Towards a Conceptual and Service-Based Adaptation Model

Martin Harrigan and Vincent Wade

Department of Computer Science, Trinity College Dublin, Ireland
{martin.harrigan, vincent.wade}@cs.tcd.ie

Abstract. Current practice in adaptation modeling assumes that concepts and relationships between concepts are the fundamental building blocks of any adaptive course or adaptive application. This assumption underlies many of the mismatches we find between the syntax of an adaptation model and the semantics of the 'real-world' entity it is trying to model, *e.g.* procedural knowledge modeled as a single concept and services or activities modeled as pockets of intelligent content. Furthermore, it results in adaptation models that are devoid of truly interactive services with workflow and dataflow between those services; it is impossible to capture the semantics of a process-oriented application, e.g. activity-based learning in education and Standard Operating Procedures (SOPs) in the workplace. To this end, we describe a representation of a conceptual and service-based adaptation model. The most significant departure from existing representations for adaptation models is the reification of services. The goal is to allow for the adaptation of the process itself and not just its constituent parts, *e.g.* an SOP can be adapted to the role or job function of a user. This expressive power will address the mismatches identified above and allow for activity-based and process-oriented adaptive applications.

1 Introduction

The cornerstone of an authoring tool-set for an adaptive course or adaptive application is the adaptation model. It is the specification of the adaptive storyline or narrative [1] that guides each user through a course or application. What makes the adaptation model so difficult to author, is the very fact that there are so many different paths each user can follow. A specification of an adaptation model serves two functions. Firstly, it determines (or is determined by) what is expressible within an authoring tool-set [1, 2, 3, 4, 5]. Secondly, it hides the intricacies of the rule-based language or low-level instruction-set of an adaptive engine, e.g. AHA! [2]. However, a problem with current adaptation models is that they are unduly focused on concepts and relationships between concepts [6, 7, 8]. Procedural knowledge, interactive services and activities are difficult, if not impossible, to model. Adaptation models require greater expressive power in order to adequately model these possibilities.

To better illustrate this short-coming, consider adaptation models within an educational or learning environment. Activity is an important part of learning. For example,

This work was performed within the GRAPPLE Project. GRAPPLE (Generic Responsive Adaptive Personalized Learning Environment) is an EU FP7 STREP project that aims to deliver a *Technology-Enhanced Learning* (TEL) environment that guides learners through a learning experience, automatically adapting to personal preferences, prior knowledge, skills and competences, learning goals, and the personal or social context in which the learning takes place. This functionality will be provided as a set of adaptive learning services that will be integrated with existing Open Source and commercial *Learning Management Systems* (LMSs).

Active Learning [9] is a model of instruction that places the responsibility of learning on learners through practical activity. In order to assimilate the learning material, the learners must actively engage with it. Examples of active learning include class discussion and ‘think-pair-share’. Situated Learning [10] states that learning is a function of activity, context and the culture in which it takes place. It occurs as a result of social interaction, *e.g.* through workshops, role-playing, field trips, etc. Many other learning theories are bound to the notion of an activity. At the same time, learners learn best when the learning experience offered to them meets their needs and stimulates their preferred modes of learning [11]. When creating an adaptation model for such an environment, we should consider not only adaptive content and adaptive navigation, but also adaptive learning activities [12, 13]. We need to shift from learning objects and content that are ‘retrieved’ or ‘accessed’ to learning activities that are ‘experienced’.

Our conceptual and service-based adaptation model is novel in the sense that it treats services as first-class citizens. Services and workflow between those services are explicitly modeled. The goal is to provide for the adaptation of both concepts and services, *i.e.* adaptive content, adaptive navigation, adaptive services, and adaptive workflow. This is a fusion of the predominantly concept-centric models of adaptive hypermedia and the service-centric models of workflow and process-oriented systems. In order to accomplish this, we generalize the notion of a *Conceptual Relationship Type* (CRT) [14] to an *Abstract Relationship Type* (ART) to cater for both.

This paper describes an adaptation model that explicitly provides for interactive services and activities. It does not consider the theoretical or pedagogical concerns when designing an actual adaptive course or adaptive application, nor does it detail a user interface of an actual authoring tool-set. At the same time, we note that this work is undertaken within a suitable context (see earlier footnote). The paper is organized as follows. In Sect. 2 we provide definitions for commonly used terms. In Sect. 3 we enumerate the broad requirements for a conceptual and service-based adaptation model. The syntax and semantics of the abstract relationship type and adaptation model are detailed in Sect. 4. This is followed by their usage in Sect. 5. We conclude and consider future work in Sect. 6.

