ADAPTIVE E-LEARNING ENGINE WITH EMBEDDED DATAGRAPH STRUCTURE

Inventors: Guy Zaslavsky, Avihail (IL); Andrei Bobkov, Krasnodar (RU)

Filed: Feb. 24, 2015

Abstract

Embodiments of the invention provide dynamically adaptive e-learning courses built on integrated datagraph macrostructures and microstructures. For example, an e-learning course includes a number of knowledge entities, each having one or more lesson and practice components. The knowledge entities can be instantiated as nodes of a course datagraph macrostructure, linked together via edges having edge attributes that can define a course flow relationship among the knowledge entities. Each node in the datagraph macrostructure can include its own set of datagraph microstructures. For example, each knowledge entity can include lesson and/or practice datagraph microstructures that include numbers of respective step objects (e.g., lesson step objects and/or practice step objects), each associated with defined responses, and each linked (e.g., via one or more of the responses) with one or more other respective step objects via edges, thereby defining lesson and practice flow relationships among the respective step objects according to attributes of the defined responses.
We previously learned about basic four-sided shapes, like squares and rectangles. Now, we will explore triangles. A triangle is any plane figure with three straight lines and three angles.

There are three types of triangles:
1. Equilateral — All 3 sides are equal in length.
2. Isosceles — 1 of the 3 sides differs in length.
3. Scalene — All 3 sides differ in length.

Correct! Each of B, C, and F has only 3 sides.

Correct! E has 4 sides, so it cannot be a triangle.

Incorrect. B and C are triangles, but not the only ones.

Try Again.

Incorrect. B and C only have 3 sides. Which of the above are triangles?

B and C only have 3 sides. Which of the above are triangles?

A, B, C, D, and E are triangles.

There are three types of triangles:

(1) Equilateral — All 3 sides are equal in length.
(2) Isosceles — 1 of the 3 sides differs in length.
(3) Scalene — All 3 sides differ in length.

Show me basic examples of triangles.

Learn about types of triangles.

I already know about triangles.

End.
Instantiate a number of knowledge entities as nodes of a course datagraph macrostructure

Link a first knowledge entity and a second knowledge entity in the course datagraph macrostructure by instantiating a knowledge edge having a set of knowledge edge attributes that defines a course flow relationship between the first and second knowledge entities

Embed each knowledge entity in the course datagraph macrostructure with at least one lesson datagraph microstructure

Define a set of lesson responses associated with each lesson step object, each lesson response including a set of lesson response attributes

Link a first lesson step object and second lesson step object in the lesson datagraph microstructure via one of the lesson responses of the first lesson step object by instantiating a lesson edge that defines a lesson flow relationship between the first and second lesson step objects

Embed each knowledge entity in the course datagraph macrostructure with at least one practice datagraph microstructure

Link each remaining knowledge entity with at least one other knowledge entity via at least one respective knowledge edge

FIG. 6
FIG. 8
ADAPTIVE E-LEARNING ENGINE WITH EMBEDDED DATAGRAPH STRUCTURE

BACKGROUND

[0001] Embodiments relate generally to e-learning systems, and, more particularly, to computer-implemented creation and delivery of adaptive, interactive e-learning courses.

[0002] For many years, traditional classrooms have included course materials: teachers to interpret, adapt, and deliver the course materials; and students to learn from the teachers and the course materials. The effective transfer and retention of knowledge in such environments can be improved through increased student engagement and interaction with teachers and course materials, and through increased adaptation by the teachers to the needs and learning styles of the students. However, many pedagogical efforts are frustrated by limitations of traditional classroom environments. For example, it may be difficult or impossible to physically locate students in classrooms with skilled teachers; it may be difficult or impossible for a single teacher to concurrently engage with and adapt to multiple students, particularly when those students have different backgrounds, levels of knowledge, learning styles, etc.; it may be difficult to accurately, or even adequately, measure student knowledge acquisition and retention, or for teachers to adapt their teaching to implicit or explicit student feedback; it may be difficult to dynamically adapt course materials in context of static course materials (e.g., printed textbooks); it may be difficult to measure and respond to teacher or student performance across large (e.g., geographically distributed) populations; it may be difficult to measure or value respective contributions to learning by multiple teachers; etc.

[0003] With the increasing ubiquity of computers and Internet access, many attempts have been made to create effective, on-line learning environments. In most instances, these attempts are primarily an on-line implementation of a traditional classroom environment. For example, typical e-learning systems include digital versions of traditional course materials (e.g., digital text and images that mimic those of a traditional, printed textbook), digital self-assessment tools (e.g., digital flash cards, quizzes, etc.), and simple tracking (e.g., quiz scoring, tracking of which lessons have been completed, timers to track time spent, etc.). Some, more recent e-learning systems have added more sophisticated functions. For example, newer digital course materials can include hyperlinks, videos, etc.; and some on-line courses include communications functions to permit live chatting with instructors and/or other students, file access and sharing, etc. A few e-learning systems have recently begun to provide limited types of adaptation. For example, some e-learning courses can offer, or force, a review of a certain concept if a certain amount of time has elapsed since the concept was last presented to the student, or a student can be permitted to select from multiple alternative versions of a course, depending on skill level, prior knowledge, goals, etc. Even with the added capabilities facilitated by computers and the Internet, many of the limitations of traditional classrooms and pedagogical approaches frustrate the efficacy of e-learning systems.

BRIEF SUMMARY

[0004] Among other things, systems and methods are described for providing dynamically adaptive e-learning courses built on integrated datagraph microstructures and macrostructures. Embodiments operate in context of on-line e-learning environments that facilitate e-learning course creation, editing and consumption. For example, a course includes a number of knowledge entities (e.g., concepts), and each knowledge entity includes a number of embedded lesson datagraph microstructures (e.g., for acquiring knowledge of sub-concepts) and embedded practice datagraph microstructures (e.g., for assessing and/or reinforcing knowledge of sub-concepts). In some implementations, each knowledge entity includes 1-to-N lesson datagraph microstructures and 0-to-M practice datagraph microstructures. The knowledge entities can be instantiated as nodes of a course datagraph macrostructure, linked together via edges having edge attributes that can define a course flow relationship among the knowledge entities. Each node in the datagraph macrostructure can include its own set of datagraph microstructures. For example, each knowledge entity can include lesson and/or practice datagraph microstructures that each include one or more respective step objects (e.g., lesson step objects or practice step objects), where each can be associated with defined responses, and each can be linked (e.g., via one or more of the responses) with one or more other respective step objects via edges, thereby defining lesson and/or practice flow relationships among the respective step objects according to attributes of the defined responses.

[0005] According to one set of embodiments, a system is provided for building an e-learning datagraph structure. The system includes: a non-transient course data store that stores a course datagraph macrostructure having lesson datagraph microstructures embedded therein; a processor in communication with the course data store that implements a course authoring platform to receive authoring commands, translate the authoring commands to executable datagraph commands, and execute the datagraph commands with the processor. Executing the datagraph commands with the processor, instantiates a number of knowledge entities as nodes of the course datagraph macrostructure in the course data store; links each knowledge entity with at least one other knowledge entity in the course datagraph macrostructure by instantiating a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship between the knowledge entities; and embeds each knowledge entity in the course datagraph macrostructure with at least one of the lesson datagraph microstructures in the course data store by instantiating a number of lesson step objects as nodes of the lesson datagraph macrostructure, defining a set of lesson responses associated with each lesson step object, and linking each lesson step object with at least one other lesson step object in the lesson datagraph macrostructure by instantiating a respective lesson edge that defines a lesson flow relationship between the lesson step objects. In some such embodiments, the processor implements the course authoring platform to execute the datagraph commands further to: embed each knowledge entity in the course datagraph macrostructure with at least one practice datagraph microstructure in the course data store by instantiating a plurality of practice step objects as nodes of the practice datagraph microstructure, defining a set of practice responses associated with each practice step object, and linking each practice step object with at least one other practice step object in the practice datagraph microstructure by instantiating a respective practice edge that defines a practice flow relationship between the practice step objects.
According to another set of embodiments, a method is provided for building an e-learning datagraph structure. The method includes: instantiating a number of knowledge entities as nodes of a course datagraph macrostructure; linking a first knowledge entity and a second knowledge entity in the course datagraph macrostructure by instantiating a knowledge edge having a set of knowledge edge attributes that defines a course flow relationship between the first and second knowledge entities; and embedding each knowledge entity in the course datagraph macrostructure with at least one lesson datagraph microstructure. The embedding includes instantiating a number of lesson step objects as nodes of the lesson datagraph microstructure; defining a set of lesson responses associated with each lesson step object, each lesson response including a set of lesson response attributes; and linking a first lesson step object and second lesson step object in the lesson datagraph microstructure via one of the lesson responses of the first lesson step object by instantiating a lesson edge that defines a lesson flow relationship between the first and second lesson step objects.

