О МОДЕЛИ ЦИФРОВЫХ ИНФОРМАЦИОННЫХ СИСТЕМ*

А. Е. Гуськов
Институт вычислительных технологий СО РАН, Новосибирск, Россия
e-mail: guskov@ict.nsc.ru

In this paper a few models of digital information systems are considered. A model of patterned multi-styled information systems is formulated; its advantages and disadvantages are identified. Results of practical implementation of the model are presented.

Введение

В настоящее время уже смело можно говорить о том, что Интернет утвердился в роли главного поставщика всевозможной информации. Любая сколько-нибудь крупная компания считает своим долгом иметь Интернет-представительство. Газеты, журналы, библиотеки стремятся предоставить наиболее полный и удобный доступ к своим электронным ресурсам. Для решения подобных задач создаются различные информационные системы (ИС), взаимодействие с которыми пользователи осуществляют в том числе через Web как наиболее популярную и доступную информационную среду.

При разработке ИС большую роль играет то, каким образом публикуется информация, насколько она доступна для различных пользователей. Чаще всего ИС оказывается спроектированной для взаимодействия только с одним типом клиентов, например человеком. Точнее, она не предусматривает единого механизма работы с разнородными клиентами. Поэтому при реализации таких механизмов информационные системы превращаются в разрозненный набор компонентов, которые настроены на конкретного пользователя и нередко дублируют друг друга. Например, библиотечная система может предоставлять web-доступ к своим архивам для Интернет-посетителей, отдельный интерфейс для обслуживания собственных серверов, находящихся в удаленных филиалах библиотеки, набор сервисов для взаимодействия с другими библиотеками и еще несколько компонентов для обслуживания автоматических поисковых агентов с разными задачами и функциональностью.

В этой статье предлагается общая модель цифровых информационных систем, устраняющая, в частности, указанный недостаток. Сначала даются базовые определения и описываются некоторые свойства информационных систем. После описания характеристик общей модели строится частная модель шаблонных мультистилевых информационных систем и производится анализ ее адекватности и применимости. Рассматриваются некоторые вопросы практической реализации модели.

*Работа выполнена при частичной поддержке Президентской программы НШ (№ 2314.2003.19).

1. Общая модель цифровых информационных систем

Информационную систему можно определить как множество $\{r_i\}_{i\in\chi}$ связанных между собой ресурсов:

$$is = \langle\{r_i\}_{i\in\chi}\rangle = \langle R_\chi\rangle,$$

где $\chi \subset \{uid\}$ — подмножество уникальных идентификаторов. Ресурс является *ресурсом*, если он имеет уникальный идентификатор[1]. В этом определении ИС ничего не говорится о природе или структуре ресурсов, подразумевается лишь наличие абстрактных связей, которые и объединяют их в систему, позволяя говорить о такой совокупности как о едином целом. Очевидно, в класс систем, определенных указанным образом, попадает большое число объектов различной природы, например библиотечные каталоги или web-сайты.

Прежде всего следует заметить, что понятие ресурса носит слишком общий характер, и поэтому основным объектом манипуляции информационных систем является частный случай ресурса — *документ*. На данном этапе определим документ d как ресурс, обладающий структурой и содержащий информацию о реальном мире.

Электронные документы должны иметь ряд преимуществ по сравнению с документами другой природы, поскольку, как правило, обладают автоматизированными средствами каталогизирования, хранения, поиска и изменения. Более того, процедура получения твердой копии электронного документа заметно проще, чем обратная процедура.

Можно сказать, что здесь документы информационной системы рассматриваются как объекты независимого происхождения и существующие независимо друг от друга. Наиболее наглядный пример — это первые web-сайты, представляющие собой набор HTML-страниц, каждая из которых хранилась в виде отдельного статичного файла. Недостатки такой организации быстро стали очевидными — при большом количестве документов сайт становится плохо управляемым и не способен быть постоянным источником актуальной информации, поскольку требует больших человеческих ресурсов для обновления. Поэтому сайты с хранилищем документов на основе файлов, как правило, используются для создания систем с небольшим числом документов, содержащих редко обновляемую информацию.

Поскольку управление большим количеством документов по отдельности представляется неэффективным, необходимо выделить подкласс информационных систем, в которых операции можно будет осуществлять с целыми группами документов. Один из наиболее хорошо зарекомендовавших себя подходов к созданию ИС основывается на понятии коллекции. *Коллекцией* называется множество документов, имеющих одинаковую структуру и одну и ту же тематическую направленность.

Прежде чем перейти к более строгому определению коллекции, следует пояснить, что такое структура документа. Один из главных признаков информационной системы (или электронной библиотеки) — существование средств ведения каталогов документов, которые позволяют реализовать функции “эффективного” поиска и классификации документов. Для организации каталога документов ИС используется понятие *метаинформации* или *метаданных* документа.