2 Definitions

This section establishes definitions for commonly used terms in the paper.

A *Resource Model* (RM) [15] models all assets (concept instances and service instances) that are available for use. These assets comprise text, images, sound, video, chat services, email services, forum services, exercises, references, datasets, etc. The actual assets can be contained in a closed-corpus repository or retrieved from an open-corpus repository, such as the Web. However, every modeled asset must provide meta-data in the form of a required set of controlled attributes. These attributes can specify, for instance, the difficulty of the content (e.g. introductory, intermediate, expert), the language, the media format, etc. Service instances also need to expose their inputs and outputs.

A *Domain Model* (DM) [6, 15] comprises a *Concept Domain Model* (CDM) and a *Service Domain Model* (SDM). A CDM models abstract concepts and relationships between them. The actual concept instances are modeled in an RM. Concept spaces, taxonomies, ontologies, and topic maps are all considered to be DMs. An SDM is an analogous model for services. It models abstract services and relationships between them. The actual service instances are modeled in an RM. Services can be domain-specific in their very nature whereas other services can be domain-independent but can be parameterized so that they appear domain-specific, *e.g.* 'classDiscussion' can apply to any domain, but when the discussion is parameterized with a particular topic, it becomes domain-specific.

An *Abstract Relationship Type* (ART) is a relationship that can be instantiated between concepts, between services, or between concepts and services and provides for adaptive behavior. It is a generalization of a *Conceptual Relationship Type* (CRT) [14], in that its parameters can be concepts or services. An authoring tool-set will provide a toolbox or palette of commonly used ARTs. ARTs are distinct from the relationships in a DM; ARTs provide for adaptive behavior, DM relationships do not. However, in some cases ARTs can be derived from relationships in a DM, allowing DM relationships to be mapped to particular ARTs. This is explored in Sect. 4.1 when discussing the `derivedFrom` element.

A *conceptual and service-based Adaptation Model* (AM) is an assembly of concepts and services retrieved from a DM, with relationships between them, a workflow specification guiding the execution of the services, and provisions for adaptive content, navigation, service selection, workflow, and presentation. The relationships are instantiated ARTs whose parameters are bound to the assembled concepts and services.

A *User Model* (UM) [6] is a store of *attributes* describing individual users and groups of users; it permits software to adapt to their characteristics. It is sometimes useful to distinguish between a *UM schema* – how the UM is logically structured, and the actual data stored in a particular UM. A simple *overlay UM* stores, for each concept (resp. service) in a DM, a user's knowledge or familiarity with that concept (resp. service).

A *strategy* is a combination of ARTs that can only be instantiated within some predefined pattern. It allows for the aggregation of ARTs in order to implement, for example, quality control procedures, WebQuests, the Jigsaw learning technique, etc. The predefined pattern can permit a certain degree of customization, *e.g.* not all ARTs are mandatory. This is explored in Sect. 4.1 when discussing the `preConditions` and `postConditions` elements.

This proliferation of models does not lead to undue complexity [16]; in fact, it promotes separation of concerns and allows, for example, DM experts to focus on DMs and educational technologists and instructional designers to focus on ARTs and CRTs of a pedagogical nature. In some cases, they may need to collaborate.

In the following sections, we refer to an example from the educational or learning domain based on the student peer review process. This example is presented in detail in Sect. 5

3 Requirements

In this section we enumerate some broad requirements for a conceptual and service-based AM. These are based on recent interviews with potential users of an Adaptive Learning System [17, 18] (an important application area for conceptual and service-based adaptation models) and on previous experience with such models [12, 19, 1, 20, 21].

1. Concepts and services should be seamlessly integrated. Early approaches [21] provide distributed, intelligent services that are embedded in concepts, that is, the concepts contain the services. However, this does not support any form of information flow between those services. The services amount to isolated, disconnected applets.
2. The representation should allow an author to create, manipulate, delete, and organize all aspects of an AM. A canvas or workspace, into which an author can 'drag-and-drop' graphical representations of concepts, services and their relationships, supports the inherently dynamic and creative process of designing an AM. The representation should also include the spatial layout of an AM. Although this information does not

