На правах рукописи

СИЛКИНА Надежда Сергеевна

МЕТОДЫ ОРГАНИЗАЦИИ СИСТЕМ ЭЛЕКТРОННОГО ОБУЧЕНИЯ НА ОСНОВЕ СТРУКТУРНО-ИЕРАРХИЧЕСКОГО ПОДХОДА

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Челябинск – 2019

Работа выполнена на кафедре системного программирования
ФГАОУ ВО «Южно-Уральский государственный университет
(национальный исследовательский университет)»

Научный руководитель: СОКОЛИНСКИЙ Леонид Борисович,
доктор физ.-мат. наук, профессор,
проректор по информатизации ФГАОУ ВО
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
(г. Челябинск)

Официальные оппоненты: ЗЫКИНА Анна Владимировна,
доктор физ.-мат. наук, профессор,
заведующий кафедрой «Прикладная математика
и фундаментальная информатика» ФГБОУ ВО
«Омский государственный технический
университет» (г. Омск)

УСТИНОВ Владимир Алексеевич,
кандидат физ.-мат. наук,
директор Управления корпоративного
ИТ-обучения и инноваций ФГАОУ ВО
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
(г. Екатеринбург)

Ведущая организация: ФГБОУ ВО «Оренбургский государственный
университет» (г. Оренбург)

Защита состоится 12 февраля 2020 г. в 11:00 часов на заседании диссертационного совета Д 212.298.18 при ФГАОУ ВО «Южно-Уральский государственный университет (национальный исследовательский университет)» по адресу: 454080, г. Челябинск, пр. Ленина, 76, ауд. 1001.

С диссертацией можно ознакомиться в библиотеке Южно-Уральского государственного университета и на сайте:
https://www.susu.ru/ru/dissertation/d-21229818/silkina-nadezhda-sergeevna.

Автореферат разослан «___» декабря 2019 г.

Ученый секретарь
диссертационного совета М.Л. Цымблер

Общая характеристика работы

Актуальность темы. В настоящее время наблюдается динамичное развитие электронного образования. Сегодня образовательные учреждения предлагают сотни тысяч различных интернет-курсов. На начальном этапе развития электронного обучения достаточно было эффективно представить учебный материал, используя современные средства информационных технологий. В настоящий момент на передний план выходит комплексный, системный подход, результатом которого должно явится формирование единой информационной среды образования, охватывающей весь учебный процесс.

В контексте формирования единой информационной среды образования становится очень важной возможность передачи от одного участника образовательного процесса другому не только целых интернет-курсов, но и их отдельных частей. Возможность такого обмена позволит образовательным учреждениям создавать единые банки знаний и максимально эффективно многократно использовать их в своей работе. Международные стандартизирующие организации уделяют этому вопросу большое внимание, одним из базовых стандартов на структуру и представление элементов контента электронных учебных курсов (ЭУК) является международный стандарт SCORM[1], обеспечивающий возможность переноса элементов контента из одного ЭУК в другой на физическом уровне. Однако, до сих пор отсутствуют стандарты, определяющие принципы формирования дидактической структуры ЭУК. Это ограничивает возможность переноса удачных дидактических блоков между различными ЭУК. Системы, реализованные на основе стандарта SCORM, позволяют переносить части, из которых состоят такие курсы. Таким образом может быть разрушена дидактическая структура переносимого блока. Отсутствие подобного стандарта препятствует получению максимального эффекта при внедрении электронного образования в высшей школе.

В соответствии с вышеизложенным является *актуальной* задача разработки модели, методов и алгоритмов электронного обучения, поддерживающих дидактическую структуру образовательного контента.

Цель и задачи исследования. Цель диссертационной работы состояла в разработке новой модели электронного обучения, позволяющей фиксировать дидактическую структуру учебного материала, осуществлять контроль дидактической полноты электронного учебного курса, эффективно реализовывать образовательную программу в рамках электронного обучения и проверять ее на соответствие образовательному стандарту, а также переносить учебный материал из одного электронного курса в другой с сохранением его дидактической структуры. Для достижения этой цели необходимо было решить следующие задачи.