Метаданные можно разделить на два типа: описательные и структурные. *Описательные метаданные* содержат семантическую информацию о документе — его название, автор, тематическая категория. *Структурные метаданные* определяют синтаксис, с помо-

[1]Это определение, по нашему мнению, является наиболее точным переводом определения, опубликованного WWW-консорциумом “A resource can be anything that has identity” [1].

щью которого записываются описательные метаданные.

Ранее документ был определен как ресурс, содержащий информацию и обладающий структурой. Структура документа задается набором элементов, между которыми могут быть определены отношения. Далее будем считать, что структура документа описывается его структурными и описательными метаданными.

Метаданные документа могут быть указаны различными способами. Один из них известен еще со времен языка разметки текстов SGML[2], где для каждого SGML-документа существовало его DTD-определение[3], которое строго специфицировало структуру разметки. Позже концепция DTD легла в основу XML-схем — нового средства описания структуры разметки, снабженного большей функциональностью, чем его предок. Как и DTD-описание, XML-схема определяла элементы структуры документа и отношения между ними. Добавление возможностей описания качественно новых типов отношений привело к созданию технологии RDFS[4], а еще позже — языка онтологических описаний OWL[5].

Определение метаданных документа, согласно рекомендациям OSI[6], должно осуществляться посредством использования схем данных. Схемы данных являются спецификациями элементов структуры документа и их семантического значения. Например, схема Dublin Core декларирует метаданные для составления самых общих описаний электронных документов. Отметим, что рекомендации группы DCMI [5] предлагают, чтобы все другие существующие и создаваемые схемы данных включали в себя обязательный набор элементов Dublin Core для унификации доступа к ним.

Перейдем к описанию формальной модели ИС и дадим несколько определений.

Определение 1. *Скриптом назовем функцию* $g : A \to D$*, где* A *— некоторая параметризация множества документов данной коллекции. Каждому элементу* $\alpha \in A$ *скрипт сопоставляет документ* $d = g(\alpha)$*, структура которого идентична структуре других документов этой коллекции, а содержание документа определяется параметром* α*.*

Каждый документ информационной системы задается парой (*структура*, *наполнение*), где структура задается с помощью функции-скрипта, а наполнение определяется аргументом, передающимся этой функции.

Обозначим множество всех документов ИС

$$D_X = \{d_i\}_{i \in X} \subset D,$$

где $X \subset \{uid\}$, D — множество всех документов. Определим *коллекцию документов* γ как множество

$$K_\gamma = \{d \in D : \exists \alpha \in \mathcal{A}^n, g_\gamma(\alpha) = d\}, \gamma \in \Gamma.$$

Здесь $\alpha = \{\alpha_1, ..., \alpha_n\} \in \mathcal{A}^n$ является кортежем из n параметров, определяющих документ в этой коллекции, $\mathcal{A}^n$ — декартово произведение из n пространств значений всех параметров; функция $g_\gamma : \mathcal{A}^n \to D$ есть *скрипт*, который отображает кортеж параметров α в документ коллекции; Γ — множество коллекций информационной системы; каждой коллекции γ однозначно соответствует скрипт g_γ.

Заметим прежде всего, что обозначение $\mathcal{A}^n$ в известной степени условно. В данном случае не предполагается, что все параметры однородны и их значения принадлежат одному

[2]SGML — Standard Generalized Markup Language [2].
[3]DTD — Document Type Definition.
[4]RDFS — Resource Description Framework Schema [3].
[5]OWL — Ontology Web Language [4].
[6]OSI — Open System Interconnection.

и тому же множеству A, поскольку они могут иметь булевские, числовые, строковые и другие типы. Здесь A являет собой своего рода собирательный образ множеств значений различных типов, а $\mathcal{A}^n$ — декартово произведение любой их конечной комбинации. Также явно не указывается, но в дальнейшем предполагается, что все коллекции имеют свои собственные наборы параметров $\mathcal{A}^n$, не зависящие друг от друга.

Теперь мы можем определить *коллекционную информационную систему* is_K как информационную систему, состоящую из коллекций документов:

$$is_K = \langle \Gamma, \mathcal{A}^n, G \rangle \sim \langle D_X \rangle,$$

где $G = \{g_\gamma\}_{\gamma \in \Gamma}$ — множество скриптов. Это соотношение в точности означает, что

$$D_X = \bigcup_{\gamma \in \Gamma} \{g_\gamma(\alpha) : \alpha \in \mathcal{A}^n\}.$$

2. Мультистилевые информационные системы

Рассмотрим понятие документа, определенного нами как носитель информации, обладающий некоторой структурой. Для передачи документа между его отправителем и получателем должна существовать договоренность о том, в какой форме он будет представлен, каким образом будет организован поток (слов, знаков, символов), передаваемый через канал связи, и как его нужно интерпретировать. Действительно, один и тот же документ может быть отображен различными способами: для текстовых устройств, для графических устройств с низким или высоким расширением, для специализированных агентов, которые "понимают" информацию только в определенной схеме данных. Поэтому любому документу должно соответствовать одно или несколько его представлений (образов), каждое из которых согласовано с некоторым форматом.

