log. In particular, when the ETF is specified as a standalone deliverable, the VPA designers and scoring experts[3] can cleanly define and quality-assure the information they need from VPA sessions in forms they find most useful for assessment purposes, and the software developers will be notified clearly about what information is expected for assessment and can choose their own ways to save the data as long as they can deliver the specified ETF.

Evidence Trace File Specification

There are two major components for the specification of an EFT. The first is a data model that governs the structure and data type of the ETF. The second is VPA-specific contents that are stored into the skeleton specified by the data model.

The standard way to specify a data model is to provide a schema file, which one can use to specify the data structure and types. The schema file also allows for a compliance check, that is, an automated check on whether a given ETF meets the structure and data types as specified by the schema file. In principle, there could be infinitely many possible data structures and data type conventions for different VPAs, although a data model that can be applied to a range of VPAs would be advantageous. For example, Hao, Smith, Mislevy, von Davier, and Bauer (2016) proposed a data model for a generic range of VPAs. This data model specifies a set of fixed attributes and a set of extensible attributes for each test-taker session of the VPA, the typical unit of analysis. Figure 1 shows the structure of the data model (see Hao et al., 2016, for details).

The specific form of the schema file can be specified in different ways, depending on the particular programming convention of a given project.XML and JSON are currently the most popular formats to store log data. (The appendix A and B give the XML and JSON schemata that correspond to the data model.) Given the schema file, computer code can be developed to check an ETF file's compliance with the schema file; other analysis pipelines can be developed to parse the evidence file to locate or further investigate evidence (e.g., as implemented in the ETS glassPy assessment data analytics solution; Hao et al., 2016).

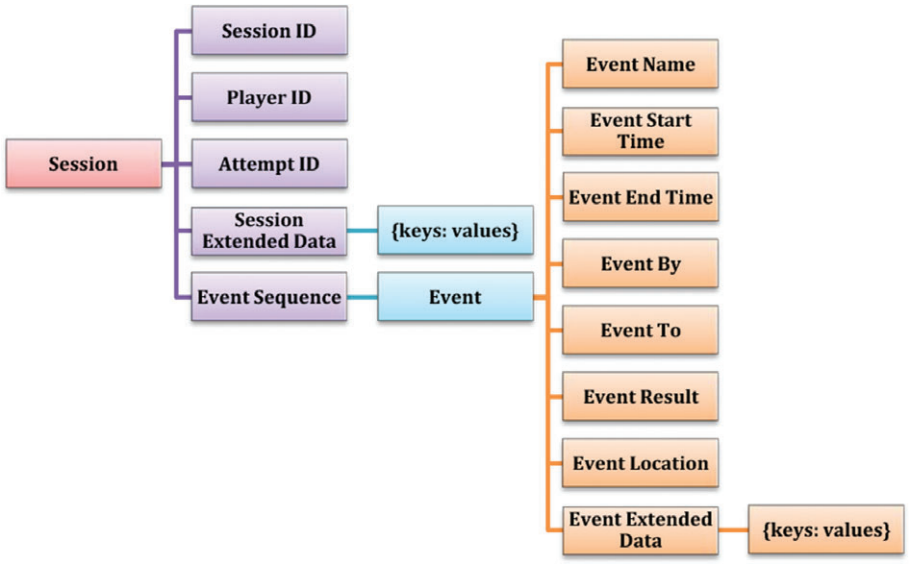


Figure 1 Structure of a generic data model for the evidence trace file of virtual performance assessment. Note: The attempt ID indicates the number of times the person has tried/attempted/playedthe VPA.

The second component of an ETF is the VPA-specific content to be stored in the file. As shown in the bottom right of Figure 1, the main content of an ETF is a sequence of events to be defined for the VPA application (which may be reused in other VPA applications whenever appropriate).

Although VPAs can vary considerably, there are some general guidelines about what eventsare to be captured. Specifically, test taker-triggered activities can be roughly put into two types. The first type involves the direct operations on elements in the VPA designed for assessment, such as clicks on buttons, moves of game elements, typed responses, checked options, and verification of the VPA system status—quite specifically, instances of meaningful actions test takers take in situations with particular features. The entry for any such activity will include timing information and the identification of the elements of the VPA that are involved and, as needed, values of variables for relevant aspects of the state of the environment before and after the activity.

A challenge regarding the last of these is determining what constitutes a sufficient description of VPA states to associate with a given action. The same test taker action can take different evidentiary meanings depending on prior actions or the state of the simulation. Thus an appropriate set of attributes and parameters for each state needs to be established for each activity in collaboration between the members of the interdisciplinary design team for the VPAs. In early stages of development, finer grained, lower level events may need to be included in the ETF so patterns among them can be further explored to determine exactly which actions and system state variables are needed to produce the required evidence (e.g., DiCerbo et al., 2015; Halverson & Owen, 2014; Roberts, Chung, & Parks, 2016).The subsequent operational version of the ETF may then need to contain only higher level computed events, if the required computation is straightforward to calculate.

The second type of test taker-triggered event involves activities that do not directly act on the VPA elements designed for assessment, such as clicks on the screen and typing on the keyboard. These may be of use in exploratory analyses to discover such patterns as inattention, random activity, and gaming the system (Baker, Corbett, Koedinger, & Wagner, 2004).