1. Разработать структурно-иерархическую дидактическую (СИД) модель электронного обучения.

[1] SCORM 2004 4th Edition. Advanced Distributed Learning (ADL) Initiative. 2009.

2. Разработать операции СИД модели над электронными энциклопедиями, образовательными программами и курсами.
3. Разработать алгоритмы анализа качественных характеристик образовательных программ и электронных учебных курсов с использованием операций СИД модели.
4. Разработать программную поддержку СИД модели.

Методы исследования. Методологической основой диссертационного исследования являются теория множеств и теория графов. При проектировании и реализации программного обеспечения применялись методы объектно-ориентированного проектирования и язык UML.

Научная новизна. Научная новизна работы заключается в том, что впервые разработана структурно-иерархическая дидактическая модель электронного обучения, позволяющая осуществлять автоматический контроль качества электронных учебных курсов и переносить образовательный контент из одного курса в другой с сохранением дидактической структуры.

Теоретическая ценность работы состоит в том, что в ней предложено формальное описание структурно-иерархической дидактической (СИД) модели электронного обучения и даны формализованные описания основных операций над элементами СИД модели.

Практическая ценность работы заключается в том, что была реализована программная поддержка СИД модели, а именно была реализована библиотека операций СИД модели, реализованы алгоритмы анализа образовательных курсов и образовательных программ, реализован прототип программной системы ECoD (Electronic Course Designer), предоставляющий интерфейс для создания электронных учебных курсов.

Степень достоверности результатов. Для предложенной СИД модели дано формальное математическое описание с использованием теории множеств и теории графов. СИД модель и все предложенные алгоритмы реализованы на языке PHP с использованием СУБД MySQL. Эффективность СИД модели подтверждена проведенными экспериментами.

Апробация результатов исследования. Основные положения диссертационной работы, разработанные модель, методы и алгоритмы докладывались автором на следующих международных и всероссийских научных конференциях:

– Международная конференция «40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO'2017» (22–26 мая 2017 г., Хорватия, г. Опатия);
– Международная научно-практическая конференция «Новые информационные технологии в образовании» (24–27 февраля 2009 г., г. Екатеринбург);
– Всероссийская научно-методическая конференция «Инновационные технологии обучения: проблемы и перспективы» (29–30 марта 2008 г., г. Липецк);

– Международная научно-практическая конференция «Новые информационные технологии в образовании» (26–28 февраля 2008 г., г. Екатеринбург).

Публикации. По теме диссертации опубликованы 6 печатных работ. Работы [1–5] опубликованы в журналах, включенных ВАК в перечень изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора и кандидата наук. Работа [6] опубликована в издании, индексируемом в Scopus. В рамках выполнения диссертационной работы получено одно свидетельство Роспатента об официальной регистрации программ для ЭВМ и баз данных [7].

Личный вклад. В работах 1–3 научному руководителю Соколинскому Л.Б. принадлежит постановка задачи, Силкиной Н.С. – все полученные результаты. В работе 4 Евдокимовой А.С. принадлежит описание примера применения модели для обучения параллельным системам баз данных (раздел 5, стр. 95–96), Силкиной Н.С. принадлежат все остальные разделы статьи. В работе 6 Ивановой О.Н. принадлежит описание требований ФГОС ВО к уровню учебных результатов выпускников (раздел II, стр. 685), Силкиной Н.С. принадлежат все остальные разделы статьи.

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения и библиографии. В приложении 1 приведены основные обозначения, используемые в диссертации. В приложении 2 приведено описание схемы LOM. В приложении 3 приведена спецификация API среды выполнения электронного обучения. В приложении 4 приведен пример содержания модуля «Площадь квадрата» миникурса «Площадь многоугольника». Объем диссертации составляет 166 страниц, объем библиографии – 117 наименований.