Определение 2. *Форматом ϕ называется контекстно-свободная грамматика, задающая множество образов документов (или, другими словами, язык образов документов) L_ϕ.*

Определение 3. *Образом документа в формате ϕ называется последовательность символов $d^\phi \in L_\phi$, задаваемая тройкой (структура, наполнение, формат).*

Рассмотрим понятие *внутреннего представления документа* (ВПД) как некоторый универсальный образ документа, имеющий специфичный *внутренний формат* ϕ_{int}. Универсальность здесь понимается в том смысле, что из этого образа можно получить образ того же документа в любом формате. При этом универсальный образ не включает в себя сведений о других форматах. Можно сказать, что внутреннее представление документа является описанием его содержания, которое отображается в различные образы документа в зависимости от требуемого формата.

Определение 4. *Внутреннее представление документа есть образ документа в формате ϕ_{int}. Соответственно, язык образов документов $L_{\phi_{int}}$ называется языком внутреннего представления документов.*

Определение 5. *ВПД-функция есть функция $c : \mathcal{A}^n \to L_{\phi_{int}}$, отображающая кортеж параметров в ВПД.*

Определение 6. *Стиль для формата ϕ есть функция $s_\phi : L_{\phi_{int}} \to L_\phi$, возвращающая образ документа по его внутреннему представлению.*

Теперь можно сказать, что *образ документа в формате* ϕ есть $d_\phi = s_\phi(c(\alpha))$ и он задается тройкой (c, α, s_ϕ).

Данные определения носят достаточно абстрактный характер, поскольку они не делают никаких предположений о том, как их следует реализовывать на практике. Информационные системы, использующие эту модель, должны иметь собственный язык внутреннего представления и реализацию всех функций.

Определение 7. *Мультистилевая информационная система (МИС) есть пятерка*

$$is_M = \langle \Gamma, C, S, \Phi, \mathcal{A}^n \rangle.$$

Здесь $C = \{c_\gamma\}_{\gamma\in\Gamma}$ — ВПД-функции; $S = \{s_{\gamma,\phi}\}_{\gamma\in\Gamma,\phi\in\Phi}$ — стилевые функции; Φ — конечное множество допустимых форматов документов. В мультистилевых системах получение документа из коллекции $\gamma \in \Gamma$ по данному кортежу параметров $\alpha \in A$ в формате ϕ обеспечивается с помощью композиции преобразований $s_{\gamma,\phi} \circ c_\gamma(\alpha)$.

Полученную функцию $df_{\gamma,\phi}(\alpha) = s_{\gamma,\phi} \circ c_\gamma(\alpha)$ назовем *функцией документоформирования*. В большинстве информационных систем запрашиваемый документ создается серверными программами (скриптами), которые являются не чем иным, как реализациями документоформирующих функций $df_{\gamma,\phi}$. Это наиболее простой и широко используемый подход при разработке информационных систем "с нуля". Его основной недостаток заключается в том, что коллекции имеют фиксированный конечный формат документов (обычно, HTML) и поэтому ориентированы только на один тип пользователя, которым почти всегда является человек. При этом логическая структура и оформление документа оказываются неотделимыми друг от друга. Другими словами, функция документоформирования не может быть разложена на ВПД-функцию и функцию стиля. Такие системы мы будем называть *скриптовыми информационными системами*, определяемыми как четверка:

$$is_S = \langle \Gamma, \mathcal{A}^n, DF, \Phi \rangle,$$

где $DF = \{df_{\gamma,\phi}\}_{\gamma\in\Gamma,\phi\in\Phi}$.

Определение 8. *Две информационные системы называются эквивалентными, если эквивалентны множества создаваемых ими образов документов.*

Утверждение. *Класс скриптовых информационных систем эквивалентен классу коллекционных информационных систем: $IS_S = IS_K$.*

Доказательство: 1. $IS_S \subset IS_K$. Для каждой $\langle \Gamma, \mathcal{A}^n, DF, \Phi \rangle \in IS_S$ рассмотрим $\Gamma' = \{\gamma_\phi : \gamma \in \Gamma, \phi \in \Phi\}$ — $|\Phi|$-кратное расширение множества коллекций Γ. Тогда $\langle \Gamma', \mathcal{A}^n, G \rangle$, где $G = \{g_{\gamma'} = df_{\gamma,\phi} : \gamma' \in \Gamma', \gamma' = \gamma_\phi\}$, — информационная система из класса IS_K, эквивалентная исходной.