In addition to the test taker-triggered events, other system-triggered events can be defined in an ETF specification, including

- actions taken by other test takers, as in a VPA involving communication or collaboration;
- actions taken by the VPA simulation itself, as in a change in a patient's condition occurring over time in a medical simulation independent of the test taker's actions;
- prespecified reports of the simulation system status, such as a "heartbeat" recording at regular intervals of the values of a set of system-status variables that will be needed in subsequent processing to interpret test-taker actions (a set could include, for example, variables for simulation conditions such that certain configurations identify that a particular dog is not barking [Mislevy, 2018]); and
- the "result" of a test-taker action can be a pointer to an accompanying more complex object that is itself a work product, such as an audio file or a graphics file containing a figure the test taker drew on a pressure-sensitive tablet.

Evidence-Centered Design and Evidence Trace Files

While the references on ECD cited in the introduction provide additional detail, a brief sketch of the main components in one of the design layers within the ECD framework is sufficient to locate potential use of ETFs within the ECD framework. The relevant components are the proficiency model (student model), evidence models, and task models of the ECD conceptual assessment framework (CAF). Potential locations for defining ETFs are noted in this discussion and illustrated in the examples in the following three sections.

Figure 2 depicts the CAF models. They contain specifications for elements, messages, and information that are needed in an operational assessment. The proficiency model (1) contains variables representing aspects of test takers' capabilities in a measurement model. The assessor's belief about their values for a given test taker is based on actions in the assessment situation (2) producing work products (4). Key features of the assessment situation are expressed as task model variables (3). Evidence identification processes (5) evaluate the work product(s) to produce values of observable variables (6). In the measurement model, conditional probability distributions (7) for possible values of the observable variables are modeled as depending on some subset (8) of the proficiency variables.

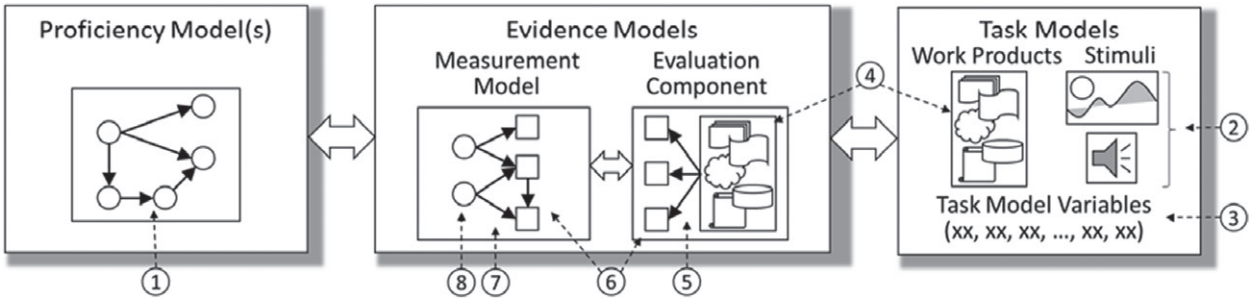


Figure 2 The main models of the evidence-centered design conceptual assessment framework.

The models in Figure 2 are meant to convey structure rather than process flow, but the necessary relationships among the components do hold implications about operational processes. The correspondence is most clear in a familiar paper-and-pencil multiple-choice test. The test taker is given a test form with distinct items, each consisting of stimuli and answer choices (2). They have been written previously by a test developer, each containing features (3) meant to elicit information bearing on the targeted proficiencies. The test taker interacts with the items and marks a choice for each item, producing the work product (4) consisting of a vector of selections that correspond one-to-one with items. This vector of responses could be expressed in the ETF structure, with each item-level data element containing an item identifier and a response and, if useful, item-feature codes. The evaluation procedure (5) is matching the test taker's selections with the correct choices to provide a string of observable variables (6) in the form of 0 s for wrong answers and 1 s for right ones. This string of evaluated responses could also be expressed in the ETF structure, to be passed to measurement-model processing.

Whereas familiar multiple-choice tests have predefined discrete tasks and stimulus materials, work products, and observable variables, a VPA typically presents a test taker with a digital simulation environment (2) with which to interact, and perhaps with other test takers as well. The simulation takes actions as well, changing some features of the environment (3) in response to the test takers' actions. There can be many different forms of initial work products (4), such as logs of low-level time-stamped key strokes and digital artifacts such as site designs or video or audio media recordings, and a given VPA may produce data of multiple types. The evidence-bearing information in some or all of these outcomes may be expressible as an ETF. Evidence evaluation (5) often consists of multiple stages of processing, as we indicate below.

Thus, the use of an ETF structure can be employed not only in the file provided to assessment analysts, but also to move information among the processes within phases of evidence identification. For example, some events may be determined as functions or configurations of lower level events (e.g., Kerr, Andrews, & Mislevy, 2016).The same ETF structure can often be used for the input, output, or both intermediate evidence-identification processes. Using the same data structure makes for efficiency in use, improved communication, and better quality control. Moreover, in research projects or in early stages of development, evidence-evaluation procedures may not be fully specified. The exact set of observable variables that will be needed for the higher level measurement modeling and processes needed to produce them are subject to exploratory analyses and refinement based on data from early trials. It can be useful in these cases to define ETFs at multiple levels to serve multiple purposes more efficiently. Some of these possibilities can be illustrated with the following examples.

Building an Evidence Trace File Around a Q-Matrix

A fundamental distinction of VPAs from games or simulations is that they are meant for assessing targeted proficiencies from evidence based on the test taker's activities in the environment. This feature generally adds additional complexity to the game development. For assessment, there must be a clearly stated scope of activities that provides evidence of having the proficiency (or proficiencies) as well as simulation elements that provide an environment that engages interaction with the proficiencies and motivates the interaction.

