Event-Based Keyframing: Transforming Observation Data into Compact and Meaningful Form

Robert Wray, Robert Bridgman, Joshua Haley, Laura Hamel, and Angela Woods

1 Introduction

Learning outcomes are improved when a learning environment is responsive to the capabilities, preferences, and needs of individual learners [1, 2]. However, implementing such responsive learning systems is a significant design and engineering challenge [3]. Creating a learner-adaptive system generally increases the complexity and cost of the learning environment. Further, because each learning environment is unique, these additional costs recur when developing adaptive learning capabilities in a new application or domain. These costs slow research progress; creating an effective adaptive learning environment is expensive and time-consuming. More importantly, they also limit reaching full human potential via better and more widespread training technologies.

This chapter describes one aspect of the adaptive training systems engineering challenge more thoroughly. We describe past work in terms of solution requirements and outline how that work fell short of meeting the challenge. We then introduce a new approach, which combines several elements of previous solutions. This "event-based keyframing" employs methods from artificial intelligence (AI) to create a "learner-centric interpretation" of the current learning situation. A machine-understandable interpretation offers multiple benefits that allow it to mitigate some of the complexity and engineering cost for developing adaptive learning systems. We illustrate how we are using this approach in a number of distinct application domains. Successful use in multiple applications highlights the generality and value of a solution that is not tied to a particular domain, learning environment, or algorithmic methods of adaptation.

R. Wray (✉) · R. Bridgman · J. Haley · L. Hamel · A. Woods
Soar Technology, Ann Arbor, MI, USA
e-mail: wray@soartech.com

H. R. Arabnia et al. (eds.), *Advances in Artificial Intelligence and Applied Cognitive Computing*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70296-0_17

2 Systems Requirements for Adaptive Learning

We focus, in this chapter, on the requirements for dynamic adaptation within a practice environment. Many practice environments are implemented with simulations of the task or performance environment at various levels of fidelity. Increasingly simulation (or "virtual") environments are also being integrated with real or "live" environments to enable an integrated mix of live, virtual, and constructive actors in complex, realistic training and practice environments. These learning environments are realistic and effective for training. However, they are also complex systems with many dependencies and interactions. Our challenge is to develop algorithms and software that can "steer" this complex system so that learning needs for an individual learner at an individual point in time can be met.

Figure 1 illustrates a high-level decomposition of the functions required for dynamic adaptation in these environments. This figure is drawn from prior work developing algorithms and software to support dynamic adaptation in training [4]. Learners interact with a simulation environment that provides the opportunity to practice various skills. We have applied this design pattern across a wide range of learning tasks, including perceptual skills for observation, awareness, and response to social cues; cross-cultural communication; tactical aviation; terrain understanding; emergency response; and others. Inputs from the practice environment are used by a "director" to determine if and when to perform tailoring actions. The concept of a director guiding action in a simulation or game occurs repeatedly in simulation environments. It is apt especially when the practice environment includes synthetic or nonplayer characters (NPCs) that can be directed to perform specific actions that will support overall system goals.

To achieve individualized adaptation, the director performs three functions:

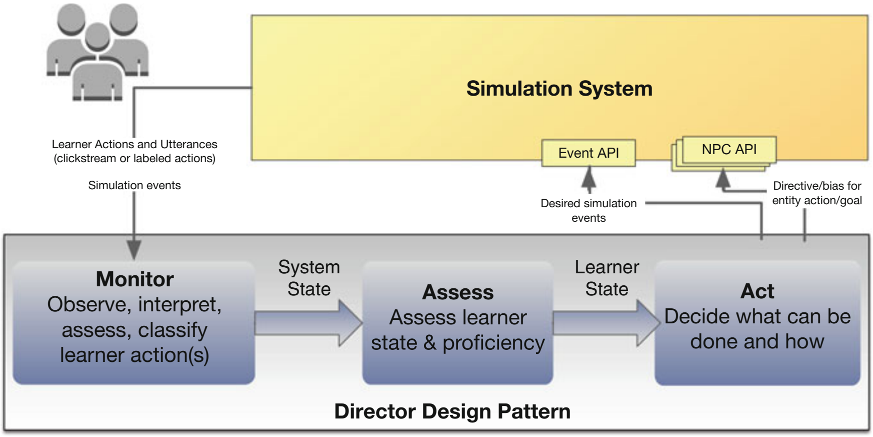
Simulation System
Event API
NPC API
Learner Actions and Utterances (clickstream or labeled actions)
Simulation events
Desired simulation events
Directive/bias for entity action/goal
Monitor
Observe, interpret, assess, classify learner action(s)
System State
Assess
Assess learner state & proficiency
Learner State
Act
Decide what can be done and how
Director Design Pattern