2. $IS_S \supset IS_K$. Возьмем $\Phi = \{\phi\}$ ($|\Phi| = 1$) и $df_{\gamma,\phi} = g_\gamma$, тогда $\langle \Gamma', \mathcal{A}^n, \{df_{\gamma,\phi}\}, \Phi \rangle$ — искомая система из класса IS_S. Утверждение доказано. □

Прямым следствием утверждения и данных определений является

Следствие. *Между классами информационных систем имеют место следующие отношения*:

$$IS_M \subset IS_S = IS_K \subset IS.$$

Первая часть доказательства утверждения демонстрирует недостатки скриптовых информационных систем по сравнению с мультистилевыми. Каждая коллекция должна быть тиражирована столько раз, сколько различных образов должны иметь порождаемые ей документы.

В этом смысле подход, применяемый для создания МИС, более адекватен. Первое, что необходимо отметить, в МИС выделяются две составляющие образа электронного документа: *внутреннее представление документа* и *стиль*. Физическое разделение этих

двух компонентов является существенным достоинством, поскольку операции модификации содержания и изменения стиля, как правило, не связаны между собой и, более того, могут выполняться различными специалистами в области информационных технологий и web-дизайна.

Вторая примечательная деталь состоит в том, что существование *нескольких стилевых представлений документов* делает МИС доступной для большего числа различных типов клиентов, не прибегая к дублированию описаний коллекций. ВПД-функции являются единственным информационным источником, имеющим минимальную избыточность, что облегчает их модификацию. Добавление нового формата представления информационной системы заключается в определении стилей для соответствующих коллекций, а не в фактическом дублировании всей системы, как это требуется в скриптовых информационных системах.

Наконец, последняя важная особенность МИС заключается в наличии *явного представления содержащейся в ней информации*. Именно оно позволяет не только адаптировать документы для запрашивающих клиентов, но и создавать дополнительные сервисы, которые было бы трудно реализовать в другой архитектуре. Например, явно выраженное внутреннее представление документа позволяет автоматически его классифицировать и отнести к некоторой категории. Таким образом, становится возможным создание подсистем каталогизации и поиска документов, что является важной частью информационных систем.

3. Модель создания документов

Рассмотрим ВПД-функцию c_γ, которая определяет процедуру составления внутреннего представления документа по данному кортежу параметров. В скриптовых системах она может реализовываться с помощью серверных скриптов (Perl, PHP, ASP, JSP, Servlets) как программный алгоритм, который формирует образ документа как последовательность символов, удовлетворяющих требуемому формату. Автор видит такой подход неконструктивным, поскольку такие скрипты обычно оперируют символьными данными, а не информационными объектами.

Определение 9. *Для каждой коллекции $\gamma \in \Gamma$ в языке внутренних представлений документов выделим элемент $p_\gamma \in L_{\phi_{int}}$, который назовем информационным шаблоном.*

Определение 10. *Трансформер контента — это функция $tr : \mathcal{A}^n \times L_{\phi_{int}} \to L_{\phi_{int}}$.*

Определение 11. *Функция наполнения есть $f_\gamma(\alpha, p_\gamma) = tr_{n_\gamma}(\alpha, tr_{n_\gamma - 1}(\alpha, ...tr_1(\alpha, p_\gamma)...))$, такая, что $c_\gamma(\alpha) = f_\gamma(\alpha, p_\gamma)\ \forall \alpha \in \mathcal{A}^n$.*

Информационный шаблон описывает базовую структуру документа, которая обрабатывается и заполняется реальными данными функцией наполнения. Функция наполнения есть композиция трансформеров контента, каждый из которых является логически независимым преобразованием шаблона. Для преобразования ВПД трансформер использует параметры запроса α и инструкции — фрагменты информационного шаблона, специфицирующие деятельность того или иного трансформера.

На рис. 1 представлена предложенная модель создания образов документов. При поступлении запроса система выбирает соответствующий информационный шаблон p_γ, набор трансформеров $tr_1, ...tr_{n_\gamma}$ и функцию стиля $s_{\gamma,\phi}$ для требуемого формата документов ϕ. На первом этапе к шаблону последовательно применяются трансформеры, которые изменяют его, наполняя специфичной для этого документа информацией. В результате бу-