Содержание работы

Во введении приводится обоснование актуальности темы и степень ее разработанности; формулируются цели и задачи исследования; раскрываются новизна, теоретическая и практическая значимость полученных результатов; формулируется методологическая основа диссертационного исследования; дается обзор содержания диссертации.

Первая глава, «Модели электронного обучения», посвящена обзору научных исследований по актуальным направлениям развития современных подходов к созданию электронных учебных курсов. Производится анализ моделей электронного обучения и промышленных стандартов в этой области.

Во второй главе, «СИД модель», дается общее и формальное описание структурно-иерархической дидактической (СИД) модели электронного обучения. Определяются основные операции 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.

Значением компоненты модуля является *коллекция*, представляющая собой список (упорядоченную последовательность) элементов, имеющих соответствующий дидактический тип. 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

$$w = \langle id_{component}, T, \xi, \lambda \rangle,$$

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 $\langle id_{encyclopedia}, M \rangle$, где $id_{encyclopedia}$ – уникальный идентификатор энциклопедии, M – коллекция (упорядоченный список) модулей энциклопедии.

Модуль μ представляет собой тройку:

$$\mu = \langle id_{module}, id_{encyclopedia}, \vec{w} \rangle,$$

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

Компетенция представляет собой четверку

$$\langle source, type, number, description \rangle.$$

Здесь *source* – уникальный идентификатор источника, содержащий перечень компетенций (в качестве такого идентификатора может выступать URL); *type* – тип компетенции с возможными значениями из множества {УК, ОПК, ПК}, где УК обозначает универсальные компетенции, ОПК – общепрофессиональные компетенции, ПК – профессиональные компетенции; *number* – порядковый номер компетенции; *description* – наименование компетенции.

Обозначим $\mathbb{K}$ – множество всех возможных компетенций, $\mathbb{K}_{univer}$ – подмножество универсальных компетенций, $\mathbb{K}_{genprof}$ – подмножество общепрофес-

сиональных компетенций и $\mathbb{K}_{prof}$ – подмножество профессиональных компетенций. Множество компетенций обладает следующими свойствами:

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

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

Образовательный стандарт S представляет собой семерку

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

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.

Образовательная программа Q представляет собой четверку

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

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:

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

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, $G = \langle V, E \rangle$ – граф-план курса (см. определение ниже); $f : V \to R$ – однозначное инъективное отображение, сопоставляющее каждой вершине $v \in V$ граф-плана G некоторый модуль $\mu \in R$: $f(v) = \mu$.

Для задания иерархической структуры курса используется граф-план. *Граф-план* представляет собой связный ациклический граф $G = \langle V, E \rangle$ с выделенной вершиной $\bar{v}$, где $V = \{ v_l \mid l = 1,...,b \}$ – множество всех вершин граф-плана, $E = \{ e_p \mid p = 1,...,d \}$ – множество всех ребер граф-плана.

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

$$\lambda_{avg} = \frac{\sum_{\mu \in f(V)} \sum_{i=1}^{7} \mu.w_i.\lambda}{|V|}.$$

Дисбаланс курса определяется по формуле

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

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.

Основными операциями над энциклопедиями являются следующие:
CREATE_ENCYCLOPEDIA – создание энциклопедии;
GET_ENCYCLOPEDIA_<atr> – доступ к атрибуту энциклопедии;
SET_ENCYCLOPEDIA_<atr> – изменение значения атрибута энциклопедии;
FETCH_MODULE – доступ к текущему модулю энциклопедии;
INSERT_MODULE – добавление в энциклопедию нового модуля;
DELETE_MODULE – удаление модуля из энциклопедии;
<COMPONENT>_COLLECTION – доступ к коллекции элементов компоненты <COMPONENT> модуля;
<COMPONENT>_CREDIT – трудоемкость компоненты <COMPONENT> модуля.