concern the semantics of an AM, it can be suggestive to an author of the directionality and temporal sequence of the model.
3. The entities and relationships that an author has in mind when creating an AM (introductory explanations, exercises, online discussions, prerequisites, roles etc.) should be the same entities and relationships that they are controlling and manipulating in the representation [22]. The model needs to operate at an appropriate level of abstraction. In practical terms, we would like an author to create, for example, an AM that describes the student peer review process by assembling concepts and services like 'introductoryResource', 'reviewResources', 'discussion', and 'writeReport' and instantiating relationships like 'prerequisite' and 'followedBy'. The service 'discussion' may be realized in any number of ways depending on what service instances or resources are available. However, when creating an AM, it is more appropriate for an author to manipulate graphical artifacts entitled `discussion`. Similarly, an encapsulated relationship entitled 'prerequisite' is more intuitive to an author than the adaptive behavior (adaptive sequencing and presentation [23, 24]) it embodies.
4. Adaptive behavior should be abstracted and refactored [25] into reusable, flexible relationships. Concepts and services can be reused in many different AMs. The same should be true for relationships like 'prerequisite', 'analogousTo', 'conformsTo', 'isSometimesFollowedBy', etc. This is a noticeable departure from existing representations and authoring tool-sets [7] that mix classical semantic relationships like 'is-a' and 'has-a' with application relationships like 'prerequisite' and 'analogousTo' which also embody adaptive behavior.
5. Practitioners appreciate the complexity and difficulty in authoring an AM [17, 18]. This is the 'price of adaptivity' [7]. However, they also seek an authoring tool-set that can produce AMs without the intervention of specialists. To overcome this difficulty, the tool-set should provide as much advice and support as possible, e.g. during author-time, the act of instantiating a relationship in an AM should only be allowed under certain 'terms-and-conditions'. This intelligence on the part of the tool-set is only possible if the representation for each relationship includes such conditions. However, the benefit is that the author can be advised at an early stage when something is askew.

These requirements, namely the seamless integration of concepts and services (R1), the 'mix-and-match' paradigm (R2), an appropriate level of abstraction (R3), the abstraction of reusable adaptive behavior (R4), and as much advice and support at author-time as possible (R5) underlie many of the decisions in the following section.

4 Syntax and Semantics

In this section we present the syntax and semantics of the ART and the AM. ARTs are instantiated in an AM so it is impossible to present one without the other. Fig. 1 summarizes their content. The ART schema comprises six elements described below. The AM schema comprises five elements also described below. An AM and a DM are linked through the AM's `entities` element: it refers to concepts and services in the DM. An AM and an ART are linked through the AM's `relationships` element: it contains instantiations of ARTs.

4.1 The Abstract Relationship Type (ART)

ARTs are the glue that bind together the concepts and services in an AM. They are also wrappers around the specification of adaptive behavior; they guide the author to the

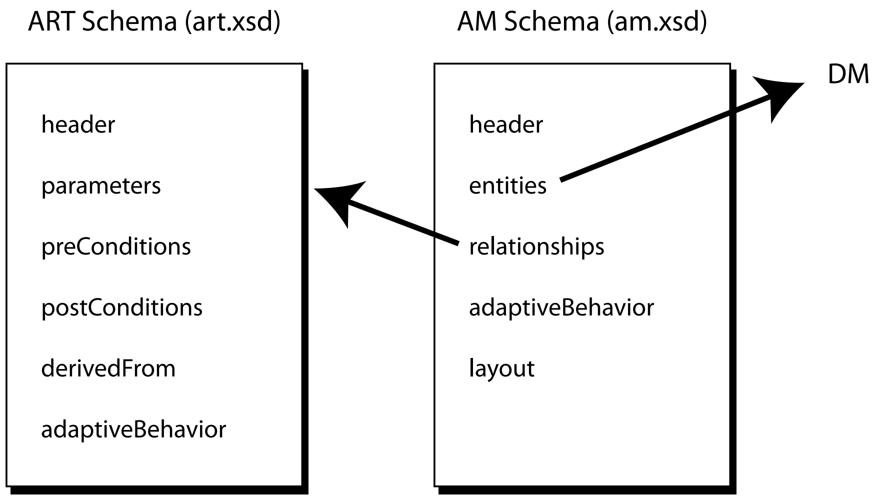

Fig. 1. The ART and AM representations are described by XML Schema.

appropriate places in which to include this behavior. If more than one end point of an ART is a service, then it also controls the workflow between those services, *i.e.* sequencing, branching, merging, etc. By refactoring [25] the adaptive behavior into reusable ARTs, we hope to improve the non-functional properties of an AM; namely its readability, maintainability, and extendability. An authoring tool-set should provide an author with a toolbox of commonly used ARTs. If an authoring tool-set is pedagogically-oriented, *i.e.* responsible for producing adaptive courses in a learning environment, then the ARTs should be pedagogical in nature, *e.g.* 'prerequisite', 'analogous-to', 'criticism-of', etc. However, these are by no means the only type of ART we envision. For example, if an authoring tool-set is responsible for producing *Standard Operating Procedures* (SOPs) for a workplace, then the ARTs should cater for quality-control, *e.g.* 'conforms-to', 'requires-accreditation', etc. The author may also need to tweak existing ARTs or create new ARTs from scratch. Each ART comprises `header`, `parameters`, `preConditions`, `postConditions`, `derivedFrom`, and `adaptiveBehavior` elements. These are specified using an XML Schema [26] and are explained in the following subsections.