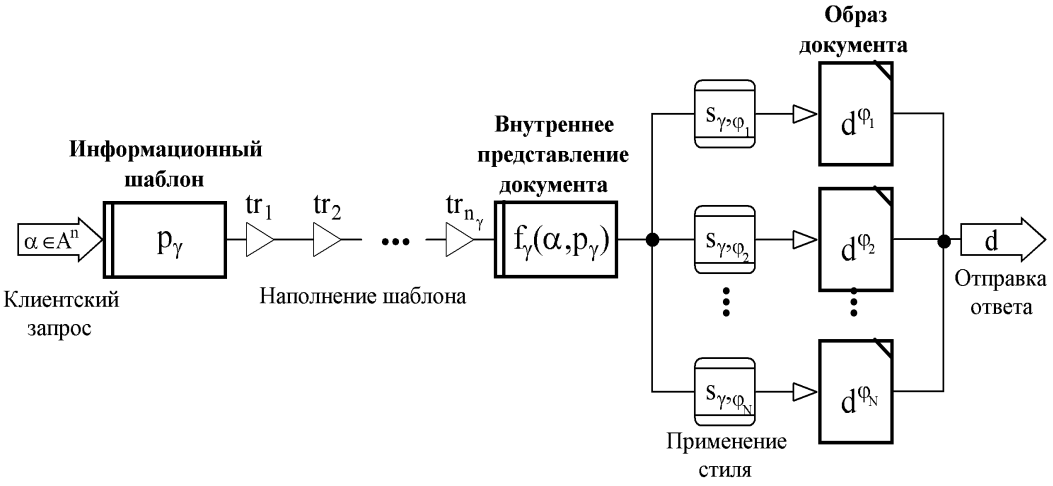

Рис. 1. Модель создания образов документов.

дет получено внутреннее представление документа (т. е. образ документа в формате ϕ_{int}), которое согласно выбранному стилю преобразуется к конечному требуемому формату ϕ.

Определение 12. *Шаблонной мультистилевой информационной системой называется шестерка*

$$is_{PM} = \langle \Gamma, P, Tr, S, \Phi, \mathcal{A}^n \rangle,$$

где $P = \{p_\gamma\}_{\gamma \in \Gamma}$ *— множество информационных шаблонов;* $Tr = \{tr_{\gamma,i}\}_{i=1..N_\gamma, \gamma \in \Gamma}$ *— множество трансформеров для каждой коллекции.*

Очевидно, шаблонная мультистилевая информационная система наследует все характеристики мультистилевой системы (т. е. $IS_{PM} \subset IS_M$), о которых говорилось ранее. Прежде чем понять, какими еще свойствами обладают такие системы, рассмотрим подробнее, что представляют собой трансформеры. Трансформер — это функция, преобразующая внутренне представление документа (или информационный шаблон). Каждый трансформер функционирует независимо, хотя и может изменять любые части шаблона, в том числе те, которые были изменены другими трансформерами. Таким образом, трансформер должен реализовывать логически завершенное действие над ВПД. Приведем примеры таких преобразований.

1. *Трансформер подстановки* просматривает ВПД в поисках определенных инструкций, которые заменяет на соответствующие им данные. Вариации этого трансформера могут использоваться в различных целях. Во-первых, таким образом в ВПД переносятся любые значения параметров запроса, которые затем могут быть использованы при составлении инструкций для других трансформеров. Во-вторых, в документ могут быть подставлены данные, взятые из внешних источников.

Примечательно, что с помощью трансформера подстановки можно создавать локализованные версии документов, т. е. документы, адаптированные для чтения пользователями из разных стран. Так, текстовые строки на естественном (русском или английском) языке должны находиться не в исходном шаблоне, а в специальном хранилище, из которого трансформер согласно инструкции выберет ту, которая будет соответствовать требуемой локализации. Отображение чисел, дат и других данных тоже может отличаться в разных локализациях и поэтому должно быть отформатировано соответственно.

2. В определенном смысле информационные системы являются посредником между недоступной напрямую базой данных и Интернет-пользователями. Поэтому центральное

место в реализации модели информационных систем должен занимать *трансформер взаимодействия с хранилищами данных*. Заметим, что в роли хранилища могут выступать не только традиционные реляционные базы данных, но и файловые архивы, серверы Z39.50, каталоги LDAP, различные сервисы.

Для функционирования такого трансформера в инструкциях информационного шаблона должны быть отображены параметры запроса к хранилищу (откуда и какие нужно взять данные) и то, каким образом следует отобразить полученные результаты. Кроме этого, необходимо учесть возможность выполнения нескольких последовательных запросов, каждый из которых может использовать результаты предыдущих. Далее обсуждать вопросы реализации этого трансформера не имеет смысла, поскольку в значительной степени она зависит от типа используемого хранилища данных, а также от требований к функциональности трансформера и должна обсуждаться в контексте определенной информационной системы.

3. В качестве еще одного примера можно привести *трансформер обработки условий*, который реализует логическую обработку содержаний на самом примитивном уровне. Он функционирует на основе трансформера подстановки, заменяя инструкции на данные в зависимости от выполнения или невыполнения указанного условия. Кроме того, трансформер может удалять фрагменты содержания, которые не должны присутствовать в данном документе при выполнении определенного условия.

Очевидно, что каждая информационная система по-своему специфична и невозможно создать универсальный комплект трансформеров, который бы подошел для создания любой из них. Тем не менее можно создать стандартный набор, на основе которого было бы проще реализовать все особенности новой информационной системы, дополнив его специализированными трансформерами. Выше приведены трансформеры, которые должны войти в стандартный набор.

4. Сравнительный анализ моделей

Рассмотрим некоторые достоинства и недостатки предложенной модели шаблонных мультистилевых информационных систем по сравнению с другими моделями. К сожалению, автору статьи неизвестно об аналогичных исследованиях и их результатах, поэтому при анализе единственно возможным представляется сравнение со скриптовыми информационными системами.

