View metadata, citation and similar papers at core.ac.uk
brought to you by CORE

МОДЕЛЬНО-ОРИЕНТИРОВАННОЙ ПОДХОД К РАЗРАБОТКЕ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

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

Одной из основных проблем модельно-ориентированного подхода разработки пользовательских интерфейсов является сложность разработки инструментальных средств, позволяющих поддерживать данную методологию на всех этапах жизненного цикла программного продукта. Проведен анализ применимости теории графов для формализации модели веб-приложения. Теория графов предоставляет средства для формализованного описания модели взаимодействия и обеспечивает предоставление точных математических зависимостей между компонентами. Предложена схема моделирования веб-приложения.

Ключевые слова: *граф, модель, модельно-ориентированный подход, трансформация, веб-приложение.*

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

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

MODEL-BASED APPROACH TO USER INTERFACE DEVELOPMENT

One of the main problems of model-driven user interface development approach is the complexity of the development of tools that maintain this methodology at all stages of the software lifecycle. Applicability of graph theory to formalize the Web application model was analyzed. Graph theory provides a means for a formalized description of the model of interaction and ensure the provision of precise mathematical relationships between the components. A scheme for modeling Web applications was proposed.

Keywords: *graph, model, model driven approach, transformation, web application.*

1. Введение

Объектом исследования является процесс модельно-ориентированной разработки веб-приложения. Специфика предлагаемого подхода к проектированию веб-приложений заключается в разработке и использовании формализованного метода моделирования, основанного на развитии графовых представлений.

Детализация представления веб-приложения должна доходить до уровня, необходимого для генерации кода веб-приложения на конкретную платформу. Рациональным подходом будет являться постепенная детализация каждого аспекта моделирования по мере согласования модели верхнего уровня. Такой подход значительно снизит сложность процесса моделирования системы и дальнейшую поддержку моделей. Наиболее распространенным вариантом построения модели взаимодействия пользователя с системой является использование теории графов.

Использование теории графов для моделирования 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.

Теория графов предоставляет средства для формализованного описания модели взаимодействия и обеспечивает предоставление точных математических зависимостей между компонентами

Использование теории графов для описания взаимодействий в ИС было положено в работе Уорда Канингема и Кента Берка [1]. В ней для снижения сложности понимания программ написанных на языке Smalltalk а также для обнаружения, локализации и устранения ошибок применяются графы. Графы в данной работе используются для определения и классификации сообщений с помощью которых происходит взаимодействие объектов во время выполнения программ. Более широкое применение графов было предложено в работе Кайна [2] где на ряду с использованием графов для описания вызовов методов было предложено применять графы для определения наследования, агрегации и использования объектов. Такой подход применим для объектно-ориентированных программ и приводит к устранению дублирования кода и сложности понимания системы в целом за счет установления ответственности типов графа за спецификацию аспектов поведения программы. Так же как и в [1] Эллис в своей работе [3] применил теорию графов для спецификации объектно-ориентированных парадигм. Основой для спецификации проектных метрик в работе Чидамбера [4] являются элементы теории графов. Бураков В.В. в своей работе [5] предлагает метод моделирования программного средства для решения задач оценивания и управления качеством и адаптации модели к изменяющимся целям моделирования. Таким образом теория графов широко применяется в работах по программной инженерии как российских так и зарубежных авторов.

С целью формализации задач модельно-ориентированной разработки пользовательских интерфейсов предлагается ввести ряд надстроек в графовых представлениях, позволяющих описывать совместную трансформацию связанных моделей

2. Схема моделирование веб-приложения

При модельно-ориентированной разработке Web-интерфейсов, ориентированных на обработку данных, выделяют три уровня: уровень данных, который описывает структуру данных предметной области; уровень навигационных структур и уровень пользовательского интерфейса. Кроме того, необходимо предусмотреть возможность описания настройки и адаптации содержимого и представления Web-интерфейса под конкретного пользователя. Уровень представления отделяет задание содержимого Web-страниц от их стилей и форматирования, позволяя задавать различные представления для одного и того же информационного наполнения. Уровень настройки отделяет ее от других аспектов проектирования, облегчая работу разработчикам, которым не нужно принимать во внимание, что может быть подвергнуто настройке для конкретного пользователя, на этапе задания навигационных структур или интерфейса пользователя.