According to another set of embodiments, a system is provided for consuming an e-learning datagraph structure. The system includes a non-transient course data store having an e-learning datagraph structure stored thereon, the datagraph structure; the datagraph structure having: a number of knowledge entities stored as nodes of a course datagraph macrostructure in the course data store, each knowledge entity being linked with at least one other knowledge entity in the course datagraph macrostructure via a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship among the knowledge entities; and a number of lesson datagraph microstructures, each embedded in a respective knowledge entity in the course datagraph macrostructure and having a number of lesson step objects stored as nodes of the lesson datagraph microstructure, each lesson step object being associated with a set of lesson responses and linked with at least one other lesson step object in its lesson datagraph microstructure via a respective lesson edge, so that the lesson edges define a lesson flow relationship between the lesson step objects of the lesson datagraph microstructure. The system further includes a first processor in communication with the course data store that implements a first instance of a course consumption platform to: determine a profile of a first student; and adaptively generate and display a first instance of an e-learning course from the e-learning datagraph structure according to the profile of the first student. The system further includes a second processor in communication with the course data store that implements a second instance of the course consumption platform to: determine a profile of a second student; and adaptively generate and display a second instance of the e-learning course from the e-learning datagraph structure according to the profile of the second student, the second instance being different from the first instance.

The present disclosure is described in conjunction with the appended figures:

FIG. 1 shows an illustrative e-learning environment, according to various embodiments;

FIG. 2 shows an illustrative course datagraph macrostructure that includes a particular course flow relationship among its knowledge entities, as defined by knowledge edges between those knowledge entities;

FIG. 3 shows a portion of a course datagraph macrostructure that includes an illustrative lesson datagraph microstructure and an illustrative practice datagraph microstructure embedded in a knowledge entity, according to various embodiments;

FIG. 4 shows an illustrative screen shot of a course datagraph macrostructure editing environment, according to various embodiments;

FIG. 5 shows an illustrative screen shot of a lesson datagraph microstructure editing environment, according to various embodiments;

FIG. 6 shows a flow diagram of an illustrative method for building a dynamic e-learning system on an embedded datagraph structure, according to various embodiments;

FIG. 7 shows an illustrative computational system for implementing one or more systems or components of systems, according to various embodiments; and

FIG. 8 shows another illustrative computational system for implementing one or more systems or components of systems, according to various embodiments.

In the appended figures, similar components and/or features may have the same reference label. Further, various components of the same type may be distinguished by following the reference label by a second label that distinguishes among the similar components. If only the first reference label is used in the specification, the description is applicable to any one of the similar components having the same first reference label irrespective of the second reference label.

DETAILED DESCRIPTION

In the following description, numerous specific details are set forth to provide a thorough understanding of the present invention. However, one having ordinary skill in the art should recognize that the invention may be practiced without these specific details. In some instances, circuits, structures, and techniques have not been shown in detail to avoid obscuring the present invention.

With the increasing ubiquity of computers and Internet access, many attempts have been made to create effective, on-line learning environments. For example, many traditional e-learning systems provide digital versions of traditional course materials, including digital versions of textbooks, some enhanced with videos, hyperlinks, integrated access to reference materials, on-line help, etc. Some traditional e-learning systems further provide self-practice and self-assessment capabilities, such as digital flashcards, timers, scored tests, and review questions. Some traditional e-learning systems also provide communications functions, such as on-line communities and social networking functions, live chatting with instructors and/or other students, etc. Still, in spite of added capabilities facilitated by computers and the Internet, most traditional e-learning approaches have not strayed for beyond the capabilities of traditional classroom environments. Accordingly, the efficacy of such approaches continues to be hindered by many of the same limitations present in traditional classroom environments. For example, on-line e-learning systems have typically been touted as a way to provide more students with more flexibility as to when and where they can learn (e.g., on-line access to courses can put students in front of teachers, regardless of their geographic separation). However, traditional on-line e-learning systems have done little to address issues, such as facilitating a single teacher’s concurrently engagement with and adapta-
tion to multiple students, particularly when those students have different backgrounds, levels of knowledge, learning styles, etc.: accurately, or even adequately, measuring student knowledge acquisition and retention, and facilitating teachers’ adapting of their teaching to implicit or explicit student feedback; dynamically adapting course materials to different students and/or other course contexts; measuring and/or responding to teacher or student performance across large (e.g., geographically distributed) populations; measuring and/or valuing respective contributions to learning by multiple teachers, etc.

Among other things, embodiments provide dynamically adaptive e-learning course systems for facilitating e-learning course creation, editing, and consumption. As described herein, courses can be created using novel, embedded datagraph structures, including course datagraph macrostructures with embedded microstructures. For example, courses can be defined by nodes and edges of directed graph macrostructures, in which, each node includes one or more directed graph microstructures for defining respective lesson and practice step objects of the courses. The content and attributes of the nodes and edges can adaptively manifest higher level course flow relationships and lower level lesson and practice flow relationships. Various embodiments can exploit such embedded datagraph structures to provide various features, such as facilitation of dynamic course creation and increased course adaptability: improved measurement of student knowledge acquisition and retention, and of student and teacher performance enhanced monitoring and responsiveness to implicit and explicit student feedback; and access to, exploitation of, measurement of, and/or valuation of respective contributions to learning by multiple teachers, including across multiple courses, disciplines, geographies, etc.; etc.

For the sake of context, FIG. 1 shows an illustrative e-learning environment 100, according to various embodiments. As illustrated, the e-learning environment 100 can include course authoring platforms 160 and course consumption platforms 170 in communication with course data stores 140, for example, over one or more networks 150. Some embodiments further include a course backend processor 180 for performing any backend and/or otherwise centralized functions. For example, in some implementations, some or all datagraph functionality described herein is performed at the course backend processor 180, and the course authoring platforms 160 and course consumption platforms 170 implement thin clients, or the like, to facilitate interaction by course authors and students with the functionality of the course backend processor 180 (e.g., by providing graphical user interfaces, etc.). The course authoring platforms 160, course consumption platforms 170, and course backend processor 180 can be implemented as any suitable computational system, such as a server computer, desktop computer, laptop computer, tablet computer, smart phone, dedicated portable electronic device, etc. The networks 150 can include any suitable communications networks, including wired and/or wireless communications links, secured and/or unsecured communications links, public and/or private networks, mesh networks, peer-to-peer networks, hub and spoke networks, etc. The course data stores 140 can include any suitable types of storage, including one or more dedicated data servers, cloud storage, etc.

While some embodiments are designed to operate in context of typical computing environments, features are embodied in and facilitated by novel datagraph structures that transform the environment into an adaptive e-learning system. As illustrated, the course data stores 140 can store one or more course datagraph macrostructures 105. Each of the course datagraph macrostructures 105 can include one or more embedded course datagraph microstructures 130. Each datagraph structure includes datagraph nodes 110 linked together by datagraph edges 120. Each datagraph node 110 has node attributes 115, and each datagraph edge 120 has edge attributes 125.