Как уже было отмечено, предложенная модель обладает всеми достоинствами мультистилевых информационных систем, а также некоторыми дополнительными положительными характеристиками:

— наличие отдельных описаний содержания (ВПД) и стиля документа;

— явное представление семантики описываемой информации;

— возможность нескольких стилевых представлений информации для различных типов клиентов;

— использование уже готовых трансформеров сводит создание новой ИС к описаниям шаблонов и стилей, позволяя разработчикам сосредоточиться на ее информационном наполнении, а не на кодировании скриптов;

— предложенная модель способствует большей структуризации ИС, разбивая процесс создания на логически обособленные составляющие и упорядочивая получение различных образов одного документа.

Несомненно, предложенная модель является заметно более сложной по сравнению со скриптовой моделью. Это уже само по себе недостаток, поскольку затрудняет восприятие такой архитектуры, что не может не сказаться на успешности ее внедрения. Кроме этого, можно выделить еще два существенных недостатка.

1. Использование этой модели необоснованно и весьма неэффективно для разработки информационных систем, ориентированных на конкретный тип клиента, например на пользователей стандартных браузеров. В этом случае время создания коллекции может увеличиться в несколько раз по сравнению с скриптовым способом.

2. Наличие двух последовательных этапов генерации образа документа негативно сказывается на производительности системы, а именно на времени создания каждого образа. Этот фактор является критическим, поскольку информационная система должна обеспечивать быстрый отклик на поступивший запрос. Предложенная реализация модели подтвердила существование этой проблемы, однако найденные технологические решения позволили добиться приемлемого времени отклика.

В целом можно заключить, что предложенная модель обладает рядом интересных свойств, перспективность которых следует проверить на практике. Основные недостатки модели говорят, прежде всего, о необходимости тщательного анализа применимости модели в том или ином случае. Остальные недостатки носят в большей степени технологический характер и могут быть исправлены при реализации информационных систем.

5. Реализация модели

Для оценки практической применимости предложенной модели была создана система публикации документов SMART (System for Managing Application based on RDF Technology) [6, 7]. Функционально система представляет собой web-сервер, который при поступлении клиентского запроса должен инициировать процесс создания образа документа на основе модели шаблонных мультистилевых информационных систем, по завершении которого полученный образ должен быть отправлен запросившему его пользователю.

При создании системы SMART прежде всего необходимо было выбрать технологическую базу для реализации элементов модели. Ключевым вопросом являлся формат внутреннего представления документов $L_{\phi_{\text{int}}}$, поскольку именно в этом свойстве заключается ее основное отличие от других моделей создания информационных систем. В предыдущей работе [8] автор обосновал использование технологии RDF [9] в качестве наиболее подходящего языка для описания информации. Таким образом, в качестве формата $L_{\phi_{\text{int}}}$, используемого для описания информационных шаблонов и ВПД, был выбран RDF.

Поскольку RDF в значительной степени является дочерним языком по отношению к языку разметки XML, он может использовать целый ряд программных средств, разработанных для последнего. В частности, любая XML-сериализация RDF-документа может быть подвергнута XSLT-преобразованию [10], результатом которого будет новый XML-документ, созданный согласно инструкциям XSLT-шаблона. Являясь стандартным средством для преобразования XML-документов, XSLT-шаблоны были использованы в системе SMART для реализации функций стиля. Таким образом, в системе стало возможным создавать образы документов в любом формате семейства XML, в том числе HTML, WML и RDF. Этот список охватывает пользователей обычных браузеров, владельцев карманных компьютеров, программных агентов, функционирующих согласно идеологии Semantic Web (`http://www.w3.org/2001/sw`), и других клиентов, работающих с XML-форматом.

Система SMART была разработана на основе технологии *servlets*, для реализации отдельных компонентов было выбрано свободно распространяемое программное обеспечение: среда разработки Java SDK 1.4 и сервер web-приложений Tomcat [11]. Для операций над RDF-документами использовались библиотека Jena — разработка лаборатории Hewlett Packard [12], библиотеки Xerces и Xalan для работы с XML-документами и XSLT-преобразованиями. Роль трансформеров в системе играют Java-классы, реализующие интерфейс `ModelTransformer` (здесь под моделью подразумевается RDF-модель). Таким образом, запуск трансформера является вызовом метода `transform` с текущим RDF-шаблоном в качестве параметра. По окончании работы метод возвращает измененный шаблон, который передается следующему трансформеру.