The `header` Element The `header` identifies an ART (`identifier`, `author`, `name`, `description`, `keywords`) and provides rudimentary support for versioning (`majorVersion`, `minorVersion`) and permissions (`permissions`). `identifier` is a *Universally Unique IDentifier* (UUID) [27] allowing each authoring tool-set to uniquely identify information without significant central coordination. Authors are identified in the same way. `name`, `description` and `keywords` are intended to help an author decide when and where an ART is appropriate. The version number is separated into two fields, `majorVersion` and `minorVersion`. The former is initialized to one and is incremented when the 'interface' to an ART changes, *i.e.* the number or types of parameters (specified below) are changed. The latter is initialized to zero and is incremented for every other change in an ART. It is reset to zero when `majorVersion` is incremented. An instantiated ART can be updated to a new minor version automatically. However, an instantiated ART can be updated to a new major version only through the intervention of an author. `created` and `lastUpdated` are stored in Coordinated Universal Time (UTC) but can be translated to and from local times by an authoring tool-set. The `permissions` element determines the access rights to an ART. It follows a Unix-like mechanism. ARTs are managed using three distinct classes: `author`, `group`, and `world`. Each can have `read`, `write`, and/or `publish` permissions. There are, of course, many alternatives, including role-based access control [28]. It is important to note that the model does not enforce these permissions but only allows for their representation. It is left to an authoring tool-set or a local or remote persistence service to enforce the rules.

The `parameters` Element An ART is instantiated between concepts and services in an AM. The `parameters` element specifies the number and types of concepts and services that it can be bound to. Each `parameter` is grouped under one of three distinct classes: `sources`, `targets`, and `undirected`. The first comprises parameters that are 'antecendents' of an ART; the second comprises parameters that are 'consequents' of an ART. For example, a 'prerequisite' ART might describe entities $X_1, X_2, \ldots, X_s$ as being prerequisites for entities $Y_1, Y_2, \ldots, Y_t$. $X_1, X_2, \ldots, X_s$ are classed as `sources` and $Y_1, Y_2, \ldots, Y_t$ are classed as `targets`. The third class, `undirected`, comprises parameters that take part in an undirected ART. For example, a 'similar-to' ART might describe an entity X as being similar to an entity Y. If Y must also be similar to X then both X and Y are classed as `undirected`.

Each `parameter` has its own `identifier` (UUID), `name`, `type`, `minOccurs`, and `maxOccurs` elements. `type` can have one of four values: `concept`, `service`, or `either`. They specify whether the parameter can be bound to a concept, service or either. The `minOccurs` and `maxOccurs` elements afford several possibilities including mandatory (`minOccurs` $= 1$), optional (`minOccurs` $= 0$), and unbounded (`minOccurs` $= 1$ and `maxOccurs` $= -1$) parameters. An ART must contain at least one `parameter`. An ART with exactly one `parameter` can be considered a tag or loop. It adds adaptive behavior to single a entity.

The `preConditions` and `postConditions` Elements The `preConditions` and `postConditions` elements are the 'terms-and-conditions' of instantiating an ART in an AM. The `preConditions` must hold before an ART is instantiated and the `postConditions` must hold after. All conditions are structural and can be checked at author-time. They provide an authoring tool-set with some intelligence to guide an author. We enumerate and describe the conditions which can appear within either the `preConditions` or `postConditions` tags:

1. `minDegree`: This is the minimum number of relationships that an entity bound to a parameter of an ART can take part in.
2. `maxDegree`: This is the maximum number of relationships that an entity bound to a parameter of an ART can take part in.
3. `directedCycle`: This determines whether instantiations of an ART can induce a directed cycle. For example, it should not be possible for instantiations of a 'prerequisite' ART to induce a directed cycle: X is a prerequisite for Y, Y is a prerequisite for Z and Z is a prerequisite for X.
4. `isParameterOfART`: This constrains an entity bound to a parameter of an ART into being bound to a parameter of some other ART. This forces certain ARTs to be instantiated in sequences or in combination with other ARTs, thus providing an implementation of our earlier definition of a strategy (see Sect. 2).

