УДК 004.588 DOI: 10.14529/cmse140401

МОДЕЛИ И СТАНДАРТЫ ЭЛЕКТРОННОГО ОБУЧЕНИЯ

Н.С. Силкина, Л.Б. Соколинский

Статья представляет собой обзор моделей и стандартов, используемых в современных системах электронного обучения. Описывается общая концептуальная модель среды электронного обучения. Рассматриваются: модель данных для взаимодействия с электронными образовательными объектами; модель накопления контента, определяющая структуру образовательных объектов, способы их поиска и передачи между различными обучающими системами, а также способы упаковки контента; модель среды выполнения, определяющей структуру прикладного программного интерфейса для управления образовательными объектами; модель упорядочивания иерархического образовательного контента; модель компетенций, используемая для спецификации знаний, умений и навыков в системах электронного обучения. Также дается обзор стандарта SCORM, объединяющего в себе комплекс моделей электронного обучения.

Ключевые слова: электронное обучение, e-learning, модель метаданных, модель данных, модель накопления образовательного контента, модель среды выполнения, модель простого упорядочивания, модель компетенций, SCORM.

Введение

Под электронным обучением (e-Learning) понимаются все формы обучения с помощью компьютеров, которые имеют методический характер и направлены на построение у субъекта обучения (учащегося) системы знаний с учетом его индивидуального опыта, практики и подготовки. При этом информационные и телекоммуникационные системы, прежде всего Интернет и мультимедиа, используются в качестве основной платформы реализации процесса обучения [1].

Одним из наиболее популярных подходов к электронному обучению на сегодняшний день является создание образовательных модулей на основе динамических Web-страниц. При этом имеется большая диспропорция между временем, затрачиваемым на производство образовательного модуля, и временем его использования при обучении, поскольку традиционные технологии обучения не очень просто воспроизводятся при помощи современных авторских инструментальных средств [2]. Использование единых моделей и стандартов позволит снизить стоимость разработки электронных учебных курсов за счет использования уже существующих образовательных модулей и легкости переноса образовательного контента из одной системы управления обучением в другую.

Одним из факторов популярности электронного обучения является возможность развертывания приложения электронного обучения и даже всего пакета университетского курса (учебный план, лекции, приложения регистрации, загружаемые книги, методические пособия и т.д.) на смартфонах, планшетных ПК и других подобных устройствах, что позволяет студентам по-настоящему быть мобильными, а лучшие лекции профессоров может видеть и изучать более широкая аудитория [3].

Популярность электронного обучения выросла также благодаря снижению затрат на инфраструктуру и стремлению получить качественное образование по приемлемой для студентов цене, удобству и гибкости [4].

Так прием в колледжи США на курсы электронного обучения в 2008 году был на 16,9% выше, чем в 2007-м, при общем числе 4,6 млн студентов, и более половины колледжей и

университетов в США имеют онлайн-курсы или программы электронного обучения. Открытый университет Великобритании играет важную роль в системе образования Евросоюза, наблюдается также значительный рост интереса к электронному обучению в Азии, в частности в Японии, Корее, Китае и Индии. Университетский колледж Университета Мэриленда является крупнейшим в мире образовательным учреждением, имеющим программу онлайнового образования, этот университет предлагает свыше 40 программ для получения степеней бакалавра и магистра, причем это обучение можно целиком получить в режиме онлайн [3].

Однако, на сегодняшний день отсутствуют обзоры международных стандартов и моделей в области электронного обучения как на русском, так и на английском языках. Также отсутствуют полные описания международных моделей и стандартов в области электронного обучения на русском языке.

Настоящая статья в определенной мере призвана восполнить существующий пробел.

1. Концептуальная модель среды электронного обучения

Концептуальная модель среды электронного обучения [5] представлена в виде схемы, изображенной на рис. 1. В основе модели лежит база данных (база знаний), содержащая *образовательный контент* — структурированную совокупность информации различного типа в цифровом виде (текст, графика, видео и т.д.), которая представляет собой некоторый учебный материал. Среда обучения включает в себя также специальное программное обеспечение, которое представлено в виде *системы управления обучением* (LMS) и *сервиса времени выполнения* (RTS).

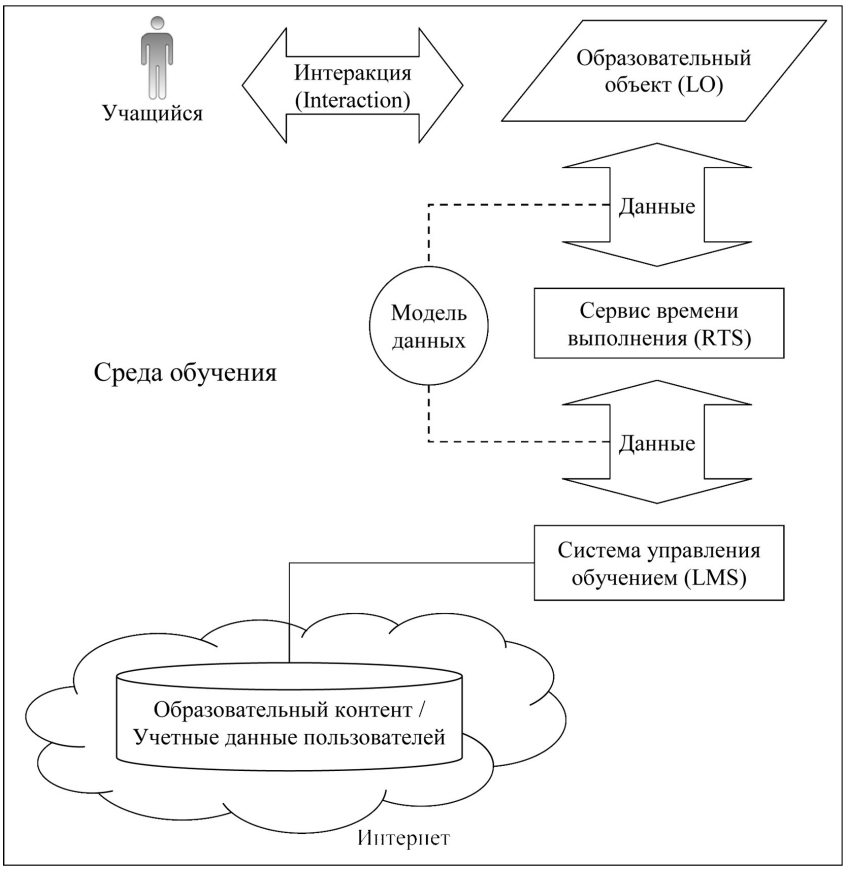

Рис. 1. Концептуальная модель среды электронного обучения

Образовательный контент, лежащий в основе схемы электронного обучения, представляет собой электронные образовательные ресурсы, распределенные в сети Интернет. Основным структурным элементом образовательного контента является *образовательный объект* (*Learning Object*) или кратко — *LO*. Объект LO представляет собой совокупность «оцифрованного» учебного материала, обладающего некоторой методической целостностью и представляющего собой единицу знания, передаваемую от LMS к учащемуся. LO обладает внутренней структурой и может включать в себя не только учебный материал, но и некоторые *инструктивные события* (*instructional events*), реализуемые в виде процедурного кода (скриптов). Например, объект LO может быть представлен в виде HTML-страницы, в которую внедрен видеоклип и скрипт ECMA, написанный в соответствии со стандартом IEEE 1484.11.2-2003 [6].

Система управления обучением (*Learning Management System*) или кратко — *LMS* представляет собой программное обеспечение, которое включает в себя функции регистрации и авторизации учащихся, администрирования образовательными объектами, управления процессом обучения, анализа результатов обучения, а также планирование и протоколирование действий учащегося.

Сервис времени выполнения (*Runtime Service*) или кратко - *RTS* представляет собой программное обеспечение, которое управляет выполнением и доставкой образовательного контента, и которое может включать в себя такие функции как распределение ресурсов, диспетчеризация процессов, управление вводом-выводом, обработка данных. Отметим, что в некоторых реализациях обучающих систем функции RTS могут интегрироваться в LMS.

Взаимодействие учащегося с обучающей системой реализуется на основе веб-протоколов и разбивается на шаги — *интеракции* (*interactions*). В ходе интеракции учащемуся доставляется образовательный объект, с которым он взаимодействует. При этом ему может демонстрироваться тот или иной учебный материал, и при определенных обстоятельствах учащийся может вводить некоторую ответную информацию в образовательный объект.

Данные, которыми обмениваются LMS, RTS и LO, должны представляться в соответствии с *моделью данных* (см. п. 3), являющейся внешней по отношению к LMS и RTS. Такой подход служит основой мобильности LO по отношению к различным системам электронного обучения. Модель данных должна поддерживать данные об учащемся (уникальный идентификатор учащегося, имя учащегося и др.), его предпочтения (уровень громкости при воспроизведении аудиоматериалов, язык обучения, скорость демонстрации образовательного контента и др.), данные об интеракции (тип), данные о взаимодействии с образовательным объектом (был ли осуществлен доступ к LO в текущем взаимодействии и по какой причине взаимодействие было прекращено: таймаут, приостановка обучения, нормальное завершение изучения LO или завершение обучения вообще), статус LO (завершено ли изучение), суммарное время изучения LO, набранные баллы и др.

2. Модель данных

Модель данных для взаимодействия с образовательными объектами (*Data Model for Content Object Communication*) была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.11.1-2004 [5]. Соответствующая XML-схема описана в стандарте IEEE 1484.11.3-2005 [7].

Данная модель используется объектами LO для получения от LMS данных, необходимых для их работы, а также для обратной передачи данных из LO в LMS. Структура модели представлена на рис. 2. Модель включает в себя следующие основные элементы данных:

- *comments_from_learner*: комментарии учащегося;
- *comments_from_lms*: комментарии LMS;
- *completion_status*: статус завершения;
- *completion_threshold*: граница завершения;
- *credit*: зачет;
- *entry*: вход;
- *exit*: выход;
- *interactions*: интеракции;
- *learner_id*: идентификатор учащегося;
- *learner_name*: имя учащегося;
- *max_time_allowed*: максимальное доступное время;
- *progress_measure*: степень освоения материала;
- *score*: набранный балл;
- *time_limit_action*: действие при превышении лимита времени.

```
content_object_communication: record
(
   comments_from_learner: array(0..249) of comment_type,
   comments_from_lms: array(0..99) of comment_type,
   completion_status: completion_status_type,
   completion_threshold: real(10,7) range(0..1),
   credit: state(credit, no_credit),
   data_model_version: characterstring(iso-10646-1),
   entry: state(ab_initio, resume, _nil_),
   exit: state(timeout, suspend, logout, normal, _nil_),
   interactions: bag of interaction_type,
   launch_data: characterstring(iso-10646-1),
   learner_id: long_identifier_type,
   learner_name: localized_string_type(250),
   learner_preference_data: learner_preference_type,
   lesson_status: state(passed, completed, failed,
                    incomplete, browsed, not_attempted),
   location: characterstring(iso-10646-1),
   max_time_allowed: timeinterval(second,10,2),
   mode: state(browse, normal, review),
   objectives: set of objective_type,
   progress_measure: progress_measure_type,
   raw_passing_score: real(10,7),
   scaled_passing_score: real(10,7) range(-1..1),
   score: score_type,
   session_time: timeinterval(second,10,2),
   success_status: success_status_type,
   suspend_data: characterstring(iso-10646-1),
   time_limit_action: state(exit_message, continue_message,
                    exit_no_message, continue_no_message),
   total_time: timeinterval(second,10,2),
)
```

Рис. 2. Структура модели данных

Элемент *comments_from_learner (комментарии учащегося)* представляет собой массив, состоящий из 250 элементов типа comment_type. Его значениями являются комментарии, получаемые от учащегося. В этих комментариях могут содержаться отзывы, пожелания и замечания учащегося относительно изучаемого LO.

