УДК 004.588 **DOI: 10.14529/cmse190405**

СТРУКТУРНО-ИЕРАРХИЧЕСКАЯ ДИДАКТИЧЕСКАЯ МОДЕЛЬ ЭЛЕКТРОННОГО ОБУЧЕНИЯ

© 2019 Н.С. Силкина, Л.Б. Соколинский

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: SilkinaNS@susu.ru, Leonid.Sokolinsky@susu.ru

Поступила в редакцию: 22.10.2019

В данной статье описывается оригинальная структурно-иерархическая дидактическая (СИД) модель электронного обучения. В основе модели лежит четырехуровневая методическая база знаний. Первый уровень включает в себя комплекс электронных учебных энциклопедий по различным областям знаний. Второй уровень включает в себя электронные учебные курсы. Модель поддерживает структурирование электронного учебного курса по дидактическим компонентам (вертикальное слоение) и уровням детализации (горизонтальное слоение). Третий уровень включает в себя комплекс рабочих учебных программ. Четвертый уровень включает в себя комплекс ФГОС ВО. Отличительной особенностью СИД модели является деление образовательных объектов на дидактические компоненты. Это позволит, во-первых, производить автоматическую верификацию дидактической полноты электронного учебного курса. Во-вторых, включать части одного курса в другой без потери дидактической структуры. В-третьих, выделять из электронного учебного курса отдельный дидактический слой и на его основе автоматически формировать специализированные учебно-методические материалы: конспекты лекций, сборники задач, экзаменационные тесты и др. Описаны основные операции СИД модели, на основе которых предложены алгоритмы по анализу образовательных программ и электронных учебных курсов. В заключении дается краткая сводка результатов и направления дальнейших исследований.

Ключевые слова: электронное обучение, e-learning, электронный учебный курс, модель электронного обучения, образовательный объект, дидактическая структура, СИД модель.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Силкина Н.С., Соколинский Л.Б. Структурно-иерархическая дидактическая модель электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 56–83. DOI: 10.14529/cmse190405.

Введение

Идея использования компьютеров в учебном процессе нашла свое выражение еще в 60-е годы в форме концепции программированного обучения [1], которая была предложена американским психологом Б.Ф. Скиннером. В соответствие с этой концепцией весь учебный материал делился на небольшие порции, а процесс обучения — на шаги. В ходе одного шага учащийся осваивал одну порцию материала. Программа, управляющая обучением, поддерживала только линейную последовательность шагов. Позднее Н. Краудером был предложен алгоритм разветвленного программированного обучения [2]. Основным отличием данного подхода является введение индивидуальных путей прохождения учебного материала. Путь для каждого учащегося определяет сама программа в процессе обучения, основываясь на ответах учащихся. Следующий этап развития программированного обучения связан с использованием гипертекстовых и мультимедийных технологий [3], которые, по существу, стали базовой технологической платформой образовательного контента. Это дало возможность использовать в качестве образовательных объектов

не только текстовую информацию, но также графику, аудио и видеоинформацию. Усложнение внутренней структуры образовательного контента породило проблему повторного использования элементов электронных учебных курсов в других электронных обучающих системах. Необходимость решения этой проблемы стимулировала развитие универсальных моделей электронного обучения и стандартов, разработанных на их основе.

Множество моделей электронного обучения можно представить в виде иерархии классов, изображенной на рис. 1. Первый уровень представлен *моделями данных*, предназначенными для обмена данными между образовательными объектами (Learning Objects, LO) и системой управления обучением (Learning Management System, LMS). LO использует модель данных для того, чтобы получить от LMS информацию, которая позволит ему выполнить требуемые обучающие функции. LMS использует модель данных для того, чтобы получать от LO информацию, которая позволит ей правильно управлять объектами LO. Таким образом, модель данных должна описывать структуру информации, которая может быть передана в и получена от LO. Однако, модель данных не должна специфицировать как, когда и в каком направлении информация может передаваться. К моделям данного уровня относится *Модель данных для взаимодействия с образовательными объектами* (*Data Model for Content Object Communication*) [4].

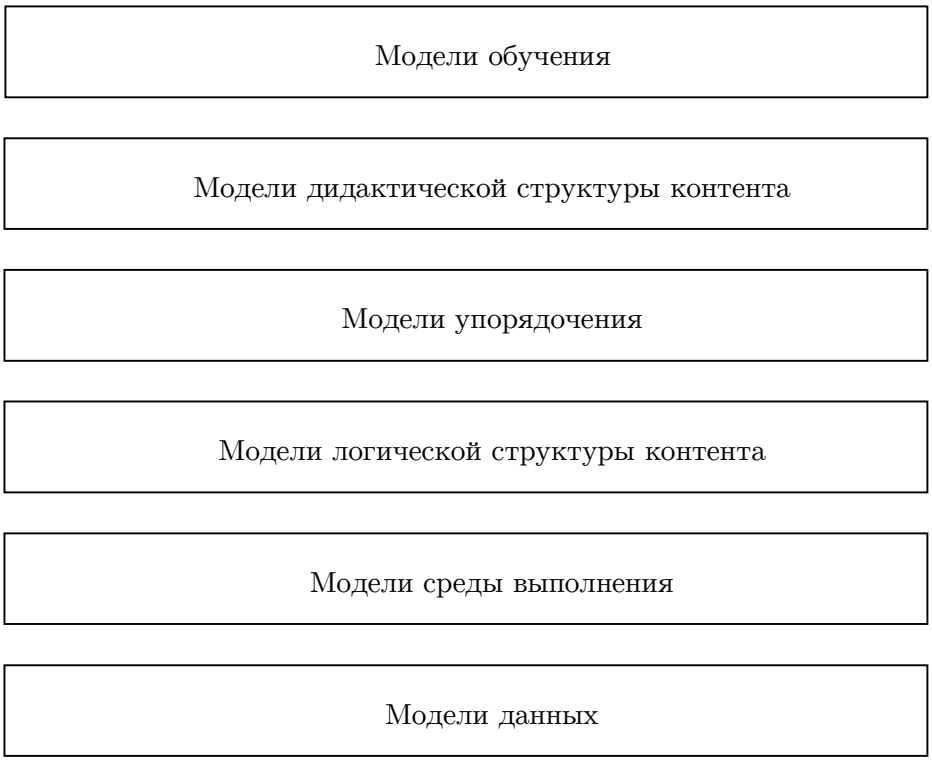

Рис. 1. Иерархия классов моделей электронного обучения

Второй уровень представлен *моделями среды выполнения*, которые описывают взаимодействие LO и LMS через прикладной программный интерфейс (Application Program Interface, API). Модель среды выполнения позволяет обеспечить совместимость LO и LMS, чтобы каждая система электронного обучения могла взаимодействовать с LO таким же образом, как и любая другая, поддерживающая ту же модель среды выполнения. Данная модель должна обеспечивать доставку требуемых ресурсов учащемуся, за-

пуск LO, отслеживание и обработку информации о действиях учащегося. Примером моделей данного уровня может служить *Модель среды выполнения* (*Run-Time Environment, RTE*) стандарта SCORM [5].

Следующий уровень представлен *моделями логической структуры контента*, согласно которым образовательный контент представляет собой некую структурно-иерархическую организацию (как правило, в виде графа или дерева). К ним относится *Модель структуры контента стандарта* (*Content Aggregation Model, CAM*) SCORM [6], *карта знаний* (*Concept Map*) [7], *граф содержания* [8].

Следующий уровень представлен моделями, которые позволяют определить последовательность изучения LO, в том числе повторное изучение. К таким моделям относится *Модель упорядочения и навигации* (*Sequencing and Navigation, SN*) стандарта SCORM [9], а также упорядочивание образовательного контента в *модели CDCGM* [10].

На пятом уровне иерархии классов находятся *модели дидактической структуры контента*. Примером может служить *Модель компетенций* (*Data Model for Reusable Competency Definitions*), описываемая стандартом IEEE 1484.20.1-2007 [11]. Также к этому классу может быть отнесена *модель динамического контента* (*Dynamic Content Model, DCM*) [12].

На верхнем уровне иерархии находятся модели обучения, которые можно разделить на три группы: модели адаптивного обучения, модели коллаборативного обучения и модель электронной учебной энциклопедии [13, 14, 15].

Адаптивное обучение — это способ организации учебного процесса с учетом индивидуального уровня подготовки учащегося до начала обучения или в процессе обучения, при котором направление дальнейшего обучения (график и интенсивность) определяется по результатам завершения предыдущих курсов или их частей. К этому классу моделей можно отнести *модель KFS* [16–18], *модель содержания учебного материала Соловова* [19, 20] и *модель CDCGM* [21].

Коллаборативное обучение — это такой способ организации обучения, при котором обучение происходит в процессе совместного решения учебных задач, осуществления взаимообмена знаниями. Одной из самых популярных систем, реализующих подобную модель, является LMS с открытым кодом Moodle (http://moodle.org), которая широко известна в мире и используется более чем в 100 странах, в том числе и в России.