The `derivedFrom` Element An author creates an AM by assembling collections of concepts and services and instantiating ARTs between them. An authoring tool-set should assist the author in this endeavor. Consider the case where an author copies two entities, X and Y, from a DM into an AM and there already exists a DM relationship between them, *e.g.* X 'is-a' Y. It may sometimes be desirable and expedient for ARTs to be instantiated automatically or semi-automatically (through a dialogue with the author) in response to this, *e.g.* Y is a prerequisite of X can be derived from the fact that X 'is-a' Y. The derivations are specified within the `derivedFrom` element. This approach differentiates between user-agnostic DM relationships that are present in concept spaces, taxonomies, ontologies, and topic maps and ARTs that have adaptive behavior and references to UM attributes associated with them while allowing the latter to be automatically derived from the former. This allows, for example, the automatic derivation of adaptive navigation structures, e.g. concept-based hyperspaces [7].

The `adaptiveBehavior` Element The final element, `adaptiveBehavior`, specifies the adaptive behavior of an ART using an appropriate adaptation language [29, 30] within a single CDATA block. This language should be independent from the DM, adaptive engine, and target platform. It can reference the parameters from the `parameters` element above and can query and update the attributes of the UM. At publish-time (when the AM is translated to the low-level instruction-set of an adaptive engine), references to parameters are substituted with the identifiers of the entities they are bound to.

4.2 The Conceptual and Service-Based Adaptation Model (AM)

The AM is a model into which concepts and services are assembled and ARTs are instantiated. It is a mapping between the 'knowledge space' and what is eventually presented to a user [7]. An author creates an AM in a declarative fashion; he or she declares the high-level structure of an AM. The entities and relationships that the author has in mind when preparing an AM are the same artifacts that he or she is controlling and manipulating in the workspace, *e.g.* an introductory explanation, exercise, online discussion, etc. The adaptive behavior of each ART and an AM as a whole (also specified in an adaptation language) provide an imperative list of actions that realize the author's intent. Each AM comprises `header`, `entities`, `relationships`, `adaptiveBehavior`, and `layout` elements. These are specified using an XML Schema [26] and are explained in the following subsections.

The `header` Element The `header` element is identical to the `header` of an ART; it identifies an AM (`identifier`, `author`, `name`, `description`, `keywords`) and provides rudimentary support for versioning (`majorVersion`, `minorVersion`) and permissions (`permissions`).

The `entities` Element The building blocks of an AM are its `entities`. Each `entity` can be either a concept or a service from a DM and is identified through an `identifier`. It references the `identifier` and `version` of the concept or service (if they exist). Each `entity` also includes a list of `controlledAttributes` along with values. These place further restrictions on the actual concept or service instance that can be retrieved for each entity when an AM is published. For example, suppose an author includes a service from a DM entitled 'discussion'. However, suppose also that there are a number of different service instances or implementations that offer this functionality but vary in form. For example, in one instance, an instant-messaging portlet is provided, in another, a discussion forum or bulletin board system is used. One standardized possibility for the controlled list of attributes the Dublin Core [31] elements. In the case of 'discussion', an attribute named 'duration' with a value of 'short' would restrict the actual service instances that can be retrieved for this entity to those that persist for a short period of time.

The `relationships` Element The `relationships` element is the container for instantiated ARTs. Each instantiated ART is identified through an `identifier`. It also references the `identifier` and `version` of an ART it is an instantiation of. It maps the `parameters` of an ART to the `entitys` of an AM. This mapping determines the entities that the instantiated ART is gluing together. For example, suppose a 'conforms-to' ART has two parameters: a service X (source parameter) and a concept Y (target parameter). When an author instantiates this ART between, say, an interactive 'generalLedger' service and the 'doubleEntryBookkeeping' concept, X is bound to 'generalLedger' and Y is bound to 'doubleEntryBookkeeping'. This is identical to the binding of arguments to function parameters in programming languages. An authoring tool-set can ensure at author-time that the `type`, `minOccurs` and `maxOccurs` children of each `parameter` element of each instantiated ART are satisfied. Furthermore, it can also ensure at author-time that the `preConditions` and `postConditions` of each instantiated ART are satisfied. An instantiated ART must be bound to at least one `entity`. An instantiated ART that is bound to exactly one `entity` can be considered a tag or loop. It adds adaptive behavior to a single entity.

The `adaptiveBehavior` Element The `adaptiveBehavior` element specifies the adaptive behavior of an AM using an appropriate adaptation language [29, 30] within a single CDATA block. This behavior is independent of any of the individually instantiated ARTs. It can reference the individual `entity` and `relationship` elements above and can query and update the attributes of the UM. For example, it could save and retrieve the last known position of a user within the structure of an AM or it could change the role or qualifications of a user at the appropriate time.