In some VPAs, assessment designers can specify the relationship between test taker – triggered events and targeted proficiencies using a representation known in the assessment literature as a Q-matrix (introduced by Tatsuoka, 1983).The Q-matrix establishes the mapping between situations in the VPAs and targeted proficiencies. The columns of a Q-matrix represent aspects of proficiency, typically at the level they will be included in the measurement model (i.e., those represented

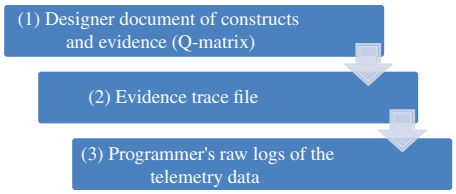


Figure 3 Relationship between the evidence trace file and other data files in the workflow.

by (1) in Figure 2) or at a finer grain size to help assessment designers determine features to include in VPA situations. The rows represent VPA situations, the salient features of which are expressed as values of task-model variables, represented by (3) in Figure 2. VPA situations may be predefined, such as preset activities or phases of challenges, or as evidence-bearing situations that arise as a student interacts with the VPA (as in the HyDrive example addressed later in the Use Cases for Evidence Trace Files by HyDrive section).The cell entries of the Q-matrix indicate the extent to which the event of that row is expected to contain information about the aspect of proficiency of that column — often simply 0 or 1 but possibly more varied. (Q-matrices can be augmented by additional information such as assessment developers' expectations about the amount of information or the degree of challenge in the interaction with regard to the proficiency, such as information about the parameters in the measurement model; Almond, 2010).

Shute, Ke, and Wang (2017) illustrated the use of Q-matrix in a game-based assessment of middle school mathematics topics called E-Rebuild. They used the Q-matrix to both guide VPA design and specify the observable variables, proficiency variables, and their interrelationships in the item-level modules of their Bayes net measurement model. They noted that a useful feature of the Q-matrix summary is to indicate to assessment designers where and how often information was being elicited for which proficiencies, so they could see if additional scenarios were needed to provide sufficient opportunities to exhibit each proficiency.

In defining an ETF for such an assessment, an assessment developer can define an attribute in the data model for each evidence-bearing event that consists of the entries of the row of the Q-matrix for that event. However, if the development of the VPA software is being provided by external programmers, it may be premature to ask the programmers to provide these values in early stages of the project. This early analysis of initial data may result in changes to the Q-matrix columns or cell entries as proficiencies or situation features may be added, dropped, or revised and the assignment of cell entries for events may be adjusted.

Figure 3 shows the relationship among three kinds of data files in a VPA that is built around a Q-matrix. (1) Assessment developers create a provisional Q-matrix in planning and early implementation. Its contents reflect information that is in the ECD proficiency, evidence, and task models, or in early light-weight sketches that are developed in connection with early design work and prototyping. (2) This forms the basis of defining an ETF, which programmers who code the simulation environment have as a requirement in their development work. (3) The software programmers may define log files for their development purposes that differ from the ETF, but because they must produce the ETF, these log files will not cause conflicts with assessment developers' purposes. As noted in the preceding paragraph, however, it can be the case that in early development stages, revisions to the VPA design may necessitate revisions in the ETF definition. These revisions should be minimal at later stages of the development process.

Use Case for an Evidence Trace File in SimCityEDU: Pollution Challenge

SimCityEDU: Pollution Challenge! is a learning game with integrated formative assessment developed by GlassLab in partnership with the Electronic Arts, Pearson, and ETS (Mislevy et al., 2014). In a simulation environment adapted from the SimCity commercial game, students play the role of mayor in a series of four challenges addressing environmental impact while balancing the employment needs and the happiness of the city's residents. This VPA provides formative assessment information about students' ability to problem solve and explain the relationships in complex systems. SimCityEDU: Pollution Challenge! was one of the projects in which experience suggested the usefulness of an ETF. The following discussion is used to highlight the role it would play and its location in the design and operation of this VPA.

Figure 4 summarizes evidence evaluation processes that are represented by the simple arrows (5) in Figure 2. The long, shaded boxes represent processing stages, which may themselves contain multiple processes. This coarse level of detail

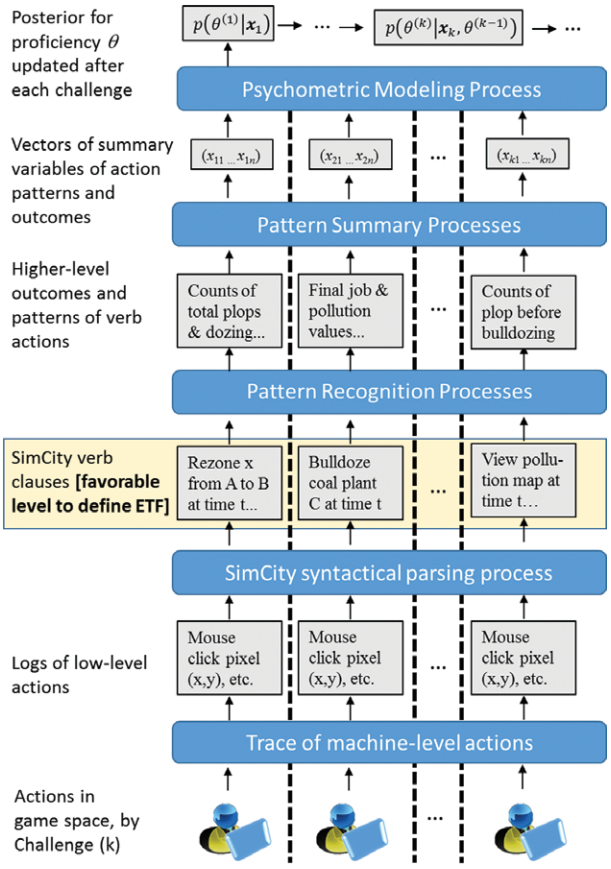


Figure 4 Stages of evidence evaluation processes in SimCityEDU: Pollution challenge!

is sufficient to indicate the location for employing an ETF. The columns represent the series of distinct challenges, with a student interacting with the VPA depicted at the bottom. The student's actions are, at the lowest level, points, drags, clicks, characters typed, and so on. At this level, the system programmers created and analyzed logs of student and system events for developing, testing, and debugging the simulation code. It is not generally necessary for assessment designers to interact with these log files if the information they need can be expressed in ETFs.

