ДИАГРАММЫ СОСТОЯНИЙ ПРИ МОДЕЛИРОВАНИИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА RIA ОБУЧАЮЩИХ СИСТЕМ

УДК 004.4'22

Анна Александровна Пупыкина,
аспирант каф. математических методов обработки информации Российского государственного гуманитарного университета (РГГУ)
Тел.: 8 (8482) 50-68-44
Эл. почта: anna.pupykina@gmail.com

Анна Евгеньевна Сатунина,
к.э.н., доцент, профессор каф. Математических и естественно-научных дисциплин филиала Российского государственного гуманитарного университета (РГГУ) в г. Балашиха
Эл. почта: aesat@mail.ru

Предложены требования, предъявляемые к методам проектирования пользовательского веб-интерфейса обучающей среды. Раскрыты особенности проектирования пользовательского интерфейса на основе событийной модели алгоритма. Проведен сравнительный анализ модельно-ориентированных подходов разработки RIA с точки зрения используемых моделей и элементов. В работе приведен пример использования диаграммы состояний UML для моделирования взаимодействия между клиентской и серверной частью RIA, выявлены недостатки.

Ключевые слова: модель, модельно-ориентированный подход, автоматная модель, диаграмма состояний, веб-приложение.

Anna A. Pupykina,
Post-graduate student, the Department of Mathematical methods of information processing, Russian State University for the Humanities (RSUH)
Tel.: 8 (8482) 50-68-44
E-mail: anna.pupykina@gmail.com

Anna E. Satunina,
PhD in Economics, Professor, the Department of Math and science education, Russian State University for the Humanities (RSUH) in Balashiha
E-mail: aesat@mail.ru

USING STATE MACHINE DIAGRAMS FOR DESIGNING WEB-BASED USER INTERFACE OF EDUCATIONAL SOFTWARE

Requirements applicable to the methods for designing web-based user interface of educational software are proposed. The features of user interface design based on the event model algorithm are revealed. A comparative analysis of model-driven approaches to RIA development from the perspective of the used models and the used elements was done. The paper shows how to use UML state diagrams to model the interaction between client and server part of RIA, revealed disadvantages.

Keywords: *model, model driven approach, statechart, state machine diagrams, web application.*

1. Введение

Эффективность функционирования сложных программных систем зависит от качественной реализации средств человеко-компьютерного взаимодействия. Поэтому разработка пользовательского интерфейса часто становится неотъемлемой частью разработки программного продукта. Создание пользовательского интерфейса при разработке программных систем занимает до 40% от всего времени разработки, поэтому обеспечение качества интерфейсов является необходимым требованием для обеспечения итогового качества системы.

2. Веб-приложения с графическим пользовательским веб-интерфейсом в системах поддержки образовательного процесса

В последнее время широкое распространение получили информационные системы в обучении, реализующие новые образовательные технологии и создающие новую электронную обучающую среду. Эффективность обучающей среды в целом во многом обуславливается рядом факторов, одним из которых является эффективность и качество используемых средств обеспечения взаимодействия преподавателя и обучаемого. В электронной обучающей среде это «пользовательский интерфейс», который наряду с обучающим контентом – залог оптимальной обучающей среды.

Опираясь на требования, предъявляемые к компьютерным обучающим системам и программам, сформулированные в [1], можно составить список требований предъявляемых к методам проектирования пользовательского веб-интерфейса обучающей среды:

– наличие технологии оформления страниц экрана с применением графики, цвета, и пр.

– возможность создания многоуровневой навигации в обучающей программе, то есть возможен не только «линейный» (последовательный, шаг за шагом) порядок выполнения программы.

– наличие способов проектирования системы подсказок, ссылок на дополнительные материалы, выходы на иные информационные материалы.

– наличие средств разработки мотивирующих и информирующих сообщений.

– наличие способов создания средств взаимодействия пользователя с обучающей программой.

– наличие методов проектирования средств адаптации системы обучения к обучающемуся.

– возможность создания протокола обучения с сохранением состояния.