Элемент *comments_from_lms (комментарии LMS)* представляет собой массив, состоящий из 100 элементов типа *comment_type*. Его значениями являются комментарии, которые LMS намеревается передать учащемуся, осуществляющему взаимодействие с LO. В этих комментариях может содержаться аннотация изучаемого LO или отзывы других учащихся, работавших с этим LO ранее.

Элемент *completion_status (статус завершения)* показывает, завершил ли учащийся изучение данного LO, и может принимать следующие значения:

- *completed*: учащийся освоил LO;
- *incomplete*: учащийся не в достаточной мере освоил LO;
- *not_attempted*: учащийся не предпринимал попытки изучить данный LO, либо работал с LO столь малое время, что это может быть квалифицировано как «не предпринимал попытки»;
- *unknown*: элементу не присваивалось значение.

Значение элемента *completion_status* может устанавливаться образовательным объектом, определяться LMS путем сопоставления *степени прогресса* (см. элемент *progress_measure*) с *границей завершения* (см. элемент *completion_threshold*), задаваемой инструктором на основе поставленных целей, либо вычисляться каким-то иным образом.

Элемент c*ompletion_threshold (граница завершения)* содержит вещественное значение в диапазоне от 0 до 1, с которым LMS сравнивает степень прогресса учащегося в освоении LO, для определения того, может ли образовательный объект рассматриваться как завершенный. Элемент *completion_threshold* разработан для использования в сочетании со *степенью прогресса* (см. ниже элемент *progress_measure*). Например, если *completion_threshold* для данного LO составляет 0.85 и учащийся достиг значения *progress_measure* 0.90, то *completion_status* этого LO для данного учащегося может быть установлен в состояние *completed*.

Элемент *credit (кредит)* определяет, является ли изучение данного LO обязательным для получения зачета. Он может принимать следующие значения:

- *credit*: необходим для зачета;
- *no_credit*: не является необходимым для зачета.

Элемент *entry (вход)* показывает, был ли ранее учащимся осуществлен доступ к объекту LO. Он может принимать следующие значения:

- *ab_initio*: доступа к LO не было;
- *resume*: учащийся работал с LO ранее и при выходе учащегося элемент *exit* (см. ниже) получил значение *suspend*;
- *_nil_*: информация о предшествующем доступе отсутствует.

Элемент *exit (выход)* определяет, по какой причине взаимодействие с объектом LO было прекращено. Он может принимать следующие значения:

- *timeout*: завершение по причине превышения лимита времени, задаваемого элементом *max_time_allowed* (см. ниже);
- *suspend*: учащийся временно приостановил работу с LO с намерением к нему вернуться;

- *logout*: LO сигнализировал о намерении учащегося завершить работу с обучающей средой, частью которой является этот LO;
- *normal*: нормальное завершение работы с LO;
- *_nil_*: условия выхода не определены.

Элемент данных *interactions (интеракции)* содержит информацию, имеющую отношение к тестированию или оценке знаний учащегося. Экземпляр записи *interaction_type* (см. рис. 3) в обязательном порядке должен включать в себя идентификатор интеракции. Если экземпляр записи *interaction_type* включает в себя элементы *correct_responses (правильные ответы)* или *learner_response (ответ учащегося)*, то он также должен включать в себя *type (вид тестового задания)*. Все остальные компоненты являются необязательными.

```
interactions: bag of interaction_type,
type interaction_type = record
(
   id: long_identifier_type,
   type: state( true_false, multiple_choice, fill_in,
          long_fill_in, likert, matching, performance,
          sequencing, numeric, other ),
   objectives_id: array(0..9) of long_identifier_type,
   time_stamp: date_time_type,
   correct_responses: correct_responses_type,
   weighting: real(10,7),
   learner_response: learner_response_type,
   result: choice (state (result_state, numeric) )
          of (
                result_state: state( correct, incorrect,
                     unanticipated, neutral ),
                numeric: real(10,7),
          ),
   latency: timeinterval(second,10,2),
   description: localized_string_type(250),
)
```

Рис. 3. Структура интеракций

Элемент *type (тип)* показывает вид тестового задания и может принимать следующие значения:

- *true_false*: задание вида «да/нет»;
- *multiple_choice*: задание с множественным выбором;
- *fill_in*: задание с кратким ответом (одно-два слова);
- *long_fill_in*: задание с развернутым ответом (предложение или абзац);
- *likert*: задание со шкалой ответов вида «полностью не согласен», «скорее не согласен», «нейтрален», «скорее согласен», «полностью согласен»;
- *matching*: задание на установление соответствия между элементами двух множеств;
- *performance*: пошаговое решение задачи;
- *sequencing*: задание на упорядочение элементов множества;
- *numeric*: задание с числовым ответом;
- *other*: прочие типы.

Элемент *correct_ responses* описывает правильные ответы данной интеракции. Структура элемента *correct_ responses* изображена на рис. 4 и представляет собой описание правильных ответов заданного вида тестового задания.

```
correct_responses: correct_responses_type,
type correct_responses_type = choice
(
   state (true_false, multiple_choice, fill_in, long_fill_in,
   likert, matching, performance, sequencing, numeric, other ),
)
of
(
   true_false: state( true, false ),
   multiple_choice: set of set of short_identifier_type,
   fill_in: bag of record
      (
         case_matters: boolean,
         order_matters: boolean,
         match_text: array(0..9) of loclized_string_type(250),
       ),
   long_fill_in: bag of record
      (
         case_matters: boolean,
         match_text: localized_string_type(4000),
      ),
   likert: short_identifier_type,
   matching: bag of bag of record
      (
         source: short_identifier_type,
         target: short_identifier_type,
      ),
   performance: bag of record
      (
         order_matters: boolean,
         answers: array(0..124) of record
           (
              step_name: short_identifier_type,
              step_answer: choice (state( literal, numeric )) of
                 (
                    literal: characterstring(iso-10646-1),
                    numeric: record
                       (
                          min: real(10,7),
                          max: real(10,7),
                        ),
                  ),
            ),
      ),
   sequencing: bag of array(0..35) of short_identifier_type,
   numeric: record
      (
          min: real(10,7),
          max: real(10,7),
       ),
   other: characterstring(iso-1046-1),
)
```

Рис. 4. Структура правильного ответа

Элемент *lerner_response* описывает ответы учащегося. Структура элемента *lerner_response* подобна структуре элемента *correct_responses*, за исключением атрибута *numeric*, который задается точным значением.

Элемент *result* описывает результат интеракции и может принимать следующие значения:

- *correct*: верный ответ;
- *incorrect*: неверный ответ;
- *unanticipated*: ответ не требовался;
- *neutral*: ответ ни правильный, ни не правильный.

Элемент *progress_measure (степень прогресса)* содержит вещественное значение в диапазоне от 0 до 1, показывающее, в какой мере учащийся продвинулся в изучении LO.

Элемент *max_time_allowed (максимальное доступное время)* содержит максимально возможное количество секунд, задающих время в течении которого учащийся может работать с LO.

Элемент *time_limit_action (действие при превышении лимита времени)* указывает LO, что нужно сделать при превышении максимального времени доступа, элемент может принимать следующие значения:

- *exit_message*: учащийся информируется о превышении времени и доступ завершается;
- *continue_message*: учащийся информируется о превышении времени, но доступ не завершается;
- *continue_no_message*: учащийся не получает сообщение о превышении времени и продолжает работать;
- *exit_no_message*: завершение доступа без вывода каких-либо сообщений.

Применение стандартизированной модели данных для обмена информацией между LMS и LO позволяет использовать без модификаций существующие объекты LO в различных системах LMS, поддерживающих данную модель.

3. Модели накопления контента

Модели накопления контента описывают компоненты, используемые в образовательных системах, способы их обмена и описания для поиска и использования, а также способы упаковки контента.

3.1. Модель метаданных

Модель метаданных образовательных объектов LOM (*Learning Object Metadata*) предназначена для описания структуры и свойств образовательных объектов LO. Обычно это описание выполняется на языке XML. Модель LOM была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.12.1-2002 [8]. Соответствующая XML-схема описана в стандарте IEEE 1484.12.3-2005 [9].

Основной целью LOM является обеспечение повторного использования LO, поддержка открытости и интероперабельности образовательных объектов в контексте LMS. По существу LOM описывает структуру и семантику атрибутов, которые должны быть определены при описании LO. В их число входят следующие: тип LO, автор, владелец, условия использования, формат, педагогические атрибуты и др.

Схема модели LOM изображена на рис. 5. Схема LOM представляет собой иерархию атрибутов, с помощью которых формируется мета-описание LO. Все атрибуты первого уровня делятся на девять категорий:

- *General (общая)* включает в себя атрибуты, описывающие общие свойства LO;
- *Lifecycle (жизненный цикл)* включает в себя атрибуты, описывающие текущее состояние LO и историю его изменений;
- *Meta-Metadata (мета-метаданные)* включает в себя атрибуты, относящиеся к самому экземпляру метаданных, но не к образовательному объекту, который он описывает;
- *Technical (техническая)* включает в себя атрибуты, описывающие технические требования и технические характеристики LO;
- *Educational (образовательная)* включает в себя атрибуты, определяющие образовательные и педагогические характеристики LO;
- *Rights (права)* включает в себя атрибуты, определяющие права на интеллектуальную собственность и условия использования LO;
- *Relation (связь)* включает в себя атрибуты, описывающие взаимосвязи образовательного объекта с другими образовательными объектами;
- *Annotation (аннотация)* включает в себя атрибуты, содержащие комментарии о том, каким образом следует использовать образовательный объект в процессе обучения, и информацию о том, кем и когда эти комментарии были созданы;
- *Classification (классификация)* включает в себя атрибуты, описывающие положение LO в определенной классификационной системе.