As used herein, a “course” generally refers to a set of related “knowledge entities,” which can generally include any relatively high-level concepts of the course. Each knowledge entity can include embedded lesson datagraph microstructures, used herein to refer generally to objects that facilitate the acquisition of knowledge relating to particular topics or sub-concepts within the course; and embedded practice datagraph microstructures, used herein to refer generally to objects that facilitate the reinforcement, assessment, and/or refreshment of knowledge relating to particular topics or sub-concepts within the course. For the sake of illustration, a course could be designed to teach basic college writing skills. Each knowledge entity can correspond to a concept, such as “Fundamentals of Good Writing” or “Fundamentals of Critical Thinking and Rhetoric.” Within the “Fundamentals of Good Writing” knowledge entity, there may be a number of lesson datagraph microstructures directed, for example, to “Sentence Structure,” “Paragraph Structure,” or “Essay Organization.” There may also be a number of practice datagraph microstructures that may track the lesson datagraph microstructures, and may be directed, for example, to “Practice Sentence Structure”; or they may combine, split, reorganize, parse, or otherwise derive from the lesson datagraph microstructures, such as “Practice Elements of Sentences” or “Practice Basic Organization in Writing.”

In typical embodiments, the course data stores 140 can include large numbers of datagraph nodes 110 created by multiple contributors in association with multiple courses and/or as stand-alone datagraph nodes 110 (e.g., a stand-alone concept, virtual textbook chapter, etc.). A course represents a specific subset of all these datagraph nodes 110, along with all their associated objects (e.g., datagraph edges 120, node attributes 115, edge attributes 125, etc.). Creation of a course, then, can include selection of datagraph nodes 110 and linking of those nodes by datagraph edges 120. In some instances, course creation can further include generation of additional datagraph nodes 110, generation (or adaptation, modification, etc.) of node attributes 115 and/or edge attributes 125, etc. Accordingly, some implementations permit a course to be contained within another course, multiple courses to intersect (i.e., share one or more datagraph nodes 110), etc. Further, proper formulation of the datagraph nodes 110 and datagraph edges 120 as course datagraph macrostructures 105 with embedded course datagraph microstructures 130 can facilitate automatic adaptation, and even generation, of courses for particular contexts. For example, consumption of a course by a student using a course consumption platform 170 can include compilation and consumption of the datagraph nodes 110 and their embedded course datagraph microstructures 130.

The node attributes 115 of the datagraph nodes 110 can be used to express any concrete or abstract aspect of the knowledge domain or of the process for acquiring the knowledge. For example, terms, like “concept” are used herein to
generally include any aspects of a knowledge domain, such as factual data, lines of reasoning, learning preferences, proofs, principles, definitions, overviews, rules, “thinking skills” (e.g., how to approach an aspect), “performance skills” (e.g., how to apply an aspect), “work orders” (e.g., tools and techniques for handling multi-stage problems, such as solving a differential equation), context (e.g., representing groups of knowledge entities and/or sub-groups of knowledge entities), specific lessons, areas of interest (e.g., “sports,” “finance,” etc.), mental tendencies (e.g., a tendency to make careless arithmetic calculation mistakes of a certain type, to make certain grammatical errors of a certain type, to jump to certain false conclusions, to misunderstand certain types of data, etc.), misconceptions (e.g., based on societal prejudices, common misunderstandings, perpetuated falsehoods, etc.), thinking styles (e.g., visual thinking or learning, auditory thinking or learning, embodied thinking or learning, etc.), and/or any other aspect of a knowledge domain. Different types of knowledge domains lend themselves to different types of aspects. For example, a course relating to automotive repair will likely rely more heavily on embodied learning, image-based (or video-based) content, step-by-step processes, diagnostic algorithms, etc.; while a course on French literature will likely rely more heavily on visual learning, textual content, critical reasoning, etc.

[0026] In some embodiments, the node attributes 115 of the datagraph nodes 110 can include various types of metadata. Some metadata in the node attributes 115 can be used to describe the respective datagraph node 110. For example, the metadata can include a title of the datagraph node 110 (e.g., a long form, a short form, etc. for different contexts), a summary (as described below), a creator identifier (e.g., the name or other identifier of the instructor who created the node), etc. Other metadata in the node attributes 115 can be used to automatically adapt the datagraph node 110 to particular contexts. For example, the metadata can include default and/or adaptive assumptions for types of student profiles, such as certain alternative forms of content of the datagraph node 110 that can be generated based on different styles of learning, different platform capabilities (e.g., different hardware and/or software of the course consumption platform 170), different knowledge levels of particular students (e.g., whether the concept(s) included in the datagraph node 110 are considered new knowledge, a review of prior knowledge, a preview of future knowledge, etc.), different student context (e.g., student age, geography, affiliation, gender, socioeconomics, etc.), etc.

[0027] In some embodiments, the node attributes 115 can include a “summary.” The summary can be used to automatically generate summaries of the aspects of the respective datagraph nodes 110. In certain implementations, the summary in the node attributes 115 is used to automatically summarize the aspects of a course or lesson at the end of a particular knowledge flow. In other implementations, the summary in the node attributes 115 is used to auto-populate a global or personalized summaries list. For example, the summary can include one or more sets of text populated by a course creator to summarize key concepts covered in a course or a set of lessons. Upon reaching a trigger event (e.g., at the end of a particular lesson or course, at a point in a lesson or course flow specified by the course creator or instructor, when requested by a student (e.g., by pressing a “summarize” button or based on student preferences), after some learning time has elapsed, after some number of concepts has been consumed and/or practiced, after some amount of time has elapsed since last interacting with a particular concept, and/or at any other suitable time), a summary can be generated from the relevant node attributes 115. The summary can include any suitable information, depending on the type of summary and/or when it is being generated. For example, the summary can include a collection of one or more sets of text stored in the node attributes 115 of the group of datagraph nodes 110 being summarized; and/or the summary can include additional monitoring information, such as indications of when a student first consumed a particular concept, when the student apparently understood the concept, when the student last reviewed the concept, how long the student spent learning and/or reviewing the concept, what other concepts are related to a learned concept (e.g., as prerequisites, as similar concepts, as next concepts, etc.), where the student’s present knowledge fits into an overall course or learning plan, etc.

[0028] As described above, the datagraph nodes 110 are linked by datagraph edges 120, each having edge attributes 125. The datagraph edges 120 can effectively define a “flow relationship” (i.e., an ontological relationship) through a set of datagraph nodes 110. Some portions of the flow relationship can be defined as static. For example, in a static portion of a flow relationship, the flow between datagraph nodes 110 is always the same, regardless of the context of the datagraph nodes 110 in a course, lesson plan, etc.; regardless of the type of student; regardless of the student’s interaction with the datagraph nodes 110 and/or other datagraph nodes 110; etc. Other portions of the flow relationship can be defined as dynamic. For example, in a dynamic portion of a flow relationship, the flow between datagraph nodes 110 is dependent on the context of the datagraph nodes 110 in a course, lesson plan, etc.; dependent on the type of student; dependent on the student’s interaction with the datagraph nodes 110 and/or other datagraph nodes 110; etc. In many instances, portions of the flow relationship can be defined as “limited dynamic,” or the like. For example, in a limited dynamic portion of a flow relationship, the flow between datagraph nodes 110 can be mostly static, except for limited variances that can be triggered by special student context (e.g., falling outside a predetermined “normal” range, etc.), special learning context (e.g., a datagraph node 110 for a knowledge entity is being consumed outside its originally intended course context, etc.), and/or in other particular cases. Even though each particular datagraph edge 120 may define relationships that do not, themselves define a flow between their source and target nodes, the datagraph edges 120 are still considered as defining a flow relationship at a higher level. For example, certain relationships defined by a datagraph edge 120 (e.g., a variant or special/general case relationship) may define, if not consumption flows from its source to its destination node, but rather which of multiple nodes will be presented to a student during consumption of the datagraph; such that the datagraph edge 120 still helps to define the overall flow relationship among the nodes.