Электронные энциклопедии [22] представляют собой базу знаний по некоторой предметной области (или сразу нескольким областям). Учебный материал в электронной энциклопедии представляется в виде описания понятий (статей), каждое понятие может «опираться» на другие понятия. Примерами реализации такого подхода являются электронные энциклопедии по линейной алгебре Линеал (http://lineal.guru.ru), по параллельному программированию Параллель (http://parallel.guru.ru), а также свободная энциклопедия по языкам программирования Прогопедия (http://progopedia.ru) и свободная энциклопедия Wikipedia (http://www.wikipedia.org). Статьи в электронной энциклопедии во многих случаях имеют линейный (алфавитный) порядок, однако, на ряду с этим встречаются энциклопедии, в которых словарные статьи организованы по иерархическому принципу [23, 24]. Подробнее все рассмотренные модели проанализированы авторами в работах [25, 26].

Целью данной работы является построение структурно-иерархической дидактической модели электронного обучения (или СИД модели). Данная модель относится к пятому

уровню иерархии, представленной на рис. 1. В основе модели лежит четырехуровневая методическая база знаний. Первый уровень включает в себя комплекс электронных учебных энциклопедий по различным областям знаний. Верхний уровень включает в себя комплекс ФГОС ВО. Третий уровень включает в себя комплекс рабочих учебных программ. Второй уровень включает в себя электронные учебные курсы. Модель поддерживает структурирование электронного учебного курса по дидактическим компонентам (вертикальное слоение) и уровням детализации (горизонтальное слоение). Отличительной особенностью этой модели является поддержка деления образовательных объектов на дидактические компоненты. Это позволит, во-первых, производить автоматическую верификацию дидактической полноты электронного учебного курса. Во-вторых, включать части одного курса в другой без потери дидактической полноты. В-третьих, выделять из электронного учебного курса отдельный дидактический слой и на его основе автоматически формировать самостоятельные учебно-методические материалы: конспекты лекций, сборники задач, экзаменационные тесты и др.

Статья организована следующим образом. В разделе 1 описываются требования к структурно-иерархической дидактической модели. Раздел 2 посвящен формальному описанию СИД модели. В разделе 3 дается описание основных операций СИД модели. В разделе 4 описываются основные алгоритмы по анализу образовательных программ и электронных учебных курсов на базе СИД модели. Заключение содержит выводы и направления дальнейших исследований.

1. Требования к модели дидактической структуры контента

Модель дидактической структуры контента должна удовлетворять следующим основным требованиям:

1) учитывать специфику образовательных стандартов «ФГОС 3++»;
2) определять дидактические типы для образовательных объектов;
3) поддерживать модульный подход при формировании образовательного контента;
4) предусматривать возможность создания дидактически структурированных хранилищ образовательных объектов в определенной предметной области (электронных энциклопедий);
5) предусматривать возможность создания дидактически структурированных электронных курсов, а также обеспечивать возможность их автоматизированного создания на основе электронных энциклопедий и других электронных курсов;
6) обеспечивать совместимость с существующими стандартами в области электронного обучения.

Рассмотрим указанные требования более подробно.

1.1. Образовательные стандарты «ФГОС 3++»

Федеральные государственные образовательные стандарты высшего профессионального образования «ФГОС 3++» представляют собой совокупность требований, обязательных при реализации основных образовательных программ высшего профессионального образования образовательными учреждениями, имеющими государственную аккредитацию. Федеральным законом «Об образовании в Российской Федерации» от 29 декабря 2012 года № 273-ФЗ утверждена структура федерального государственного образовательного стандарта (ФГОС). Каждый ФГОС включает три группы требований:

1) требования к структуре основных образовательных программ (в том числе соотношению обязательной части основной образовательной программы и части, формируемой участниками образовательных отношений) и их объему;
2) требования к условиям реализации основных образовательных программ, в том числе кадровым, финансовым, материально-техническим и иным условиям;
3) требования к результатам освоения основных образовательных программ.

Согласно Федеральному закону образовательная программа — это комплекс основных характеристик образования (объем, содержание, планируемые результаты), организационно-педагогических условий и в случаях, предусмотренных настоящим Федеральным законом, форм аттестации, который представлен в виде учебного плана, календарного учебного графика, рабочих программ учебных предметов, курсов, дисциплин (модулей), иных компонентов, а также оценочных и методических материалов. Основная образовательная программа (ООП) разрабатывается образовательным учреждением на основе ФГОС и определяет цели, задачи, планируемые результаты, содержание и организацию образовательного процесса. Требования к результатам освоения основных образовательных программ подготовки бакалавров (магистров, специалистов) в целом и их разделов формулируется в виде компетенций как в области профессиональной, так и социально-личностной деятельности [27].

1.2. Дидактические типы образовательных объектов

Дидактическим элементом в педагогике называют логически самостоятельную часть учебного материала, по своему объему и структуре соответствующую таким компонентам содержания как понятие, теория, закон, явление, факт, объект и др. [28, 29]. Наряду с этим понятием в педагогике существует также понятие *фонда оценочных средств* (*ФОС*), который представляет собой совокупность *контрольно-измерительных материалов* (*КИМ*) и методов их использования [29]. Примерами КИМ являются типовые задачи (задания), контрольные работы, тесты и др. С точки зрения обучения в высшей школе фонд оценочных средств позволяет оценить достижение запланированных в образовательной программе результатов обучения и уровень сформированности компетенций, заявленных в образовательной программе дисциплины. Таким образом, можно выделить следующие основные дидактические типы:

1) теоретическое (гипертекстовое) описание понятия;
2) вопрос с открытым ответом, проверяющий знание (понимание) теоретического описания понятия;
3) пример использования понятия при решении задач;
4) упражнение на умение применять изученное понятие при решении задач;
5) вопрос с закрытым ответом, проверяющий знание (понимание) теоретического описания понятия;
6) практическое задание, задание для самостоятельной работы, формирующие навыки использования изученного понятия при решении задач;
7) библиографическая ссылка.

Модель дидактической структуры контента в обязательном порядке должна поддерживать указанные типы.

1.3. Модульный подход при формировании образовательного контента

Модель дидактической структуры контента должна поддерживать модульный подход при формировании образовательного контента. С точки зрения дидактики *модуль* представляет собой наименьшую логическую единицу обучения, объединяющую в себе образовательный контент для формирования определенных знаний, умений и навыков. С методической точки зрения модуль соответствует определенному понятию. Каждый модуль имеет определенную *дидактическую схему*. Эта схема однозначно определяется набором атрибутов. Каждый атрибут представляет собой пару (идентификатор, дидактический тип). Значением атрибута может являться образовательный объект соответствующего типа. Далее значения атрибутов модуля будем называть *компонентами*. Некоторые компоненты модуля могут быть пустыми. С точки зрения стандарта SCORM, модулю соответствует агрегация [6].

1.4. Структурированные электронные энциклопедии

Модель дидактической структуры контента должна предусматривать возможность формирования дидактически структурированных *электронных энциклопедий* (хранилищ образовательных объектов) по различным предметным областям. *Дидактическая схема* (структура) энциклопедии однозначно определяется набором атрибутов, представляющих собой пару: (идентификатор, дидактический тип). Энциклопедия может включать в себя модули, которые имеют ту же дидактическую схему, что и энциклопедия. Между родственными модулями энциклопедии возможно установление связей «смотри также». Под родственными модулями понимаются модули, связанные отношениями «должен знать». В отличие от связей «должен знать» связи «смотри также» могут идти в любом направлении (от отца к сыну и от сына к отцу), а также могут идти через поколения (от отца к внуку и далее). Кроме этого, некоторые (возможно все) родственные модули могут в энциклопедии не иметь взаимных связей. Модули энциклопедии не обязаны образовывать иерархическую структуру.

1.5. Электронные учебные курсы

Модель дидактической структуры контента должна предусматривать возможность формирования дидактически структурированных *электронных учебных курсов*. *Дидактическая схема* электронного учебного курса однозначно определяется набором атрибутов, представляющих собой пару (идентификатор, дидактический тип). Электронный учебный курс может включать в себя модули, которые имеют ту же дидактическую схему, что и курс. Все модули курса должны образовывать упорядоченную древовидную структуру. В качестве корня дерева фигурирует модуль, включающий в себя аннотацию курса. Связи в направлении от корня к листьям задают отношение «следует изучить». Линейный порядок узлов одного уровня задает порядок их изучения. По умолчанию порядок изучения модулей курса определяется прямым обходом дерева: корень, узлы первого поддерева, узлы второго поддерева и так далее. Указанный порядок соответствует модели простого упорядочивания стандарта SCORM [9].

Модель дидактической структуры контента должна обеспечивать возможность переноса (копирования) модулей из электронной энциклопедии в электронный курс и из одного электронного курса в другой.

1.6. Совместимость с существующими стандартами в области электронного обучения

Модель дидактической структуры контента должна быть совместима со стандартом SCORM. Это предполагает совместимость с моделями, которые были описаны в работах [5, 6, 9]. Совместимость модели дидактической структуры контента со стандартом SCORM позволит осуществлять импорт и экспорт образовательного контента из одной системы управления обучением в другую.