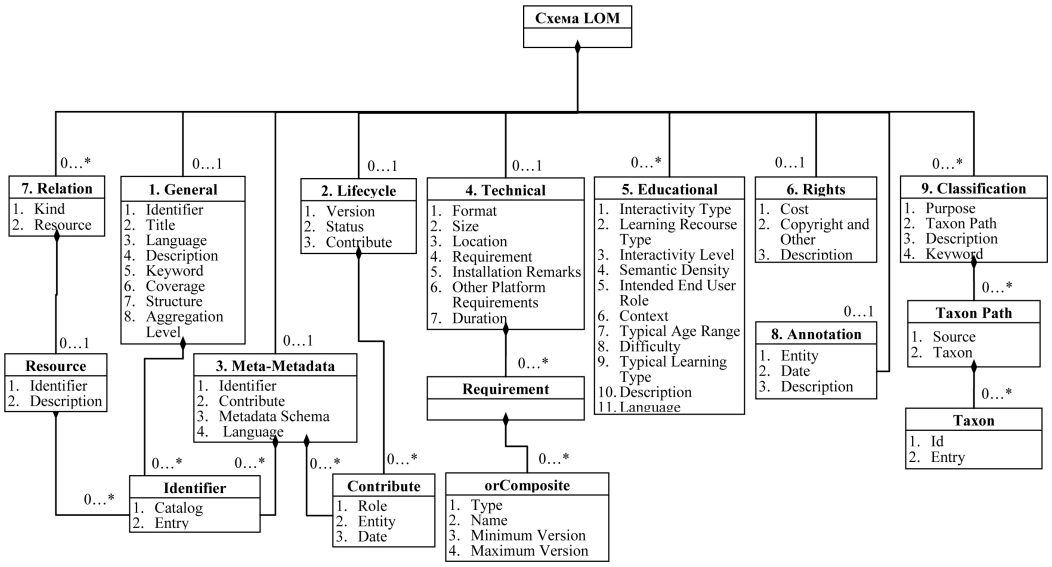

Рис. 5. Схема модели LOM

Атрибуты первого уровня могут включать в себя под-атрибуты второго уровня, которые, в свою очередь, могут включать в себя под-атрибуты третьего уровня. Семантика каждого атрибута обусловлена его положением в иерархической схеме, описывающей LO.

LOM также определяет тип данных и множество значений для каждого простого атрибута. Для многих атрибутов в качестве значения может быть введена произвольная строка символов UNICODE. Для других атрибутов вводимые значения должны выбираться из предопределенного списка или иметь специальный формат. Следующие элементы данных предусматривают специальный формат значений:

- элементы *LangString* содержат части *Language* and *String*, позволяя записывать одну и ту же информацию на разных языках;
- элементы *Vocabulary* ограничивают множество вводимых значений предопределенным списком терминов и представляют собой пару источник, значение. Источник задает определенный словарь (список терминов), а значение указывает на слово в этом словаре;
- элементы *DateTime* и *Duration* состоят из части, задающей дату или время в «машинном» формате, и части, определяющей семантику значения (например: {продолжительность в минутах, 190}).

В современном электронном обучении модель LOM является базовой для обеспечения мобильности и интероперабельности образовательных объектов.

3.2. Модель структуры контента

Модель структуры контента определяет структуру электронного учебного курса в соответствии с требованием интероперабельности. Эта модель была разработана отраслевой стандартизирующей организацией AICC (Aviation Industry Computer-based training Committee) и описана в документах CRS003 [10] и CMI001 [11]. Главная цель, которую преследовали разработчики, состояла в том, чтобы различные LMS могли бы использовать широкую гамму поставляемых электронных курсов. Дополнительным требованием была возможность сохранять структуру курса, разработанного в одной LMS, при его экспорте в другую LMS.

Схематично модель структуры контента изображена на рис. 6. В модели вводятся несколько уровней иерархии, имеющие определенные имена. Каждый уровень представляет собой определенные части учебного курса и занимает строго фиксированную позицию в иерархии. Любой элемент нижнего уровня должен являться структурной частью некоторого элемента вышестоящего уровня и не может, таким образом, превосходить его по объему. Модель наделяет уровни следующей семантикой.

Иерархия	Уровень	Структура курса
Программа (Curriculum)	1.	
Курс (Course)	2.	Блок
Глава (Chapter)	3.	Блок
Раздел (Subchapter)	4.	Блок
Модуль (Module)	5.	
Урок (Lesson)	6.	Назначаемый элемент (Assignable Unit)
Понятие (Topic)	7.	
Кадр (Frame)	8.	
Объект (Object)	9.	

Рис. 6. Модель структуры контента

Программа (Curriculum) — совокупность учебных курсов, составляющих образовательную программу.

Курс (Course) — совокупность учебного материала, обладающая дидактической целостностью, и позволяющая учащемуся освоить знания, необходимые для приобретения определенных умений и навыков.

Глава (Chapter) — значимая часть курса. Объединяет в себе несколько разделов или уроков.

Раздел (Subchapter) — значимая часть главы. Объединяет в себе несколько модулей или уроков.

Модуль (Module) — значимая часть курса, главы или раздела. Объединяет в себе несколько уроков.

Урок (Lesson) — значимая часть учебного материала, которая осваивается учащимся за одно занятие продолжительностью от 20 минут до одного часа.

Понятие (Topic) — логически самостоятельная часть урока.

Кадр (Frame) — значимый элемент графического интерфейса (окно), появляющийся на экране в определенный момент в течении урока.

Объект (Object) — компонент кадра. Объекты могут быть графическими, текстовыми или управляющими.

Электронный учебный курс имеет иерархическую структуру, включающую в себя элементы следующих трех типов:

- *Курс (Course)*;
- *Блок (Block)*;
- *Назначаемый элемент (Assignable Unit)*.

Курс представляет собой наибольшую часть учебного материала, которая может быть передана из одной системы управления обучением в другую.

Назначаемый элемент — наименьшая компонента курса, которую LMS может назначать и контролировать в процессе обучения. Обычно такой компонентой является урок.

Блок представляет собой группу назначаемых элементов и/или других блоков, логически связанных между собой. На различных уровнях иерархии блокам могут соответствовать модули, разделы или главы.

Подобная организация учебного материала обеспечивает большую гибкость при создании электронного учебного курса, поскольку позволяет описывать блоки внутри других блоков и задавать, таким образом, практически неограниченное количество уровней детализации материала.

3.3. Модель упаковки контента

Модель упаковки контента (Content Packaging Information Model) определяет стандартный набор структур данных, которые могут быть использованы для обмена образовательным контентом между различными LMS. Модель упаковки контента была разработана консорциумом IMS Global Learning Consortium и описана в спецификациях IMS [12, 13].

Общая структура модели упаковки контента изображена на рис. 7. Модель включает следующие основные структурные компоненты:

- *Логический пакет (Logical package)* представляет собой один или более *юнитов (units)* многократно используемого учебного материала. Логический пакет охватывает все множество компонент, описываемых манифестом, включая локальные компоненты и удаленные компоненты, доступные по ссылкам.
- *Пакет обмена (Interchange package)* представляет собой совокупность компонент, подлежащих обмену между различными LMS, включая манифест и другие выбранные файлы.

- *Манифест (Manifest)* представляет собой xml-документ, описывающий полный экземпляр логического пакета. Манифест может содержать ссылки на локальные или удаленные компоненты. Манифест включает в себя следующие секции:
  - *Метаданные (Metadata)* представляют собой описательную информацию о пакетах контента, логических организациях, контенте или файлах (см. п. 4.1);
  - *Организации (Organizations)* описывают логические взаимосвязи между юнитами в виде деревьев активностей (см. п. 6);
  - *Ресурсы (Resources)* представляют собой опись контента и файлов, используемых в корневом (parent) манифесте;
  - *Под-манифесты (Child-manifests)* представляют собой полноценные подчиненные манифесты, которые присутствуют или на которые имеется ссылка в корневом манифесте. Каждый под-манифест описывает полноценный логический пакет, являющийся частью объемлющего логического пакета. Под-манифест может являться корневым для под-манифеста еще более низкого уровня. Иерархия манифестов отражает иерархическую организацию образовательного контента.
- *Файлы (Files)* воплощают в себе образовательный контент, описываемый логическим пакетом.

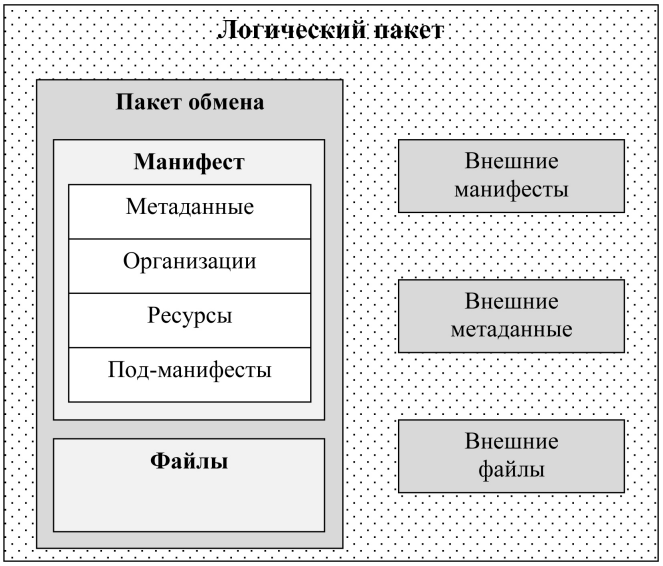

Рис. 7. Структура модели упаковки контента

Любой из описанных компонентов пакета обмена может быть внешним по отношению к нему. В этом случае пакет обмена содержит только ссылку на удаленный компонент. Пакет обмена вместе с удаленными компонентами и образует логический пакет. Детальное описание компонентов и соответствующие xml-схемы можно найти в [13].

4. Модель среды выполнения

Модель среды выполнения (Runtime Environment Model) определяет структуру *прикладного программного интерфейса (Application Program Interface)* или кратко *API*, используемого для организации взаимодействия между образовательным объектом (LO) и сервисом времени выполнения (RTS). Модель среды выполнения была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.11.2-2003 [6]. Основной целью, преследу-

емой при разработке этой модели, было обеспечение интероперабельности образовательных объектов в различных средах выполнения.

Схематично модель среды выполнения изображена на рис. 8. Общепринятой реализацией этой модели является программная среда, ориентированная на доставку контента посредством web-обозревателя, в которой все коммуникации инициируются образовательным объектом LO. Такая коммуникационная модель не предусматривает возможности коммуникаций между RTS и LO, инициированных со стороны RTS. Интероперабельность достигается путем посылки запросов через общие коммуникационные методы, реализованные в API. Организация коммуникаций и интерфейс методов являются одинаковыми для каждого экземпляра реализации API.