[0029] To define such flow relationships, the edge attributes 125 of each datagraph edge 120 can include a type, a source, and a destination. For example, as shown in FIG. 1, the source of datagraph edge 120ab is datagraph node 110a and the destination of datagraph edge 120ab is datagraph node 110b. Its edge attributes 125ab can define the type of datagraph edge 120ab in a manner that indicates a relationship in the course datagraph macrostructure 105 between datagraph node 110a and datagraph node 110b.
Another edge type is a “variant” knowledge edge 215 (labeled as “V”), indicating that the creator of the course flow relationship considers two knowledge entities 210 to be variants of each other to be selected according to defined conditions. For example, edge attributes of a variant knowledge edge 215 can determine which of alternative knowledge entities 210 to present to a student according to different styles of learning, different platform capabilities, different knowledge levels of the student, different student context, etc. As illustrated, knowledge entity 210a is shown as a prerequisite of knowledge entity 210b1 and knowledge entity 210b2, shown as variants. Depending on predefined conditions stored in the edge attributes, upon completion of knowledge entity 210a, the student will be presented with one of knowledge entity 210b1 or knowledge entity 210b2. For example, the creator of the course flow relationship desires that the student have certain knowledge going into knowledge entity 210c, but that prerequisite knowledge is not presented in knowledge entity 210a or knowledge entity 210b1. Accordingly, if the student is determined not to already have that prerequisite knowledge, the student may be presented with knowledge entity 210b2, which teaches that additional knowledge (as opposed to being presented with knowledge entity 210b1 that assumes prior knowledge of that information). As another example, the creator of the course flow relationship believes that students are more likely to grasp certain concepts in context of case studies related to their experiences. Accordingly, the creator formulates and selectively presents alternative versions of a knowledge entity based on a determined profile of the student (e.g., gender, age, likes or dislikes, etc.). Any suitable rationale or basis can derive the use of variation knowledge entities 210. While the variations are shown as separate knowledge entities 210, variations can also be implemented within a particular knowledge entity 210. For example, as described above, the node attributes of a knowledge entity 210 can include metadata that defines alternative forms of the knowledge entity 210 (e.g., different content) depending on certain characteristics, or the knowledge edges 215 can define variant knowledge entities 210 to present based on their edge attributes.

Though not shown, other edge types are possible. For example, some implementations include a “general case” knowledge edge 215, indicating that the creator of the course flow relationship considers a source knowledge entity 210 to be a general case of a destination knowledge entity 210 (and/or considers the destination knowledge entity 210 to be a special case of its source knowledge entity 210). For example, it may be understood that a particular concept is best learned through specific cases, so that a student’s knowledge of a general concept may be determined by the student’s apparent grasp of some quantity of those specific cases illustrating the general concept. Such a general case knowledge edge 215 can be used in multiple ways, for example, as a dynamic type of prerequisite generator, where the dependency is a function of student profile information. For example, a general case knowledge edge 215 can be used to link two knowledge entities 210—one as a general case of a sub-concept and the other as a specific case of the sub-concept—that are not dependent on each other (e.g., neither is a prerequisite of the other, and it is intended for the student to consume both knowledge entities 210 as part of consuming the course). For a student determined to be more of a “deductive” thinker, it can be desirable to present the general case as a prerequisite of the specific case (the student will likely have more success...
acquiring the general case first and be intuitively inclined to subsequently grasp the specific case); while for a student determined to be more of an “inductive” thinker, it can be desirable to present the specific case as a prerequisite of the general case (the student will likely have more success acquiring the specific case first and be intuitively inclined to subsequently grasp the general case).

[0035] In addition to type, source, and destination, the edge attributes can include other metadata that may relate to its type. For example, for a prerequisite knowledge edge 215, the metadata may define a “mastery threshold,” indicating a certain level of mastery required by the student in the source knowledge entity 210 before being eligible to start learning the destination knowledge entity 210.

[0036] As described above with reference to FIG. 1, various functions are facilitated by exploiting novel types of course datagraph macrostructures 105 having course datagraph microstructures 130 embedded therein. FIG. 3 shows a portion of a course datagraph macrostructure 300 that includes an illustrative lesson datagraph microstructure 305 and an illustrative practice datagraph microstructure 350 embedded in a knowledge entity 210, according to various embodiments. The lesson datagraph microstructure 305 and the practice datagraph microstructure 350 can be implementations of the course datagraph microstructure 130 of FIG. 1. In general, each datagraph structure is implemented as a directed graph, which, as described above, includes a set of nodes connected by edges having associated direction. Each course datagraph macrostructure 300 (i.e., and the corresponding macrostructure 105 of FIG. 1 and macrostructure 200 of FIG. 2) is implemented as a directed acyclic graph, which is a directed graph that has no directed cycles (i.e., there is no way to follow a set of edges from a source node and end up back at the source node). In certain instances, the course datagraph macrostructure 300 can manifest various acyclic structures, such as multi-trees, poly-trees, rooted trees, etc. Each datagraph microstructure (e.g., lesson datagraph microstructure 305, practice datagraph microstructure 350, course datagraph microstructures 130 of FIG. 1, etc.) is also implemented as a directed graph, but may or may not be acyclic. For example, some lesson datagraph microstructures 305 loop back on themselves (e.g., depending on student interaction with the lesson step objects, as described below). In a particular knowledge entity 210, the lesson datagraph microstructure 305 and practice datagraph microstructure 350 do not have to manifest the same directed graph structure (e.g., they can have different numbers and types of nodes and/or edges, one can be acyclic while the other is not, etc.).

[0037] As illustrated, the lesson datagraph microstructure 305 can include a number of lesson step objects 310 linked together by lesson edges 315. Each lesson step object 310 includes lesson step object attributes 330, and each lesson edge 315 includes lesson edge attributes 340. Each lesson step object 310 can also include a set of (i.e., one or more) associated lesson responses 320. In some embodiments, each lesson step object 310 is linked to one or more other lesson step objects 310 via a respective lesson edge 315. Similarly, the practice datagraph microstructure 350 can include a number of practice step objects 360 linked together by practice edges 365. Each practice step object 360 includes practice step object attributes 380, and each practice edge 365 includes practice edge attributes 390. Each practice step object 360 can also include a set of associated practice responses 370. In some embodiments, each practice step object 360 is linked to one or more other practice step objects 360 through one or more of its practice responses 370 via a respective practice edge 365. In some implementations, each lesson datagraph microstructure 305 can effectively represent a particular sub-concept presented via its lesson step objects 310, and each practice datagraph microstructure 350 can effectively represent a practice question, or the like, presented via its practice step objects 360.

[0038] For example, the set of lesson responses 320 or practice responses 370 can represent multiple answer choices for a question posed as part of its lesson step object 310 or practice step object 360 (e.g., “The capital of Japan is: (a) Tokyo; (b) Kyoto; (c) Okinawa; or (d) None of the above”), as a prompt posed as part of its lesson step object 310 or practice step object 360 (e.g., “Are you ready to move on?” followed by a selection or text field; or “Click here to continue”), or any other suitable set of responses. Each lesson response 320 or practice response 370 can include any suitable response data, including response content and a response target. For example, the response content can include the text or other information to be presented to the student, and the response target can indicate what should happen if that response is selected by a student. For example, if a student is asked a true/false question having “True” as the correct answer, the lesson responses can show a “True” response and a “False” response to the student (i.e., those are the respective response texts). When the student selects “True,” the response target can include an indication of a correct response (e.g., by changing appearance (such as turning green), giving audio feedback (such as playing a chime), or giving other feedback (e.g., showing a special graphic, changing a score, etc.), etc.), and the response target may also automatically proceed to a next lesson step object 310 or practice step object 360 (or provide a second level of responses, such as a selection to proceed to the next lesson step object 310 practice step object 360, or to return to the (or any) previous lesson step object 310 or practice step object 360).