2. Формальное описание СИД модели

Образовательный контент в СИД модели хранится в электронных учебных энциклопедиях (далее — просто энциклопедии). Основной структурной единицей энциклопедии является модуль. Модуль содержит образовательный контент, связанный с определенным понятием. С формальной точки зрения модуль представляет собой сложный образовательный объект, состоящий из следующих компонент:

1) `concept`: теоретическое (текстовое) описание понятия;
2) `open_question`: вопрос с открытым ответом для самопроверки понимания теоретического описания понятия;
3) `example`: пример решения задачи с использованием изучаемого понятия;
4) `exercise`: задание, проверяющее умение применять изученное понятие при решении задач;
5) `close_question`: вопрос с закрытым ответом для проверки освоения теоретического описания понятия;
6) `problem`: практическое задание, формирующее навыки использования изученного понятия при решении задач;
7) `bibitem`: библиографическая ссылка.

Отметим, что модулю в стандарте SCORM соответствует разделяемый объект контента SCO. Компоненты модуля соответствуют дидактическим типам, описанным в разделе 1.2. Значением компоненты модуля является коллекция, представляющая собой список (упорядоченную последовательность) элементов, имеющих соответствующий дидактический тип. Коллекция может не содержать ни одного элемента, то есть быть пустой. Основными операциями над коллекциями являются следующие:

1) `CreateCursor(collection)` — создание курсора;
2) `Fetch(cursor)` — возвращает элемент коллекции, на который указывает курсор;
3) `Set(cursor, value)` — присваивает указанное значение элементу коллекции, на который указывает курсор;
4) `Next(cursor)` — передвигает курсор на следующий элемент коллекции;
5) `Insert(cursor)` — добавляет новый элемент непосредственно перед элементом, на который указывает курсор;
6) `Delete(cursor)` — удаляет элемент коллекции, на который указывает курсор.

Тип компоненты определяет дидактический класс ассоциированного с ним объекта SCO стандарта SCORM. Каждый такой объект SCO включает в себя ассет в виде html-фрагмента и необязательный ассет в виде функции JavaScript. Первый ассет включает в себя образовательный контент в виде гипертекста. Второй ассет содержит программный

код на языке JavaScript, реализующий интерактивные действия по взаимодействию с учащимся. В соответствии со стандартом SCORM система управления обучением LMS запускает SCO в дочернем окне или фрейме окна LMS. Описание SCO осуществляется в соответствии с моделью метаданных LOM [33]. Таким образом, объекты СИД модели имеют возможность взаимодействия с учащимся, а также могут запрашивать и передавать данные от/в LMS [4], используя стандартный API [34].

Источником образовательного контента для наполнения курсов служат энциклопедии. Энциклопедии включают в себя образовательные модули, которые, в свою очередь, состоят из дидактических компонент. Каждая компонента характеризуется своим дидактическим типом. Пусть $\mathbb{T}$ — множество всех дидактических типов. Для каждого дидактического типа $T \in \mathbb{T}$ будем обозначать через $\mathfrak{D}_T$ множество всех конечных множеств (включая пустое), состоящих из значений этого типа.

Компонента w представляет собой четверку

$$w = \left\langle id_{component}, T, \xi, \lambda \right\rangle, \tag{1}$$

где $id_{component}$ — уникальный идентификатор компоненты, T — дидактический тип компоненты, $\xi \in \mathfrak{D}_T$ — коллекция, представляющая компоненту, λ — трудоемкость компоненты (в зачетных единицах).

Энциклопедия R представляет собой пару $\left\langle id_{encyclopedia}, M \right\rangle$, где $id_{encyclopedia}$ — уникальный идентификатор энциклопедии, M — коллекция (упорядоченный список) модулей энциклопедии.

Модуль μ представляет собой тройку:

$$\mu = \left\langle id_{module}, id_{encyclopedia}, \vec{w} \right\rangle, \tag{2}$$

где id_{module} — уникальный идентификатор модуля, $id_{encyclopedia}$ — энциклопедия модуля, $\vec{w} = (w_1, \ldots, w_7)$ — вектор компонент модуля в порядке, определенном выше.

В основе СИД модели лежит понятие компетенции, которая имеет следующее формальное определение.

Компетенция представляет собой четверку $\left\langle source, type, number, description \right\rangle$. Здесь *source* — уникальный идентификатор источника, содержащий перечень компетенций (в качестве такого идентификатора может выступать URL [32]); *type* — тип компетенции с возможными значениями из множества {УК, ОПК, ПК}, где УК обозначает универсальные компетенции, ОПК — общепрофессиональные компетенции, ПК — профессиональные компетенции [30, 31]; *number* — порядковый номер компетенции; *description* — наименование компетенции. В качестве примера можно привести следующую компетенцию:

<http://fgosvo.ru/uploadfiles/ProjFGOSVO3++/Bak3++/020302_B_3plus_21062017 .pdf, УК, 3, "Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде">.

Отметим, что роль первичного ключа в общей базе данных компетенций играют первые три атрибута: $(source, type, number)$.

Обозначим $\mathbb{K}$ — множество всех возможных компетенций, $\mathbb{K}_{univer}$ — подмножество универсальных компетенций, $\mathbb{K}_{genprof}$ — подмножество общепрофессиональных компетенций и $\mathbb{K}_{prof}$ — подмножество профессиональных компетенций. Множество компетенций обладает следующими свойствами:

$$\mathbb{K} = \mathbb{K}_{univer} \cup \mathbb{K}_{genprof} \cup \mathbb{K}_{prof}, \tag{3}$$

$$\mathbb{K}_{univer} \cap \mathbb{K}_{genprof} = \varnothing, \mathbb{K}_{univer} \cap \mathbb{K}_{prof} = \varnothing, \mathbb{K}_{genprof} \cap \mathbb{K}_{prof} = \varnothing. \tag{4}$$

На основе понятия компетенции вводится следующее формальное определение образовательного стандарта.

Образовательный стандарт S представляет собой семерку

$$S = \left\langle id_{standart}, K_{ungen}, \beta_{stud}, \beta_{pract}, \beta_{crt-l}, \beta_{crt-h}, \beta_{total} \right\rangle, \tag{5}$$

где $id_{standart}$ — уникальный идентификатор стандарта; $K_{ungen} \subset \left(\mathbb{K}_{univer} \cup \mathbb{K}_{genprof} \right)$ — множество универсальных и общепрофессиональных компетенций, входящих в стандарт; β_{stud} — минимальное количество зачетных единиц, отводимых на освоение учебных дисциплин; β_{pract} — минимальное количество зачетных единиц, отводимых на практики; β_{crt-l} и β_{crt-h} — соответственно нижняя и верхняя граница для зачетных единиц, отводимых на государственную итоговую аттестацию; β_{total} — общее количество зачетных единиц образовательной программы.

На основе образовательного стандарта формируется *образовательная программа*, включающая в себя совокупность учебных дисциплин.

Образовательная программа Q представляет собой четверку

$$Q = \left\langle id_{syllabus}, id_{standart}, K_{prof}, C \right\rangle, \tag{6}$$

где $id_{syllabus}$ — уникальный идентификатор образовательной программы, $id_{standart}$ — идентификатор образовательного стандарта, $K_{prof} \subset \mathbb{K}_{prof}$ — множество профессиональных компетенций, предусмотренных образовательной программой, C — множество курсов, входящих в образовательную программу. Определение курса будет дано ниже.

Электронный учебный курс γ представляет собой восьмерку вида:

$$\gamma = \left\langle id_{course}, id_{syllabus}, \beta_{lec}, \beta_{lab}, \beta_{out}, R, G, f \right\rangle, \tag{7}$$

где id_{course} — уникальный идентификатор курса; $id_{syllabus}$ — идентификатор образовательной программы, в рамках которой создается курс; β_{lec}, β_{lab}, β_{out} — трудоемкость лекций, лабораторных (практических) занятий и самостоятельной работы студента соответственно; R — энциклопедия курса, $G = \left\langle V, E \right\rangle$ — граф-план курса (см. определение ниже); $f : V \to R$ — однозначное инъективное отображение, сопоставляющее каждой вершине $\nu \in V$ граф-плана G некоторый модуль $\mu \in R$: $f(\nu) = \mu$.

Для задания иерархической структуры курса используется граф-план. *Граф-план* представляет собой связный ациклический граф $G = \left\langle V, E \right\rangle$ с выделенной вершиной $\overline{\nu}$, где $V = \left\{ v_l \mid l = 1,...,b \right\}$ — множество всех вершин граф-плана, $E = \left\{ e_p \mid p = 1,...,d \right\}$ — множество всех ребер граф-плана. Граф-план можно рассматривать как ориентированное дерево с корнем в вершине $\overline{\nu}$ в соответствии со следующим правилом:

Ребро, инцидентное вершинам ν *и* ν', *заменяем дугой от* ν *к* ν' *тогда и только тогда, когда простой путь от* ν *к* $\overline{\nu}$ *проходит через* ν', *то есть когда он имеет вид* $(\nu_0,\nu_1,\ldots,\nu_k)$, *где* $k>0,\ \nu_0=\nu,\ \nu_1=\nu',\ \nu_k=\overline{\nu}$.