– наличие способов интеграции со сторонними программными решениями

Недостатком традиционных веб-приложений с графическим пользовательским веб-интерфейсом для применения в системах поддержки образовательного процесса является ограничение интерактивности, что является исключительно важным в образовательном процессе.

Современная технология Rich Internet Applications (RIA) позволяет обойти эти ограничения, предоставляя ряд преимуществ, например, более богатые и более эффективные графические интерфейсы, напоминающие настольные приложения [2]. RIA это сложные веб-приложения на основе архитектуры толстого клиента с асинхронным взаимодействием и набором виджетов пользовательского интерфейса. Разработка RIA требует более трудоемкого процесса проектирования. Традиционно процесс разработки RIA включает следующие этапы:

1. Анализ требований позволяет определить набор необходимых функций, предоставляемых пользователю.

2. Проектирование RIA охватывает моделирование архитектуры приложения, контента, навигации, элементов пользовательского интерфейса и сценариев.

3. Реализация RIA основывается на интеграции специальных библиотек, которые предоставляют виджеты пользовательского интерфейса.

4. Тестирование созданного приложения может проводиться вручную или с использованием средств автоматизации тестирования (например, Selenium, FlexMonkey и др.).

Технологии разработки RIA является новым направлением в области разработки программного обеспечения. Новые языки и фреймворки (например, ICEFaces, Dojo, GWT, Silverlight, Adobe Flash) свидетельствуют о быстром развитии RIA, но отсутствие полной документации является главной проблемой для создания унифицированных модельно-ориентированных подходов разработки RIA. Унификация процессов проектирования и разработки RIA позволит снизить зависимость от конкретной платформы. Проектирование и разработка RIA в составе сложного программного продукта обеспечит интеграцию с серверами обработки бизнес-логики, что снизит нагрузку на серверную часть веб-приложения.

3. Проектирование пользовательского веб-интерфейса на основе событийной модели алгоритма

Наиболее распространенным алгоритмом работы пользовательского интерфейса для RIA является алгоритм, основанный на событийной модели (схема представлена на рисунке).

Посылать событие диспетчеру и принимать его можно в асинхронном режиме. Это позволяет распараллеливать процессы генерации и обработки событий. Для обмена событиями между такими процессами используется очередь событий. К преимуществам систем, построенных на базе событийной модели можно отнести следующее:

1. Весь код, который приводит к изменению состояния элементов приложения, вынесен в независимые друг от друга обработчики событий.

2. Распараллеливание обработчиков разных событий (и в некоторых случаях независимых обработчиков одного события).

3. Расширяемость приложения за счет регистрации новых обработчиков для существующих событий и добавления новых событий.

4. Применима модель распределенных вычислений (обработчики событий могут быть распределены между отдельным узлам сети)

При проектировании пользовательского интерфейса на основе событийной модели алгоритма каждый элемент должен иметь хотя бы один «Прослушиватель Событий», которые в зависимости от внешнего воздействия (изменения состояния элемента) влияет на другие элементы. Те элементы тоже имеют свои «Прослушиватели Событий», которые тоже в зависимости от контекста оказывают воздействие на другие. Это усложняет логику работы интерфейса, т.к. трудно отследить, как изменение состояния одного элемента повлияет на остальные элементы. Даже незначительная модификация приводит к необходимости отслеживать все зависимости между элементами интерфейса, что увеличивает сложность не только процесса разработки интерфейса, но и процесса обеспечения качества.