[0039] Each practice step object 360 or lesson step object 310 (or their respective practice responses 370 or lesson responses 320) can, in some implementations, be considered as a prompt of the system having static content resources (e.g., text, images, video, interactive animations, etc.) and/or dynamic content resources (e.g., content that adapts to other tags, contexts, etc.). For example, a dynamic content resource can include a special “congratulate” tag that translates in runtime into a specific intensity of congratulating the student for a correct answer depending on a streak of success and difficulty level of the question (e.g., “Wow, that’s 3 correct answers in a row!! You’re amazing!”). Some dynamic content resources include tags. The tags can facilitate conditional content insertions (e.g., a “snippet” to be displayed only on the first event in which the tag gets a rendering request); peripheral instructions (e.g., a feature tooltip to appear on mouse-over, or a countdown-timer to be used for the containing step); etc. Some implementations include a content chunk bank that stores “chunks” that can be inserted (e.g., using tags) to facilitate centralized updating and/or localization or translation. Each object or response can also be associated with metadata. Some such metadata can be used to create Boolean conditions. For example, a lesson step object 310 can be configured as “try again” lesson step object 310, so that there must be at least one “correct” lesson response 320 underneath the lesson step object 310, and any incorrect les-
son responses 320 will cause the system to automatically bring the student (after reaching the end of the directed graph path by following the incorrect response) back to the try-again lesson step object 310. Other such metadata can define response style types (e.g., does the step accept free-text, or multiple-choice answers, or a slider, or checkboxes, or a numerical value, etc.). Other such metadata can include correlations to the mental attributes and performance factors; whether a user can expand the set of responses for an object after choosing one of the responses (e.g., for inspecting the other choices, viewing the “path not chosen,” etc.); defining whether a particular object is a “root object,” meaning it is the beginning of a flow (e.g., within a knowledge entity 210); showing a countdown timer for the object and/or which response should be the implied user choice when the time expires; whether to show an auto-diagnostic menu (e.g., if the user makes a mistake and/or the user gets the next response right); whether to show a time awareness tool for the object; etc. Some implementations permit multiple variations for each lesson step object 310 or practice step object 360, each of which having different associated metadata, in a way that can allow the flow algorithms to choose an optimal step variation per student in a given point in time.

[0040] Embodiments of lesson responses 320 and/or practice responses 370 can include any suitable possible choice by a student as a reaction to a particular lesson step object 310 or practice step object 360 prompt. The responses may or may not be limited by an instructor or creator of the response or object. For example, the response may allow for any free-form response, free-form responses only in a particular format, only a selection of a limited range of predetermined options, etc. A lesson response 320 or a practice response 370 can contain any of the types of rich content (e.g., dynamic content resources, metadata, etc.) as that of its lesson step object 310 or practice step object 360. In some implementations, the responses are permitted to have additional functionality to facilitate processing of various response types. For example, a response can include mathematical formulae or computer code (e.g., scripts, etc.) to define and/or carry-out rule sets for matching various user reactions to and/or interactions with the responses and associated objects (e.g., a response may contain code that defines a match if the student enters free-form text indicating any prime number greater than 100). The rich content of the responses can also consider contextual and/or other information separate from the particular student interaction with the response. For example, a response can be accepted as a match if its code determines at runtime that more than two minutes has elapsed since the preceding step was displayed, and no response has been explicitly selected (i.e., a “ran out of time” default can be set to pedagogically focus on timing strategy and solution approach, rather than on the core content). Some examples of response metadata include degree of correctness (e.g., ranging from totally incorrect to fully correct with a spectrum between); matching type (e.g., whether a strict match is required, or whether a free-response step should permit a match with differences in whitespace, punctuation, minor spelling errors, etc.); how to factor in the user’s response time and/or confidence level with the correctness to generate an overall performance estimation; how to present the response after it gets clicked (e.g., whether to collapse the set of responses and leave the chosen response visible, to hide all, etc.); whether the response should be navigable as part of the “path not chosen” (e.g., this can be manual or automatic, such as if the subsequent steps are part of a long diagnostic flow and would not make sense for a stand-alone peek; etc.

[0041] In general, the lesson step objects 310 are intended to increase a student’s level of knowledge of a certain concept, and the practice step objects 360 are intended to develop (e.g., and measure) a student’s level of knowledge of a certain concept. For example, while lesson step objects 310 often include questions, review materials, and/or other content that can result in the student “practicing” a particular concept; the primary objective of the lesson step objects 310 is to add to the student’s knowledge. Similarly, while a student’s interaction with practice step objects 360 can certainly help a student learn underlying concepts, the primary objective of the practice step objects 360 is to further develop the student’s understanding of previously seen concepts (e.g., and to measure the student’s knowledge level with respect to those concepts). Still, much information about the student can be gleaned by monitoring the student’s interaction with both lesson step objects 310 and practice step objects 360. As discussed herein, the embedded datagraph structures allow such monitoring to enhance course adaptations. For example, information gleaned from monitoring student interactions with the lesson step objects 310 can cause adaptations in presentation to that student (and/or to other students) of future lesson step objects 310 and/or adaptations in presentation of practice step objects 360 within that knowledge entity 210, can propagate up to cause adaptations in presentation to that student (and/or to other students) of other microstructures of the knowledge entity 210 and/or in other knowledge entities 210, can further propagate to alter profiles and/or other information relating to that student or other students (e.g., for use in characterizing a student’s proclivities, learning style, preferred learning times or apparent attention span, etc.), can be used to measure effectiveness of a particular lesson step objects 310 or course objects 360 (e.g., on its own, relative to others in its knowledge entity 210, relative to its course context, relative to alternative approaches to the same or similar concepts, etc.) or effectiveness of a particular contributor (e.g., how the creator of that course or object compares to himself/herself and/or to other contributors, etc.).

[0042] Some embodiments include additional functionality for lesson step objects 310 and/or practice step objects 360. For example, practice step objects 360 can be categorized and assigned to “baskets.” Each lesson step object 310 or knowledge entity 210 can contain one or more baskets, each having zero or more assigned practice step objects 360. Each basket can represent a difficulty level, learning type, or any other useful categorization. The basket assignment can be implemented at the knowledge entity 210 level (e.g., using the node attributes of knowledge entities 210 or edge attributes of knowledge edges 215), at the object level (e.g., using practice step object attributes 380 or practice edge attributes 365), or using course-level metadata or other metadata. For example, a basket can be implemented as a separate type of object or effectively as the result of attributes of other objects. The baskets can be used, for example, to dynamically and automatically generate a set of appropriate practice step objects 360 according to a student’s interactions with related lesson step objects 310 and/or based on other information (e.g., a student’s overall measured knowledge level, learning style, time elapsed since last review, etc.).

[0043] For the sake of illustration, FIGS. 4 and 5 show a portion of an example introductory course on geometry. FIG. 4 shows an illustrative screenshot 400 of a course datagraph
macrostructure editing environment, according to various embodiments. The course datagraph macrostructure editing environment can be displayed on a course authoring platform 160, as illustrated in FIG. 1. The particular illustrated course datagraph macrostructure is intended to illustrate certain functionality of a particular implementation and is not intended to be limiting. As shown, the illustrative course datagraph macrostructure includes three knowledge entities 210: “Introductory Shapes”; “4-Sided Shapes”; and “Triangles.” The “4-Sided Shapes” and “Triangles” knowledge entities 210 are shown as part of a “Basic Shapes” group 410. Dashed box 415 indicates that a course author has selected the “Triangles” knowledge entity 210c.

In response to this selection, a window 420 can populate with information about the selected knowledge entity 210c. The window 420 can include any useful controls, information, menus, etc. The illustrated window 420 includes various controls and/or information relating to the knowledge entity 210c: contents. For example, the window 420 shows controls, including “Publish” for publishing a new or edited version of the knowledge entity 210c: to the course for visibility by students, “Preview” for allowing the course author to see what a student would see when interacting with that knowledge entity 210c: “Edit” for accessing various editing functions for the knowledge entity 210c: (e.g., including adding lesson step objects to an embedded lesson datagraph microstructure), and “Add Practice Items” for associating the knowledge entity 210c: with practice step objects in an embedded practice datagraph microstructure. The window 420 can also include controls and information relating to the knowledge edges associated with the knowledge entity 210c. For example, the window 420 shows that the knowledge entity 210c: is “in the Basic Shapes group” and is “a dependent of 4-Sided Shapes” (e.g., the “4-Sided Shapes” knowledge entity 210d is a prerequisite to the “Triangles” knowledge entity 210c).

FIG. 5 shows an illustrative screenshot 500 of a lesson datagraph microstructure editing environment, according to various embodiments. For example, the lesson datagraph microstructure for the “Triangles” knowledge entity 210c: is displayed to the course author in response to selecting to edit the “Triangles” knowledge entity 210c: in the interface shown in FIG. 4. As shown, the lesson datagraph microstructure includes three lesson step objects 310 having associated lesson responses 320, linked together to form a directed graph that defines a lesson flow relationship. Some implementations include a start node 510 and an end node 520 for the lesson datagraph microstructure, for example, to help define the flow relationship and/or to make the flow relationship more intuitive for the author. The particular illustrated lesson datagraph microstructure is intended to illustrate certain functionality of a particular implementation and is not intended to be limiting.