Это означает, что направления дуг в ориентированном дереве полностью определяется положением корневой вершины $\overline{\nu}$, то есть их можно не указывать, если корень дерева обозначен явно.

Уровнем вершины $v\in V$ граф-плана $G=\langle V,E\rangle$ назовем длину простого пути от корня $\overline{\nu}$ до узла ν. Уровень корня $\overline{\nu}$ равен нулю. Множество вершин граф-плана, расположенных на уровне i от корня $\overline{\nu}$, назовем *i-тым ярусом* граф-плана и обозначим V_i.

Средняя трудоемкость λ_{avg} *модулей* курса γ вычисляется по следующей формуле

$$\lambda_{avg}=\frac{\sum\limits_{\mu\in f(V)}\sum\limits_{i=1}^{7}\mu.w_i.\lambda}{\left|V\right|}. \tag{8}$$

Дисбаланс курса определяется по формуле:

$$\delta=\frac{\sum\limits_{\mu\in f(V)}\left|\lambda_{avg}-\sum\limits_{i=1}^{7}\mu.w_i.\lambda\right|}{\left|V\right|}. \tag{9}$$

3. Основные операции СИД модели

Операции, предусмотренные в СИД модели, можно разделить на следующие группы:

1) операции над энциклопедиями;
2) операции над стандартами;
3) операции над образовательными программами;
4) операции над курсами и граф-планами.

Рассмотрим каждую из этих групп подробно.

3.1. Операции над энциклопедиями, модулями и компонентами

Для создания новой (пустой) энциклопедии используется операция:

```
encyclopedia = CREATE_ENCYCLOPEDIA().
```

Для работы с модулями энциклопедии используется внутренний курсор, указывающий на текущий модуль. Доступ к текущему модулю энциклопедии осуществляется следующим образом:

```
module = FETCH_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на первый модуль энциклопедии осуществляется с помощью операции:

```
RESET_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на следующий модуль энциклопедии осуществляется с помощью операции:

```
NEXT_MODULE(encyclopedia).
```

Перемещение внутреннего курсора на предыдущий модуль энциклопедии осуществляется с помощью операции:

```
PRIOR_MODULE(encyclopedia).
```

Для добавления в энциклопедию нового пустого модуля используется операция:

```
INSERT_MODULE(encyclopedia).
```

При этом текущим становится вновь добавленный модуль.

Удаление из энциклопедии текущего модуля выполняется с помощью операции:

DELETE_MODULE(encyclopedia).

После удаления текущим становится модуль, следующий за удаленным.

Для копирования содержимого модуля module1 в модуль module2 используется операция:

COPY_MODULE(module1, module2),

где module1 — модуль-донор, module2 — модуль-реципиент.

Доступ к коллекции элементов компоненты <COMPONENT> модуля module осуществляется следующим образом:

collection = <COMPONENT>_COLLECTION(module),

где в качестве <COMPONENT> фигурируют префиксы: CONCEPT, OPEN_QUESTION, EXAMPLE, EXERCISE, CLOSE_QUESTION, PROBLEM или BIBITEM. Работа с элементами коллекции, представляющей указанную компоненту, осуществляется с помощью операций над коллекциями, приведенными ранее.

Трудоемкость компоненты <COMPONENT> модуля module может быть получена аналогичным образом:

credit = <COMPONENT>_CREDIT(module).

3.2. Операции над стандартами

Для создания образовательного стандарта используется операция:

standart = CREATE_STANDART(description, min_stud, min_pract, min_crt, max_crt, total),

где min_stud — минимальное количество зачетных единиц, отводимых на освоение учебных дисциплин; min_pract — минимальное количество зачетных единиц, отводимых на практики; min_crt и max_crt — соответственно нижняя и верхняя граница для зачетных единиц, отводимых на государственную итоговую аттестацию; total — общее количество зачетных единиц образовательной программы. При создании стандарта неявно создаются курсоры для работы с коллекциями универсальных и общепрофессиональных компетенций. Доступ к атрибутам стандарта осуществляется с помощью операции GET_<atr>(standart), где в качестве <atr> указывается одно из следующих значений: DESCRIPTION, MIN_STUD, MIN_PRACT, MIN_CRT, MAX_CRT, TOTAL — названия соответствующих атрибутов стандарта.

Доступ к атрибуту универсальной компетенции осуществляется с помощью операции UC_GET_<atr>(standart), где в качестве <atr> указывается одно из следующих значений: NUMBER — порядковый номер компетенции; DESCRIPTION — наименование компетенции.

Изменение значения атрибута универсальной компетенции осуществляется с помощью операции UC_SET_<atr>(standart, value).

Перемещение внутреннего курсора на первую универсальную компетенцию стандарта осуществляется с помощью операции:

RESET_UC(standart).

Перемещение курсора на следующую универсальную компетенцию осуществляется с помощью операции:

NEXT_UC(standart).

Перемещение курсора на предыдущую универсальную компетенцию осуществляется с помощью операции:

```
PRIOR_UC(standart).
```

Для добавления в стандарт новой универсальной компетенции используется операция:

```
INSERT_UC_INTO_STANDART(standart, number, description),
```

где `number` — номер компетенции, `description` — наименование компетенции.

Удаление универсальной компетенции из стандарта выполняется с помощью операции:

```
DELETE_UC_FROM_STANDART(standart).
```

Аналогичным образом определяются операции для работы с общепрофессиональными компетенциями, только вместо суффикса `UC` используется суффикс `GPC`.

3.3. Операции над образовательными программами

Образовательная программа включает в себя (см. раздел 2) следующие атрибуты:

1) ссылка на образовательный стандарт;
2) перечень профессиональных компетенций, покрываемых образовательной программой;
3) перечень курсов, предусмотренных образовательной программой.

Для создания образовательной программы используется операция:

```
edu_prog = CREATE_EDU_PROG(standart),
```

где `standart` — указатель на стандарт, на основе которого разрабатывается образовательная программа. При создании образовательной программы неявно создаются курсоры для просмотра коллекции профессиональных компетенций и коллекции курсов.

Для получения указателя на стандарт, которые является базовым для образовательной программы `edu_prog` используется операция:

```
standart = GET_STANDART(edu_prog).
```

Операции для работы с профессиональными компетенциями образовательной программы определяются аналогично набору операций для работы с универсальными компетенциями стандарта, у которых вместо суффикса `UC` используется суффикс `PC`, а вместо указателя на стандарт `standart` — указатель на образовательную программу `edu_prog`.

Для работы с курсами образовательной программы используется внутренний курсор, указывающий на текущий курс.

Перемещение внутреннего курсора на начало списка учебных курсов образовательной программы осуществляется с помощью операции:

```
RESET_COURSE(edu_prog).
```

Перемещение курсора на следующий курс осуществляется с помощью операции:

```
NEXT_COURSE(edu_prog).
```

Перемещение курсора на предыдущий курс осуществляется с помощью операции:

```
PRIOR_COURSE(edu_prog).
```

Удаление текущего курса из образовательной программы выполняется с помощью операции:

```
DELETE_COURSE(edu_prog).
```

Для добавления в образовательную программу нового (пустого) курса используется операция:

```
INSERT_COURSE(edu_prog).
```

Получение указателя на текущий курс осуществляется с помощью операции:

```
course = FETCH_COURSE(edu_prog).
```

Доступ к атрибуту текущего курса осуществляется с помощью операции `COURSE_GET_<atr>(course)`, где в качестве `<atr>` указывается одно из следующих постфиксов: `TITLE` — название дисциплины; `LECTURE` — трудоемкость лекций; `LABORATORY` — трудоемкость лабораторных (практических) занятий; `OUTWORK` — трудоемкость самостоятельной работы студента.

Изменение значения атрибута текущего курса осуществляется с помощью операции `COURSE_SET_<atr>(course, value)`.

3.4. Операции над курсами и граф-планами

При создании нового курса автоматически создаются пустая *энциклопедия курса* и граф-план, состоящий из одной вершины (корня).

Доступ к энциклопедии курса осуществляется с помощью операции:

```
encyclopedia = GET_ENCYCLOPEDIA(course),
```

где `course` — курс соответствующей образовательной программы.