Fig. 1 Functional decomposition for adaptive tailoring of simulation-based practice

- *Monitoring*: The system must interpret what the learner is doing in the environment. Generally, monitoring requires some interpretation of learner actions as well as interpretation the context/situation in which the learner is acting.
- *Assessing*: The system uses its understanding of what is happening to assess the individual's learning state. Monitoring concerns identifying what is happening; assessing concerns what an understanding of the current situation reveals about the learner's progress and needs (e.g., in terms of the learning goals of the system). Its role is to translate from system state to learner state.
- *Acting*: The system state and the learner state provide a context for acting to address learning needs. Action results in adaptation that changes the environment in some way. Adaptation can include extrinsic (outside of the simulation) direct feedback, coaching, and intrinsic adaptation of the simulation (also known as pedagogical experience manipulation).

There are many different methods and algorithms that can be used to realize these functions, including long-standing cognitive and learner modeling approaches and task networks for planning actions and more recent, data-driven approaches that learn to perform these functions via data mining and machine learning. Regardless of the specific approach(es) adopted, however, one of the immediate challenges of building an adaptive learning system is incorporating, integrating, and unifying the various data streams that come from the other components of the learning environment. Examples of these data are enumerated in Table 1, based on both our direct experience in building adaptive learning systems and previous analyses of the requirements for monitoring and interpretation of the learning context [5].

The table enumerates the wide variety of inputs that are used to inform monitoring. We identify three characteristics of the inputs which vary from source to source.

Frame specifies the perspective or point of view of the input. In this table, we distinguish between two distinct frames. *Contextual* inputs are those that describe the environment, the situation (including the presence and actions of others), and other elements of the learning context (e.g., learning objectives). *Learner-centric* inputs directly encode information about the learner, such as activities and actions.

Representation distinguishes between largely *continuous* inputs (e.g., an analog sensor) and largely *discrete* ones. To achieve targeted adaptation, continuous and discrete data must be integrated and fused.

Interpretability characterizes whether meaning is incorporated into the inputs prior to its presentation to the learning system. For example, a raw sensor feed typically requires some translation process to extract meaning (*translatable*). A symbolic or discrete input may also require translation but the process is typically more reliable and simpler; we use *interpretable* in the table to make this distinction. Finally, some inputs may be presented with formally defined semantics underlying the input. We term directly machine-interpretable inputs as *system-understandable*.

Table 1 Examples of various data needed for monitoring and interpretation of a learner

Name/description	Frame	Representation	Interpretability
Simulation state (whole)	Contextual	Discrete	Interpretable
Current state of the simulation environment. Often represented as a "window" or "frame"			
Simulation state (part)	Contextual	Discrete	Interpretable
Representation of one element or entity in the simulation. Often used to communicate state across a network (e.g., DIS PDU)			
Physical state	Contextual	Continuous	Translatable
Summary of the current physical state of some object or entity (in simulation or live). For example, telemetry conveys location as f(t)			
Sensor state	Contextual	Continuous	Translatable
Outputs/status of a particular domain sensor. For example, the contacts a radar is detecting			
Speech	Contextual	Continuous	Translatable
Speech produced in the course of the practice, including learner speech and interactions with others. For example, radio communications	& learner		
Learner sensor	Learner	Continuous	Translatable
Input from a sensor external to the simulation that supports interpretation of learner state. Examples include eye tracking, mouse tracking, postural sensors, facial expression, etc.			
Activity data	Learner	Discrete	Interpretable
Input from the learning environment that classifies the activities of the learner. Protocols (e.g., xAPI) increasingly provide self-describing activity data that can be immediately understood			Understandable
Learning system state	Learner	Discrete	Interpretable
Feedback inputs from other components within the learner-adaptation system (e.g., estimates of learner proficiency, directives generated).			Understandable