An effective location for defining an ETF to articulate between the software developers and the assessment experts is the next level up, labeled "SimCity verb clauses." These are student actions in terms of semantically meaningful acts in the game space. Rezoning a parcel of land from one category to another is an example; checking the pollution map is another; bulldozing a power plant is another. These clauses can be expressed in the basic form of the ETF events described previously. The game interface can include multiple ways to carry out such actions, which are identified and evaluated in the SimCityEDU program code in the processes labeled the "SimCity syntactical parsing process." In SimCityEDU: Pollution Challenge! this distinction is not evidentially relevant and need not be captured in the ETF. (Such distinctions might be relevant in other VPAs, depending on the intended inferences and the information needed to support them.) The SimCityEDU software also creates a heartbeat event summarizing a vector of system-state variables every 30 seconds, which would also be included in the ETF.

SimCity verb clauses contain information that provides evidence about the level of systems thinking represented by a student's play in a challenge, but verb clauses alone need further analysis to determine just what that evidence is. DiCerbo et al. (2015) describe some of the analyses carried out with data at the verb-clause and heartbeat level to determine patterns of verb-clause events that provide evidence of students' understanding and strategies. These analyses were exploratory investigations at the process level labeled "Pattern recognition processes." Their finalized forms were summarized in the form of values of observable variables, indicated as (6) in Figure 2. These in turn were the input variables in the measurement modeling process, the vector of evidence used to inform belief about the level of systems thinking the student

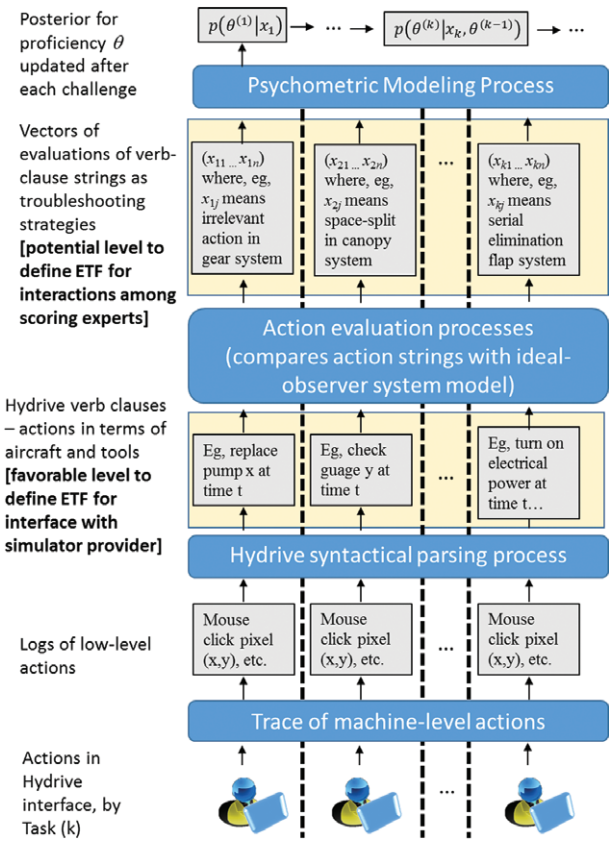


Figure 5 Stages of evidence evaluation processes in HyDrive.

had displayed in the challenge. The key point is the encapsulation of software programmers' data needs and assessment developers' data needs, which can be achieved clearly and efficiently through the specification of an ETF.

Use Cases for Evidence Trace Files in HyDrive

HyDrive was a coached practice system designed to help Air Force trainees learn to troubleshoot the hydraulics system of the F-15 aircraft (Steinberg & Gitomer, 1996).The problems, the interfaces, and the feedback in HyDrive are built around information-processing notions: defining an active path in a problem space, carrying out test procedures, and applying strategies such as space-splitting and serial elimination. Assessment is carried out continuously as trainees carry out information-gathering and repair actions in the simulation space (Mislevy & Gitomer, 1996; Steinberg & Gitomer, 1996). HyDrive, like SimCityEDU, was a project in which experience suggested the usefulness of an ETF. The following discussion highlights the roles the ETF structure could play and its locations in the design and operation of this VPA.

Figure 5 shows stages of processing from trainees' interactions in the simulation environment, suggested at the bottom of the figure, to measurement modeling, at the top. The processing in the evaluation component of the ECD evidence model, represented by the arrow labeled (5) in Figure 2, encompasses the trace of machine-level actions through the action evaluation processes.

As in SimCityEDU, trainee's interactions with the aircraft simulation produce a trace of low-level actions that are not, in themselves, semantically meaningful. The software programmers needed this information—and in fact information at even finer levels of detail—to develop and troubleshoot code. However, the assessment developers could again begin from the level of syntactically parsed verb clauses, such as replacing Pump x at Time t or checking Gauge y at Time z—basic ETF events. This level is thus a first location at which an ETF could be defined for HyDrive. Its roles are specifying to a programmer the data form that the VPA code must produce and designating the information that will be captured and conveyed for subsequent assessment purposes, such as refining evidence-evaluation routines and analyzing trainees' solution strategies.