As illustrated, the lesson datagraph microstructure can begin at the start node 510 and proceed to a first lesson step object 310a. The lesson step object 310a presents certain content (introductory and basic information about triangles) and prompt language, and includes associated lesson responses 320a. The lesson responses 320a provide the student with three different paths through the lesson: one that is more visual and links to lesson step object 310b, one that is more textual and links to lesson step object 310c, and one that effectively skips the lesson by linking to the end node 520. Following the link to lesson step object 310a, the student is now presented with a set of images of triangles and non-triangles, a prompt, and a set of lesson responses 320b relating to the prompt. Two of the three answer choices 530 presented in the lesson responses 320b are incorrect, and includes related content and a further interactive prompt to “try again.” The third answer choice 530c is correct; selecting that choice provides explanatory text and links to a next lesson step object 310c. Lesson step object 310c: teaches more information about triangles and has no associated lesson responses 320c. After it is determined that the student has consumed lesson step object 310c: (e.g., after some time, after the student clicks a “continue” button, or in any suitable manner), the flow relationship can continue to the end node 520 to end the lesson.

Various implementations can permit different types of objects, responses, etc. For example, some implementations can require that every lesson step object 310c: includes at least one lesson response 320c (e.g., lesson step object 310c: would not be allowed in such implementations). Further, some implementations may permit or require objects that are global or external to the datagraph structures. For example, a user interface for course consumption may include various global controls, such as menus, navigation buttons, etc.; and/or course authors may be permitted to set up features in the user interface for course consumption, such as chat windows, file management areas, etc.

While FIG. 5 is described with reference to lesson step objects 310 of a lesson datagraph microstructure, some embodiments can implement practice step objects of a practice datagraph microstructure in a similar or identical fashion. The user interface and its functionality can be substantially the same for either type of datagraph microstructure, except that particular lesson- or practice-related functions can be included. For example, rather than presenting new information to the student as with lesson step objects, the practice step objects are intended to review and test that information.

FIG. 6 shows a flow diagram of an illustrative method 600 for building a dynamic e-learning course on an embedded datagraph structure, according to various embodiments.

Embodiments begin at stage 604 by instantiating a number of knowledge entities as nodes of a course datagraph macrostructure. At stage 608, a first knowledge entity and a second knowledge entity can be linked in the course datagraph macrostructure by instantiating a knowledge edge having a set of knowledge edge attributes that defines a course flow relationship between the first and second knowledge entities. For example, as described above, a first knowledge edge attribute can be defined to identify the first knowledge entity as a source node for the knowledge edge; a second knowledge edge attribute can be defined to identify the second knowledge entity as a destination node for the knowledge edge; and a third knowledge edge attribute can be defined to identify an edge type for the knowledge edge. The edge type can define the course flow relationship between the first and second knowledge entities. For example, the edge type can be identified as a prerequisite edge that defines the course flow relationship between the first and second knowledge entities so that, at runtime, the first knowledge entity is required to be consumed prior to the second knowledge entity; or the edge type can be identified as a variant edge that defines the course flow relationship between the first and second knowledge entities so that, at runtime, a selection is automatically made between presenting the first knowledge entity and presenting...
the second knowledge entity based on criteria stored in association with the knowledge edge. In some implementations, at stage 610, each remaining knowledge entity in the course datagraph macrostructure can be linked to at least one other knowledge entity via at least one respective knowledge edge. Attributes of the knowledge entities and knowledge edges (e.g., metadata and/or other rich content) can be generated to define and link course concepts in a manner that defines the macrostructure (e.g., a directed acyclic graph) of an adaptive e-learning course.

At stage 612, each knowledge entity in the course datagraph macrostructure can be embedded with at least one lesson datagraph microstructure. Some implementations generate the lesson datagraph microstructure by stages 616-624 (e.g., and, optionally, stage 626). At stage 616, a number of lesson step objects can be instantiated as nodes of the lesson datagraph microstructure. A set of lesson responses can be defined at stage 620 in association with each lesson step object, such that each lesson response includes a set of lesson response attributes. At stage 624, a first lesson step object and second lesson step object in the lesson datagraph microstructure can be linked via one of the lesson responses of the first lesson step object by instantiating a lesson edge that defines a lesson flow relationship between the first and second lesson step objects. In some implementations, at stage 626, each remaining lesson step object in the lesson datagraph microstructure can be linked to at least one other lesson step object via at least one respective lesson edge. Attributes of the lesson step objects and lesson edges (e.g., metadata and/or other rich content) can be generated to define and link lesson concepts (e.g., course sub-concepts associated with a knowledge entity) in a manner that defines the lesson as an embedded microstructure (e.g., a directed graph) of the adaptive e-learning course defined by the course macrostructure.

At stage 628, each knowledge entity in the course datagraph macrostructure can be further embedded with at least one practice datagraph microstructure. Some implementations generate the practice datagraph microstructure in a similar manner to the generation of the lesson datagraph microstructure (e.g., illustrated by stages 616-626). For example, a number of practice step objects can be instantiated as nodes of the practice datagraph microstructure, and each can be associated with a defined set of practice responses having respective sets of practice response attributes. Each practice step object in the practice datagraph microstructure can be linked to at least one other practice step object via at least one respective practice edge, thereby defining a practice flow relationship between the practice step objects. Attributes of the practice step objects and practice edges (e.g., metadata and/or other rich content) can be generated to define and link practice concepts (e.g., practice for course sub-concepts associated with particular knowledge entities and/or lesson objects) in a manner that defines the practice as an embedded microstructure (e.g., a directed graph) of the adaptive e-learning course defined by the course macrostructure.

The methods disclosed herein include one or more actions for achieving the described method. The method and/or actions may be interchanged with one another without departing from the scope of the claims. In other words, unless a specific order of actions is specified, the order and/or use of specific actions may be modified without departing from the scope of the claims.

FIG. 7 shows an illustrative computational system 700 for implementing one or more systems or components of systems, according to various embodiments. The computational system 700 is described as a particular machine for implementing course authoring functionality, like the course authoring platform 160 described with reference to FIG. 1. Embodiments of the computational system 700 can be implemented as or embodied in single or distributed computer systems, or in any other useful way.

The computational system 700 is shown including hardware elements that can be electrically coupled via a bus 755. The hardware elements can include one or more processors (shown as central processing units, “CPU(s)” 705, one or more input devices 710 (e.g., a mouse, a keyboard, etc.), and one or more output devices 715 (e.g., a display, a printer, etc.). The computational system 700 can also include one or more storage devices 720. By way of example, storage device(s) 720 can be disk drives, optical storage devices, solid-state storage device such as a random access memory (RAM) and/or a read-only memory (ROM), which can be programmable, flash-updateable and/or the like. In some embodiments, the storage devices 720 are configured to store unpublished course data 722. For example, while course data (e.g., knowledge entities, microstructures, step objects, etc.) are being edited, unpublished or otherwise unofficial versions can be stored by certain implementations of a course authoring platform.

The computational system 700 can additionally include a communications system 730 (e.g., a modem, a network card (wireless or wired) or chipset, an infra-red communication device, etc.). The communications system 730 can permit data to be exchanged with a public or private network and/or any other system. For example, as shown in FIG. 1, the communications system 730 can permit the course authoring platform to communicate with a remote (e.g., cloud-based) course data store 140 via a public or private network 150 (shown in dashed lines for context). In some embodiments, the computational system 700 can also include a processing acceleration unit 735, which can include a DSP, a special-purpose processor and/or the like.

Embodiments can also include working memory 740, which can include RAM and ROM devices, and/or any other suitable memory. The computational system 700 can also include software elements, shown to being currently located within a working memory 740, including an operating system 745 and/or other code 750, such as an application program (which can be a client application, web browser, mid-tier application, relational database management system (RDBMS), etc.). As illustrated, a course authoring application 760 can be implemented in the working memory 740. Some implementations of the course authoring application 760 and can include an author interface 762. For example, the author interface 762 can provide graphical user interface (GUI) and/or other functionality for receiving authoring commands relating to creation, editing, publishing, and/or other authoring-related course functions. Some implementations of the course authoring application 760 and can include a datagraph compiler 764. For example, the datagraph compiler 764 can translate received authoring commands into datagraph commands for execution by the processor(s) 705 in interfacing with the datagraph structure for course authoring.