Доступ к корневой вершине граф-плана курса осуществляется с помощью операции:

```
graph_plan_root = GRAPH_PLAN(course),
```

где `course` — курс соответствующей образовательной программы. С каждой вершиной граф-плана ассоциируется коллекция дочерних вершин (возможно пустая), указатель на модуль из энциклопедии курса (возможно пустой) и коллекции номеров универсальных, общепрофессиональных и профессиональных компетенций. Для работы с дочерними вершинами для каждого узла граф-плана создается внутренний курсор, указывающий на текущую дочернюю вершину. Получение указателя на текущую дочернюю вершину узла `node` осуществляется с помощью операции:

```
child = FETCH_CHILD(node).
```

Перемещение внутреннего курсора на первую дочернюю вершину узла `node` осуществляется с помощью операции:

```
RESET_CHILD(node).
```

Перемещение внутреннего курсора на следующую дочернюю вершину узла `node` осуществляется с помощью операции:

```
NEXT_CHILD(node).
```

Перемещение курсора на предыдущую дочернюю вершину узла осуществляется с помощью операции:

```
PRIOR_CHILD(node).
```

Для добавления новой дочерней вершины узла `node` используется операция:

```
INSERT_CHILD(node).
```

Удаление дочерней вершины узла, на которую указывает курсор, выполняется с помощью операции:

```
DELETE_CHILD(node).
```

После удаления внутренний курсор устанавливается на дочернюю вершину, следующую за удаленной. Отметим, что при удалении узла граф-плана удаляется все поддерево, рекурсивно ассоциированное с удаляемым узлом.

Связывание узла граф-плана `node` с модулем `module` осуществляется следующим образом:

```
SET_LINK(node, module).
```

Для получения доступа к модулю энциклопедии курса, который ассоциирован с узлом граф-плана node, осуществляется следующим образом:

```
module = GET_LINK(node).
```

Добавление номера number в коллекцию номеров универсальных компетенций узла граф-плана node осуществляется следующим образом:

```
INSERT_UC_INTO_GP(node, number).
```

Удаление номера number из коллекции номеров универсальных компетенций узла граф-плана node осуществляется следующим образом:

```
DELETE_UC_FROM_GP(node, number).
```

Определение наличия универсальной компетенции с номером number в узле node граф-плана осуществляется следующим образом:

```
UC_ISEXIST(node, number).
```

Операции для работы с коллекциями номеров общепрофессиональных и профессиональных компетенций узла node граф-плана определяются аналогично (вместо суффикса UC используются суффиксы GPC и PC соответственно).

4. Анализ образовательных программ и электронных учебных курсов в СИД модели

В данном разделе рассматриваются основные алгоритмы по анализу образовательных программ и электронных учебных курсов на базе СИД модели.

4.1. Вычисление целостности курса

Перед использованием курса необходимо проверить его целостность. Под *целостностью* курса в СИД модели понимается тот факт, что каждый узел граф-плана *связан* с некоторым модулем. Под *коэффициентом целостности* курса понимается отношение количества *связанных* узлов к общему количеству узлов граф-плана курса. На рис. 2 представлена реализация рекурсивной функции вычисления количества связанных узлов поддерева с корнем root граф-плана курса.

```
function Amount_of_linked_nodes(root)
1)  amount = 0;
2)  if (GET_LINK(root) ≠ Null) then
3)       amount += 1;
4)  end if;
5)  RESET_CHILD(root);
6)  child = FETCH_CHILD(root);
7)  while (child ≠ Null)
8)       amount += Amount_of_linked_nodes(child);
9)       NEXT_CHILD (root);
10)      child = FETCH_CHILD(root);
11) end while;
12) return amount;
end function;
```

Рис. 2. Количество связанных узлов поддерева с корнем root

Количество связанных узлов граф-плана вычисляется в переменной amount. В строке 1 переменной amount присваивается начальное значение. В строках 2–4 значение переменной amount увеличивается на единицу, если узел root является связанным. В строке 5 выполняется операция RESET_CHILD, устанавливающая внутренний курсор на

начало списка дочерних узлов узла root. Операторы 6–11 добавляют в переменную amount количество связанных узлов в поддеревьях, соответствующих дочерним узлам. При этом используется рекурсивный вызов функции Amount_of_linked_nodes.

На рис. 3 представлена рекурсивная реализация функции вычисления общего количества узлов поддерева граф-плана с корнем root.

```
function Amount_of_nodes(root)
1)  amount = 1;
2)  RESET_CHILD(root);
3)  child = FETCH_CHILD(root);
4)  while (child ≠ Null))
5)       amount += Amount_of_nodes(child);
6)       NEXT_CHILD (root);
7)       child = FETCH_CHILD(root);
8)  end while;
9)  return amount;
end function;
```

Рис. 3. Количество узлов поддерева с корнем root

На рис. 4 представлена реализация функции вычисления коэффициента целостности курса.

```
function Consistency_rate(course)
1)  root = GRAPH_PLAN(course);
2)  return
    Amount_of_linked_nodes(root) / Amount_of_nodes(root);
end function;
```

Рис. 4. Коэффициент целостности курса

4.2. Оценка трудоемкости курса

При оценке трудоемкости курса используется понятие дидактического слоя. Под *дидактическим слоем* понимается совокупность всех компонент курса, имеющих одинаковый дидактический тип. Оценка трудоемкости электронного учебного курса в зачетных единицах осуществляется на основе оценки трудоемкости отдельных дидактических слоев:

1) Course_concept_credit(course) — трудоемкость слоя «Теория»;
2) Course_open_question_credit(course) — трудоемкость слоя «Вопросы с открытым ответом»;
3) Course_example_credit(course) — трудоемкость слоя «Примеры»;
4) Course_exercise_credit(course) — трудоемкость слоя «Упражнения»;
5) Course_close_question_credit(course) — трудоемкость слоя «Вопросы с закрытым ответом»;
6) Course_problem_credit(course) — трудоемкость слоя «Практические задания»;
7) Course_bibitem_credit(course) — трудоемкость слоя «Библиография».

Функция вычисления трудоемкости дидактического слоя в свою очередь реализуется с помощью соответствующей рекурсивной функции, вычисляющей трудоемкость слоя для поддерева граф-плана курса. На рис. 5 представлена реализация рекурсивной функции Subtree_concept_credit, вычисляющей трудоемкость слоя «Теория» для поддерева граф-плана курса.

```
function Subtree_concept_credit(root)
1)  module = GET_LINK(root);
2)  credit = CONCEPT_CREDIT(module);
3)  RESET_CHILD(root);
4)  child = FETCH_CHILD(root);
5)  while (child ≠ Null))
6)       credit += Subtree_concept_credit(child);
7)       NEXT_CHILD (root);
8)       child = FETCH_CHILD(root);
9)  end while;
10) return credit;
end function;
```

Рис. 5. Трудоемкость слоя «Теория» для поддерева с корнем `root`

Трудоемкость в зачетных единицах вычисляется в переменной `credit`. В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. Здесь предполагается, что все узлы граф-плана связаны с соответствующими модулями (это необходимо заранее проверить с помощью функции `Consistency_rate`, рассмотренной в разделе 4.1). В строке 2 переменной `credit` присваивается начальное значение. В строке 3 выполняется операция `RESET_CHILD`, устанавливающая внутренний курсор на начало списка узлов, являющихся дочерними по отношению к текущему. Операторы 4–9 добавляют в переменную `credit` трудоемкость «теории» дочерних узлов, используя рекурсивный вызов функции `Subtree_concept_credit`.

На рис. 6 представлен алгоритм вычисления трудоемкости слоя «теория» для всего курса в целом.

```
function Course_concept_credit(course)
1)  root = GRAPH_PLAN(course);
2)  credit = Subtree_concept_credit(root);
3)  return credit;
end function;
```

Рис. 6. Трудоемкость слоя «теория» для курса в целом

В строке 1 вычисляется ссылка на корень дерева граф-плана указанного курса. В строке 2 вычисляется общая трудоемкость слоя «теория». Как уже указывалось, для корректной работы функции `Course_concept_credit` предварительно необходимо проверить целостность курса с помощью функции `Consistency_rate`. Для остальных дидактических слоев алгоритмы вычисления трудоемкости строятся аналогично.

На рис. 7 представлен алгоритм вычисления трудоемкости курса в целом.

```
function Course_credit(course)
1)  credit = Course_concept_credit(course) +
         Course_open_question_credit(course) +
         Course_example_credit(course) +
         Course_exercise_credit(course) +
         Course_close_question_credit(course) +
         Course_problem_credit(course) +
         Course_bibitem_credit(course);
2)  return credit;
end function;
```

Рис. 7. Общая трудоемкость курса

4.3. Оценка сбалансированности курса

Величина дисбаланса курса, вычисляемая по формуле (9), определяет меру разброса трудоемкостей модулей и является одним из критериев оценки качества разработанного курса. Если все модули курса имеют примерно одинаковую трудоемкость, то величина дисбаланса близка к нулю. Данная величина вычисляется на основе средней трудоемкости модулей курса, для вычисления которой используется функция вычисления трудоемкости модуля, представленная на рис. 8.

```
function Module_credit(module)
1) credit = CONCEPT_CREDIT(module) +
         OPEN_QUESTION_CREDIT(module) +
         EXAMPLE_CREDIT(module) +
         EXERCISE_CREDIT(module) +
         CLOSE_QUESTION_CREDIT(module) +
         PROBLEM_CREDIT(module) +
         BIBITEM_CREDIT(module);
2) return credit;
end function;
```

Рис. 8. Трудоемкость модуля

На рис. 9 представлен алгоритм вычисления средней трудоемкости модулей для поддерева с корнем `root` граф-плана курса.

```
function Average_credit(root)
1)  amount = Amount_of_nodes(root);
2)  credit = Subtree_concept_credit(root) +
         Subtree_open_question_credit(root) +
         Subtree_example_credit(root) +
         Subtree_exercise_credit(root) +
         Subtree_close_question_credit(root) +
         Subtree_problem_credit(root) +
         Subtree_bibitem_credit(root);
3)  return credit/amount;
end function;
```

Рис. 9. Средняя трудоемкость модулей для поддерева с корнем `root`

Для корректной работы функции `Average_credit` предварительно необходимо проверить целостность курса с помощью функции `Consistency_rate`.