Основными операциями над образовательными стандартами являются следующие:
CREATE_STANDART – создание образовательного стандарта;
GET_STANDART_<atr> – доступ к атрибуту образовательного стандарта;
SET_STANDART_<atr> – изменение значения атрибута образовательного стандарта;
INSERT_UC_INTO_STANDART – добавление в образовательный стандарт универсальной компетенции;
DELETE_UC_FROM_STANDART – удаление универсальной компетенции из образовательного стандарта;
UC_GET_<atr> – доступ к атрибуту универсальной компетенции образовательного стандарта;
UC_SET_<atr> – изменение значения атрибута универсальной компетенции образовательного стандарта;
INSERT_GPC_INTO_STANDART – добавление в образовательный стандарт общепрофессиональной компетенции;

DELETE_GPC_FROM_STANDART – удаление общепрофессиональной компетенции из образовательного стандарта;
GPC_GET_<atr> – доступ к атрибуту общепрофессиональной компетенции образовательного стандарта;
GPC_SET_<atr> – изменение значения атрибута общепрофессиональной компетенции образовательного стандарта.

Основными операциями над образовательными программами являются следующие:
CREATE_EDU_PROG – создание образовательной программы;
GET_STANDART – получение указателя на образовательный стандарт, на основе которого была создана образовательная программа;
INSERT_PC_INTO_EDU_PROG – добавление в образовательную программу профессиональной компетенции;
DELETE_PC_FROM_EDU_PROG – удаление профессиональной компетенции из образовательной программы;
PC_GET_<atr> – доступ к атрибуту профессиональной компетенции образовательного программы;
PC_SET_<atr> – изменение значения атрибута профессиональной компетенции образовательного программы;
INSERT_COURSE – добавление в образовательную программу курса;
DELETE_COURSE – удаление курса из образовательной программы;
FETCH_COURSE – получение указателя на курс образовательной программы;
GET_COURSE_<atr> – доступ к атрибуту курса образовательной программы;
SET_COURSE_<atr> – изменение значения атрибута курса образовательной программы.

Основными операциями над курсами и граф-планами являются следующие:
GET_ENCYCLOPEDIA – получение указателя на энциклопедию курса;
GRAPH_PLAN – доступ к корневой вершине граф-плана курса;
FETCH_CHILD – получение указателя на дочернюю вершину узла граф-плана курса;
INSERT_CHILD – создание дочерней вершины узла граф-плана курса;
DELETE_CHILD – удаление дочерней вершины узла граф-плана курса;
SET_LINK – связывание узла граф-плана курса с модулем;
GET_LINK – получение указателя на модуль, привязанный к узлу граф-плана курса;
INSERT_UC_INTO_GP – добавление номера универсальной компетенции к коллекции компетенций узла граф-плана курса;
DELETE_UC_FROM_GP – удаление номера универсальной компетенции из коллекции компетенций узла граф-плана курса;

UC_ISEXIST – проверка наличия номера универсальной компетенции в коллекции компетенций узла граф-плана курса;
INSERT_GPC_INTO_GP – добавление номера общепрофессиональной компетенции к коллекции компетенций узла граф-плана курса;
DELETE_GPC_FROM_GP – удаление номера общепрофессиональной компетенции из коллекции компетенций узла граф-плана курса;
GPC_ISEXIST – проверка наличия номера общепрофессиональной компетенции в коллекции компетенций узла граф-плана курса;
INSERT_PC_INTO_GP – добавление номера профессиональной компетенции к коллекции компетенций узла граф-плана курса;
DELETE_PC_FROM_GP – удаление номера профессиональной компетенции из коллекции компетенций узла граф-плана курса;
PC_ISEXIST – проверка наличия номера профессиональной компетенции в коллекции компетенций узла граф-плана курса.

В третьей главе, «Анализ образовательного контента», рассматриваются алгоритмы анализа образовательных программ и электронных учебных курсов, использующие основные операции СИД модели: верификация целостности курса, оценка трудоемкости и сбалансированности курса, определение соответствия образовательной программы стандарту и оценка целостность образовательной программы.