Схема моделирования веб-приложения:

1. Разрабатывается *UML*-диаграмма классов уровня данных, определяющая сущности предметной области и отношения между ними (в терминах архитектурных шаблонов – модель).

2. Из анализа функциональных требований выделяются сущности, предоставляющие множество четко определенных автономных функций, независимых от взаимного состояния.

3. Верхний уровень абстракции модели навигационных структур разрабатывается в виде *UML*-диаграммы вариантов использования. Каждый вариант использования определяет свою специфическую логику и отвечает за отдельный пользовательский интерфейс (набор web-страниц). Моделирование прав доступа на основе ролей осуществляется с помощью связывания вариантов использования с акторами.

4. Детализация сценария в рамках каждого варианта использования, проводится на *UML*-диаграмме состояний. На этой диаграмме моделируется поведение страницы пользовательского веб-интерфейса в зависимости от действий пользователя. Связывание диаграммы состояний с определенным вариантом использования проводится определением диаграммы состояний в качестве дочернего элемента соответствующего варианта использования.

5. Вызов бизнес-логики с уровня пользовательского интерфейса должен обрабатываться контроллером, который на *UML*-диаграмме состояний задается дочерним элементом – классом. Это позволит использовать архитектурные шаблоны проектирования.

6. Уровень пользовательского интерфейса позволяет моделировать поэлементное наполнение web-страниц (в терминах архитектурных шаблонов – вид). Для этих целей каждая web-страница изображается в виде составного компонента, в который добавлены элементы графического интерфейса. Элементы целесообразно вводить с помощью механизмов расширения UML-стереотипов. Это позволит легко расширять модель платформы для использования с различными технологиями. Компонент пользовательского интерфейса задается дочерним элементом клиентского состояния, а элементам, ответственным за обработку пользовательских действий, ставится в соответствие переход диаграммы состояний.

На рисунке 1 приведена схема моделирования веб-приложения

3. Применение графов и графовых продукций в модельно-ориентированном подходе разработки веб-приложений

Переход от платформенно-независимой модели к платформенно-зависимой является трансформацией по заданной спецификации, содержащей формальное описание преобразования модели общего вида к конкретной платформе реализации.

Входными данными механизма трансформации являются:

– Метамодели каждого вида моделей;

– Множество исходных моделей;

– Описание правил трансформации.

Результаты применения механизма трансформации:

– Множество исходных моделей; преобразованных к виду пригодному для применения правил трансформации;

– Множество сгенерированных моделей, созданных в процессе трансформации.

Для описания правил трансформации представим модели в виде графов. Это позволит применять автоматическую верификацию и валидацию трансформаций.

Ориентированным графом называется граф, ребрам которого присвоено направление. Ориентиро-