In some embodiments, the course data store 140 and/or the storage device(s) 720 implement a non-transient course data store that stores a course datagraph macrostructure having lesson datagraph microstructures embedded therein. The processor(s) 705 of the computational system
implement functions of a course authoring platform as the course authoring application 760, including receiving authoring commands (via the author interface 762), translating the authoring commands to executable datagraph commands (via the datagraph compiler 764), and executing the datagraph commands with the processor(s) 705. Executing the datagraph commands can include instantiating a number of knowledge entities as nodes of the course datagraph macrostructure, linking each knowledge entity with at least one other knowledge entity by instantiating a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship between the knowledge entities; and embedding each knowledge entity with at least one of the lesson datagraph microstructures in the course data store by instantiating a number of lesson step objects as nodes of the lesson datagraph microstructure, defining a set of lesson responses associated with each lesson step object, and linking each lesson step object with at least one other lesson step object in the lesson datagraph microstructure by instantiating a respective lesson edge that defines a lesson flow relationship between the lesson step objects.

FIG. 8 shows another illustrative computational system 800 for implementing one or more systems or components of systems, according to various embodiments. The computational system 800 is described as a particular machine for implementing course consumption functionality, like the course consumption platform 170 described with reference to FIG. 1. Embeds of the computational system 800 can be implemented as or embodied in single or distributed computer systems, or in any other useful way.

The computational system 800 is shown including hardware elements that can be electrically coupled via a bus 855. The hardware elements can include one or more processors (shown as central processing units, “CPU(s)” 805, one or more input devices 810 (e.g., a mouse, a keyboard, etc.), and one or more output devices 815 (e.g., a display, a printer, etc.). The computational system 800 can also include one or more storage devices 820. By way of example, storage device(s) 820 can be disk drives, optical storage devices, solid-state storage devices such as a random access memory (RAM) and/or a read-only memory (ROM), which can be programmable, flash-updateable and/or the like. In some embodiments, the storage devices 820 are configured to store student-specific data 822. For example, some implementations of the course consumption platform can store profile information about one or more students.

The computational system 800 can additionally include a communications system 830 (e.g., a modem, a network card (wireless or wired) or chipset, an infra-red communication device, etc.). The communications system 830 can permit data to be exchanged with a public or private network and/or any other system. For example, as shown in FIG. 1, the communications system 830 can permit the course consumption platform to communicate with a remote (e.g., cloud-based) course data store 140 via a public or private network 150 (shown in dashed lines for context). In some embodiments, the computational system 800 can also include a processing acceleration unit 835, which can include a DSP, a special-purpose processor and/or the like.

Embodiments can also include working memory 840, which can include RAM and ROM devices, and/or any other suitable memory. The computational system 800 can also include software elements, shown as being currently located within a working memory 840, including an operating system 845 and/or other code 850, such as an application program (which can be a client application, web browser, mid-tier application, relational database management system (RDBMS), etc.). As illustrated, a course consumption application 860 can be implemented in the working memory 840. Some implementations of the course consumption application 860 can include a student interface 862. For example, the student interface 862 can provide graphical user interface (GUI) and/or other functionality for receiving interaction commands relating to consuming knowledge entities, microstructures, step objects, etc. (e.g., student interactions with responses, timer data, etc.), and/or other consumption-related course functions. Some implementations of the course consumption application 860 and can include a datagraph compiler 864. For example, the datagraph compiler 864 can translate received interaction commands into datagraph commands for execution by the processor(s) 805 in interfacing with the datagraph structure for dynamic course generation, course adaptation, course display, etc. Some implementations of the course consumption application 860 and can include a student profiler 866. For example, the student profiler 866 can receive explicit profile information from a student (e.g., through a profile page, an external database, etc.), monitor student interactions with course materials to infer student profile information (e.g., learning styles, strengths, tendencies, etc.), and/or develop the student profile in any other suitable manner.

In some embodiments, the course data store 140 and/or the storage device(s) 820 implement a non-transient course data store that stores an e-learning datagraph structure. The datagraph structure can include: a number of knowledge entities stored as nodes of a course datagraph macrostructure, each knowledge entity being linked with at least one other knowledge entity in the course datagraph macrostructure via a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship among the knowledge entities; and a number of lesson datagraph microstructures, each embedded in a respective knowledge entity in the course datagraph macrostructure and having a number of lesson step objects stored as nodes of the lesson datagraph microstructure, each lesson step object being associated with a set of lesson responses and linked with at least one other lesson step object in its lesson datagraph microstructure via a respective lesson edge, so that the lesson edges define a lesson flow relationship between the lesson step objects of the lesson datagraph microstructure.

The processor(s) 805 of the computational system 800 implement functions of a course consumption platform as the course consumption application 860, including determining a profile of a first student (via the student profiler 866), and adaptively generating and displaying an e-learning course from the e-learning datagraph structure according to the profile of the first student (via the student interface 862 and the datagraph compiler 864). As described above, the e-learning datagraph structure permits dynamic adaptation of the course materials to each student. For example, a first instance of the computational system 800 can use its processor(s) 805 to implement a first instance of a course consumption platform that determines a profile of a first student and adaptively generates and displays a first instance of an e-learning course from the e-learning datagraph structure according to the profile of the first student; and a second instance of the computational system 800 can use its processor(s) 805 to implement a second instance of a course con-
sumption platform that determines a profile of a second student and adaptively generates and displays a second instance of an e-learning course from the same e-learning datagraph structure according to the profile of the second student (i.e., where the second instance is different from the first instance).

It should be appreciated that alternate embodiments of computational systems 700 and 800 can have numerous variations from those described above. For example, customized hardware can be used and/or particular elements can be implemented in hardware, software (including portable software, such as applets), or both. Further, connection to other computing devices such as network input/output devices can be employed. In various embodiments a computational system, like those illustrated in FIGS. 7 and 8, can be used to implement one or more functions of the systems described with reference to FIG. 1 and/or to implement one or more methods, such as those described with reference to FIG. 6.

The functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored as one or more instructions on a tangible computer-readable medium. A storage medium may be any available tangible medium that can be accessed by a computer. By way of example, and not limitation, such computer-readable media can include RAM, ROM, EEPROM, CD-ROM, or other optical disk storage, magnetic disk storage, or other magnetic storage devices, or any other tangible medium which can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Disk and disc, as used herein, include compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk, and Blu-ray disc where discs usually reproduce data magnetically, while discs reproduce data optically with lasers.

A computer program product may perform certain operations presented herein. For example, such a computer program product may be a computer readable tangible medium having instructions tangibly stored (and/or encoded) thereon, the instructions being executable by one or more processors to perform the operations described herein. The computer program product may include packaging material. Software or instructions may also be transmitted over a transmission medium. For example, software may be transmitted from a website, server, or both. Remote source using a transmission medium such as a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technology such as infrared, radio, or microwave.

Further, modules and/or other appropriate means for performing the methods and techniques described herein can be downloaded and/or otherwise obtained by suitable terminals and/or coupled to servers, or the like, to facilitate the transfer of means for performing the methods described herein. Alternatively, various methods described herein can be provided via storage means (e.g., RAM, ROM, a physical storage medium such as a CD or floppy disk, etc.), such that a user terminal and/or base station can obtain the various methods upon coupling or providing the storage means to the device. Moreover, any other suitable technique for providing the methods and techniques described herein to a device can be utilized.

Other examples and implementations are within the scope and spirit of the disclosure and appended claims. For example, due to the nature of software, functions described above can be implemented using software executed by a processor, hardware, firmware, hardwiring, or combinations of any of these. Features implementing functions may also be physically located at various positions, including being distributed such that portions of functions are implemented at different physical locations. Also, as used herein, including in the claims, “or” as used in a list of items prefaced by “at least one of” indicates a disjunctive list such that, for example, a list of “at least one of A, B, or C” means A or B or C or AB or AC or BC or ABC (i.e., A and B and C). Further, the term “exemplary” does not mean that the described example is preferred or better than other examples. As used herein, a “set” of elements is intended to mean “one or more” of those elements, except where the set is explicitly required to have more than one or explicitly permitted to be a null set.