The `layout` Element Finally, the `layout` element provides a graphical representation of an AM. This representation is based on a *graph model.* A *graph* is a mathematical structure typically used to model the relationships between objects. It consists of *nodes* or *vertices* linked together by (possibly *directed*, *weighted*) *edges.* A graph can be defined by a list of vertices and edges, by using a matrix structure or simply by a *drawing* of the graph. A drawing of a graph is a visual representation of its vertices and edges. We describe one such representation. A polygonal or circular shape depicts a vertex with an optional label printed inside or close to the shape. An edge is depicted by a polyline or curve which connects the shapes and may also have an optional label. If an edge has an associated direction it is decorated with an arrow to indicate this direction. A 'good' drawing of a graph not only defines the graph but also effectively aids its comprehension by authors. Our graph drawings are also subject to *constraints* [32] that reflect the domain of AMs, *e.g.* their layout can be suggestive of the directionality and temporal sequence of the model. Graphs provide a uniform interface over heterogeneous models with standard graphical operations and easily-understood semantics. The same interface can be used for both creating and manipulating existing AMs. Indeed, we expect high-quality AMs to evolve rather than be completely designed from the ground up. Therefore, the interface must cater for both creation and maintenance.

A vertex can represent either a concept or a service. An edge represents an ART. Any subgraph can be contracted to a single vertex. It can be expanded again as necessary.

An entire graph represents an AM. An author creates an AM by dragging vertices and edges from a palette and adding them to a graph. For example, adding a vertex that represents some concept and linking it with an edge to some service indicates that the concept and the service are related. The service might be a realization of the concept or it might conform to the principles of the concept, etc. The type of relationship will be indicated by the type and labeling of the edge. Vertices and edges can also be removed. The actual drawing or layout of a graph does not concern the semantics of an AM. It is purely an aid for the author when comprehending the structure and contents of an AM. The interface enforces constraints during author-time. For example, when dragging an ART from a palette and adding it to a graph, the edge can only be added between vertices whose type matches those of the ART's parameters.

We use GraphML [33] to store our layout data. Our AM schema imports the GraphML schema and the contents of the `layout` element adhere to this schema. GraphML can also be extended using either attributes to attach scalar values or key/data pairs to attach structured data to a graph. This reduces the burden on us, allowing us to focus solely on the representation of an AM and not on the graphical and topological aspects.

5 Example Usage

The LADiE project [34] has developed a broad set of use case scenarios for activity-based e-Learning. One such scenario requires students to review a set of resources relating to a specific topic. This review is followed by a discussion on the topic, which is guided by the teacher. The students then write and submit a report based on their discussion. There is also the possibility that a student can re-examine the resources, and discuss the problem while writing the report at the same time. The list of steps is:

1. Teacher briefs students on the activity.
2. Students log into the system and access the resources.
3. Students discuss the problem.
4. Students write report.
5. Students submit report.
6. System notifies teacher that report has been submitted.

We wish to adapt both the content and the workflow of these activities to the needs of the students. The content can be adapted based on various adaptive axes such as the students' prior knowledge. In addition to this, the services can be tailored to the needs of the students as well as to their context. For example, discussion between groups of students can be supported by various different services. A bulletin board service might be appropriate if the activity is expected to run over a long period of time while a chat room service might be more appropriate for a relatively short activity where all of the students are online. Similarly, the writing and submission tasks can employ a choice of services.

For the purpose of this example, we assume that the topic is *Digital Rights Management* (DRM) and that the author has access to a DM containing concepts like 'DRM', 'DRMTechnologies', and 'DRMLegislation' and services like 'reviewResources', 'discussion', 'writeReport', and 'submitReport'. The author drags each of these entities into an AM. He instantiates a 'prerequisite' ART between 'DRM' and 'DRMTechnologies' and between 'DRM' and 'DRMLegislation'. He instantiates a 'isLegalIssue' ART whose only parameter is 'DRMLegislation'. This ensures that the 'DRMLegislation' concept is only presented to the class if they have an interest in law. The author instantiates an 'andSplit' ART between 'DRMTechnologies' and the services 'reviewResources', 'discussion'