На рис. 10 представлена реализация функции, вычисляющей для поддерева с корнем `root` суммарное отклонение значений трудоемкости модулей от указанного значения. Данная функция используется в реализации функции вычисления дисбаланса курса, представленной на рис. 11.

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. В строке 2 переменной `total_deviation` присваивается начальное значение — отклонение трудоемкости текущего модуля от указанного значения. В строках 3–9 рекурсивно вычисляется сумма отклонений трудоемкости модулей, ассоциированных с дочерними вершинами текущего узла `root`.

```
function Sum_of_deviations(root, value)
1)  root_module = GET_LINK(root);
2)  total_deviation = abs(value – Module_credit(root_module));
3)  RESET_CHILD(root);
4)  child = FETCH_CHILD(root);
5)  while (child ≠ Null))
6)       total_deviation += Sum_of_deviations(child, value);
7)       NEXT_CHILD(root);
8)       child = FETCH_CHILD(root);
9)  end while;
10) return total_deviation;
end function;
```

Рис. 10. Суммарное отклонение трудоемкостей модулей от значения value для поддерева с корнем root

```
function Course_imbalance(course)
1)  root = GRAPH_PLAN(course);
2)  average = Average_credit(root);
3)  amount = Amount_of_nodes(root)
4)  return Sum_of_deviations(root, average) / amount;
end function;
```

Рис. 11. Оценка дисбаланса курса

4.4. Соответствие образовательной программы стандарту

Образовательная программа *соответствует* стандарту, если выполняются следующие два условия:

1) *ограничения на трудоемкость*: трудоемкость освоения учебных дисциплин и трудоемкость практик образовательной программы должны быть не меньше соответствующих минимальных значений, указанных в стандарте, а сумма этих значений (с учетом трудоемкости итоговой аттестации) должна быть равна общему значению трудоемкости образовательной программы, указанной в стандарте;
2) *покрытие компетенций*: все универсальные и общепрофессиональные компетенции, указанные в стандарте, покрываются курсами образовательной программы.

Алгоритм проверки ограничений на трудоемкость представлен на рис. 12.

В строке 1 вычисляется ссылка на стандарт, на основе которого создана текущая образовательная программа. В строках 2 и 3 переменным study и practice присваиваются начальные значения. В строках 4–15 вычисляются трудоемкости учебных дисциплин курса и практики соответственно. В строках 16–18 осуществляется проверка ограничений на трудоемкость.

Проверка покрытия компетенций распадается на две подзадачи: 1) проверка покрытия универсальных компетенций; 2) проверка покрытия общепрофессиональных компетенций. Для решения первой подзадачи используется рекурсивная функция Course_UC_credit(root, uc_number), реализация которой приведена на рис. 13. Эта функция вычисляет в поддереве с корнем root суммарную трудоемкость модулей, связанных с узлами, в которых присутствует номер универсальной компетенции uc_number.

```
function Credit_constraints(edu_prog)
1)  standart = GET_STANDART(edu_prog);
2)  study = 0;
3)  practice = 0;
4)  RESET_COURSE(edu_prog);
5)  course = FETCH_COURSE(edu_prog);
6)  while (course ≠ Null))
7)       practice += Course_problem_credit(course);
8)       study += Course_concept_credit(course);
9)       study += Course_open_question_credit(course);
10)      study += Course_example_credit(course);
11)      study += Course_exercise_credit(course);
12)      study += Course_close_question_credit(course);
13)      NEXT_COURSE(edu_prog);
14)      course = FETCH_COURSE(edu_prog);
15) end while;
16) if ((study ≥ GET_MIN_STUD(standart))
    and (practice ≥ GET_MIN_PRACT(standart))
    and (study + practice ≤ GET_TOTAL(standart) -
    GET_MIN_CRT(standart))
    and (study + practice ≥ GET_TOTAL(standart) -
    GET_MAX_CRT(standart))) then
17)      return true;
18) end if;
19) return false;
end function;
```

Рис. 12. Проверка ограничений на трудоемкость

```
function Course_UC_credit(root, uc_number)
1)  module = GET_LINK(root);
2)  if (UC_ISEXIST(root, uc_number)) then
3)       credit = Module_credit(module);
4)  else
5)       credit = 0;
6)  end if;
7)  RESET_CHILD(root);
8)  child = FETCH_CHILD(root);
9)  while (child ≠ Null)
10)      credit += Course_UC_credit(child, uc_number);
11)      NEXT_CHILD (root);
12)      child = FETCH_CHILD(root);
13) end while;
14) return credit;
end function;
```

Рис. 13. Покрытие универсальной компетенции uc_number в поддереве с корнем root

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом графплана. В строке 2 с помощью операции UC_ISEXIST осуществляется проверка присутствия универсальной компетенции uc_number в узле root. В зависимости от результата этой проверки, переменной credit присваивается общая трудоемкость соответствующего модуля либо ноль. В строках 7–13 рекурсивно вычисляется суммарная трудоемкость модулей, связанных с узлами, в которых присутствует универсальная компетенция uc_number.

Аналогичным образом реализуется функция Course_GPC_credit, вычисляющая трудоемкость для общепрофессиональной компетенции.

На рис. 14 представлена реализация функции проверки покрытия универсальных компетенций образовательного стандарта курсами образовательной программы.

```
function UC_coverage(edu_prog)
1)  standart = GET_STANDART(edu-prog);
2)  RESET_UC(standart);
3)  number = UC_GET_NUMBER(standart);
4)  while (number ≠ Null)
5)       credit = 0;
6)       RESET_COURSE(edu_prog);
7)       course = FETCH_COURSE(edu_prog);
8)       while (course ≠ Null))
9)            root = GRAPH_PLAN(course);
10)           credit += Course_UC_credit(root, number);
11)           NEXT_COURSE(edu_prog);
12)           course = FETCH_COURSE(edu_prog);
13)      end while;
14)      if (credit == 0) then
15)           return false;
16)      end if;
17)      NEXT_UC(standart);
18) end while;
19) return true;
end function;
```

Рис. 14. Проверка покрытия универсальных компетенций

В строке 1 вычисляется ссылка на стандарт, на основе которого реализована образовательная программа. В строках 2–18 для каждой универсальной компетенции стандарта вычисляется покрытие компетенции курсами образовательной программы. В строке 5 переменной credit присваивается начальное значение. В строках 6–13 вычисляется суммарное покрытие универсальной компетенции с номером number всеми курсами образовательной программы. Если суммарное покрытие универсальной компетенции с номером number равно нулю, тогда функция UC_coverage завершает свою работу с результатом false (строки 14–16). В строке 19 функция UC_coverage завершает свою работу с результатом true.

Реализация функции проверки покрытия общепрофессиональных компетенций реализуется аналогичным образом.

На рис. 15 представлена реализация функции оценки соответствия образовательной программы стандарту.

```
function Conformance_to_standart(edu_prog)
1)  if (Credit_constraints(edu_prog) == true)
and (UC_coverage(edu_prog) == true)
and (GPC_coverage(edu_prog) == true) then
2)       return true;
3)  else
4)       return true;
5)  end if;
end function;
```

Рис. 15. Соответствие образовательной программы стандарту

В строке 1 осуществляется проверка ограничений на трудоемкость и покрытия компетенций стандарта. При успешной проверке функция Conformance_to_standart завершает свою работу с результатом true (строка 2), иначе функция Conformance_to_standart завершает свою работу с результатом false (строка 4).

4.5. Целостность образовательной программы

Образовательная программа называется *целостностной*, если выполняются следующие условия:

1) все курсы образовательной программы являются целостными;
2) образовательная программа соответствует стандарту;
3) профессиональные компетенции образовательной программы покрываются курсами образовательной программы.

Функция вычисления целостности курса Consistency_rate(course) представлена в п. 4,1, функция вычисления соответствия образовательной программы стандарту Conformance_to_standart(edu_prog) представлена в п. 4,4. Алгоритм вычисления покрытия профессиональных компетенций курсами образовательной программы основан на рекурсивном алгоритме, реализация которого представлена на рис. 16.

```
function Course_PC_credit(root, pc_number)
1)  module = GET_LINK(root);
2)  if (PC_ISEXIST(root, pc_number)) then
3)       credit = Module_credit(module);
4)  else
5)       credit = 0;
6)  end if;
7)  RESET_CHILD(root);
8)  child = FETCH_CHILD(root);
9)  while (child ≠ Null)
10)      credit += Course_PC_credit(child, pc_number);
11)      NEXT_CHILD (root);
12)      child = FETCH_CHILD(root);
13) end while;
14) return credit;
end function;
```

Рис. 16. Покрытие профессиональной компетенции pc_number в поддереве с корнем root

В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом графплана. В строке 2 с помощью операции PC_ISEXIST осуществляется проверка присутствия профессиональной компетенции pc_number в узле root. В зависимости от результата этой проверки, переменной credit присваивается общая трудоемкость соответствующего модуля либо ноль. В строках 7–13 рекурсивно вычисляется суммарная трудоемкость модулей, связанных с узлами, в которых присутствует профессиональная компетенция pc_number.