Similar to SimCityEDU, neither individual actions nor counts of them express the evidence they may hold about a trainee's knowledge and strategies; it is only through higher level characteristics of the strings, as seen through the lens of the proficiencies at issue and how they manifest. This higher level evidence processing is implemented by HyDrive VPA's Action Evaluator module (Steinberg & Gitomer, 1996).It is a program that calculates the implication of each trainee action on potential information about the so-called active path of components that the presence of a problem has revealed; for example, if the pilot wiggled the stick and the flaps did not move, then the components between the stick and the flap actuators constitute the active path of components along which the fault lies. Troubleshooting actions along the path, such as checking component inputs and outputs and replacing components, provides information about the location of a fault. For each sequence of moves that holds the possibility of providing the trainee with information, the action evaluator determines whether each trainee's action sequence is, from an ideal information-evaluator's perspective, consistent with space-splitting (an optimal move), serial elimination (a good move), remove-and-replace (effective but inefficient), redundant, or irrelevant. It is the string of results of each of such evaluation that are the observable variables (6) in Figure 2 that are the input to the measurement model (7).

This string of observable variables is a second location in the HyDrive VPA in which data can be expressed in the structure of an ETF. These are the data in Figure 5 labeled "Vectors of evaluations of verb-clause strings as troubleshooting strategies." The events defined in the ETF at this second possible location would be inferred as troubleshooting actions, each in a given equivalent class of events (e.g., action in a flaps-system situation in which space-splitting is possible).

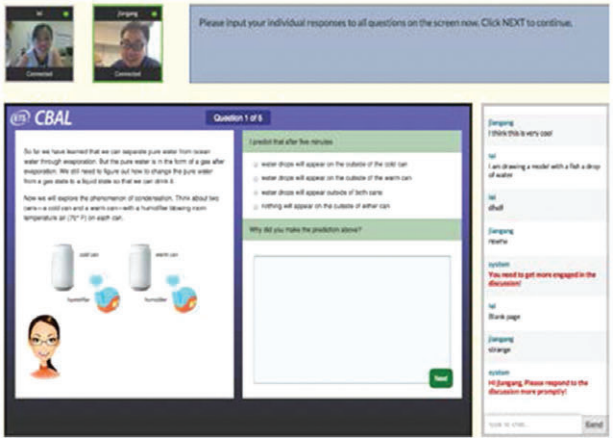
These are the key differences between ETFs defined at the two locations: The lower-level verb-clause of overt actions can be readily produced by software programmers who write the code for the VPA interface. They do not need to evaluate the evidence, but the information they are tasked with providing in the specified ETF is the minimally necessary information from trainees' actions needed for evidence evaluation. Their responsibilities and their work need not include the action evaluator. The action evaluator, on the other hand, is needed to produce the evidence contained in the higher level evaluated-action ETF. It can take the verb-clause ETF, along with its own model of the aircraft system, and calculate the inferred actions that constitute evidence about trainees' proficiencies. As noted, this output can be expressed in the structure of an ETF as well, now specified in terms of events that are inferred actions. Its contents would depend on the nature of the evidence-evaluation stage, such as the identification of a most likely value, an identification accompanied by a confidence level, or a vector of likelihoods, posterior probabilities, or fuzzy-set membership across the set of possible values. If data at this level are deemed as input to a measurement model, the structure and parameterization of which is yet to be determined, scoring experts who work on action evaluation and on measurement modeling can use this higher level ETF to transmit the data as they explore measurement-modeling alternatives.

Implementation of Evidence Trace Files in Digitally Based Assessments

So far, we have introduced the ETF and shown its location in two examples of VPAs: SimCityEDU and HyDrive. As we mentioned earlier, the ETF notion is not only applicable to VPAs, but also applicable to more general DBAs (consisting of even simpler items such as multiple choice) in which the test taker's response process information is needed for improving the assessments. For example, in a digitally delivered reading comprehension test consisting of multiple-choice questions, test takers' response process information, such as whether they visited the relevant part of the passage before answering a particular question, could potentially help us to identify whether or to what degree various test-taking approaches are reflected in test takers' performances. This information helps designers improve an assessment by either refining the assessment design or flagging items, test takers, or individual responses that are seriously affected by unintended test-taking strategies (such as rapid guessing).

To use the response process information, people need to record the response process data. Currently a common practice is to record the response information into separate logs (a set of files or a set of entries in a database).The same confusions or confrontations between assessment designers and software programmers that were described in the ETF section for VPAs appear here too. AnETF can play a beneficial role here for the same reasons. Moreover, as assessment designers can specify the format of the ETF, it is possible to develop cross-task and cross-assessment data analytics tools that make the data analyses more efficient and cost-effective. For example, such a workflow was implemented in the development of the EPCAL (Hao et al., 2017) and in the glassPy analytics solution (Hao et al., 2016).