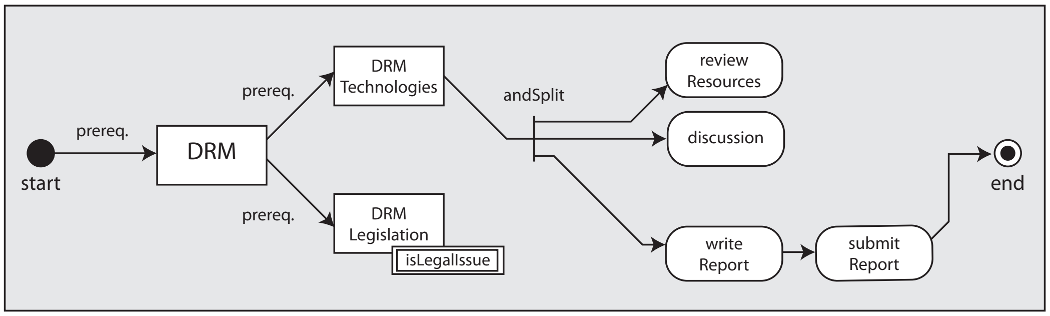

Fig. 2. The student peer review process.

and ‘writeReport’. This indicates that once a student has studied the ‘DRMTechnologies’ concept he or she can perform the next three services in parallel, switching between them as needed. These may be presented to the user as three separate windows or portlets. Of course, it is altogether possible that an instantiation of an ‘andSplit’ ART is not only distinguished by its label but also by a unique graphical depiction –one that is similar to ‘split’ constructs in visual workflow languages. Finally, the author instantiates a ‘sequence’ ART between the ‘writeReport’ and the ‘submitReport’ services. The outcome is illustrated in Fig 2.

6 Conclusions and Future Work

We have presented the basic ingredients of an AM that supports the adaptive selection, sequencing and presentation of both concepts and services. By putting services on an equal footing with concepts, we allow for adaptive service selection and adaptive sequencing of the services themselves. We have generalized the notion of CRTs [14] to cater for these new possibilities. Within the context of the GRAPPLE Project (see earlier footnote), we are developing a graph-based authoring tool-set that will be based on this AM. The tool-set will comprise a *Rich Internet Application* (RIA) that will enable authors, in particular teachers, educational technologists and instructional designers, to compose AMs representing adaptive courses and adaptive procedural simulations. Server-side functionality will enable translation of this AM to the rule-based languages or low-level instruction-sets of an adaptive engine, e.g. AHA! [2].

There are many possible extensions to the AM. For example, some DMs use a frame-like knowledge representation, *i.e.* they model the internal structure of each concept or service. Instead of assembling entire concepts or services in an AM, an author may want to mix and match parts of different concepts and services. Also, we believe that the analogy between ARTs and functions in programming languages, mentioned in Sect. 4.2, is particularly apt. We are considering the benefits of inheritance and overloading when applied to ARTs.

References

[1] Dagger, D., Wade, V., Conlan, O.: Personalisation for All: Making Adaptive Course Composition Easy. Educational Technology and Society **8**(3) (2005) 9–25

[2] De Bra, P., Smits, D., Stash, N.: Creating and Delivering Adaptive Courses with AHA! In Nejdl, W., Tochtermann, K., eds.: Proceedings of the 1st European Conference on Technology Enhanced Learning (EC-TEL'06), Springer (2006) 21–33

[3] Specht, M., Kravčík, M., Klemke, R., Pesin, L., Hüttenhain, R.: Adaptive learning environment for teaching and learning in winds. In: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02), Springer (2002) 572–575

[4] Kravčík, M., Specht, M., Oppermann, R.: Evaluation of WINDS Authoring Environment. In: Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'04), Springer (2004) 166–175

[5] Cristea, A., de Mooij, A.: Adaptive Course Authoring: My Online Teacher. In: Proceedings of the 10th International Conference on Telecommunications (ICT'03). (2003) 1762–1769

[6] De Bra, P., Houben, G., Wu, H.: AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia. In: Proceedings of the 10th ACM Conference on Hypertext and Hypermedia (HYPERTEXT'99), ACM (1999) 147–156

[7] Brusilovsky, P.: Developing Adaptive Educational Hypermedia Systems: From Design Models to Authoring Tools. In Murray, T., Blessing, S., Ainsworth, S., eds.: Authoring Tools for Advanced Technology Learning Environments. Kluwer Academic Publishers (2003) 377–409

[8] Cristea, A., Smits, D., De Bra, P.: Towards a Generic Adaptive Hypermedia Platform: A Conversion Case Study. Journal of Digital Information **8**(3) (2007)

[9] Bonwell, C., Eison, J.: Active Learning: Creating Excitement in the Classroom. AEHE-ERIC Higher Education Report 1, Washington, D.C. (1991)

[10] Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. 1st edn. Cambridge University Press (1991)