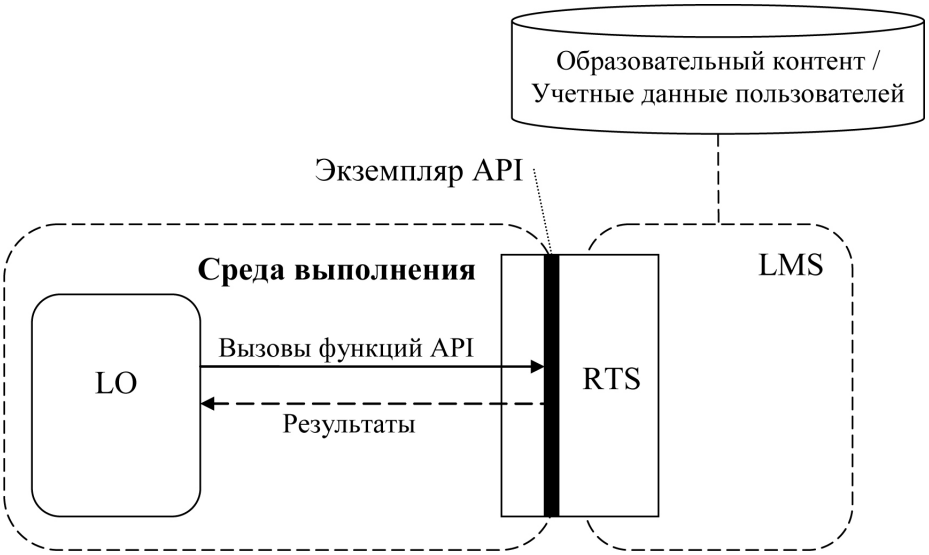

Рис. 8. Модель среды выполнения

Как показано на рис. 8, RTS реализует API в среде выполнения LO. При разработке LO разработчик включает в его реализацию средства обнаружения и коммуникации с экземпляром API. LMS или ее клиент, обеспечивающие доступ к репозиторию с образовательным контентом (локальному или удаленному), предоставляет для учащегося RTS. RTS либо доставляет учащемуся LO и осуществляет его запуск, либо загружает URI (Uniform Resource Identifier) для активизации LO посредством web-обозревателя. Сразу после своего запуска LO осуществляет поиск экземпляра API. Как только экземпляр API найден, LO инициирует сеанс связи с RTS. В ходе сеанса связи образовательный объект может запрашивать необходимые ему данные через экземпляр API. Через этот же экземпляр API RTS возвращает затребованные данные или сообщение об ошибке. В ходе сеанса связи LO может пересылать или устанавливать элементы модели данных (см. п. 3) для сохранения в репозитории. RTS может использовать элементы данных или другую сохраненную информацию в отчетах о статусе учащегося по отношению к данному образовательному объекту. В свою очередь, LO может получить детализированное сообщение об ошибке.

Образовательный объект может продолжать подобные коммуникации, запрашивая и передавая данные до момента, когда учащийся завершит изучение LO, либо учащийся прервет сеанс связи до окончания изучения LO, либо сеанс связи будет завершен аварийно (отказ системы, потеря питания или др.). В первых двух случаях LO явно сообщает экземпляру API о закрытии сеанса связи. В последнем случае, RTS не получает сигнала через экземпляр API о закрытии сеанса связи. RTS необходимо самостоятельно обработать ис-

ключительную ситуацию в соответствии со спецификациями экземпляра API. На рис. 9 представлена общая структура API.

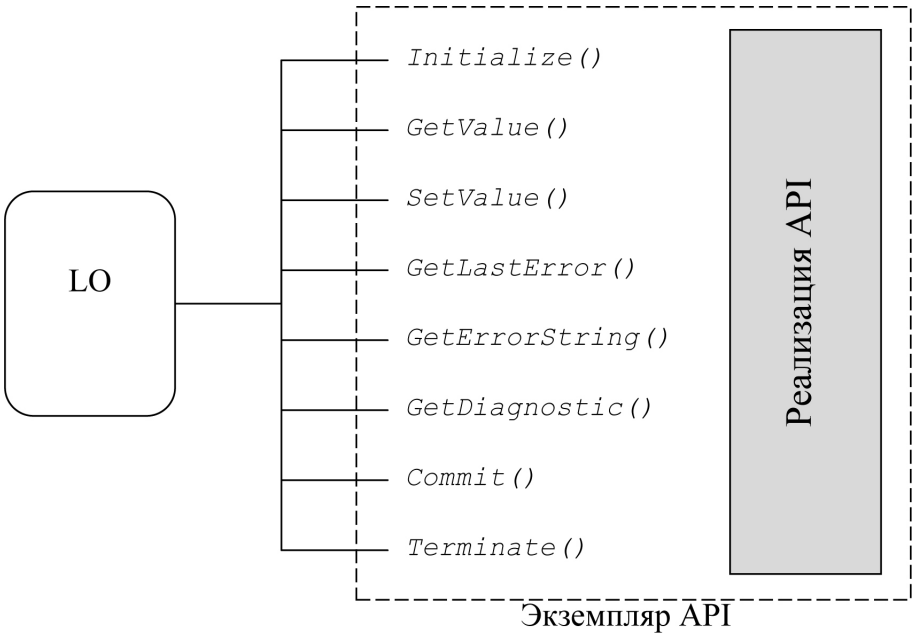

Рис. 9. Структура API

5. Модель простого упорядочения

Модель простого упорядочения (Simple Sequencing Information and Behavior Model) разработана консорциумом IMS Global Learning Consortium и описана в спецификациях IMS [14–16]. Данная модель определяет метод, с помощью которого, исходя из имеющегося преподавательского опыта, можно задать план изучения образовательного контента, на основе которого LMS упорядочивает отдельные *обучающие активности (learning activity)*, непротиворечивым образом. Разработчик образовательного контента задает относительный порядок предоставления учащемуся образовательных объектов и условия, при которых части образовательного контента выбираются, доставляются или пропускаются в процессе обучения.

Модель простого упорядочения определяет поведение и функциональность, которые должны поддерживаться стандартной LMS. Она включает правила, формирующие поток активностей в контексте определенного образовательного контента, в зависимости от результатов взаимодействия учащегося с этим контентом. Такое представление планируемого инструктивного потока может быть создано вручную или с помощью авторинговых систем, генерирующих результат, в соответствии со спецификациями данной модели. Созданное представление упорядочивания может использоваться различными системами, разработанными для доставки инструктивных активностей учащемуся.

Простое упорядочение обозначено как *простое* потому, что оно включает ограниченный набор наиболее распространенных стратегий упорядочивания, а не потому, что спецификации этой модели являются простыми. Простое упорядочение не охватывает все возможные аспекты, связанные с проблемой упорядочивания учебного материала в LMS. В частности, простое упорядочение не включает в себя, но и не запрещает использовать: упорядочивание с использованием искусственного интеллекта; упорядочивание на основе плана заня-

тий; упорядочение, требующее данные от внешних закрытых систем и сервисов (например, упорядочивание на базе встроенной имитационной модели); коллаборативное (совместное) обучение; индивидуальное обучение; синхронизацию между параллельными обучающими активностями и др.

Модель простого упорядочения опирается на понятие обучающей активности. *Обучающая активность (learning activity)* — это отдельное задание (instructional event), являющееся объектом упорядочивания. Процесс взаимодействия учащегося с активностью делится на последовательные во времени интервалы, называемые *попытками*. Попытка считается *успешной*, если учащийся выполнил все шаги задания, представленного в данной активности. Учащийся может завершить попытку досрочно, не выполнив всех шагов задания, и начать следующую попытку. Выполнение попытки может прерываться на некоторое время, в течение которого активность будет находиться в *приостановленном* состоянии. Впоследствии учащийся может возобновить прерванную попытку. Таким образом, одна попытка может распространяться на несколько сеансов работы учащегося с LMS. При анализе результатов взаимодействия учащегося с активностью, система учитывает количество попыток, их успешность, продолжительность взаимодействия с активностью и абсолютную продолжительность активности (см. рис. 10).

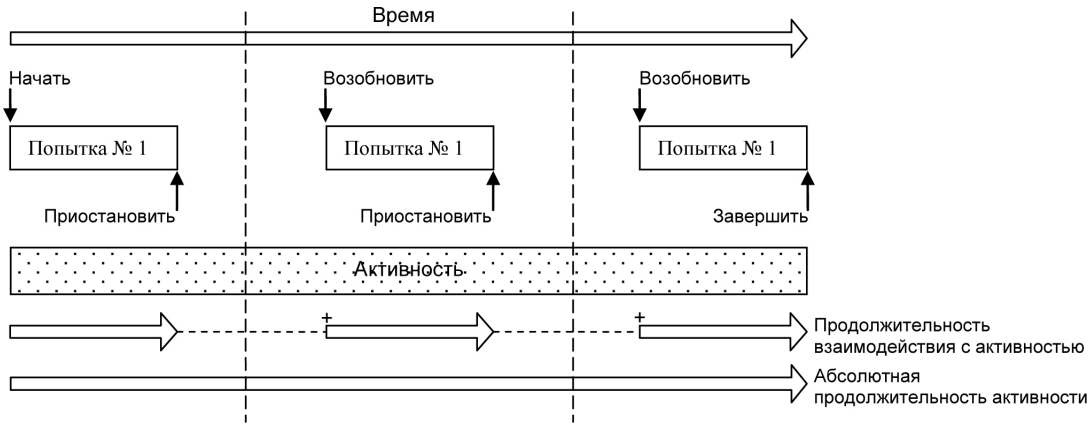

Рис. 10. Временная диаграмма взаимодействия с активностью

Отдельные активности могут группироваться для формирования активности более высокого уровня и включать в себя несколько уровней подчиненных активностей, образуя *дерево активностей (activity tree)*. Пример дерева активностей приведен на рис. 11. Здесь, например, активность AA является частью активности A, а активность AABA является частью активности AAB, которая является частью активности AA.

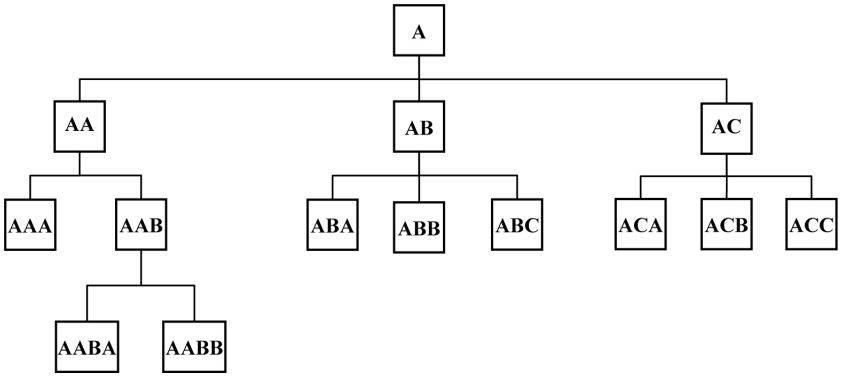

Рис. 11. Дерево активностей

Различным уровням иерархии в дереве активностей могут соответствовать различные концептуальные метки. Например, в рамках одного учебного курса активности A соответствует курс в целом, AA является уроком, AAB — этапом урока, а AABB — шагом этапа. С листовыми активностями связываются образовательные ресурсы, представляющие собой контент, который должен быть доставлен учащемуся.

В качестве канонического порядка обхода дерева активностей модель простого упорядочения предполагает обход дерева в прямом порядке (pre-order traversal). В соответствии с указанным порядком сначала нужно посетить корень, а затем слева-направо рекурсивно совершить обход всех поддеревьев. Для примера на рис. 11 порядок обхода узлов в этом случае будет следующим: A, AA, AAA, AAB, AABA, AABB, AB, ABA, ABB, ABC, AC, ACA, ACB, ACC. Маршрут обхода дерева по умолчанию может быть изменен с помощью *правил упорядочивания (sequencing rules)*, созданных разработчиком образовательного контента. Обход дерева начинается при поступлении запроса упорядочивания, который, в свою очередь, инициируется учащимся с помощью навигационных событий, либо системой доставки контента самостоятельно. Правила упорядочивания вычисляются во время выполнения и могут зависеть от результатов предшествующей работы учащегося. В каждый момент времени учащемуся всегда доставляется только одна активность, с которой может быть связано произвольное количество образовательных ресурсов.