The EPCAL platform was designed by ETS and programmed by a software vendor, Pragmatic Solutions (PS). Most software vendors have their own data platforms that can capture the data and provide some basic analytics such as descriptive




```
<gameLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <session>
    <sessionID>4570</sessionID>
    <teamID>jiangang_jj</teamID>
    <playerID>...</playerID>
    <attemptID>1</attemptID>
    <sessionExtData>...</sessionExtData>
    <eventSequence>
      <event>
        <eventName>question</eventName>
        <eventStartTime>2017-11-30T20:18:29Z</eventStartTime>
        <eventEndTime>2017-11-30T20:18:29Z</eventEndTime>
        <eventBy>jj</eventBy>
        <eventTo>cbal-1-0</eventTo>
        <eventResult>2</eventResult>
        <eventLocation>slide2-step0</eventLocation>
        <eventResultAltData/>
        <eventExtData>...</eventExtData>
      </event>
      <event>
        <eventName>question</eventName>
        <eventStartTime>2017-11-30T20:18:29Z</eventStartTime>
        <eventEndTime>2017-11-30T20:18:29Z</eventEndTime>
        <eventBy>jj</eventBy>
        <eventTo>cbal-1-1</eventTo>
        <eventResult>huinuin</eventResult>
        <eventLocation>slide2-step0</eventLocation>
        <eventResultAltData/>
        <eventExtData>
          <pair>...</pair>
          <pair>...</pair>
        </eventExtData>
      </event>
```

Figure 6 Left: Screenshot of the user interface of EPCAL. Right: A snippet of the evidence trace file that records the response process information for assessment purposes.

statistics. For example PS has its own data platform, called "Leverage." However, as most of the software vendors are not experts in educational assessment or psychometrics, the data analytics components of their data platforms are rarely tuned to the more specialized needs of data exploration to support a complex assessment. Therefore, a more specialized analytics capability is generally needed for those who design and deliver assessments. In this regard, the ETF not only plays a middleman's role to interface the assessment-relevant information and the software vendor's existing data platform, but it also facilitates the development of an efficient and cost-effective data analytics system from the assessment designer's side.

In the implementation of the EPCAL project, we developed an ETF schema in XML format following the data model as shown in Figure 1. We then specified the response process information we needed for our assessment purposes into the ETFschema and included the corresponding ETFs in vendor specifications. Figure 6 shows the user interface of the EPCAL system (left) and a snippet of an ETF (right) that records response process information from a given session. Upon receiving the ETFs, our data analytics pipeline first performs an automatic compliance check against our schema. We notify the programmers to fix the issues if the ETF fails this first "sanity check." After passing this check, the ETFs are analyzed in ways that serve the assessment needs. An analytics system with a library of convenient functions has been developed to make this process efficient and cost-effective (e.g., glassPy; Hao et al., 2016). This existing analytics tool allows us to process the delivered ETFs efficiently and to provide feedback to the vendors regarding missing or redundant information through the exchange of ETFs. Such a workflow has reduced the extra work that would be needed to parse the arbitrarily structured log files created at programmer's convenience. This effort becomes nontrivial if there are a number of different tasks. On the other hand, the programmers are better informed about what needs to be delivered rather than busy with patch-making processes that complicate the logs in formats they feel are inconvenient.

Discussion

In current practice, both assessment designers and software programmers often refer to the file that captures telemetry information from a VPA as "the log file," even though their notions of the term can be fundamentally different given the nature of what data they need and how they will use the information. The assessment designers' goal is to identify the evidence from VPA interactions to assess the proficiency of the test taker. They need to make sure all the information that contains the required evidence is captured and can be retrieved in a convenient way. Imposing conventions to structure the way such a log file is saved will make their work more efficient. On the other hand, software developers' goal is to ensure that the VPA functions properly and that the data they need are captured for monitoring, refining, and debugging the VPA. With such a goal in mind and without knowledge of assessment-argument requirements, software programmers tend to save a log file in a way that makes their work easier, either by simply dumping the telemetry information line by line or storing it in a structure they are familiar with but that is not attuned to assessment development and analysis.

If there is only to be a single log file, debates arise between assessment designers and software programmers regarding the scope of the process data to be captured and how to record it. These debates, complicated by the constraints due

to budgets and schedule, can escalate into conflicts. For programmers, saving the telemetry data into a log file with a specific structure and complications for assessment purposes creates extra work that is generally not explicitly defined or spelled out as a deliverable. Programmers' flexibility to save the data may also be limited by their available infrastructure and programmer experience. The assessment designer who has received a log file that has been developed to suit the programmers' convenience faces the additional complication of parsing the log files that may not even contain all of the necessary information that could have been provided.(The authors have worked on projects in which it took months for assessment analysts to fully understand, quality-check, and recode log files so the analyses needed for assessment purposes could be conducted — sometimes resulting in going back to the software providers for rework or even losing an experiment or study due to flawed data collection.)

Developing VPA is an inherently challenging endeavor. We suggest that a simple and actionable change may help significantly: rethinking the use of the term log file. It has a deep-rooted meaning among software programmers — a meaning that does not take into account the requirements of the VPA to serve as both an assessment and a game or simulation. As such, we suggest introducing the term "ETF," with a basic form described in the ETF section. Using ETF has the following benefits:

- ensuring data requirements for assessment purposes are recognized early in the design process;
- capitalizing on assessment developers' knowledge of assessment-argument requirements, in ways consistent with ECD principles;
- serving to specify delivery requirements to software programmers, so that assessment data needs are met without their having to understand the technical details of the assessment aspect of the VPA; and
- providing a standard form of input to develop cross-task analytics solutions for analyzing VPA data efficiently and cost-effectively (such as the Hao et al., 2016, GlassPy package).

Notes

1 For brevity we will use "log" or "log file" to refer to both the database and the standalone file modes of recording, as the distinction is not relevant to present purposes.