3 Insights from Past Experiences

As suggested by the diversity and breadth of data types in the table, creating a generalized adaptation capability depends on developing an approach to monitoring that can translate/interpret learner and contextual data into a dynamically updating understanding of the progression of learning. In earlier attempts to address this challenge across previous efforts and learning domains (as outlined previously), we have identified several important insights that further expand and refine the requirements introduced in the previous section. This section briefly summarizes these “lessons learned.”

Inseparability of learner and context: Some of initial attempts to fuse learner data focused the development of standalone middleware that could capture learning state [5]. In practice, we have concluded that learning context is essential to interpreting learner state. For example, when attempting to debrief or review learner actions, some capture of the context in which those actions occurred is needed for human understanding and assessment of the action. Because the context itself also requires translation and interpretation (as in the table), monitoring needs to perform these tasks together because they depend on one another. Thus, solutions need to aggregate learner and context data.

Common representation of situation and activity: Aggregation of context and learner data leads to a requirement for a common representation that expresses what is happening in the environment, regardless of whether it is relevant to the current learning situation. For example, some situational change or event may not be relevant to the learner’s activity in the moment, but may have significant bearing on future learner activities (e.g., an actor makes a decision that is not immediately visible to the learner but that will result in some interaction with the learner at a later time in the learning exercise). Just as it is difficult to separate learner and context for fusion, a common representation that allows representation of learner and environmental activity supports an integrative perspective on learner activity and the capability to reason about what has, is, and will happen without requiring the system to understand the learning implications as those occurrences take place.

Capture of activity at human timescales: The need for common representation pushes solutions toward fine-grained capture of the situation. For example, if the largest update frequency is 200 Hz, then data capture tends to occur at that frequency for all system elements. High frequency capture introduces two needs for the adaptive learning system that are not generally functional requirements for the target capability. First, it creates requirements for data throughput and storage that are marginally useful for the adaptation algorithms. These algorithms will either be required to sample the data or perform an additional abstraction. Second, for the purposes of using data-driven/machine-learning approaches, increased resolution can increase complexity of the machine-learning challenge. Data updates at timescales faster than human reaction times results in noise that the machine-learning system must learn to ignore (greatly increasing the data requirements for training the system). Because adaptive learning system will act on a human timescale, we have concluded that the data moving thru the adaptive system should be captured and abstracted at a comparable timescale, typically from 100 ms to seconds, even when incoming data presents with much higher frequency.

4 Event-Based Keyframing

As a consequence of the previous efforts to provide adaptive training and lessons derived from these experiences, we are now developing and evaluating a new approach. This *event-based keyframing* approach consists of the following:

1. *A generalized event representation*: We have developed a new knowledge representation that seeks to capture the occurrences (or "events") on-going in a dynamic scenario. The representation is "generalized" both in the sense that it integrates across multiple modalities within an application as well as being readily extensible to new application domains.
2. *Event-recognition and keyframing*: While the representation describes what has occurred, event-recognition and keyframing algorithms define the processes and mechanisms by which instances of the event representation (loosely, an "event") is generated in the system. Keyframing is a complementary process that marks some event-instances as being particularly important for understanding how the learner is progressing.

Figure 2 illustrates the event-based keyframing approach conceptually. As in Fig. 1, a breadth of inputs of various types of various periodicities are presented to the monitoring system. These inputs are combined and integrated in the aggregation and event recognition component. The output of this process are instances of the event representation, resulting in a sequence of "events" produced at human timescales (e.g., roughly every second). The result of aggregation is a syntactically regular, semantically consistent, discrete summary of the activity occurring in the environment. In the diagram, various types of events (see next section) are indicated using different colors.