В качестве пробного проекта был реализован прототип пользовательской части уже существующей информационной системы "Конференции" [13], которая в настоящее время находится по адресу `http://www.nsc.ru/ws`. Она состоит из нескольких коллекций, содержащих различную информацию о конференциях, которые проводятся институтами СО РАН: Participant и Report предоставляют данные об участниках и докладах, Part_list и Rep_list — документы с соответствующими списками, Section — расписание работы конференции для каждой секции, Conf_list — списки конференций, проводимых в указанной организации в определенном году. Новая версия информационной системы имела такую же функциональность и сохраняла старую структуру коллекций и документов, но была реализована полностью на основе системы SMART. После этого был проведен сравнительный анализ некоторых характеристик новой и старой информационных систем.

5.1. Тест трудоемкости разработки информационных систем

Для проверки тезиса об увеличении размеров исходных описаний коллекций были исследованы PHP-скрипты старой версии, RDF-описания и стили XSLT нового прототипа (табл. 1). Заметим, что сравнение физических размеров соответствующих файлов не является в полной мере показательным — даже программы, реализующие один и тот же алгоритм, но написанные на разных языках, могут ощутимо различаться размерами. Более того, в количество килобайт, указанных в таблице, входят символы форматирования исходного кода для облегчения его чтения и понимания разработчиком. Поэтому был введен еще один параметр сравнения — логический размер, который равен количеству основных синтаксических единиц описания. В PHP-скриптах синтаксической единицей является оператор, в RDF-документах — утверждение, в XSLT-документах — XML-тег. Сравнение этих элементов, несомненно, имеет смысл, поскольку именно в таких терминах

Таблица 1

Размеры исходных описаний коллекций

	PHP-скрипт		Содержание (RDF)		Стиль (XSLT)	
Имя коллекции	кбайт	Операторы	кбайт	Утверждения	кбайт	XML-теги
Participant	3.4	80	3.6	40	1.8	30
Part_list	3.3	80	4.1	50	1.7	20
Report	3.6	80	3.0	40	0.8	10
Rep_list	3.5	80	4.9	60	1.1	20
Section	5.0	120	7.0	80	2.5	30
Conf_list	3.0	70	4.8	60	1.2	20
Допол. библиотеки	5.5	120	—	—	14.5	200

мыслит разработчик, создавая новую коллекцию.

Нужно сказать, что приведенные цифры носят весьма условный характер, но все же они позволяют оценить разницу в трудоемкости описания коллекций тем или иным способом. Из таблицы видно, что физический размер RDF-описаний, как правило, на 20–50 % больше, чем у исходников PHP, а суммарный физический размер RDF и XSLT и вовсе может двукратно превосходить соответствующие PHP-скрипты. Вместе с тем следует отметить, что логические размеры в обоих случаях примерно одинаковые. В частности, это означает, что средний размер оператора PHP в 2–2.5 раза меньше, чем средний размер RDF-утверждения. Действительно, используемая XML-нотация для записи RDF достаточно громоздка, однако на сегодняшний день это оптимальный способ хранения и передачи RDF-документов в виде файлов.

Нельзя не отметить еще одну деталь. Размер стилевых описаний, как физический, так и логический, в несколько раз меньше, чем размеры скриптов. Следовательно, добавление новых форматов образов документов эффективнее осуществлять в SMART-системе, так как для этого необходимо лишь создание нового стиля. Таким образом, подтвержден тезис о том, что предложенную архитектуру предпочтительнее использовать для информационных систем с несколькими форматами представлений документов.

5.2. Тест производительности системы SMART

Другой недостаток, выявленный при теоретическом анализе модели, связан с возможным увеличением времени создания образов документов. Было создано несколько тестов производительности системы SMART (табл. 2), в которых фиксировались время генерации ВПД (этап I) и время применения стиля (этап II). Всего было проведено 8 тестов для документов, ВПД которых имеет разные размеры, выражающиеся в количестве утверждений. Кроме того, в таблице указано количество простых SQL-запросов (хранилище данных реализовано на основе реляционной СУБД MySQL) типа `SELECT`, которые выполнены при создании страницы. Каждый тест проводился 10 раз, в таблицу заносилось среднее время. Колебания значений относительно среднего составили до 40 %. Для сравнения в последнем столбце приведено время генерации аналогичных страниц с использованием традиционных PHP-скриптов.

Из этой таблицы можно сделать следующие выводы.

1. Время работы первого этапа зависит от размера ВПД и от количества выпол-

Таблица 2

Тесты производительности. Время генерации HTML-документов, с

Тест	Intel P1, 233 МГц RAM 128 Мбайт			Intel P3, 1 ГГц RAM 256 Мбайт			Intel P4, 2 ГГц RAM 512 Мбайт			PHP
	I	II	Всего	I	II	Всего	I	II	Всего	
50 утв., 3 запр.	0.6	0.2	0.8	0.1	0.1	0.2	0.05	0.05	0.1	0.1
100 утв., 4 запр.	0.8	0.3	1.1	0.2	0.2	0.4	0.1	0.1	0.2	0.1
270 утв., 18 запр.	3.8	0.7	4.5	0.6	0.6	1.2	0.3	0.3	0.6	0.2
530 утв., 23 запр.	3.8	0.9	4.7	0.7	0.9	1.6	0.3	0.8	1.1	0.2
650 утв., 2 запр.	2.2	1.2	3.4	0.5	1.2	1.7	0.1	1.0	1.1	0.2
1000 утв., 62 запр.	6.8	1.9	8.5	1.3	1.5	2.8	0.5	1.3	1.8	0.4
1300 утв., 68 запр.	9.2	2.4	11.6	1.7	2.1	3.8	0.6	1.9	2.6	0.6
1800 утв., 68 запр.	13.1	3.1	16.2	2.4	2.8	5.2	0.7	2.3	3.0	0.6