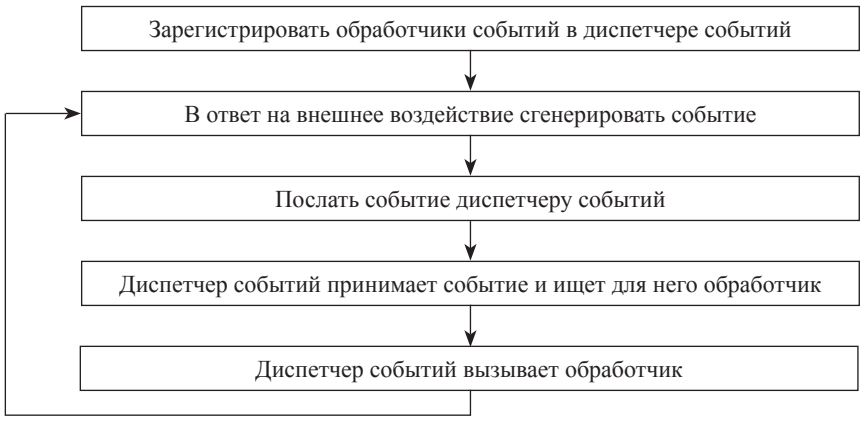

Рис. 1. *Схема работы пользовательского интерфейса приложения, основанный на событийной модели*

Эту проблему можно решить, представив пользовательский интерфейс как множество состояний и переходов между ними. Автоматная модель позволит абстрагировать компоненты автомата от реализации элементов интерфейса на конкретной технологической платформе. Автомат может подключаться к разным реализациям интерфейса, технологиям вывода и использоваться с разными объектами программы, используя модельно-ориентированный подход (Model-Driven Development, MDD).

Использование MDD при разработке приложения позволяет сосредоточиться на бизнес-логике приложения, а не на реализации низкоуровневого функционала. MDD относится к семейству подходов разработки, основанных на использовании модели в качестве первичного артефакта в жизненном цикле программного продукта. Инструменты MDD позволяют автоматически транслировать высокоуровневые спецификации в качественный код и документацию, что дает преимущество в существенной экономии времени, необходимого для реализации информационных систем.

Согласно модельно-ориентированной архитектуре (Model Driven Architecture, MDA) [3], созданной консорциумом OMG и основанной на идеях MDD, можно выделить следующие уровни моделей:

− платформо-независимая модель – автоматная модель пользовательского интерфейса, описываю-

щая состав, структуру и поведение системы в терминах близких к предметной области;

− платформо-зависимая модель – описывает состав, структуру и поведение системы с указанием средств реализации на конкретной платформе. Модель создается на основе PIM и модели платформы.

− модель платформы описывает элементы пользовательского интерфейса. Для упрощения поддержки различных технологий RIA, модель платформы целесообразно создавать как иерархию метамоделей, это позволит вводить новые технологии за счет расширения метамодели пользовательского веб-интерфейса.

Большинство инструментов модельно-ориентированной разработки для описания моделей используют UML (унифицированный язык моделирования) [4]. UML применим для моделирования как статических, так и динамических аспектов системы. Автоматная модель объединяет статические и динамические аспекты программы.

Пользовательский интерфейс может быть представлен как суперпозиция автоматов его элементов. Автомат представим как шестерку $A = (Z, X, Y, f, h, z_0)$, где

Z – множество состояний автомата (внутренний алфавит);

X – множество входных символов (входной алфавит);

Y – множество выходных символов (выходной алфавит);

f – функция переходов, $f: Z \times X \to Z$;

h – функция выходов, $h: Z \to Y$;

$z_0 \in Z$ – начальное состояние.

Если автомат находится в некотором состоянии $z \in Z$, то выход автомата определяется функцией выхода h. Выход автомата интерпретируется в данном случае как реакция пользовательского интерфейса, которая, может привести к изменению состояния элементов пользовательского интерфейса. В каждый момент времени автомат читает входное слово, которое является действий со стороны пользователя и/или воздействием со стороны элементов пользовательского интерфейса. По прочитанному входному слову и функции переходов определяется состояние в следующий момент времени.