Note that the sequence is being generated from right to left. Additionally, the system creates event instances to mark the "start" and "end" of a real-world occurrence that has duration/is not instantaneous (in contrast to representing the

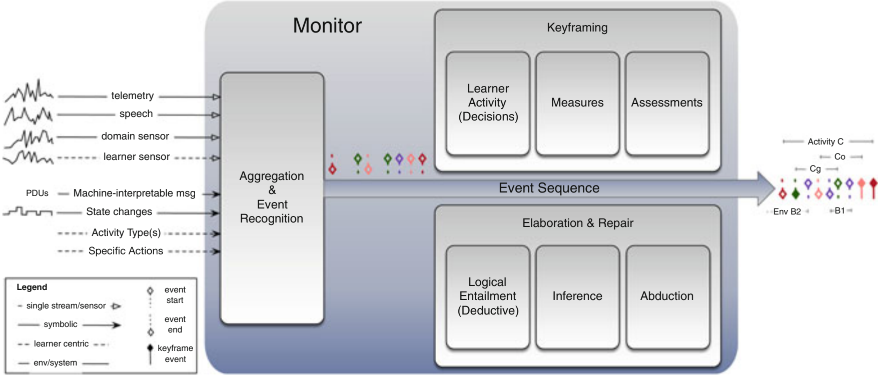

Fig. 2 Decomposition of monitoring functions to support event-based keyframing

event instance as a duration). Using a consistent representation of real-world events as "impulses" of various types has several advantages for monitoring. Further, as suggested by the labels on the output from the monitor, collections of events can be grouped. In the example, there is some *Activity C* that the learner is doing that has sub-activities C_o and C_g. Additionally, in the environment, some occurrence *B* has begun and ended (*B1*) and new, similar occurrence (*B2*) has begun, but has not yet concluded. This very simple example suggests how the event representation allows the integration of learner and environment occurrences within the common representation, which, as above, we have identified as an important requirement for adaption algorithms.

Having generated events, keyframing processes determine if any of the event instances generated in the sequence thus far have direct relevance to interpreting learner activity. The keyframing process looks at the sequence thus far (rather than only the most recently generated instance), allowing it to mark or identify event instances earlier in the observation stream as keyframes. Keyframes can also trigger measurement and assessment activities during keyframing, which then can result in additional annotation.

Finally, the monitoring process also now includes a "elaborate and repair" process that analyzes the event/keyframe sequence being produced and attempts to "fill in" any gaps it identifies in the sequence. As we discuss further below, a significant advantage of the common and consistent representation is that we can identify and use general algorithms that operate on event sequences (and the properties of the event representation) to perform logical entailment and various kinds of inference without requiring additional encoding of the properties of the domain.

4.1 Event Representation and Event Recognition

This section briefly outlines the event representation, further detailed in [6], in the context of the event-based keyframing approach. Events that occur in the learning environment are represented as *event instances* that conform to an *event representation.* Events may include environmental events (the entry of an actor the learner will need to interact with) and learner events (the learner acts, such as turning to face the new actor). As the system recognizes events, it produces a series of event instances, resulting in an event sequence. This event sequence forms the primarily method for communicating what is happening in the environment to all downstream components.

Individual event instances are represented as a defined set of "slots" and "fillers" for events of particular type. Slots include a subject (the primary actor(s) in an event), an event type or "verb" which captures what kind of activity or occurrence is being recorded ("turning toward"). Some event types include an object (what is being turned toward?) and event-specific parameters (e.g., the new direction that the learner is facing). The event types are organized into a taxonomy, drawing from formal ontology [7], which supports inheritance, property evaluation, and other functions. For example, “TurnToward” may be a specialization of the more general event type “Turn” which is where a “facing-direction” property is defined.

Adopting this formal knowledge representation for the event representation, provides several important advantages. First, the library of event types can be created progressively, and grow with additional applications. We first developed “TurnToward” events in the context of aviation training, but the same event type can be used for a first-person game perspective as well. Second, the event sequence provides a compact summary of learning activity, that can be expanded by inspection of the event instances. In an initial experiment, we discovered that the pattern of event types alone was sufficient for some pattern detection in a learning context [6]. Third, the event representation provides a well-defined and consistent target for additional algorithms that can support interpretation and adaptation, as we discuss in the next subsections.