Правила упорядочивания ассоциируются с кластерами узлов в дереве активностей. Под *кластером* в модели простого упорядочения понимается выделенный узел и все его сыновья (см. рис. 12). Действия правила упорядочивания никогда не выходят за пределы кластера. По умолчанию, учащийся может выбрать любую дочернюю активность в кластере, однако, мы также можем задать режим *рекомендуемого маршрута (guided flow)*, при котором учащийся будет совершать обход кластера в прямом порядке, как это показано на рис. 13. Правило, которое задает такую возможность, связывается с родительским узлом кластера. Если дочерние активности в кластере являются листьями, с которыми ассоциированы образовательные ресурсы, они будут доставляться учащемуся последовательно в указанном порядке.

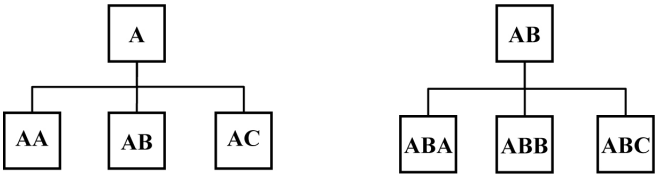

Рис. 12. Примеры кластеров

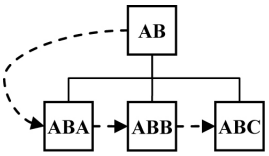

Рис. 13. Обход кластера в прямом порядке

Основными управляющими режимами, определяющими порядок обхода кластера, являются *выбор (choice)* и *маршрут (flow)*, каждый из которых может быть активирован или деактивирован. Если режим выбор деактивирован, то учащийся не может выбирать произвольный порядок обхода кластера, но должен следовать рекомендуемому маршруту.

Если режим маршрут деактивирован, то учащийся должен сам определять порядок обхода кластера. Оба режима могут быть активированы одновременно, в этом случае учащийся может выбирать активности произвольно или предпочесть рекомендуемый маршрут. Однако оба режима не могут быть одновременно деактивированы. Для всех кластеров в дереве активностей могут быть специфицированы одинаковые управляющие режимы, либо они могут отличаться у разных кластеров.

Более сложное навигационное поведение может быть организовано с помощью *правил упорядочивания (sequencing rules)*, назначаемых различным узлам дерева активностей на различных уровнях иерархии. Каждое такое правило задает условия, при которых ассоциированный с ним узел должен быть пропущен. На рис. 14 приведен пример обхода дерева активностей при активированном режиме маршрут. Штриховкой помечены узлы, для которых «сработало» правило «пропустить узел».

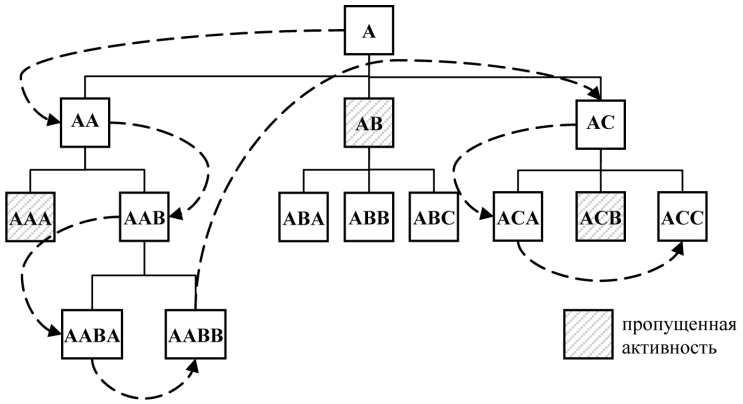

Рис. 14. Пример обхода дерева активностей с пропуском узлов

Многие правила упорядочивания, определенные в модели простого упорядочения, базируются на данных, полученных путем отслеживания результатов взаимодействия учащегося с активностями. С каждой активностью в дереве активностей связывается группа *атрибутов мониторинга*, которая делится на две подгруппы: 1) *атрибуты цели*, содержащие информацию о степени достижения цели обучения; 2) *атрибуты взаимодействия*, содержащие количественную информацию о действиях учащегося по отношению к данной активности.

Примерами атрибутов цели являются: *статус достижения цели (objective satisfied status)* — логический атрибут, показывающий, достигнута ли дидактическая цель, предусмотренная для данной активности, или нет; *нормализованная мера достижения цели (objective normalized measure)* — вещественный атрибут, принимающий значения из отрезка [-1, 1], который показывает в какой мере учащийся приблизился к достижению цели, ассоциированной с данной активностью.

Примерами атрибутов взаимодействия являются: *продолжительность взаимодействия с активностью (activity experienced duration)* — атрибут, показывающий совокупное время взаимодействия учащегося с активностью без учета времени, когда активность находилась в приостановленном состоянии (см. рис. 10); *абсолютная продолжительность активности (activity absolute duration)* — атрибут, показывающий совокупное астрономическое время взаимодействия учащегося с активностью с учетом времени, когда активность находилась в приостановленном состоянии (см. рис. 10); *количество попыток (activity attempt count)* — атрибут, показывающий количество попыток выполнения активности (нулевое значение атрибута означает, что активность еще ни разу не была запущена учащимся,

положительное значение означает, что активность находится в процессе выполнения, либо завершена учащимся); *статус завершения попытки (attempt completion status)* — логический атрибут, показывающий, была ли завершена попытка выполнения активности; *мера завершения попытки (attempt completion amount)* — вещественный атрибут, принимающий значения из отрезка [0, 1], который показывает, как далеко учащийся находится от завершения попытки.

Атрибуты мониторинга родительского узла могут вычисляться на основе анализа значений атрибутов мониторинга дочерних активностей. Это достигается путем использования *правил мониторинга (rollup rulles)*. На рис. 15 приведен пример правила мониторинга, которое вычисляет значение атрибута *статус достижения цели* для активности AB на основе значений атрибутов *статус достижения* цели дочерних активностей кластера. При этом некоторые дочерние активности кластера могут иметь пометку «не учитывать», что позволяет не учитывать значения атрибутов мониторинга данных активностей при вычислении правил мониторинга.

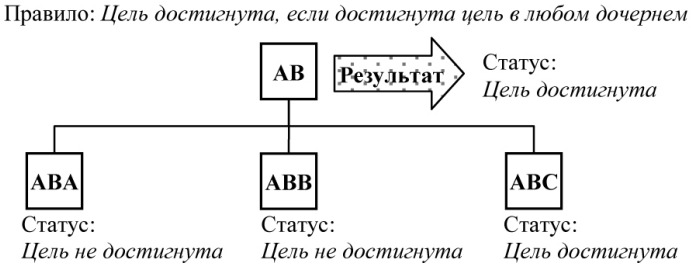

Рис. 15. Пример правила мониторинга для расчета статуса активности

Все LMS, реализующие модель простого упорядочения, выполняют следующую последовательность действий, называемую *процессом простого упорядочивания (simple sequencing process)*.

Начало сессии упорядочивания

1. Учащийся инициирует сеанс работы с LMS и устанавливает контекст, выбирая конкретный модуль электронного учебного курса.
2. LMS инициирует *сессию упорядочивания* с помощью навигационного запроса *«Start (Начать)», «Resume All (Возобновить все)»* или *«Choice (Выбрать)»*.
3. *Блок навигации (navigation behavior)* транслирует навигационный запрос «Start», «Resume All» или «Choice» в соответствующий запрос упорядочивания и выполняет его. Сессия упорядочивания «официально» начинается, когда найдена активность для предоставления учащемуся (успешно завершены шаги 4 и 5).

Цикл упорядочивания

4. Используя значения атрибутов мониторинга, *блок упорядочивания (sequencing behavior)* выполняет обход дерева активностей и определяет, какая активность (узел в дереве активностей) должна быть доставлена учащемуся на данном шаге.
5. *Блок доставки (delivery behavior)* определяет, могут ли быть доставлены учащемуся образовательные ресурсы, ассоциированные с выбранной активностью: активность должна быть листовой; предыдущая попытка взаимодействия с активностью должна быть завершена; активность не должна быть *заблокированной (disabled)*, и, в случае положительного ответа, подготавливает для доставки необходимые образовательные ресурсы.

Если выбранная активность не может быть предоставлена учащемуся, тогда общий процесс упорядочивания останавливается и ожидает следующего навигационного запроса (шаг 8).

6. Процесс упорядочивания простаивает, ожидая навигационных запросов пока учащийся взаимодействует с образовательным ресурсом.
7. Во время взаимодействия учащегося с образовательным ресурсом активность может передавать значения, которые используются для обновления значений атрибутов мониторинга.
8. Учащийся, LMS или активность инициирует навигационное событие, такое как *«Continue (продолжить)»* , *«Previous (предыдущий)»*, *«Choose activity X (выбрать активность X)»*, *«Abandon (отказаться)»*, *«Exit (выйти)»* и др.
9. LMS информирует процесс упорядочивания о произошедшем навигационном событии путем формирования навигационного запроса.
10. Если навигационный запрос показывает, что учащийся хочет завершить сессию упорядочивания блок навигации транслирует навигационный запрос в *запрос завершения (termination request)*, и сессия упорядочивания завершается. В противном случае блок навигации транслирует навигационный запрос в запрос упорядочивания.
11. Если навигационный запрос порожден в результате нормального или досрочного завершения активности, активность может обновить значения определенных атрибутов мониторинга, после чего активность завершает свою работу. Далее LMS активизирует *блок учета (rollup behavior)* для анализа всех изменений, произошедших в результате взаимодействия учащегося с активностью. Блок учета обновляет значения атрибутов мониторинга данной активности и всех ее предков в дереве активностей.
12. Процесс повторяется, начиная с шага 4, до тех пор, пока сессия упорядочивания не завершится.

Схематично процесс простого упорядочивания изображен на рис. 16.

6. Стандарт SCORM

Стандарт *SCORM (Sharable Content Object Reference Model)* [17–20] объединяет в себе модели, которые были описаны в разделах 3-6. SCORM описывает организацию и структуру образовательного контента и систему управления обучением, позволяет обеспечить совместимость образовательных объектов (LO) и возможность их многократного использования. Образовательные объекты SCORM могут включаться в разные учебные курсы и использоваться разными LMS, поддерживающими стандарт SCORM, независимо от того, кем, где и с помощью каких средств они были созданы.

На рис. 17 изображены компоненты образовательного контента SCORM от самых маленьких (assets) до самых крупных (учебных планов).

Образовательные объекты в SCORM могут быть двух типов: *asset (образовательный элемент)* и *разделяемый объект контента SCO (Shareable content object)*.

Объект *asset* — это электронное представление аудиовизуальных средств, используемых при обучении. В качестве объекта asset могут фигурировать текст, картинка, веб-страница, аудио- или видеоматериал и др. Объекты asset не могут взаимодействовать с LMS напрямую. Объекты asset могут быть многократно использованы как в различных контекстах одного электронного учебного курса, так и в разных электронных учебных курсах.