Д.Харелом была предложена модификация диаграммы состояний и переходов, названная «картой состояний» (State chart) [5], позволяющая в некоторых случаях более компактно описывать поведение автоматов реагирующих (реактивных) систем. Основными особенностями карт состояний являются:

− гиперсостояния (суперсостояния), объединяющие несколько состояний, имеющих идентичную реакцию на одно и то же событие;

Таблица 1

Модельно-ориентированные подходы разработки RIA

Метод	Модели	Элементы
OOH4RIA*[6]	Модель представления	Виджеты, контейнеры и другие объекты, такие как текстовые поля и кнопки
	Модель взаимодействия	стереотипные состояния, такие как снимки экрана, виджеты взаимодействия, области распараллеливания, условия и действия
OOWS*[7]	Модель пользователя	Типы пользователей, взаимодействующих с системой
	Модель навигации	Ориентированный граф, узлы которого представляют собой контекст навигации, а дуги навигационные ссылки
UWE-R[8]	Модели аналогичны моделям UWE	Расширен набор элементов UWE с помощью механизма наследования для моделирования RIA
WebML[9]	Модель гипертекста	Клиентские и серверные модули, страницы, селекторы, и операции
	Модель данных	Диаграмма сущность-связь
RUX [10]	Абстрактное представление	Конекторы (соединители), среда, представления, обобщения, ассоциации и агрегации
	Конкретное представление	контроллеры, разметка, навигаторы, обработчики, временное представление
ADRIA[11]	Модель структуры	Пространство взаимодействий, ввод, вывод, действия, ассоциации, навигация
	Модель поведения	диаграммы состояний с действиями вызывающими клиентские или серверные методы, области распараллеливания
	Модель данных	клиентские и серверные сущности, клиентские, серверные и межуровневые отношения
Martinez-Ruiz [12]	Модель задач и предметной области	Последовательность задач пользователя
	Абстрактная модель пользовательского интерфейса	Элементы пользовательского интерфейса без определения средств взаимодействия
	Реальная модель пользовательского интерфейса	Виджеты RIA фреймворков (Flex, XAML, Laszlo)
UWE [13]	Модель контента	Диаграмма классов UML для представления сущностей предметной области
	Модель навигации	Класс навигации – для спецификации узла доступного для посещения пользователем или системой Навигационная ссылка – спецификация ссылки используемой для доступа к объекту навигации
	Модель презентации	Структурные элементы презентации: группы, альтернативы, страницы
	Модель процессов	Класс процесса – не навигационный класс. Ссылка процесса – для соединения класса навигации и класса процесса Диаграмма деятельности UML для описания процесса

Таблица 2

Описание метода ADRIA, с точки зрения используемых элементов и поддержки стандартов OMG

Модели	Элементы	MDE
Модель структуры	Пространство взаимодействий, ввод, вывод, действия, ассоциации, навигация	Метод предоставляет средства анализа и проектирования RIA. Не поддерживает парадигму MDA
Модель поведения	диаграммы состояний с действиями, вызывающими клиентские или серверные методы, области распараллеливания	
Модель данных	клиентские и серверные сущности, клиентские, серверные и межуровневые отношения	

− обобщение переходов из всех состояний, охватываемых гиперсостоянием, в указанное состояние;

− произвольная глубина вложения гиперсостояния;

− объединение последовательных и параллельных состояний в гиперсостояния;

− псевдосостояния (условные, терминальные, исторические), которые не являются реальными состояниями;

− указание в вершинах, соответствующих состояниям и гиперсостояниям, деятельностей (do); однократно выполняемых при входе в вершину действий (entry); однократно выполняемых при выходе из вершины действий (exit);

− указание на дугах логических условий, вызывающих переход событий, однократно выполняемых на переходе действий и формируемых на переходе событий.

Впервые карты состояний были предложены Харелом в 1987 году. Впоследствии на основе карт состояний Харела были созданы диаграмма конечных автоматов (State Machine Diagrams) и диаграмма деятельности (Activity Diagrams) UML.

4. Применение диаграмм состояний в модельно-ориентированных подходах разработки RIA

На основе анализа модельно-ориентированных подходов разработки RIA с точки зрения используемых моделей и элементов (таблица 1) можно сделать вывод: большинство подходов используют структурные диаграммы для проектирования клиент-серверного взаимодействия (диаграммы классов, диаграммы сущность-связь). Исключением является метод ADRIA.