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. 1 представлена реализация рекурсивной функции вычисления количества связанных узлов поддерева с корнем root граф-плана курса. 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. 2 представлен рекурсивная псевдокод реализации функции вычисления общего количества узлов поддерева граф-плана с корнем root.

На рис. 3 представлен псевдокод реализации функции вычисления коэффициента целостности 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. 1. Количество связанных узлов поддерева 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. 2. Количество узлов поддерева с корнем root.

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

Рис. 3. Коэффициент целостности 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».

Алгоритм вычисления трудоемкости дидактического слоя в свою очередь реализуется с помощью соответствующей рекурсивной функции, вычисляющей трудоемкость слоя для поддерева граф-плана курса. На рис. 4 представлена реализация рекурсивной функции Subtree_concept_credit, вычисляющей трудоемкость слоя «Теория» для поддерева граф-плана курса. 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. 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. 5 представлен алгоритм вычисления трудоемкости слоя «теория» для всего 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. 6 представлен алгоритм вычисления трудоемкости курса в целом.

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

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

Рис. 5. Трудоемкость слоя «Теория» для 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. 6. Общая трудоемкость курса.

Величина дисбаланса курса, вычисляемая по формуле 1, 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. 7.

```
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. 7. Трудоемкость модуля.

На рис. 8 представлен алгоритм вычисления средней трудоемкости модулей для поддерева с корнем root граф-плана курса. Для корректной работы функции Average_credit предварительно необходимо проверить целостность курса с помощью функции Consistency_rate.

На рис. 9 представлен псевдокод реализации 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. 10. 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.

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.

```
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. 8. Средняя трудоемкость модулей для поддерева с корнем 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. 9. 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. 10. Оценка дисбаланса курса.

Алгоритм проверки ограничений на трудоемкость представлен на рис. 11. 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.

```
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. 11. 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. 12. Покрытие универсальной компетенции uc_number в поддереве с корнем root.

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. 12. Эта функция вычисляет в поддереве с корнем root суммарную трудоемкость модулей, связанных с узлами, в которых присутствует номер универсальной компетенции uc_number. В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. 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. 13 представлен псевдокод реализации функции проверки покрытия универсальных компетенций образовательного стандарта курсами образовательной программы. В строке 1 вычисляется ссылка на стандарт, на основе которого реализована образовательная программа. В строках 2–18 для каждой

```
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. 13. Проверка покрытия универсальных компетенций.

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. 14 представлен псевдокод реализации функции оценки соответствия 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).

```
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 false;
5)      end if;
end function;
```

Рис. 14**.** Соответствие образовательной программы стандарту.

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) описана выше, функция вычисления соответствия образовательной программы стандарту Conformance_to_standart(edu_prog) также представлена выше. Алгоритм вычисления покрытия профессиональных компетенций курсами образовательной программы основан на рекурсивном алгоритме, реализация которого представлена на рис. 15. В строке 1 вычисляется ссылка на модуль, ассоциированный с текущим узлом граф-плана. 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. 16 представлен псевдокод реализации 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, 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.

```
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. 15. Покрытие профессиональной компетенции pc_number в поддереве с корнем root.