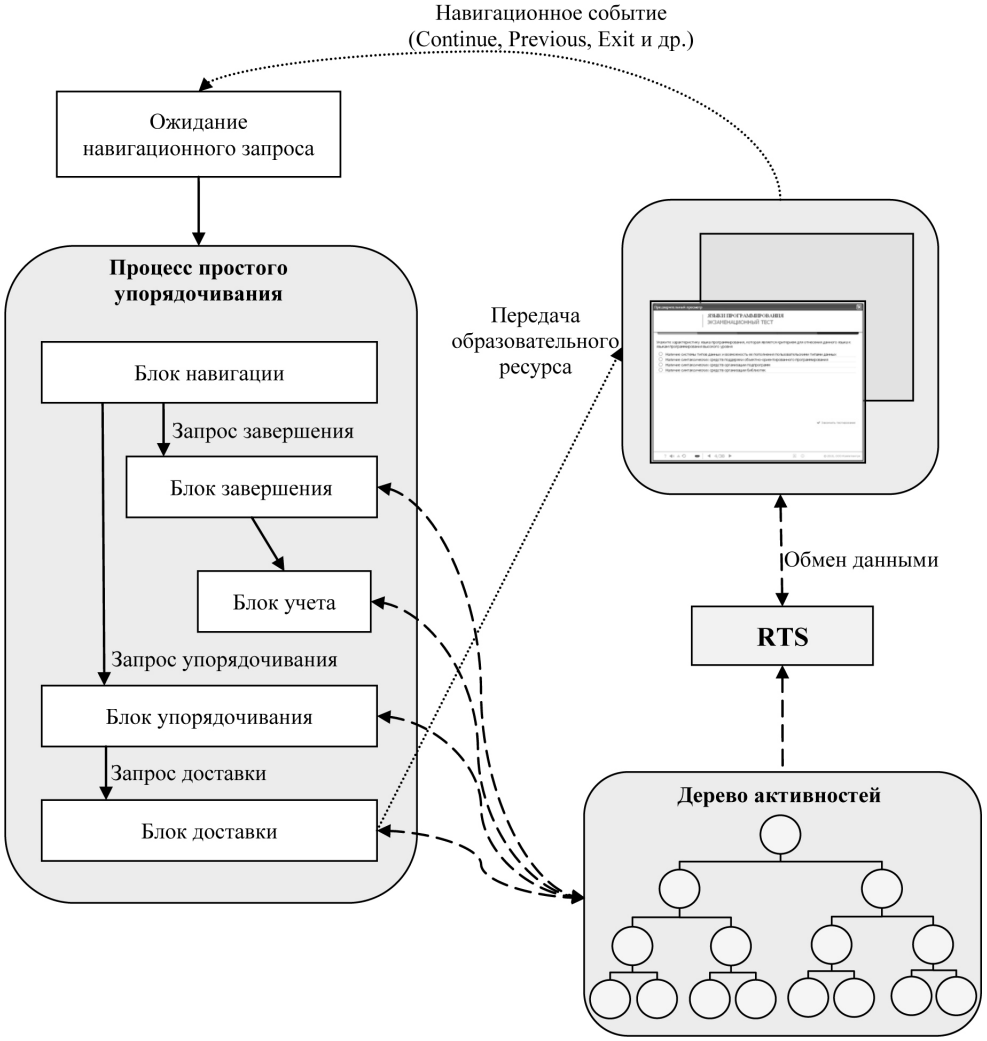

Рис. 16. Цикл упорядочивания

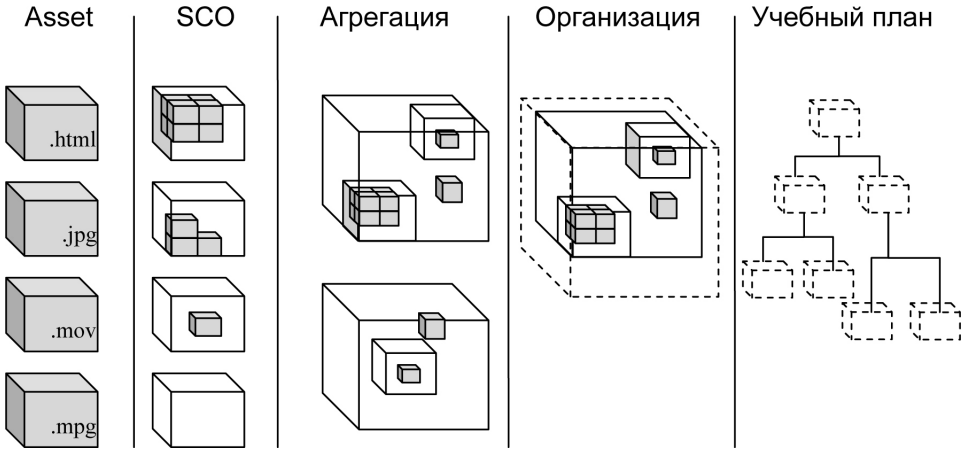

Рис. 17. Компоненты образовательного контента в SCORM

Разделяемый объект контента SCO представляет собой коллекцию из одного или более объектов asset и является наименьшей логической единицей обучения, которая может быть доставлена учащемуся с помощью LMS. С точки зрения реализации SCO является веб-приложением, которое взаимодействует с LMS посредством стандартного API (см. п. 5). SCO может получать от LMS и сохранять в LMS значения параметров мониторинга

(см. п. 6). Например, он может сохранить в LMS, такие значения, как *оценка* или *статус завершения*, или получить от LMS, такие значения, как *имя учащегося*.

Агрегация (aggregation) — это коллекция связанных обучающих активностей. Агрегация может включать в себя объекты SCO и вложенные агрегации. Агрегация соответствует кластеру в дереве активностей, описываемому в модели простого упорядочения (см. п. 6). С точки зрения реализации агрегация — это структура, описываемая в *манифесте SCORM*, где правила упорядочивания применяются к коллекции логически связанных объектов SCO или вложенных агрегаций. Структура манифеста SCORM основана на подходе, описанном в разделе 4.3. На рис. 18 изображен пример агрегации «Методы сортировки», которая включает в себя шесть вложенный объектов SCO. Каждый из вложенных объектов SCO также может представлять собой агрегацию.

Рис. 18. Пример агрегации

Организация (organization) — это раздел пакета контента, описываемого моделью упаковки контента (см. п. 4.3), в котором определяются логические связи между образовательными объектами для формирования древовидной структуры. Организация вцелом описывает структуру образовательного контента, которую автор намерен поставлять в виде единого пакета контента. Каждая организация является высокоуровневой агрегацией и рассматривается как корень агрегации. Изначально модель упаковки контента предусматривала возможность включать в один пакет несколько организаций, однако в действующей версии стандарта SCORM эта возможность не поддерживается, и в пакет можно включать только одну организацию. Организация контента является аналогом дерева активностей, описываемого в разделе 6.

Учебный план (curriculum) представляет собой древовидную структуру, узлами которой могут являться независимые компоненты. В качестве таких компонент могут фигурировать курсы, уроки, тесты и др. Для их представления могут использоваться организации SCORM. Описание структуры учебного плана выходит за рамки стандарта SCORM.

В SCORM весь образовательный контент в конце концов представляется в виде пакета контента. Пакет контента может включать единственный объект SCO или состоять из сотен таких объектов. Это зависит от разработчика образовательного контента и цели его создания. Пакет контента SCORM представляет собой zip-файл, который включает в себя все необходимое для доставки образовательного контента учащемуся, а именно манифест SCORM и файлы, в которых хранятся образовательные объекты SCO и asset, описываемые в файле манифеста. *Манифест SCORM* представляет собой xml-файл, содержащий описания всех объектов SCO и asset, которые включены в пакет контента, описания логических взаимосвязей между образовательными объектами, описания правил упорядочивания и метаданных. Как уже указывалось выше, структура манифеста SCORM основана на подходе, описанном в разделе 4.3.

Метаданные — это информация, которая описывает, чем является образовательный контент. Метаданные описывают отдельные образовательные объекты (asset и SCO) и па-

кеты контента. Метаданные позволяют разработчикам курсов осуществлять поиск контента или образовательных объектов и определять, будут ли эти объекты полезны, прежде чем они будут загружены или разработчик курса запросит права доступа. Метаданные SCORM основаны на модели метаданных, описанной в разделе 4.1.

Стандарт SCORM позволяет разработчику курса задавать порядок изучения образовательных объектов. Данный механизм может предоставлять учащемуся некоторую свободу при выборе образовательного объекта для изучения либо может жестко ограничивать этот порядок. Подробнее данный механизм описан в разделе 6.

Распространенным шаблоном проектирования электронных учебных курсов является наличие обязательного условия для некоторой обучающей активности. Когда доступность одной обучающей активности зависит от некоторых внешних условий, таких как степень завершения набора активностей или степень достижения цели набора активностей. Например, на рис. 19 показана блок-схема с серией уроков и последующего контрольного теста. Тест является заблокированным для учащегося до тех пор, пока Урок 1 и Урок 2 не будут им завершены.

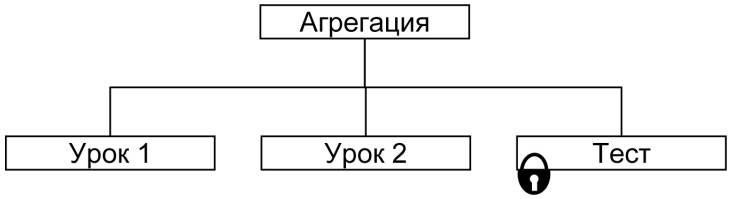

Рис. 19. Схема обучения из двух лекций, за которыми следует контрольный тест

Для реализации такого шаблона необходимо создать агрегацию уроков (кластер), добавить правило мониторинга, задать цели и создать предусловие для прохождения теста (см. раздел 6). Пример реализации кластера представлен на рис. 20. Кластер включает в себя

```
<item identifier="CLUSTER-ONE" isvisible="true" >
  <title>Lesson Aggregation</title>
  <item identifier="LESSON1" identifierref="RES-SCO1"
                    isvisible="true">
    <title>Child 1</title>
  </item>
  <item identifier="LESSON2" identifierref="RES-SCO2"
                    isvisible="true">
    <title>Child 2</title>
  </item>
  <item identifier="ASSESSMENT" identifierref="RES-SCO3"
                    isvisible="true">
    <title>Child 3</title>
    <!-- Pre-Condition Rule on Assessment (реализация представлена
на рис. 23)-->
  </item>
  <imsss:sequencing>
    <imsss:controlMode flow="true" choice="true"/>
    <!--Rollups: (реализация представлена на рис. 22)
             1)If any are incomplete, the cluster is incomplete
             2)If all are completed, the cluster is complete-->
    <!-- Map Aggregation's Primary Objective (реализация
представлена на рис. 21)>
```

Рис. 20. Пример реализации кластера

три дочерних элемента (у третьего элемента задано предусловие), описание правил мониторинга и описание целей. Правила мониторинга описывают следующие условия: «если хотя бы один из уроков не пройден, тогда кластер не завершен» и «если все уроки пройдены,