Various changes, substitutions, and alterations to the techniques described herein can be made without departing from the technology of the teachings as defined by the appended claims. Moreover, the scope of the disclosure and claims is not limited to the particular aspects of the process, machine, manufacture, composition of matter, means, methods, and actions described above. Processes, machines, manufacture, compositions of matter, means, methods, or actions, presently existing or later to be developed, that perform substantially the same function or achieve substantially the same result as the corresponding aspects described herein may be utilized. Accordingly, the appended claims include within their scope such processes, machines, manufacture, compositions of matter, means, methods, or actions.

What is claimed is:

1. A system for building an e-learning datagraph structure, the system comprising:

- a non-transient course data store that stores a course datagraph macrostructure;

- a set of processors in communication with the course data store that implements a course authoring platform to receive authoring commands, translate the authoring commands to executable datagraph commands, and execute the datagraph commands with the set of processors to:

  - instantiate a plurality of knowledge entities as nodes of the course datagraph macrostructure in the course data store;

  - link each knowledge entity with at least one other knowledge entity in the course datagraph macrostructure by instantiating a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship between the knowledge entities; and

  - embed at least one of the knowledge entities in the course datagraph macrostructure with at least one lesson datagraph microstructure in the course data store by instantiating a plurality of lesson step objects as nodes of the lesson datagraph microstructure, defining a set of lesson responses associated with each lesson step object, and linking each lesson step object with at least one other lesson step object in the lesson datagraph microstructure by instantiating a respective lesson edge that defines a lesson flow relationship between the lesson step objects.

2. The system of claim 1, wherein a first lesson step object is linked with a second lesson step object in the lesson datagraph microstructure via one of the lesson responses of the first lesson step object.
3. The system of claim 1, wherein instantiating each knowledge edge comprises:
defining a first knowledge edge attribute to identify a first knowledge entity as a source node for the knowledge edge;
defining a second knowledge edge attribute to identify a second knowledge entity as a destination node for the knowledge edge; and
defining a third knowledge edge attribute to identify an edge type for the knowledge edge, such that the edge type defines the course flow relationship between its source and destination nodes.

4. The system of claim 3, wherein the edge type is identified as a prerequisite edge that defines the course flow relationship between the source and destination nodes so that, at runtime, the first knowledge entity is required to be consumed prior to the second knowledge entity.

5. The system of claim 3, wherein the edge type is identified as a variant edge that defines the course flow relationship between the source and destination nodes so that, at runtime, a selection is automatically made between presenting the first knowledge entity and presenting the second knowledge entity based on criteria stored in association with the knowledge edge.

6. The system of claim 1, wherein the course authoring platform comprises a graphical user interface that receives the authoring commands from a human author and translates the authoring commands to datagraph commands for execution by the set of processors.

7. The system of claim 1, wherein the course datagraph macrostructure is stored as a directed acyclic graph structure, and the lesson microstructure is stored as a directed graph structure.

8. The system of claim 1, wherein the set of processors implements the course authoring platform to execute the datagraph commands further to:

- embed the knowledge entity in the course datagraph macrostructure with at least one practice datagraph microstructure in the course data store by instantiating a plurality of practice step objects as nodes of the practice datagraph macrostructure, defining a set of practice responses associated with each practice step object, and linking each practice step object with at least one other practice step object in the practice datagraph microstructure by instantiating a respective practice edge that defines a practice flow relationship between the practice step objects.

9. The system of claim 1, wherein the processor that implements the course authoring platform is communicatively coupled with the course data store over a public network.

10. A method for building an e-learning datagraph structure, the method comprising:

- instantiating a plurality of knowledge entities as nodes of a course datagraph macrostructure;
- linking a first knowledge entity and a second knowledge entity in the course datagraph macrostructure by instantiating a knowledge edge having a set of knowledge edge attributes that defines a course flow relationship between the first and second knowledge entities; and
- embedding each knowledge entity in the course datagraph macrostructure with at least one lesson datagraph microstructure by:
  - instantiating a plurality of lesson step objects as nodes of the lesson datagraph microstructure;
  - defining a set of lesson responses associated with each lesson step object, each lesson response including a set of lesson response attributes; and
  - linking a first lesson step object and second lesson step object in the lesson datagraph microstructure via one of the lesson responses of the first lesson step object by instantiating a lesson edge that defines a lesson flow relationship between the first and second lesson step objects.

11. The method of claim 10, wherein instantiating the knowledge edge comprises:

- defining a first knowledge edge attribute to identify the first knowledge entity as a source node for the knowledge edge;
- defining a second knowledge edge attribute to identify the second knowledge entity as a destination node for the knowledge edge; and
- defining a third knowledge edge attribute to identify an edge type for the knowledge edge, such that the edge type defines the course flow relationship between the first and second knowledge entities.

12. The method of claim 11, wherein the edge type is identified as a prerequisite edge that defines the course flow relationship between the first and second knowledge entities so that, at runtime, a selection is automatically made between presenting the first knowledge entity and presenting the second knowledge entity based on criteria stored in association with the knowledge edge.

13. The method of claim 1, wherein the edge type is identified as a variant edge that defines the course flow relationship between the first and second knowledge entities so that, at runtime, a selection is automatically made between presenting the first knowledge entity and presenting the second knowledge entity based on criteria stored in association with the knowledge edge.

14. The method of claim 10, further comprising:

- linking each remaining one of the plurality of knowledge entities with at least one other of the plurality of knowledge entities in the course datagraph macrostructure by instantiating a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship between the respective knowledge entities.

15. The method of claim 10, wherein the embedding is further by:

- linking each remaining one of the plurality of lesson step objects with at least one other of the plurality of lesson step objects in the lesson datagraph microstructure by instantiating a respective lesson edge having a respective set of lesson edge attributes that defines a lesson flow relationship between the respective lesson step objects.

16. The method of claim 10, further comprising:

- storing the course datagraph macrostructure in a non-transient storage medium.

17. The method of claim 10, wherein the course datagraph macrostructure is a directed acyclic graph structure, and the lesson datagraph microstructure is a directed non-acyclic graph structure.

18. The method of claim 10, further comprising:

- embedding each knowledge entity in the course datagraph macrostructure with at least one practice datagraph microstructure by:
  - instantiating a plurality of practice step objects as nodes of the practice datagraph microstructure;
defining a set of practice responses associated with each practice step object, each practice response including a set of practice response attributes; and
linking a first practice step object and second practice step object in the practice datagraph microstructure via one of the practice responses of the first practice step object by instantiating a practice edge that defines a practice flow relationship between the first and second practice step objects.

19. A system for consuming an e-learning datagraph structure, the system comprising:
a non-transient course data store having an e-learning datagraph structure stored thereon, the datagraph structure comprising:
a plurality of knowledge entities stored as nodes of a course datagraph macrostructure in the course data store, each knowledge entity being linked with at least one other knowledge entity in the course datagraph macrostructure via a respective knowledge edge having a respective set of knowledge edge attributes that defines a course flow relationship among the knowledge entities; and
a plurality of lesson datagraph microstructures, each embedded in a respective knowledge entity in the course datagraph macrostructure and having a plurality of lesson step objects stored as nodes of the lesson datagraph microstructure, each lesson step object being associated with a set of lesson responses and linked with at least one other lesson step object in its lesson datagraph microstructure via a respective lesson edge, so that the lesson edges define a lesson flow relationship between the lesson step objects of the lesson datagraph microstructure;
a first processor in communication with the course data store that implements a first instance of a course consumption platform to:
determine a profile of a first student; and
adaptively generate and display a first instance of an e-learning course from the e-learning datagraph structure according to the profile of the first student; and
a second processor in communication with the course data store that implements a second instance of the course data store that implements a second instance of the course consumption platform to:
determine a profile of a second student; and
adaptively generate and display a second instance of the e-learning course from the e-learning datagraph structure according to the profile of the second student, the second instance being different from the first instance.

20. The system of claim 19, wherein each of the first and second processor is communicatively coupled with the course data store over a public network.