[11] Conlan, O., Hockemeyer, C., Wade, V., Albert, D., Gargan, M.: An Architecture for Integrating Adaptive Hypermedia Services with Open Learning Environments. In Barker, P., Rebelsky, S., eds.: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA'02), AACE (2002) 344–350

[12] Conlan, O., O'Keeffe, I., Brady, A., Wade, V.: Principles for Designing Activity-Based Personalized eLearning. In Spector, J., Sampson, D., Okamoto, T., Kinshuk, Cerri, S., Ueno, M., Kashihara, A., eds.: Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT'07), IEEE Computer Society (2007) 642–644

[13] O' Keeffe, I., Brady, A., Conlan, O., Wade, V.: Just-In-Time Generation of Pedagogically Sound, Context Sensitive Personalized Learning Experiences. International Journal on E-Learning **5**(1) (2006) 113–127

[14] De Bra, P., Aerts, A., Rousseau, B.: Concept Relationship Types for AHA! 2.0. In Richards, G., ed.: Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn'02), AACE (2002) 1386–1389

[15] Conlan, O., Wade, V., Bruen, C., Gargan, M.: Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning. In De Bra, P., Brusilovsky, P., Conejo, R., eds.: Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'02), Springer (2002) 100–111

[16] Conlan, O., Wade, V.: Evaluation of APeLS - An Adaptive eLearning Service Based on the Multi-Model, Metadata-Driven Approach. In De Bra, P., Nejdl, W., eds.: Pro-

ceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'04), Springer (2004) 291–295

[17] Harrigan, M., Kravčík, M., Steiner, C., Wade, V.: What Do Academic Users Want from an Adaptive Learning System? Technical Report, Trinity College Dublin (2009) https://www.cs.tcd.ie/publications/tech-reports/reports.09/TCD-CS-2009-06.pdf.

[18] Harrigan, M., Kravčík, M., Steiner, C., Wade, V.: What Do Academic Users Really Want from an Adaptive Learning System? In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP'09) (to appear), Springer (2009)

[19] O'Keeffe, I., Conlan, O., Wade, V.: A Unified Approach to Adaptive Hypermedia Personalisation and Adaptive Service Composition. In Wade, V., Ashman, H., Smyth, B., eds.: Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'06), Springer (2006) 303–307

[20] Wilson, S., Blinco, K., Rehak, D.: An e-Learning Framework: A Summary. In: Proceedings of the Advancing Learning Technology Interoperability Lab Conference (ALT-I-LAB'04). (2004)

[21] Brusilovsky, P.: KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In Feldman, S., Uretsky, M., Najork, M., Wills, C., eds.: Proceedings of the 13th International Conference on World Wide Web – Alternate Track Papers and Posters (WWW'04), ACM (2004) 104–113

[22] Rosch, E.: Natural Categories. Cognitive Psychology **4**(3) (1973) 328–350

[23] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction **6**(2–3) (1996) 87–129

[24] Brusilovsky, P.: Adaptive Hypermedia. User Modeling and User Adapted Interaction **11**(1–2) (2001) 87–110

[25] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley (1999)

[26] W3C: XML Schema http://www.w3.org/XML/Schema.

[27] Leach, P., Mealling, M., Salz, R.: A Universally Unique IDentifier (UUID) URN Namespace. Technical Report RFC 4122, The Internet Engineering Task Force (IETF) (2005) http://tools.ietf.org/html/rfc4122.

[28] Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer **29**(2) (1996) 38–47

[29] Hendrix, M., De Bra, P., Pechenizkiy, M., Smits, D., Cristea, A.: Defining Adaptation in a Generic Multi Layer Model: CAM: The GRAPPLE Conceptual Adaptation Model. In Dillenbourg, P., Specht, M., eds.: Proceedings of the 3rd European Conference on Technology Enhanced Learning (EC-TEL'08), Springer (2008) 132–143

[30] Stash, N., Cristea, A., De Bra, P.: Adaptation Languages as Vehicles of Explicit Intelligence in Adaptive Hypermedia. International Journal of Continuing Engineering Education and Life-Long Learning **17**(4–5) (2007) 319–336

[31] The Dublin Core Metadata Initiative: http://www.dublincore.org.

[32] Tamassia, R.: Constraints in Graph Drawing Algorithms. Constraints **3**(1) (1998) 87–120

[33] Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., Marshall, S.: GraphML Progress Report. In Mutzel, P., Jünger, M., Leipert, S., eds.: Proceedings of the 9th International Symposium on Graph Drawing (GD'01), Springer (2001) 501–512

[34] JISC: Learning Activity Design in Education: LADiE. Technical report, The Joint Information Systems Committee (JISC) (2006) http://www.jisc.ac.uk/publications.