тогда кластер завершен». Примеры реализации цели, правил мониторинга и предусловия представлены на рис. 21, рис. 22 и рис. 23, соответственно.

```
<!-- Map Aggregation's Primary Objective >
<adlseq:objectives>
  <adlseq:objective objectiveID = "obj-primary">
    <adlseq:mapInfo targetObjectiveID = "obj-global-lessons"
                writeCompletionStatus="true" />
  </adlseq:objective>
</adlseq:objectives>
```

Рис. 21. Пример реализации целей

```
<!--Rollups: 1)If any are incomplete, the cluster is incomplete
             2)If all are completed, the cluster is complete-->
<imsss:rollupRules rollupObjectiveSatisfied="true"
                   rollupProgressCompletion="true">
  <imsss:rollupRule childActivitySet="any">
    <imsss:rollupConditions conditionCombination="any">
      <imsss:rollupCondition operator="not" condition="completed"/>
    <imsss:rollupConditions>
      <imsss:rollupAction action="incomplete"/>
  </imsss:rollupRule>
  <imsss:rollupRule childActivitySet="all">
    <imsss:rollupConditions conditionCombination="any">
      <imsss:rollupCondition condition="completed"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="complete"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Рис. 22. Пример реализации правил мониторинга

```
<!-- Pre-Condition Rule on Assessment -->
<imsss:sequencing>
 <imsss:sequencingRules>
  <imsss:preConditionRule>
    <imsss:ruleConditions conditionCombination="any">
      <imsss:ruleCondition operator="not" condition="completed"
           referencedObjective="obj-prereqs" />
      <imsss:ruleCondition operator="not"
           condition="activityProgressKnown"
           referencedObjective="obj-prereqs" />
    </imsss:ruleConditions>
    <imsss:ruleAction action="disabled" />
  </imsss:preConditionRule>
 </imsss:sequencingRules>
<imsss:rollupRules rollupObjectiveSatisfied="true"
           rollupProgressCompletion="true" />
 <imsss:objectives>
  <imsss:primaryObjective objectiveID="obj-primary"
           satisfiedByMeasure="false"/>
  <imsss:objective objectiveID="obj-prereqs"
           satisfiedByMeasure="false">
    <imsss:mapInfo targetObjectiveID="obj-global-lessons"
           readSatisfiedStatus="true"/>
  </imsss:objective>
 </imsss:objectives>
<imsss:deliveryControls objectiveSetByContent = "true"/>
<adlseq:objectives>
 <adlseq:objective objectiveID = "obj-prereqs">
    <adlseq:mapInfo targetObjectiveID = "obj-global-lessons"
           readCompletionStatus="true" />
  </adlseq:objective>
 </adlseq:objectives>
</imsss:sequencing>
```

Рис. 23. Пример реализации предусловия

Более детальные сведения о разработке учебных курсов на базе стандарта SCORM можно найти в [19].

7. Модель компетенций

Модель компетенций используется для спецификации *знаний, умений и навыков (ЗУНов)* в LMS и в компетентностных профилях учащихся. Модель компетенций предоставляет средства для общепонятного описания компетенций, являющихся составной частью учебного плана, для спецификаций ЗУНов, необходимых для начала обучения (learning prerequisites), и ЗУНов, получаемых в результате обучения (learning outcomes). Модель компетенций может использоваться для обмена ЗУНами между различными LMS, системами управления персоналом (human resource management system), образовательным контентом, хранилищами компетенций и другими аналогичными системами [21].

Модель компетенций была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандартах IEEE 1484.20.1-2007 [22] и IEEE SRCM [23]. Стандарт IEEE 1484.20.1-2007 содержит формальное описание *структуры определения компетенции*. Стандарт IEEE SRCM определяет информационную модель для представления связей между различными компетенциями в виде ориентированного ациклического графа, называемого *графом компетенций*.

Структура определения компетенции изображена на рис. 24 и включает в себя следующие основные атрибуты:

- *identifier*: идентификатор компетенции;
- *title*: название компетенции;
- *description*: описание компетенции;
- *definition*: определение компетенции.

Identifier представляет собой глобальный идентификатор компетенции, который должен быть уникален во всех системах, хранящих или обрабатывающих данную компетенцию.

Title представляет собой понятное для человека название компетенции. Название может быть продублировано на нескольких языках. В качестве названия компетенции, например, может быть текстовая строка «Знание английского языка».

Description представляет собой понятное для человека описание компетенции, которое предназначено только для интерпретации человеком. Описание может быть продублировано на нескольких языках. В качестве описания компетенции, например, может быть текстовая строка «Владение письменным и устным английским языком».

Definition представляет собой структурированное описание компетенции, которое включает в себя непустой набор *утверждений (statements)*.

Statement (утверждение) описывает отдельное свойство компетенции и включает в себя следующие элементы:

- *statement_ id*: идентификатор утверждения,
- *statement_ name*: имя утверждения,
- *statement_ text*: описание утверждения,
- *statement_ token*: лексема.

Statement_ id представляет собой уникальную метку в рамках данного определения.

Statement_ name представляет собой имя утверждения, которое должно быть уникальным в рамках данного определения. В качестве имени утверждения может быть исполь-

```
reusable_competency_definition: record
(
   identifier: long_identifier_type,
   title: bag of langstring_type(1000),
   description: bag of langstring_type(4000),
   definition: definition_type,
   metadata: metadata_type,
)

definition_type = record
(
    model_source: characterstring(iso-10646),
   statement: statement_type,
)

statement_type = record
(
   statement_id: long_identifier_type,
   statement_text: bag of langstring_type(1000),
   statement_token: vocabulary_type,
)