```
function PC_coverage(edu_prog)
1)      RESET_PC(edu_prog);
2)      number = GET_PC_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. 16. Покрытие профессиональных компетенций курсами образовательной программы.

```
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. 17. Целостность образовательной программы.

Реализация функции вычисления целостности образовательной программы представлена на рис. 17. 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.

В четвертой главе, «Программная поддержка СИД модели», описывается процесс проектирования и реализации программной системы ECoD (Electronic Course Designer), предназначенной для создания электронных учебных курсов и обеспечивающей их использование в системах управления обучением, поддерживающих стандарт SCORM. Описываются эксперименты по применению системы ECoD.

Диаграмма прецедентов системы ECoD представлена на рис. 18. Программная система ECoD включает в себя восемь компонент. Компонент «Менеджер» программной системы ECoD реализует операции СИД модели и обеспечивает взаимодействие с базой данных. Компонент «СИД анализ» предоставляет пользователю интерфейс для выполнения операций для анализа образовательных программ и электронных учебных курсов. Компоненты «Редактор энциклопедий», «Редактор компетенций», «Редактор стандартов», «Редактор

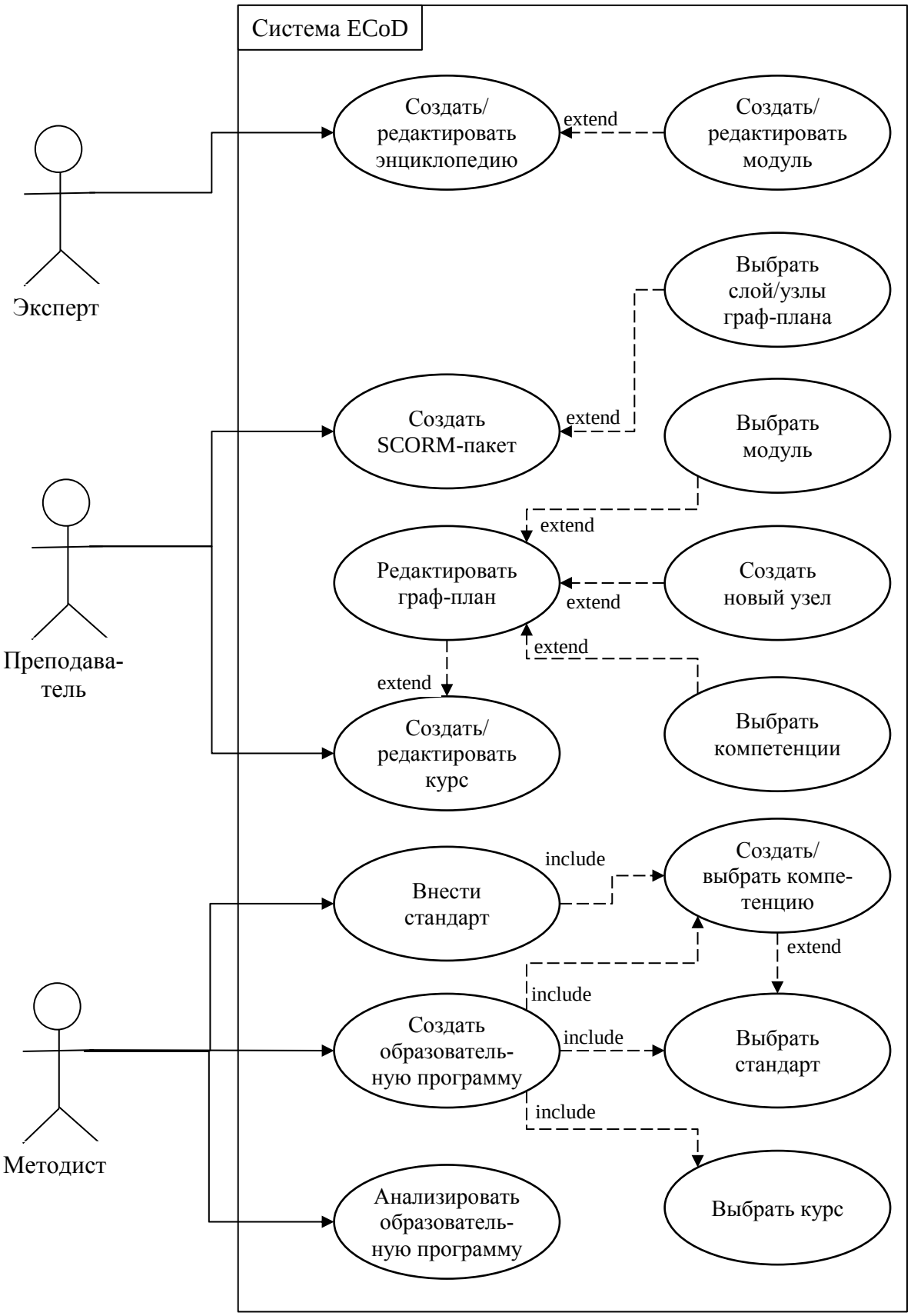

Рис. 18. Диаграмма прецедентов системы ECoD.

образовательных программ» и «Редактор курсов» предоставляют пользователю интерфейс для выполнения операций с соответствующими сущностями СИД модели. Компонент «Генератор SCORM-пакета» предоставляет пользователю интерфейс для генерации SCORM-пакета, который может быть запущен в любой системе управления обучением, поддерживающий данный стандарт. Программная система ECoD была реализована на языке PHP с использованием СУБД MySQL, исходные тексты свободно доступны в сети Интернет по адресу: https://github.com/zhnadya81/ECoD.

Было проведено два эксперимента по применению системы ECoD. В качестве первого эксперимента было выполнено создание миникурса «Площадь многоугольника». Целью эксперимента являлась проверка работоспособности сгенерированного пакета SCORM в системе дистанционного образования «Электронный ЮУрГУ», созданной на базе системы управления обучением MOODLE. Миникурс «Площадь многоугольника» в формате пакета SCORM был апробирован на студентах первого. В ходе апробации студенты освоили теоретический материал, выполнили тест, содержащий контрольные вопросы для проверки усвоения материала, изучили примеры решения задач, выполнили контрольные задания и прошли итоговое тестирование. При работе с миникурсом ошибок в работе системы выявлено не было. В качестве второго эксперимента был выполнен анализ образовательной программы, которая была сгенерирована программно. Результаты тестирования совпали с ожидаемыми.