Для моделирования таких функции RIA, как клиентские вычисления, синхронизация между объектами клиента и сервера, метод ADRIA предлагает объектно-ориентированное аналитическое проектирование с использованием моделей задач, областей взаимодействия и конечных автоматов. Этот метод фокусируется на проектировании событий, вызванных действиями пользователя.

В таблице 2 приведено описание метода ADRIA, с точки зрения используемых элементов и поддержки стандартов OMG (MDA, MOF).

Диаграмма состояний используется только для создания модели поведения. Моделирование проходит в три этапа [14]:

1. Разделение общего пространства взаимодействия на серверные и клиентские объекты.

2. Определение порядка обмена сообщениями. Клиентские и серверные объекты взаимодействуют за счет обмена хорошо структурированными сообщениями. Запрос на сервер обычно приводит к необходимости обновления состояния клиентского объекта. Поэтому сообщение должно содержать идентификатор клиентского объекта, вызванного метода серверного объекта и переданные параметры.

3. Определение параллелизма и синхронизации.

Диаграмма состояний используется для моделирования взаимодействия между клиентской и серверной частью веб-приложения. Однако использование диаграммы состояний ограничивается моделированием и последующей генерацией, т.е. не используется теория автоматов для анализа и валидации решений.

Метод ADRIA делает акцент на моделировании клиент-серверного взаимодействия RIA, не предлагая высокоуровневых абстракций для задания полнофункциональных интерфейсов и детальной модели пользовательского интерфейса в стиле редакторов экранных форм. Использование подобных редакторов, которые в настоящее время представлены практически в любой среде разработки ПО (в том числе не коммерческих), позволяет снизить трудоемкость разработки пользовательского интерфейса. Для сложных информационных систем уровня предприятия создание детальной модели пользовательского интерфейса на верхних уровнях абстракции приведет к неоправданной громоздкости моделей и потере их наглядности. Поэтому моделирование пользовательского интерфейса целесообразно проводить на нескольких уровнях абстракции, т.к. это позволит создавать лаконичные, хорошо читаемые модели с последующей эффективной кодогенерацией.

Целью модельно-ориентированные подходов разработки RIA является более эффективное создание качественных веб-приложений. Однако их применение для реальных систем приводит к необходимости создания и ведения через весь цикл разработки сложных и громоздких моделей. При этом ограничение технологических платформ приводит к потере гибкости средств автоматической кодогенерации. В результате использование этих подходов для создания реальных информационных систем не дает выигрыша в производительности.

Выходом из такой ситуации может быть ограничение класса целевых приложений и создание гибкой системы поддержки различных технологических платформ. Типизация пользовательского интерфейса позволит ввести базовые модели, на основе которых возможно более

быстрое и удобное создание моделей пользовательского интерфейса. Гибкая система поддержки технологических платформ позволит использовать модельно-ориентированный метод разработки RIA с последующей автоматической кодогенерацией на актуальную, современную платформу.

5. Выводы

Информатизация образования должна привести к повышению качества образования; увеличению степени доступности образования; росту мобильности студентов и преподавателей. В информационной системе поддержки образовательного процесса, где одним из основных пользователей является студент, к пользовательскому интерфейсу должно уделяться особое внимание. Пользовательский интерфейс обучающей среды, с которой взаимодействуют студенты, оказывает непосредственное влияние на мотивацию, скорость восприятия материала, утомляемость и другие важные показатели. Поэтому разработка интерфейса обучающей среды требует научно обоснованный, взвешенный и продуманный системный подход, отвечающий требованиям стандартов по юзабилити. Сложный интерфейс обучающей среды, на освоение которого требуются значительные затраты усилий, не соответствует целям внедрения информационных систем в процесс обучения, т.к. отвлекает обучающегося от основного предмета изучения, не мотивирует преподавателей разрабатывать эффективные электронные курсы в такой программной системе. Такие системы поддержки образовательного процесса не могут быть признаны качественными.