vocabulary_type = record
(
   source: characterstring(iso-10646),
   value: characterstring(iso-10646),
)
```

Рис. 24. Определение компетенции

зованы *condition (условие)*, *action (действие)*, *standard (стандарт)*, *outcome (результат)*, *criteria (критерий)*.

Statement_ text представляет собой неструктурированное текстовое описание свойства компетенции, представленного данным утверждением. В качестве описания *statement_ text* может быть, например, текстовая строка «Задано множество целых чисел в диапазоне от 1 до 49».

Лексема *statement_ token* представляет собой словарную лексему, которая включает в себя *значение лексемы (value)* и *идентификатор схемы лексемы (source)*. Словарная лексема позволяет использовать зафиксированные словарные термины вместе (или вместо) со свободным текстовым описанием утверждения *statement_ text*.

Граф компетенций, описываемый стандартом IEEE SRCM, представляет собой ориентированный ациклический граф. Узлы графа компетенции могут ссылаться на различные формальные описания компетенций, структура которых была описана выше.

Граф компетенций представляет собой *нестрогую* иерархию узлов в том смысле, что узел графа компетенций может иметь более одного родительского узла. Если узел графа компетенций A имеет сыновей B и C, то из этого следует, что компетенция A состоит из суб-компетенций B и C, или, что компетенции B и C составляют A. На рис. 25 приведены четыре примера графа компетенций. Стрелка задает связь отец-сын.

Узел графа компетенций может ссылаться как на формальное определение компетенции, так и на другие графы компетенций. Указанная структура модели представлена на рис. 26.

Узлы графа компетенций могут иметь специальные правила, определяющие поведение системы при анализе графа компетенций. В правилах могут быть использованы следующие методы обработки компетенций:

- *метод all*,
- *метод any*,
- *метод fraction*,
- *метод units*,
- *метод mean*,
- *метод other*.

Метод all определяет правило, при котором для приобретения компетенции данного узла должны быть приобретены все компетенции, связанные с дочерними узлами. Этот метод используется по умолчанию.

Метод any определяет правило, при котором для приобретения компетенции данного узла должна быть приобретена хотя бы одна компетенция, связанная с одним из дочерних узлов.

Метод fraction определяет правило, при котором для приобретения компетенции данного узла должна быть приобретена определенная доля компетенций, ассоциированных с дочерними узлами. Значение доли является вещественным числом из отрезка [0, 1]. Значение 0 означает, что для приобретения компетенции не требуется приобретать дочерние компетенции, значение 1 эквивалентно методу all, значение 0.5 означает, что для приобретения компетенции требуется приобрести 50% дочерних компетенций.

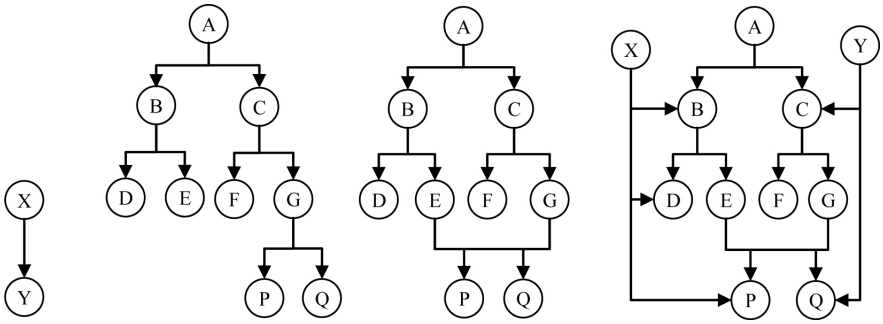

Рис. 25. Примеры графа компетенций с разным уровнем сложности

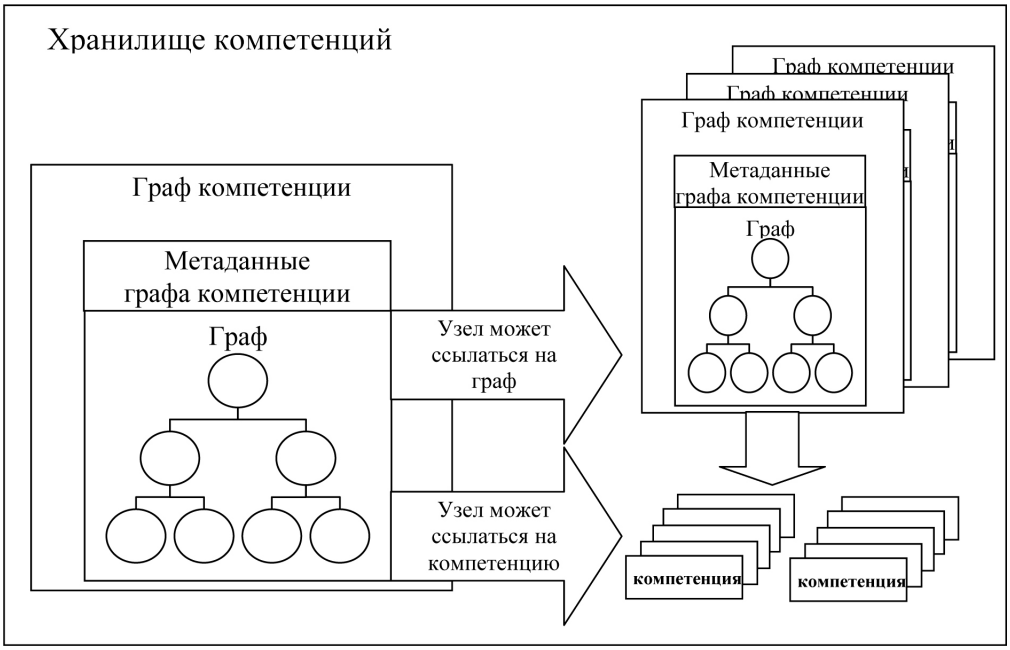

Рис. 26. Структура модели

Метод units определяет правило, при котором для приобретения компетенции данного узла должно быть приобретено определенное количество компетенций, ассоциированных с дочерними узлами. Количество является целочисленным значением. Значение 0 означает, что для приобретения компетенции не требуется приобретать дочерние компетенции, значение 1 эквивалентно *методу any.*

Метод mean определяет правило, при котором вычисляется мера приобретения компетенции, ассоциированной с данным узлом, как усредненное значение мер приобретения компетенций, связанных с дочерними узлами. Факт приобретения компетенции, ассоциированной с данным узлом, определяется путем сравнения вычисленной меры с заданным пороговым значением (proficiencyRequired).

Метод other определяет нестандартное правило, которое должно быть определено в профиле приложения.

Заключение

В работе предложен обзор основных международных стандартов на структуру и представление элементов образовательного контента в области электронного обучения. Базовым стандартом здесь является SCORM. Он обеспечивает возможность переноса элементов контента из одного электронного учебного курса в другой на физическом уровне.

Следует отметить, что до сих пор отсутствуют стандарты, определяющие принципы формирования дидактической структуры электронных учебных курсов. Это ограничивает возможность переноса методических наработок из одного электронного учебного курса в другой и препятствует получению максимального эффекта при внедрении электронного обучения в высшей школе. Авторами в настоящее время ведется работа по созданию высокоуровневой дидактической модели электронного учебного комплекса UniCST [24–26]. Модель предусматривает двухуровневую методическую базу знаний. Первый уровень предполагает создание комплекса электронных энциклопедий по различным областям знаний. Второй уровень предполагает создание электронных учебных курсов на основе существующих энциклопедий путем экспорта учебных блоков и организации их в иерархическую структуру, адекватно реализующих рабочие учебные программы в соответствии с образовательными стандартами направлений третьего поколения. Модель поддерживает структурирование учебного материала по способам представления образовательного контента. Учебный блок представляет собой набор компонент. Модель поддерживает стандартный набор таких компонент: теоретическое описание понятия; пример, иллюстрирующий те или иные отличительные черты понятия; упражнение для самостоятельного выполнения; тестовое задание; слайд презентации; библиографическая ссылка. Каждая компонента обладает своими дидактическими возможностями. Модель допускает возможность расширения набора стандартных дидактических способов представления учебного материала с учетом специфики каждой конкретной специальности или направления подготовки. Один и тот же блок учебного материала электронной энциклопедии может быть использован в разных электронных учебных курсах.

Литература

1. Tavangarian, D. Is e-Learning the Solution for Individual Learning? / D. Tavangarian, M. Leypold, K. Nōlting, M. Rōser // Electronic Journal of e-Learning. — 2004. — Vol. 2, No. 2. — URL: http://www.ejel.org/volume-2/vol2-issue2/v2-i2-art4-tavangarian.pdf (дата обращения: 02.10.2010).
2. Friedland, G. Architecting Multimedia Environments for Teaching. / Friedland G., Pauls K. — IEEE Computer Society, 2005. — Vol. 38, No. 6. — pp. 57–64.
3. Рут, С. Оправданно ли электронное обучение? / С. Рут // СУБД. — Открытые системы, 2010. — No. 03. — URL: http://www.osp.ru/os/2010/03/13001922/ (дата обращения: 02.09.2013).
4. Ли, К. и др. Новые Internet-технологии электронного обучения. // СУБД. — Открытые системы, 2009. — No. 07. — URL: http://www.osp.ru/os/2009/07/10456963/ (дата обращения: 02.09.2013).
5. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
6. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.
7. IEEE 1484.11.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication. 2005.
8. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (дата обращения: 28.01.2013).
9. IEEE 1484.12.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. 2005.
10. CRS003. Hierarchy of CBT Terms for AICC Publications.1992. URL: http://archive.aicc.org/docs/tech/crs003.rtf (дата обращения: 15.03.2011).
11. CMI001. CMI Guidelines for Interoperability AICC. 2004. URL: http://archive.aicc.org/docs/tech/cmi001v4.pdf (дата обращения: 15.03.2011).
12. IMS Content Packaging Information Model. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_infov1p2pd2.html (дата обращения: 04.06.2011).
13. IMS Content Packaging XML Binding. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_bindv1p2pd2.html (дата обращения: 04.06.2011).
14. IMS Simple Sequencing Best Practice and Implementation Guide, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bestv1p0.html (дата обращения: 10.06.2011).
15. IMS Simple Sequencing Information and Behavior Model, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html (дата обращения: 10.06.2011).

16. IMS Simple Sequencing XML Binding, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bindv1p0.html (дата обращения: 10.06.2011).

17. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.

18. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.

19. SCORM 2004 4th Edition. SCORM Users Guide for Programmers. Advanced Distributed Learning (ADL) Initiative. 2011.

20. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.

21. Rifon L.A. Standardising competency definitions for engineering education. Proceedings of Global Engineering Education Conference (EDUCON), Amman, Jordan, 4–6 April 2010. IEEE, 2010. P. 52–58.

22. IEEE 1484.20.1. Standard for Learning Technology – Data Model for Reusable Competency Definitions. 2007.

23. IEEE Draft Standard on Simple Reusable Competency Map. 2006. URL: http://ieeeltsc.files.wordpress.com/2009/03/reusablecompetencymapproposal.pdf (дата обращения: 28.01.2013).

24. Силкина, Н.С. Модель образовательного стандарта третьего поколения на основе компетентностного подхода для систем электронного обучения / Н.С. Силкина, А.С. Евдокимова // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». — 2011. — № 37(254). Вып. 10. — С. 90–98.

25. Силкина, Н.С. Система UniCST — универсальная среда электронного обучения / Н.С. Силкина, Л.Б. Соколинский // Системы управления и информационные технологии. — 2010. — № 2. — С. 81–86.

26. Жигальская, Н.С. Моделирование дидактической структуры электронных учебных комплексов // Вестник Южно-Уральского государственного университета. Серия «Математическое моделирование и программирование». — 2008. — № 27(127). Вып. 2. — С. 4–9.

Силкина Надежда Сергеевна, ст. преподаватель кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), zhnadya@rambler.ru.

Соколинский Леонид Борисович, д.ф.-м.н., профессор, проректор по информатизации, Южно-Уральский государственный университет (Челябинск, Российская Федерация), Leonid.Sokolinsky@susu.ru.

Поступила в редакцию 7 июля 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 5–35

DOI: 10.14529/cmse140401

E-LEARNING MODELS AND STANDARDS

N.S. Silkina, L.B. Sokolinsky

The paper is a review of models and standards used in modern e-learning systems. The general concept model of e-learning environment is considered. The review also regards these topics: the data model for interaction with learning objects; the content aggregation model which describes the structure of learning objects, the ways of searching, packing and transmitting them between different learning systems; the run-time environment model, which defines an API for learning objects management; the sequencing model for learning content; the competency model, which is used to specify knowledge, skills and abilities in e-learning systems. The paper also reviews SCORM, which integrates a system of e-learning models.

Keywords: e-learning, metadata model, data model, content aggregation model, run-time environment, simple sequencing information and behavior model, competency model, SCORM.

References

1. Tavangarian D., Leypold M., Nōlting K., Rōser M. Is e-Learning the Solution for Individual Learning? Electronic Journal of e-Learning. 2004. Vol. 2, No. 2. URL: http://www.ejel.org/volume-2/vol2-issue2/v2-i2-art4-tavangarian.pdf.
2. Friedland G., Pauls K. Architecting Multimedia Environments for Teaching. IEEE Computer Society. 2005. Vol. 38, No. 6. pp. 57–64. DOI: 10.1109/MC.2005.181.
3. Ruth S. Is E-Learning Really Working? The Trillion-Dollar Question. IEEE Internet Computing. 2010. Vol. 14,, No. 2. pp. 78–82. DOI: 10.1109/MIC.2010.46.
4. Li Q., Lau R., Leung R., Li F., Lee V., Wah B., Ashman H. Emerging Internet Technologies for E-Learning. IEEE Internet Computing. 2009. Vol. 13, No. 4. pp. 11–17. DOI: 10.1109/MIC.2009.83.
5. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
6. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.
7. IEEE 1484.11.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication. 2005.
8. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf.
9. IEEE 1484.12.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. 2005.
10. CRS003. Hierarchy of CBT Terms for AICC Publications.1992. URL: http://archive.aicc.org/docs/tech/crs003.rtf (дата обращения: 15.03.2011).
11. CMI001. CMI Guidelines for Interoperability AICC. 2004. URL: http://archive.aicc.org/docs/tech/cmi001v4.pdf (дата обращения: 15.03.2011).

12. IMS Content Packaging Information Model. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_infov1p2pd2.html (дата обращения: 04.06.2011).
13. IMS Content Packaging XML Binding. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_bindv1p2pd2.html (дата обращения: 04.06.2011).
14. IMS Simple Sequencing Best Practice and Implementation Guide, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bestv1p0.html (дата обращения: 10.06.2011).
15. IMS Simple Sequencing Information and Behavior Model, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html (дата обращения: 10.06.2011).
16. IMS Simple Sequencing XML Binding, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bindv1p0.html (дата обращения: 10.06.2011).
17. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
18. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
19. SCORM 2004 4th Edition. SCORM Users Guide for Programmers. Advanced Distributed Learning (ADL) Initiative. 2011.
20. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
21. Rifon L.A. Standardising competency definitions for engineering education. Proceedings of Global Engineering Education Conference (EDUCON), Amman, Jordan, 4–6 April 2010. IEEE, 2010. P. 52–58.
22. IEEE 1484.20.1. Standard for Learning Technology – Data Model for Reusable Competency Definitions. 2007.
23. IEEE Draft Standard on Simple Reusable Competency Map. 2006. URL: http://ieeeltsc.files.wordpress.com/2009/03/reusablecompetencymapproposal.pdf (дата обращения: 28.01.2013).
24. Silkina N.S., Evdokimova A.S. Model' obrazovatel'nogo standarta tret'ego pokolenija na osnove kompetentnostnogo podhoda dlja sistem jelektronnogo obuchenija [A Model of the Third Generation Educational Standard on the Basis of Competency Approach for E-learning Systems]. Bulletin of South Ural State University, "Mathematical Modelling, Programming & Computer Software" Series. 2011. Vol. 37(254). No. 10. pp. 90–98.
25. Silkina N.S., Sokolinsky L.B. Sistema UniCST — universal'naja sreda jelektronnogo obuchenija [UniCST System — a Universal E-learning Environment]. Sistemy upravlenija i informacionnye tehnologii [Control Systems and Information Technologies]. 2010. No. 2. pp. 81–86.
26. Zhigalskaya N.S. Modelirovanie didakticheskoj struktury jelektronnyh uchebnyh kompleksov [Didactic Structure Modeling of E-learning Systems]. Bulletin of South Ural State University, "Mathematical Modelling, Programming & Computer Software" Series. 2008. Vol. 27(127). No. 2. pp. 4–9.

Received July 7, 2014.