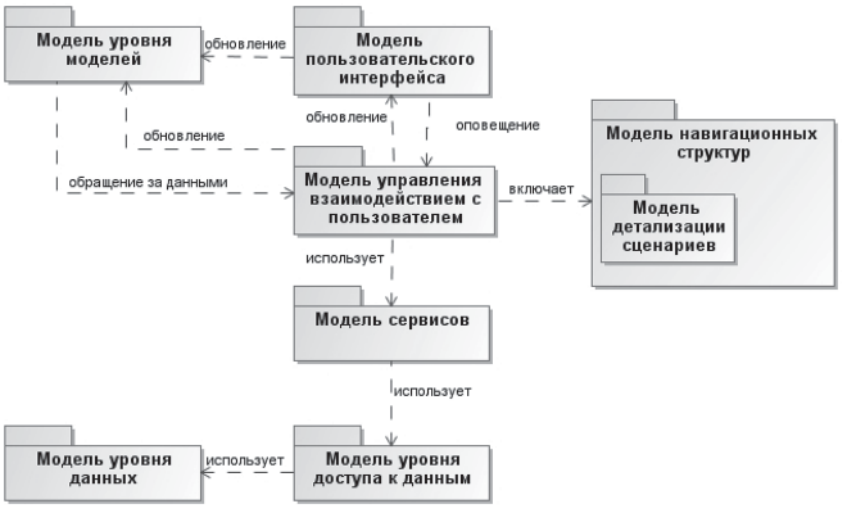

Рис. 1. *Схема моделирования веб-приложения*

ванный граф $G = (V, E, s, t)$ состоит из конечного множества вершин V и конечного множества ребер (дуг) E. Каждое ребро соединяет упорядоченную пару вершин. Вершины обозначаются $v_1, v_2, \ldots v_n \ldots$ а ребра – $e_1, e_2 \ldots, e_m \ldots$ Если $e_k = (v_i, v_j)$, то v_i называется начальной вершиной ребра e_k, а v_j – конечной. Ребра, связывающие одну и ту же упорядоченную пару вершин, называются параллельными. Если вершина v_i является одновременно начальной и конечной вершиной, ребро называется петлей. Функции $s: E \to V$ и $t: E \to V$ для каждого ребра ставят в соответствие в точности одну начальную и одну конечную вершины.

Помеченным графом называется граф, вершинам или дугам которого присвоены какие-либо метки. Пусть L (VL, EL), R (VR, ER) – пара непересекающихся множеств меток и ролей соответственно. (L, R)-помеченный граф $G = (g, l, r)$, где

- $g = (V, E, s, t)$ – граф;
- $l = (vl: V \to VL, el: E \to EL)$ – функции расстановки меток вершинам и ребрам соответственно;
- $r = (vr: V \to VR, er: E \to ER)$ – функция отображения вершин на множество ролей. Метки определены для идентификации вершин и ребер, роли для описания контекста использования сущности

Помеченным типизированным графом называется помеченный граф, вершины и ребра которого типизированы. Тип вершины или ребра представляет собой совокупность атрибутов (свойств), приписываемых вершинам и ребрам: $T = (VT, ET)$ где VT-множество предопределенных типов вершин и ET-множество предопределенных типов ребер. (L, R)-помеченный T-типизированный граф $G = (g, type)$, где g – (L, R)-помеченный граф, а $type = (vt: V \to VT, et: E \to ET)$ –функции, ставящие в соответствие тип для каждой вершины и ребра.

Определим множество типов вершин VT и множество типов ребер ET. Множество VT представляет собой множество вида ($type_{id}$, $type_{name}$, A), где $type_{id}$ – идентификатор типа, $type_{name}$ – имя типа, A – подмножество множества доступных атрибутов $Attributes$. Множество ET представляет собой ($type_{id}$, $type_{name}$, A, directed, restricted, vt_h, vt_t), где $type_{id}$ – идентификатор типа; $type_{name}$ – имя типа; $A \subseteq Attributes$; $directed \in \{true, false\}$ – флаг направленности ребра; $restricted \in \{true, false\}$ –флаг определенности типов вершин, связанных с ребром; $vt_h \in VT$ – тип исходящей вершины ребра; $vt_t \in VT$ – тип входящей вершины ребра. Для любого типа $T \in ET$ если $restricted(T) = true$, то значения vt_h и vt_t определены; если же $restricted(T) = false$, то значения vt_h и vt_t не определены. Для направленных ребер с определенными начальными и конечными вершинами будем применять сокращенную запись: $type_{name}: vt_h \to vt_t$.

Описание графовой трансформации заключается в задании продукций. В левой части продукции указываются вершины и ребра, которые должны быть в составе исходного графа, а в правой — представление графа после применения продукции. Для детализации процесса применения правил вводится понятие частичного морфизма [5].

Продукция $p : (L \xrightarrow{r} R)$ состоит из шаблона заменяемого графа в левой части L, шаблона графа назначения в правой части R и частичного морфизма p, описывающего процесс трансформации, т.е. набор преобразований приводящий к включению графа R в качестве подграфа графа G. Применение правил трансформации может зависеть от набора условий накладываемых на исходный или трансформированный граф. В таких случаях применяется условная продукция $\hat{p} : (L \xrightarrow{p} R), cond(p)$, где p – частичный графовый морфизм и $cond(p)$ условия применения.