Эффективность современных обучающих систем, основанных на использовании компьютерных и телекоммуникационных технологий, достигается, в том числе, за счет соблюдения высокого качества и отсутствия ограничений доступа к образовательным материалам и услугам. Использование веб-ориентированных пользовательских интерфейсов в обучающей среде позволит обеспечить наиболее удобный и независимый доступ к обучающим материалам и услугам. Недостатком традиционных веб-приложения с графическим пользовательским веб-интерфейсом для применения в системах поддержки образовательного процесса является ограничение интерактивности. Были рассмотрены Rich Internet Applications (RIA), позволяющие обойти эти ограничения. Они обладают рядом преимуществ, например, более богатыми и более эффективными графическими интерфейсами, напоминающими настольные приложения, однако разработка RIA является трудоемким процессом. Инструменты MDD позволяют снизить зависимость от конкретной платформы, автоматическая трансляция высокоуровневых спецификаций в качественный код и документацию дает преимущество в существенной экономии времени. В работе проанализирована возможность использования теории автоматов в методах проектирования пользовательских веб-интерфейсов.

Для создания качественного интерактивного управляемого событиями пользовательского веб-интерфейса необходима модель управления состояниями. Использование для этих целей модели, построенной на теории конечных автоматов, является целесообразным по ряду причин:

− позволяет упростить логику пользовательского интерфейса за счет формальной структуризации приложения;

− позволяет в явном виде определить состояния приложения и задать варианты поведения при переходах приложения из одного состояния в другое;

− позволяет наглядно представить состояния, в которых может находиться приложение;

− позволяет использовать несколько конечных автоматов, для каждого из которых определяется собственный набор вариантов поведения приложения;

− централизация и инкапсуляция управления состояниями.

В работе приведен пример использования диаграммы состояний UML для моделирования взаимодействия между клиентской и серверной частью RIA, выявлены недостатки.

За счет использования модельно-ориентированного подхода разработки RIA на основе автоматной модели возможно создание гибкой, адаптируемой информационной системы поддержки образовательного процесса.

Литература

1. Алисейчик П.А. Компьютерные обучающие системы / П.А. Алисейчик и др. // Интеллектуальные системы. – М.: изд-во РГГУ, 2004. – Т. 8, вып.1–4. – С. 5–44.

2. Busch M. Rich Internet Applications State-of-the-Art: Technical Report / M. Busch, N. Koch; Ludwig-Maximilians-Universität. – München, 2009.

3. Object Management Group, MDA Guide Version 1.0.1 [Электронный ресурс]. – Электрон. дан. – [2003] – Режим доступа : http://www.omg.org/cgi-bin/doc?omg/03-06-01

4. Object Management Group (OMG). Unified Modeling Language (UML), Version 2.3 [Электронный ресурс]. – Электрон. дан. – [2010] – Режим доступа : http://www.omg.org/spec/UML/2.3

5. Harel D. Statecharts: A visual formalism for complex systems / D. Harel // Science of Computer Programming. – 1987. – Vol. 8, No. 3. – pp. 231–274.

6. Meli´a S. A model-driven development for gwt-based rich internet applications with ooh4ria / S. Meli´a and others // Proc. 8th Int Conf on Web Engineering (ICWE 2008), IEEE. – 2008 – pp. 13–23.

7. Valverde F. Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach / F. Valverde, O. Pastor // Proc. 10th Int Conf on Web Information Systems Engineering (WISE ‘09). – 2009.

8. Machado L., Filho O., Ribeiro J. UWER: an extension to a Web Engineering methodology for Rich Internet Applications / L. Machado, O. Filho, J. Ribeiro // WSEAS Transactions on Information Science and Applications. – 2009. – Vol. 6, Is. 4. – pp. 609–610.

9. Comai S. RIA concepts in a web modeling language / S. Comai and others. // Proc. 15th Int Conf on World

Wide Web (WWW ‘06), ACM. – 2006 – pp. 907–908.