На рис. 17 представлена реализация функции проверки покрытия профессиональных компетенций курсами образовательной программы.

В строках 1–17 для каждой профессиональной компетенции вычисляется покрытие компетенции курсами образовательной программы. В строке 4 переменной credit присваивается начальное значение. В строках 5–12 вычисляется суммарное покрытие профессиональной компетенции с номером number всеми курсами образовательной программы. Если суммарное покрытие профессиональной компетенции с номером number равно нулю,

```
function PC_coverage(edu_prog)
1)  RESET_PC(edu_prog);
2)  number = PC_GET_NUMBER(edu_prog);
3)  while (number ≠ Null)
4)       credit = 0;
5)       RESET_COURSE(edu_prog);
6)       course = FETCH_COURSE(edu_prog);
7)       while (course ≠ Null))
8)            root = GRAPH_PLAN(course);
9)            credit += Course_PC_credit(root, number);
10)           NEXT_COURSE(edu_prog);
11)           course = FETCH_COURSE(edu_prog);
12)      end while;
13)      if (credit == 0) then
14)           return false;
15)      end if;
16)      NEXT_PC(edu_prog);
17) end while;
18) return true;
end function;
```

Рис. 17. Покрытие профессиональных компетенций курсами образовательной программы

тогда функция PC_coverage завершает свою работу с результатом false (строки 13–15). В строке 18 функция PC_coverage завершает свою работу с результатом true.

Реализация функции вычисления целостности образовательной программы представлена на рис. 18.

```
function Consistency(edu_prog)
1)  RESET_COURSE(edu_prog);
2)  course = FETCH_COURSE(edu_prog);
3)  while (course ≠ Null))
4)       if (Consistency_rate(course) ≠ 1) then
5)            return false;
6)       end if;
7)       NEXT_COURSE(edu_prog);
8)       course = FETCH_COURSE(edu_prog);
9)  end while;
10) if (Correctness_eduprog(edu_prog) == true))
and (PC_coverage(edu_prog) == true) then
11)      return true;
12) else
13)      return false;
14) end if;
end function;
```

Рис. 18. Целостность образовательной программы

В строках 1–9 производится оценка целостности всех курсов образовательной программы. Если коэффициент целостности хотя бы одного курса отличается от 1, функция Consistency завершает свою работу с результатом `false`. В строках 10–14 осуществляется проверка соответствия образовательной программы стандарту и покрытия профессиональных компетенций образовательной программы курсами образовательной программы. В зависимости от этой проверки функция Consistency завершает свою работу с результатом `false` или `true`.

Заключение

В статье определены требования к модели дидактической структуры контента. Представлено формальное описание структурно-иерархической дидактической модели электронного обучения, отличительной особенностью которой является поддержка деления образовательных объектов на дидактические компоненты. Определены основные операции СИД модели, на основе которых построены алгоритмы анализа качественных характеристик образовательных программ и электронных учебных курсов. В рамках дальнейших исследований авторы предполагают разработать прототип программной системы для создания электронных учебных курсов на базе СИД модели, с помощью которой выполнить верификацию СИД модели.

Исследование выполнено при финансовой поддержке Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

Литература

1. Скиннер Б.Ф. Наука об учении и искусство обучения // Программированное обучение за рубежом: Сб. статей / Под ред. И.И. Тихонова. М.: Высшая школа. 1968. С. 32–46
2. Краудер Н.А. О различиях между линейным и разветвленным программированием // Программированное обучение за рубежом: Сб. статей / Под ред. И.И. Тихонова. М.: Высшая школа. 1968. С. 58–67.
3. Engelbart D.C. Toward Augmenting the Human Intellect and Boosting our Collective IQ // Communications of the ACM. 1995. Vol. 38, no. 8. P. 30–33. DOI: 10.1145/208344.208352.
4. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
5. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
6. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
7. Novak J.D., Canas A.J. The theory underlying concept maps and how to construct them. Technical Report IHMC CmapTools 2006–01 Rev 2008–01. Florida Institute for Human and Machine Cognition, USA. 2008. URL: http://cmap.ihmc.us/docs/theory-of-concept-maps (дата обращения: 10.07.2019).
8. Соловов А.В. Математические модели содержания и процессов электронного обучения // Телекоммуникации и информатизация образования. 2006. № 4. С. 20–37.
9. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
10. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms // Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007 (Oct., 29–31, 2007). Vol. 2. P. 111–116. DOI: 10.1109/ICTAI.2007.14.
11. IEEE 1484.20.1. Standard for Learning Technology — Data Model for Reusable Competency Definitions. 2007.

12. Kristensen T., Lamo Y., Hinna K.R., Hole G.O. Dynamic Content Manager — A New Conceptual Model for E-Learning // Proceedings of the International Conference on Web Information Systems and Mining, WISM '09. Springer-Verlag, Berlin, Heidelberg, 2009. P. 499–507. DOI: 10.1007/978-3-642-05250-7_52.
13. Брызгалов П.А. Система «Ареола» — программная оболочка для создания электронных энциклопедий // Вычислительные методы и программирование. 2005. Т. 6, № 2. С. 22–26.
14. Брызгалов П.А,. Воеводин В.В., Воеводин Вл.В. О некоторых проблемах компьютеризации знаний // Научный сервис в сети Интернет: Тезисы докладов Всероссийск. науч. конф. (Новороссийск, 18–23 сентября 2000 г.). М.: Изд-во МГУ, 2000.
15. Cunningham W., Leuf B. The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Professional, 2001. 464 p.
16. Курганская Г.С. Модель представления знаний и система дифференцированного обучения через Интернет на его основе // Известия Челябинского Научного Центра. 2000. Вып. 2. С. 84–88.
17. Курганская Г.С. Облачные технологии интернет-образования на основе KFS модели представления знаний // Вестник Бурятского государственного университета. 2013. № 9. С. 69–75.
18. Курганская Г.С. Развитие методик адаптивного обучения: опыт применения системы «ГЕКАДЕМ» // Интернет и современное общество: Труды XI Всероссийской объединенной конференции IMS–2008 (Санкт-Петербург, 28–30 октября 2008 г.). СПб.: Изд-во С.-Петерб. ун-та. 2008. С. 65–67.
19. Соловов А.В. Математическое моделирование содержания, навигации и процессов электронного обучения в контексте международных стандартов и спецификаций. Лекция-доклад // Труды Всероссийской научно-практической конференции с международным участием «Информационные технологии в обеспечении нового качества высшего образования» (Москва, НИТУ «МИСиС», 14–15 апреля 2010 г.). М.: Исследовательский центр проблем качества подготовки специалистов, 2010. 52 с.
20. Соловов А.В. Проектирование компьютерных систем учебного назначения. Гриф «Рекомендовано Госкомитетом по высшему образованию РФ к изданию». Самара: СГАУ, 1995. 138 с.
21. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms // Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence ICTAI 2007 (Oct., 29–31, 2007). Vol. 2. P. 111–116. DOI: 10.1109/ICTAI.2007.14.
22. Калитина В.В. Электронная энциклопедия как средство повышения уровня запоминания учебного материала // Вестник Красноярского государственного педагогического университета им. В.П. Астафьева. 2013. № 1 (23). С. 111–114.
23. Воеводин В.В. Открытая энциклопедия свойств алгоритмов AlgoWiki: от мобильных платформ до экзафлопсных суперкомпьютерных систем // Вычислительные методы и программирование: новые вычислительные технологии. 2015. Т. 16, № 1. С. 99–111. DOI: 10.26089/NumMet.v16r111.
24. Voevodin V., Antonov A., Dongarra Ja. AlgoWiki: an open encyclopedia of parallel algorithmic features // Supercomputing Frontiers and Innovations. 2015. Vol. 2, no. 1. P. 4–18. DOI: 10.14529/jsfi150101.