Система продукций позволяет наглядно описывать преобразования, которые происходят в моделях при выполнении над ним заданных трансформаций. Такой подход может использоваться как для описания гомогенных трансформаций, так и для перевода модели в другую метамодель. При этом графы L_i и R_i продукции p_i, а также промежуточные графы G_i будут содержать элементы (вершины и дуги) из обеих метамоделей.

4. Заключение

Применение модельно-ориентированного подхода для разработки веб приложений, использование декларативного языка спецификации пользовательского интерфейса и средств автоматической генерации приведет к снижению стоимости и времени разработки. Модельно-ориентированный подход к разработке веб-приложения, предоставляющий автоматическую генерацию интерфейса по декларативным, высокоуровневым моделям позволит ослабить технологическую привязку разрабатываемого интерфейса к конкретной платформе. Однако сам процесс разработки правил трансформации требует формализации моделей для применения автоматической верификации и валидации механизмов трансформации. В работе предложена схема моделирования веб-приложения, рассмотрена теория графов в контексте применения для формализации трансформации моделей.

Литература

1. Cunningham W. A Diagram for Object-Oriented Programs / W. Cunningham, K. Beck .//OOPSLA ‘86 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data. – 1986. – Vol. 21, issue 11. – P. 361–367. – Mode of access : http://dl.acm.org/citation.cfm?id=28734

2. Kleyn M.F. GraphTrace – Understanding Object-Oriented Systems Using Concurrently Animated Views / M.F. Kleyn, PP.C. Gingrich //OOPSLA ‘88 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data . – 1988. – Vol. 23, issue 11. – P. 191–205. – Mode of access : http://dl.acm.org/citation.cfm?id=62101

3. Ellis G. Object-Oriented Conceptual Graphs / Proceedings of the 3rd International Conference on Conceptual Structures [Electronic resource]. – Electronic data. – 1995. – P. 144–257. – Mode of access : http://link.springer.com/chapter/10.1007%2F3-540-60161-9_35

4. Chidamber S.R., Kemerer C.F. Towards a Metrics Suite for Object-

Oriented Design / S.R. Chidamber, C.F. Kemerer //OOPSLA ‘91 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data . – 1991. – Vol. 26, issue 11. – P. 197–211. – Mode of access : http://dl.acm.org/citation.cfm?id=117954.117970

5. Бураков В.В. Модели оценивания и алгоритмы управления качеством программных средств: автореф. дис. ... д-ра тех. наук / Бураков Вадим Витальевич. [Электронный ресурс]. – СПб, 2010. – 42 с. – Режим доступа: http://guap.ru/guap/main/avtoref_burakov.pdf

References

1. Cunningham W. A Diagram for Object-Oriented Programs / W. Cunningham, K. Beck .//OOPSLA ‘86 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data . –1986. – Vol. 21, issue 11. – P. 361–367. – Mode of access : http://dl.acm.org/citation.cfm?id=28734

2. Kleyn M.F. GraphTrace – Understanding Object-Oriented Systems Using Concurrently Animated Views / M.F. Kleyn, PP.C. Gingrich //OOPSLA ‘88 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data . – 1988. – Vol. 23, issue 11. – P. 191–205. – Mode of access : http://dl.acm.org/citation.cfm?id=62101

3. Ellis G. Object-Oriented Conceptual Graphs / Proceedings of the 3rd International Conference on Conceptual Structures [Electronic resource]. – Electronic data. – 1995. – P. 144–257. – Mode of access : http://link.springer.com/chapter/10.1007%2F3-540-60161-9_35

4. Chidamber S.R., Kemerer C.F. Towards a Metrics Suite for Object-Oriented Design / S.R. Chidamber, C.F. Kemerer //OOPSLA ‘91 Conference proceedings on Object-oriented programming systems, languages and applications [Electronic resource]. – Electronic data. – 1991. – Vol. 26, issue 11. – P. 197–211. – Mode of access : http://dl.acm.org/citation.cfm?id=117954.117970

5. Burakov V.V. Models and algorithms for estimation of quality management software: abstract of the dissertation Dr. of technical sciences / Burakov Vadim Vitaliecich. [Electronic resource]. – Spb, 2010. – 42 c. – Mode of access: http://guap.ru/guap/main/avtoref_burakov.pdf