2 This saying is based on the theme of Sir Arthur Conan Doyle's Sherlock Holmes mystery, *The Adventure of Silver Blaze*. Here is the key passage from the story: Detective Gregory: "Is there any other point to which you would wish to draw my attention?" Holmes: "To the curious incident of the dog in the night-time." Gregory: "The dog did nothing in the night-time." Holmes: "That was the curious incident." Doyle, A.C. (1892). *The Adventure of Silver Blaze*. London, England: Methuen.

3 "Scoring experts" is used as an umbrella term to include experts who carry out the data analyses and modeling that ground inference from test-takers' performances for assessment purposes. These functions can include exploratory data analysis, data mining, knowledge engineering, and psychometric modeling. There could be one or several such experts in a given VPA project, but we refer collectively to the information they need as "information for assessment purposes." Ideally their work is closely coordinated with that of the assessment designers.

References

Almond, R., Steinberg, L., & Mislevy, R. (2002). Enhancing the design and delivery of assessment systems: A four-process architecture. *The Journal of Technology, Learning and Assessment, 1*(5), 1 – 63.

Almond, R. G. (2010). I can name that Bayesian network in two matrixes! *International Journal of Approximate Reasoning, 51*, 167 – 178. https://doi.org/10.1016/j.ijar.2009.04.005

Almond, R. G., Mislevy, R. J., Steinberg, L. S., Williamson, D. M., & Yan, D. (2015). *Bayesian networks in educational assessment*. New York, NY: Springer. https://doi.org/10.1007/978-1-4939-2125-6

Baker, R. S., Corbett, A. T., Koedinger, K. R., & Wagner, A. Z. (2004). Off-task behavior in the cognitive tutor classroom: When students game the system. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 383 – 390). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/985692.985741

Barik, T., DeLine, R., Drucker, S., & Fisher, D. (2016). The bones of the system: A case study of logging and telemetry at Microsoft. *Proceedings of the 38th International Conference on Software Engineering Companion* (pp. 92 – 101). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/2889160.2889231

Behrens, J. T., Mislevy, R. J., DiCerbo, K. E., & Levy, R. (2012). An evidence centered design for learning and assessment in the digital world. In M. C. Mayrath, J. Clarke-Midura, & D. Robinson (Eds.), *Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research* (pp. 13–54). Charlotte, NC: Information Age.

DiCerbo, K., Bertling, M., Stephenson, S., Jia, Y., Mislevy, R. J., Bauer, M., & Jackson, T. (2015). The role of exploratory data analysis in the development of game-based assessments. In C. S. Loh, Y. Sheng, & D. Ifenthaler (Eds.), *Serious games analytics: Methodologies for performance measurement, assessment, and improvement* (pp. 319–342). New York, NY: Springer. https://doi.org/10.1007/978-3-319-05834-4_14

Ercikan, K. A., & Pellegrino, J. W. (Eds.). (2017). *Validation of score meaning in the next generation of assessments*. Washington, DC: National Council on Measurement in Education.

Halverson, R., & Owen, V. E. (2014). Game-based assessment: An integrated model for capturing evidence of learning in play. *International Journal of Learning Technology, 9*(2), 111–138. https://doi.org/10.1504/IJLT.2014.064489

Hao, J., Liu, L., von Davier, A. A., Lederer, N., Zapata-Rivera, D., Jkal, P., & Bakkenson, M. (2017). *EPCAL: ETS Platform for Collaborative Assessment and Learning* (Research Report No. RR-17-49). Princeton, NJ: Educational Testing Service. https://doi.org/10.1002/ets2.12181

Hao, J., Smith L., Mislevy, R., von Davier, A., & Bauer, M. (2016). *Taming log files from the game and simulation-based assessment: Data model and data analysis tool* (Research Report No. RR-16-11). Princeton, NJ: Educational Testing Service. https://doi.org/10.1002/ets2.12181

Kerr, D., Andrews, J. J., & Mislevy, R. J. (2016). The in-task assessment framework for behavioral data. In A. Rupp & J. Leighton (Eds.), *Handbook of cognition and assessment* (pp. 472–507). Hoboken, NJ: Wiley Blackwell. https://doi.org/10.1002/9781118956588.ch20

Mislevy, R. J. (2018). On integrating psychometrics and learning analytics in complex assessments. In H. Jiao & R. W. Lissitz (Eds.).*Test data analytics and psychometrics: Informing assessment practices*. Charlotte, NC: Information Age.

Mislevy, R. J., Corrigan, S., Oranje, A., DiCerbo, K., John, M., Bauer, M. I., … Hao, J. (2014).*Psychometric considerations in game-based assessment*. New York, NY: Institute of Play.

Mislevy, R. J., & Gitomer, D. H. (1996). The role of probability-based inference in an intelligent tutoring system. *User-Modeling and User-Adapted Interaction*, *5*, 253–282. https://doi.org/10.1007/BF01126112

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). Focus article: On the structure of educational assessments. *Measurement: Interdisciplinary research and perspectives, 1*, 3–62. https://doi.org/10.1207/S15366359MEA0101_02

O'Neil, H. F., Baker, E. L., & Perez, R. S. (Eds.). (2016). *Using games and simulations for teaching and assessment: Key issues*. New York, NY: Routledge.