25. Силкина Н.С., Соколинский Л.Б. Обзор адаптивных моделей электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 4. С. 61–76. DOI: 10.14529/cmse160405.
26. Силкина Н.С., Соколинский Л.Б. Модели и стандарты электронного обучения // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. Т. 3, № 4. С. 5–35. DOI: 10.14529/cmse140401.
27. Приказ Минобрнауки России «Об утверждении порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования — программам бакалавриата, программам специалитета, программам магистратуры» от 19 декабря 2013 года № 1367. URL: http://минобрнауки.рф/документы/5242 (дата обращения: 10.07.2019).
28. Новгородцева И.В. Педагогика с методикой преподавания специальных дисциплин. М. : ФЛИНТА, 2011. 378 с. URL: http://e.lanbook.com/book/2440.
29. Национальная педагогическая энциклопедия. URL: http://didacts.ru/.
30. Рудской А.И., Боровков А.И., Романов П.И., Колосова О.В. Общепрофессиональные компетенции современного российского инженера // Высшее образование в России. 2018. № 2. С. 5–18. URL: https://vovr.elpub.ru/jour/article/download/1267/1072 (дата обращения: 10.07.2019).
31. Сергеев А.Г. Компетентность и компетенции: монография. Владимир: Изд-во Владим. гос. ун-та, 2010. 107 с.
32. Berners-Lee T. Uniform Resource Locators “URL”: A Syntax for the Expression of Access Information of Objects on the Network. World Wide Web Consortium. 1994. URL: https://www.w3.org/Addressing/URL/url-spec.txt (дата обращения: 22.05.2019).
33. 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: 10.07.2019).
34. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.

Силкина Надежда Сергеевна, старший преподаватель, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинский Леонид Борисович, д.ф.-м.н., профессор, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190405

STRUCTURAL-HIERARCHICAL DIDACTIC MODEL OF E-LEARNING

© 2019 N.S. Silkina, L.B. Sokolinsky

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: SilkinaNS@susu.ru, Leonid.Sokolinsky@susu.ru

Received: 22.10.2019

This article describes the original Structural-Hierarchical Didactic (SHD) model of e-learning. The model is based on a four-level methodological knowledge base. The first level includes a set of electronic educational encyclopedias in various fields of knowledge. The second level includes e-learning courses. The model supports the structuring of e-learning course on didactic components (vertical structuring) and levels of detail (horizontal structuring). The third level includes a set of syllabuses. The fourth level includes a set of a federal state educational standard of higher education (FSES HE). A distinctive feature of the SHD model is the division of educational objects into didactic components. This will allow, firstly, to automatically verify the didactic completeness of the electronic training course. Secondly, to include parts of one course in another without losing the didactic structure. Thirdly, to select a separate didactic layer from the electronic training course and, based on it, automatically generate specialized training materials: lecture notes, task collections, exam tests, etc. The basic operations of the SHD model are described, based on which analysis algorithms are proposed syllabuses and e-learning courses. In conclusion, a brief summary of the results and directions for further research is given.

Keywords: e-learning, course, e-learning model, learning object, didactic structure, SHD model.

FOR CITATION

Silkina N.S., Sokolinsky L.B. Structural-Hierarchical Didactic Model of E-Learning. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 56–83. (in Russian) DOI: 10.14529/cmse190405.

References

1. Skinner B.F. The science of learning and the art of teaching. Programmed study abroad: Col. of Articles / Ed. by I.I. Tikhonov. M.: Higher school. 1968. pp. 32–46. (in Russian)
2. Crowder N.A. On the differences between linear and branched programming. Programmed study abroad: Col. of Articles / Ed. by I.I. Tikhonov. M.: Higher school. 1968. pp. 58–67. (in Russian)
3. Engelbart D.C. Toward Augmenting the Human Intellect and Boosting our Collective IQ. Communications of the ACM. 1995. vol. 38, no. 8. pp. 30–33. DOI: 10.1145/208344.208352.
4. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
5. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
6. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
7. Novak J.D., Canas A.J. The theory underlying concept maps and how to construct them. Technical Report IHMC CmapTools 2006–01 Rev 2008–01. Florida Institute for Human

and Machine Cognition, USA. 2008. Available at: http://cmap.ihmc.us/docs/theory-of-concept-maps (accessed: 10.07.2019).
8. Solovov A.V. Mathematical models of the content and processes of e-learning. Telecommunications and informatization of education. 2006. no. 4. pp. 20–37. (in Russian)
9. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
10. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms. Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence ICTAI 2007 (Oct., 29–31, 2007). vol. 2. pp. 111–116. DOI: 10.1109/ICTAI.2007.14.
11. IEEE 1484.20.1. Standard for Learning Technology — Data Model for Reusable Competency Definitions. 2007.
12. Kristensen T., Lamo Y., Hinna K.R., Hole G.O. Dynamic Content Manager — A New Conceptual Model for E-Learning. Proceedings of the International Conference on Web Information Systems and Mining, WISM '09. Springer-Verlag, Berlin, Heidelberg, 2009. pp. 499–507. DOI: 10.1007/978-3-642-05250-7_52.
13. Bryzgalov P.A. System "Areola" — a software shell for creating electronic encyclopedias. Computational methods and programming. 2005. vol. 6, no. 2. pp. 22–26. (in Russian)
14. Bryzgalov P.A. Voevodin V.V., Voevodin Vl.V. On some problems of computerization of knowledge. Scientific service on the Internet: Proceedings of the the All-Russian Scientific Conference (Novorossiysk, September, 18, 2000). M.: Publishing House of Moscow State University, 2000. (in Russian)
15. Cunningham W., Leuf B. The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Professional, 2001. 464 p.
16. Kurganskaya G.S. A knowledge representation model and a system of differentiated learning through the Internet. Bulletin of the Chelyabinsk Scientific Center. 2000. no. 2. pp. 84–88. (in Russian)
17. Kurganskaya G.S. Cloud technologies of Internet education based on the KFS knowledge representation model. Bulletin of the Buryat State University. 2013. no. 9. pp. 69–75. (in Russian)
18. Kurganskaya G.S. The development of adaptive learning methods: the experience of using HECADEM. Internet and modern society: Proceedings of the XI All-Russian Joint Conference IMS–2008 (St. Petersburg, October, 28–30, 2008). SPb.: Publishing House of Sankt Petersburg University. 2008. pp. 65–67. (in Russian)
19. Solovov A.V. Mathematical modeling of the content, navigation and e-learning processes in the context of international standards and specifications. Lecture report. Proceedings of the All-Russian Scientific and Practical Conference with International Participation "Information Technologies in Ensuring a New Quality of Higher Education" (Moscow, NUST "MISiS", April, 14–15, 2010). M.: Research Center for the Problems of Quality of Training of Specialists, 2010. 52 p. (in Russian)
20. Solovov A.V. Design of computer systems for educational purposes. Fingerboard "Recommended by the State Committee for Higher Education of the Russian Federation for publication". Samara: SSAU, 1995. 138 p. (in Russian)
21. De-Marcos L., Pages C., Martinez J.J., Gutierrez J.A. Competency-Based Learning Object Sequencing Using Particle Swarms. Proceedings of 19th IEEE International Conference on

Tools with Artificial Intelligence, ICTAI 2007 (Oct., 29–31, 2007). vol. 2. pp. 111–116. DOI: 10.1109/ICTAI.2007.14.

22. Kalitina V.V. Electronic encyclopedia as a tool of increasing the level of memorization of educational material. Bulletin of the Krasnoyarsk State Pedagogical University named after V.P. Astafieva. 2013. no. 1 (23). pp. 111–114. (in Russian)
23. Voevodin V.V. An open encyclopedia of the properties of algorithms AlgoWiki: from mobile platforms to exaflops supercomputer systems. Computational methods and programming: new computing technologies. 2015. vol. 16, no. 1. pp. 99–111. (in Russian) DOI: 10.26089/NumMet.v16r111.
24. Voevodin V., Antonov A., Dongarra Ja. AlgoWiki: an open encyclopedia of parallel algorithmic features. Supercomputing Frontiers and Innovations. 2015. vol. 2, no 1. pp. 4–18. DOI: 10.14529/jsfi150101.
25. Silkina N.S., Sokolinsky L.B. Survey of Adaptive E-learning Models. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 4. pp. 61–76. (in Russian) DOI: 10.14529/cmse160405.
26. Silkina N.S., Sokolinsky L.B. E-learning models and standards. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. vol. 3, no. 4. pp. 5–35. (in Russian) DOI: 10.14529/cmse140401.
27. Order of the Ministry of Education and Science of Russia "On the approval of the organization and implementation of educational activities for educational programs of higher education — undergraduate programs, specialty programs, master's programs" dated December 19, 2013. no. 1367. (in Russian)
28. Novgorodtseva I.V. Pedagogy with the methodology of teaching special disciplines. M.: FLINT, 2011. 378 p. (in Russian)
29. National pedagogical encyclopedia. Available at: http://didacts.ru/ (accessed: 10.07.2019). (in Russian)
30. Rudskoy A.I., Borovkov A.I., Romanov P.I., Kolosova O.V. General professional competences of a modern Russian engineer. Higher education in Russia, 2018. no. 2. pp. 5–18. Available at: https://vovr.elpub.ru/jour/article/download/1267/1072 (accessed: 10.07.2019). (in Russian)
31. Sergeev A.G. Competence and competencies: monograph. Vladimir: Publishing house Vladimir State University, 2010. 107 p. (in Russian)
32. Berners-Lee T. Uniform Resource Locators "URL": A Syntax for the Expression of Access Information of Objects on the Network. World Wide Web Consortium. 1994. Available at: https://www.w3.org/Addressing/URL/url-spec.txt (accessed: 10.07.2019).
33. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. Available at: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (accessed: 10.07.2019).
34. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.