The primary function of the event recognition process is to identify the specific items to fill the slot for the particular event instance being recorded. Using the example in the previous section, when the system observes the learner turning toward the new actor, then it would identify the learner as the subject of the event, the new actor as the object, and the learner’s new heading as the key parameter associated with the turn. As we discuss further in the next section, in many cases, the initial or triggering observation may be less specific than the event instance that is finally recorded.

We have explored a number of different mechanisms for recognizing events, including simple, user-defined programs that look for specific patterns [8], data-driven machine learning [9], and comparison of observed patterns to previously observed patterns [6]. Enumerating the various requirements and trade-offs for these methods is outside the scope of this paper. However, key conclusions are that event-recognition is a type of pattern-recognition process and existing pattern-recognition algorithms can be readily applied to this problem. Using these methods enables acceptably reliable recognition of events during learner activity. The event representation provides a target that simplifies computational and data requirements for pattern/event recognition.

4.2 Keyframes and Keyframing

The event sequence summarizes all of the activity and occurrences taking place in the learning environment. Some of the resulting event instances will be of particular interest in interpreting or understanding the learner. We term these special event instances *keyframes* [10] and the process of identifying or marking them, *keyframing*. We first developed keyframes to allow a learner to target specific goals, actions, or outcomes when the learner was to replay a past practice experience (whether their own or the experience of another). Providing indices to the “important

Table 2 Examples of keyframes and keyframing processes

Type	Description	Process
Learner activity	Event in which a learner takes some identifiable action ("TurnToward" when the subject of the event is a learner)	Auto
Measurement	Event that measures some aspect of learner activity ("response time" could record the time that elapses between the new actor's entry and the learner initiating "TurnToward")	Auto
Assessment	Event that assesses some aspect of learner activity (the response time recorded for "TurnToward" was acceptable)	Auto
Learner cognitive state	Event that indicates a change in the learner's cognitive/affective state (engagement is now above the threshold)	Auto
Exercise state	Event that indicates a change in environment resulting from learner activity ("TurnToward" allows learner to observe the new actor)	Semi-auto

moments" within the log or trace of activity helped learner focus and resulted in more effective use of time.

Although useful, we have discovered that what a user considers "important" cannot be consistently anticipated. Instead, we have extended the notion of keyframes so that a system can mark various learner-specific events. Keyframes are tags or markers added to the "base" events. Table 2 lists examples of keyframes we have explored in current and prior work. Similar to the event representation, many keyframes of various types can be automatically identified and tagged. In the case of changes in the state of the learning environment itself that are due to learner action, we lack algorithms that can fully automate keyframing for these events, as that would require some causal model of the domain. However, we can employ similar mechanisms to those used for event recognition to mark key changes in the exercise state (a semi-automated solution).

Keyframes provide a means to help interpretation and adaption processes further abstract from the many sources of data input from the environment to information most likely to help the system build a model of the learner and progress and challenges in the domain. Automated keyframing provides a straightforward approach to capture keyframes, but at the cost of mixing "important" and "routine" events together. Long term, we plan to continue to research methods that can support the automatic classification of importance and relevance.

4.3 *Elaboration and Repair*

The adaptive learning system must be tolerant to noise and observation errors. Sensors will be mis-calibrated. Packets of data will be dropped. A consistent representation enables the research of general algorithms and tools that can support elaboration and refinement of event sequences as well as recognition of problems and repair of the sequence and individual instances within it. Three such elaboration/repair algorithms are as follows:

Logical entailment The formal semantics of the event representation enable deductive elaboration of the event sequence. Every "start" event has a paired "end" event such as start/stop "TurnToward." Some events can be defined as inverses of another event, thus representing the same physical event from different perspectives. For example, "Learner detects Actor" is the inverse of "Actor is-detected-by Learner." Inverse events are particularly useful for elaborating the event sequence so that all event instances are represented from the perspective of each individual entity sequence.