10. Preciado J. C. Designing rich internet applications combining uwe and rux-method / J. C. Preciado and others // Proc 8th Int Conf on Web Engineering (ICWE 2008), IEEE. – 2008 – pp. 148–154.

11. Engineering Adaptive Web Applications : Dr. rer. nat. Thesis. / Dolog, Peter. Hannover, Germany : University Library, University of Hannover, 2006. 148 p.

12. Martínez-Ruiz F.J. A First Draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications / F.J. Martínez-Ruiz and others // Proc. of the 4th Latin American Web Congress (LA-Web’06), IEEE. – 2006.

13. Kroiß C. UWE Metamodel and Profile. User Guide and Reference : Technical Report / C. Kroiß, N. Koch; Ludwig-Maximilians-Universität. – Münche. – 2008.

14. ADRIA: A Method for Abstract Design of Rich Internet Applications for the Web 2.0 [Электронный ресурс]. – Электрон. дан. – Режим доступа : https://intranet.cs.aau.dk/fileadmin/user_upload/Education/Courses/2009/DIEB/ADRIA_LongVersion.pdf

References

1. Aliseichik P.A. Computer training systems / P.A. Aliseichik etc. // Intelligent Systems – M. RGGU 2004. – Vol. 8, no.1–4. – S. 5–44.

2. Busch M. Rich Internet Applications State-of-the-Art: Technical Report / M. Busch, N. Koch; Ludwig-Maximilians-Universität. – München, 2009.

3. Object Management Group, MDA Guide Version 1.0.1 [electronic resource]. – Electron. dan. – [2003] – Access mode: http://www.omg.org/cgi-bin/doc?omg/03-06-01

4. Object Management Group (OMG). Unified Modeling Language (UML), Version 2.3 [electronic resource]. – Electron. dan. – [2010] – Access mode: http://www.omg.org/spec/UML/2.3

5. Harel D. Statecharts: A visual formalism for complex systems / D. Harel / / Science of Computer Programming. – 1987. – Vol. 8, No. 3. – Pp. 231–274.

6. Meli’a S. A model-driven development for gwt-based rich internet applications with ooh4ria / S. Meli’a and others / / Proc. 8th Int Conf on Web Engineering (ICWE 2008), IEEE. – 2008 – pp. 13–23.

7. Valverde F. Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach / F. Valverde, O. Pastor / / Proc. 10th Int Conf on Web Information Systems Engineering (WISE ‘09). – 2009.

8. Machado L., Filho O., Ribeiro J. UWER: an extension to a Web Engineering methodology for Rich Internet Applications / L. Machado, O. Filho, J. Ribeiro / / WSEAS Transactions on Information Science and Applications. – 2009. – Vol. 6, Is. 4. – Pp. 609–610.

9. Comai S. RIA concepts in a web modeling language / S. Comai and others. / / Proc. 15th Int Conf on World Wide Web (WWW ‘06), ACM. – 2006 – pp. 907–908.

10. Preciado JC Designing rich internet applications combining uwe and rux-method / JC Preciado and others / / Proc 8th Int Conf on Web Engineering (ICWE 2008), IEEE. – 2008 – pp. 148–154.

11. Engineering Adaptive Web Applications: Dr. rer. nat. Thesis. / Dolog, Peter. Hannover, Germany: University Library, University of Hannover, 2006. 148 p.

12. Martínez-Ruiz F.J. A First Draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications / FJ Martínez-Ruiz and others / / Proc. of the 4th Latin American Web Congress (LA-Web’06), IEEE. – 2006.

13. Kroiß C. UWE Metamodel and Profile. User Guide and Reference: Technical Report / C. Kroiß, N. Koch; Ludwig-Maximilians-Universität. – Münche. – 2008.

14. ADRIA: A Method for Abstract Design of Rich Internet Applications for the Web 2.0 [electronic resource]. – Electron. dan. – Mode of access: https://intranet.cs.aau.dk/fileadmin/user_upload/Education/Courses/2009/DIEB/ADRIA_LongVersion.pdf