Roberts, J. D., Chung, G. K. W. K., & Parks, C. B. (2016). Supporting children's progress through the PBS KIDS learning analytics platform. *Journal of Children and Media*, *10*, 257–266. https://doi.org/10.1080/17482798.2016.1140489

Russo, B., Succi, G., & Pedrycz, W. (2015). Mining system logs to learn error predictors: A case study of a telemetry system. *Empirical Software Engineering, 20*, 879–927. https://doi.org/10.1007/s10664-01303-2

Shute, V., Ke, F., & Wang, L. (2017). Assessment and adaptation in games. In P. Wouters & H. van Oostendorp (Eds.), *Instructional techniques to facilitate learning and motivation of serious games* (pp. 59–78). New York, NY: Springer. https://doi.org/10.1007/978-3-319-39298-1_4

Steinberg, L. S., & Gitomer, D. G. (1996). Intelligent tutoring and assessment built on an understanding of a technical problem-solving task. *Instructional Science, 24,* 223–258. https://doi.org/10.1007/BF00119978

Tatsuoka, K. K. (1983).Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement, 20,* 345–354. https://doi.org/10.1111/j.1745-3984.1983.tb00212.x

Appendix A

XML Schema of the Data Model

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<!--
  Game Log Schema
  Version 1.2
  Authors:
    Lonnie Smith (lsmith@ets.org)
    Jiangang Hao (jhao@ets.org)

    Note: this is final version as of 7/27/2015

    (c) 2014, Educational Testing Service
-->

<!-- Root element and children -->

<xs:element name="gameLog">
<xs:complexType>
<xs:sequence>
<xs:element name="session" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="sessionID" type="idType" minOccurs="1" maxOccurs="1"/>
              <xs:element name="playerID" type="idType" minOccurs="1" maxOccurs="1"/>
<xs:element name="attemptID" type="idType" minOccurs="1" maxOccurs="1"/>
<xs:element name="sessionExtData" type="dictType" minOccurs="0" maxOccurs="1"/>
<xs:element name="eventSequence" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="event" type="eventType" minOccurs="1" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>


<!-- Data type definitions -->
```

```
        <!-- ID definition. All identifiers must follow this rule -->
        <xs:simpleType name="idType">
        <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z0-9_\-]{0,}"/>
        </xs:restriction>
        </xs:simpleType>

        <!-- Timestamps must follow subset of ISO 8601 standard, be resolved to (at least) the milisecond, and must use UTC -->
        <xs:simpleType name="timestampType">
        <xs:restriction base="xs:dateTime">
        <xs:pattern value="20\d{2}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z"/>
        </xs:restriction>
        </xs:simpleType>


        <!-- Type for "extended" data (sequence of key/value pairs). Extended data is supposed to conform to the user-defined data
descriptions in the extendedSessionData element, though this isn't enforced by the schema. -->
        <xs:complexType name="dictType">
        <xs:sequence>
        <xs:element name="pair" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
        <xs:sequence>
        <xs:element name="key" type="idType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        </xs:complexType>
        </xs:element>
        </xs:sequence>
        </xs:complexType>

        <!-- events -->
        <xs:complexType name="eventType">
        <xs:sequence>
        <xs:element name="eventName" type="idType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventStartTime" type="timestampType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventEndTime" type="timestampType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventBy" type="idType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventTo" type="idType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventResult" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventLocation" type="idType" minOccurs="1" maxOccurs="1"/>
        <xs:element name="eventExtData" type="dictType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        </xs:complexType>

        </xs:schema>
```

Appendix B

JSON Schema of the Data Model

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description":"Gamelog Schema v1.2, Created by Jiangang Hao @ ETS",
  "type": "object",
  "properties": {
    "gameLog": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "sessionID": {
              "type": "string",
             "pattern":"[a-zA-Z0-9_\\-]{1,}"
          },
          "playerID": {
              "type": "string",
             "pattern":"[a-zA-Z0-9_\\-]{1,}"
          },
          "attemptID": {
            "type": "integer"
          },
          "sessionExtData": {
            "type": "object"
          },
          "eventSequence": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "eventName": {
                    "type": "string",
                   "pattern":"[a-zA-Z0-9_\\-]{1,}"
                },
                "eventStartTime": {
                    "type": "string",
                   "pattern":"20\\d{2}-\\d{2}-\\d{2}T\\d{2}:\\d{2}:\\d{2}Z"
                },
                "eventEndTime": {
                    "type": "string",
                   "pattern":"20\\d{2}-\\d{2}-\\d{2}T\\d{2}:\\d{2}:\\d{2}Z"
                },
                "eventBy": {
                    "type": "string",
                   "pattern":"[a-zA-Z0-9_\\-]{1,}"
                },
                "eventTo": {
                    "type": "string",
                   "pattern":"[a-zA-Z0-9_\\-]{1,}"
                },
                "eventResult": {
                    "type": "string"
                },
                "eventLocation": {
                    "type": "string",
                   "pattern":"[a-zA-Z0-9_\\-]{1,}"
                },
                "eventExtData": {
                  "type": "object"
                }
              },
              "required": [
                "eventName",
                "eventStartTime",
                "eventEndTime",
```

```
                "eventBy",
                "eventTo",
                "eventResult",
                "eventLocation",
                "eventExtData"
              ]
            }
          }
        },
        "required": [
          "sessionID",
          "playerID",
          "attemptID",
          "sessionExtData",
          "eventSequence"
        ]
      }
    }
  },
  "required": [
    "gameLog"
  ]
}
```

Suggested citation:

Hao, J., & Mislevy, R. (2018). *The Evidence Trace File: A Data Structure for Virtual Performance Assessments Informed by Data Analytics and Evidence-Centered Design* (Research Report No. RR-18-28). Princeton, NJ: Educational Testing Service. https://doi.org/10.1002/ets2.12215

Action Editor: James Carlson

Reviewers: Michelle LaMar and Andreas Oranje