В заключении в краткой форме излагаются основные результаты диссертационной работы, представляются отличия диссертационной работы от ранее выполненных родственных работ других авторов и рассматриваются перспективы дальнейших исследований в данной области.

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты.

1. Разработана структурно-иерархическая дидактическая (СИД) модель электронного обучения.
2. Разработаны операции СИД модели над электронными энциклопедиями, образовательными программами и курсами.
3. Разработаны алгоритмы анализа качественных характеристик образовательных программ и электронных учебных курсов с использованием операций СИД модели.
4. Разработан прототип программной системы ECoD для создания электронных учебных курсов на базе СИД модели.

Публикации по теме диссертации

Статьи в журналах из перечня ВАК

1. 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.

2. Силкина, Н.С. Структурно-иерархическая дидактическая модель электронного обучения / Н.С. Силкина, Л.Б. Соколинский // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». –2019. –Том 8, № 4. –С. 56-83.
3. Силкина, Н.С. Обзор адаптивных моделей электронного обучения Н.С. Силкина, Л.Б. Соколинский // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». –2016. –Том 5, № 4. –С. 61–76.
4. 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.
5. Жигальская (Силкина), Н.С. Моделирование дидактической структуры электронных учебных комплексов / Н.С. Жигальская (Силкина) // Вестник Южно-Уральского государственного университета. Серия «Математическое моделирование и программирование». –2008. –№ 27(127). –Вып. 2. –С. 4–9.

Статья в издании, индексируемом в Scopus и Web of Science

6. Ivanova, O.N. Competence-Oriented Model of Representation of Educational Content / O.N. Ivanova, N.S. Silkina // Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO'2017, Opatija, Croatia, May 22–26, 2017. –IEEE, 2017. –P. 791–794.

Свидетельство о регистрации программ и баз данных

7. Жигальская (Силкина) Н.С., Евдокимова А.С. Свидетельство Роспатента о государственной регистрации программы для ЭВМ «UnicstLite: программное средство для разработки электронных учебных энциклопедий на базе платформы Microsoft.NET» № 2008614487 от 03.10.2008, правообладатель: ГОУ ВПО "ЮУрГУ".

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

Издательский центр Южно-Уральского государственного университета
Подписано в печать 7.12.2019. Формат 60 × 84 1/16. Печать цифровая.
Усл. печ. л. 1,28. Уч.-изд. л. 1,0. Тираж 100 экз. Заказ ???/???.
Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, пр. Ленина, 76.