Abduction When event instances are missing or appear out of sequence, some process other than deductive closure is needed to repair the sequence. The representation itself can provide signals when the sequence is need of repair. In Fig. 2, the presence of two "start" event instances for the "green" event, without an intervening "stop" event creates such a signal. An abductive reasoner can evaluate potential hypotheses about the causes of sequence faults. Abduction can also be used to specialize event instances as further evidence accrues. In the "TurnToward" example, the system cannot know when the turn is initiated where it will end and to what end it is directed; however, when it concludes with the learner facing the new actor, abduction can specialize the "Turn" event instance to "TurnToward" based on observation/evidence.

Inference In some cases, there may not be enough evidence to assert a repair in detail. In this case, we are exploring machine learning to compile patterns from a large collection of observations and assert the presence of missing event instances (or missing details from the instance). We are using machine-learned models of individual parameters within event instances to compare and match event sequences [6].

While this component is relatively nascent in comparison to event representation and keyframing, it offers an example of the potential power and value of the integrated and abstract representational approach. Because these algorithms are based on the properties of the representation and the statistical properties of observations within a domain, they are not specific to an individual application. We anticipate further reuse across multiple applications of the adaptive learning system.

Acknowledgments The authors thank teammates for contributions to concepts and implementations reported here. We also thank Dr. Ami Bolton, Ms. Jennifer Pagan, and Dr. Heather Priest for contributions to problem definition and context that informed the approach. Opinions expressed here are not necessarily those of the Department of Defense or the sponsors of this effort, the Office of Naval Research and the Naval Air Warfare Center Training Systems Division (NAWCTSD). This work funded under contracts N68335-20-F-0549, N68335-17-C-0574, and N68335-19-C-0260.

References

1. P. J. Durlach, A. M. Lesgold (eds.), *Adaptive Technologies for Training and Education* (Cambridge, New York, 2012)
2. C.R. Landsberg, R.S. Astwood, W.L. Van Buskirk, L.N. Townsend, N.B. Steinhauser, A.D. Mercado, Review of adaptive training system techniques. Mil. Psychol. **24**, 96–113 (2012)
3. National Academy of Engineering: Grand Challenges for Engineering. National Academy of Sciences/National Academy of Engineering (2008)
4. R.E. Wray, A. Woods, A cognitive systems approach to tailoring learner practice, in *Proceedings of the Second Advances in Cognitive Systems Conference*, ed. by J. Laird, M. Klenk, (Baltimore, MD, 2013)
5. R.E. Wray, J.T. Folsom-Kovarik, A. Woods, Instrumenting a perceptual training environment to support dynamic tailoring, in *Proceedings of 2013 Augmented Cognition Conference*, ed. by D. Schmorrow, C. Fidopiastis, (Springer-Verlag (Lecture Notes in Computer Science), Las Vegas, 2013)
6. R.E. Wray, J. Haley, R. Bridgman, A. Brehob, Comparison of complex behavior via event sequences, in *2019 International Conference on Computational Science and Computational Intelligence (CSCI'19)*, (IEEE Press, Las Vegas, 2019)
7. R. Arp, B. Smith, A.D. Spear, *Building Ontologies with Basic Formal Ontology* (MIT Press, 2015)
8. R.E. Wray, C. Newton, R.M. Jones, Dynamic scenario adaption for simulation training, in *2018 International Conference on Computational Science and Computational Intelligence (CSCI'18)*, (IEEE Press, Las Vegas, 2018)
9. J. Haley, V. Hung, R. Bridgman, N. Timpko, R.E. Wray, Low level entity state sequence mapping to high level behavior via a deep LSTM model, in *20th International Conference on Artificial Intelligence*, (IEEE Press, Las Vegas, 2018)
10. R.E. Wray, A. Munro, Simulation2Instruction: Using simulation in all pases of instruction, in *2012 Interservice/Industry Training, Simulation, and Education Conference*, (NTSA, Orlando, 2012)