ненных запросов. Также наблюдается сильная зависимость от серверного оборудования. На "быстром" оборудовании производительность первого этапа сравнима с производительностью PHP-скриптов.

2. Время работы второго этапа линейно зависит от размера ВПД. Причем это время также зависит от вычислительной мощности сервера, но уже в меньшей степени: при увеличении параметров производительности примерно на порядок производительность на втором этапе улучшилась не более чем на 30 %.

3. Таким образом, на суммарную производительность двух этапов на "медленном" оборудовании влияет первый этап, а на "быстром" — второй.

4. На "быстром" оборудовании суммарная производительность системы SMART в 2–5 раз хуже производительности PHP-скриптов.

Эти выводы подтвердили гипотезу о заметном ухудшении производительности информационных систем. Однако время создания образов документов, зафиксированное на "быстром" оборудовании (до 3 с), можно считать приемлемым. Следует также заметить, что существует ряд способов дальнейшего повышения производительности: оптимизация программного кода, подсистемы кэширования документов, установка более мощного оборудования. Это позволяет заключить, что с технической точки зрения эксплуатация модели шаблонных мультистилевых информационных систем возможна.

Заключение

Рассмотрены несколько классов информационных систем и указаны некоторые их свойства. Даны определения базовых понятий ресурса, электронного документа, коллекции документов. Основное внимание уделено описанию модели шаблонной мультистилевой информационной системы. Главными особенностями предложенной модели являются:

— выделенное понятие коллекции как множества документов, имеющих одинаковую структуру и описывающих одну и ту же сущность;

— выделение в документе структурной, содержательной и презентационной частей;

— возможность представлять одну и ту же информацию различными стилями, т. е. на основе одного документа создавать его образы в различных форматах.

Указаны достоинства этой модели в сравнении с традиционной скриптовой моделью. В процессе теоретического анализа сформулированы некоторые недостатки, которые потенциально могут стать критичными для всей модели.

Приведены результаты практической реализации предложенной модели. Для исследования технологических возможностей ее применения проведены тест трудоемкости и тест производительности. Они показали, что недостатки, обозначенные в теоретическом анализе, имеют место, но не являются критичными и в целом модель шаблонных мультистилевых информационных систем может быть использована на практике.

Список литературы

[1] Berners-Lee T., Fielding R., Masinter L. Uniform Resource Identifiers (URI). Generic Syntax. RFC 2396. http://www.ietf.org/rfc/rfc2396.txt

[2] Sperberg-McQueen C.M., Burnard L. A Gentle Introduction to SGML. http://www.isgmlug.org/sgmlhelp/g-index.htm

[3] Manola F., Miller E., McBride B. RDF Vocabulary Description Language 1.0: RDF Schema; W3C Recommendation, 10 Feb. 2004. http://www.w3.org/TR/rdf-schema/

[4] McGuinness D. L., Harmelen F. OWL Web Ontology Language Overview. W3C Recommendation 10 Febr. 2004. http://www.w3.org/TR/owl-features/

[5] Dublin Core Metadata Initiative (DCMI). http://www.dublincore.org

[6] Шрайбман В.Б., Гуськов А.Е. Разработка информационных систем на основе RDF-технологии // Тр. XLI Междунар. научной студен. конф. "Студент и научно-технический прогресс", НГУ. Новосибирск, 2003. Ч. 1. С. 143–150.

[7] SMART: System for Managing Applications based on RDF Technology. http://web.ict.nsc.ru/smart

[8] Гуськов А.Е. Модель документа web-ориентированных информационных систем на основе RDF // Вест. НГУ. Сер. Информационные технологии в образовании. Новосибирск, 2004. Т. 1, вып. 2. С. 27–35.

[9] Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation. http://www.w3.org/TR/rdf-concepts/

[10] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 Nov. 1999. http://www.w3.org/TR/xslt

[11] Apache Jakarta Tomcat. http://jakarta.apache.org/tomcat/index.html

[12] Jena — A Semantic Web Framework for Java. http://jena.sourceforge.net/

[13] Федотов А.М., Гуськов А.Е. Информационная система "Конференции" // Тр. VII Междунар. конф. по электронным публикациям "EL-Pub2002". http://www.ict.nsc.ru/ws/show_abstract.dhtml?ru+45+4503

Поступила в редакцию 18 марта 2005 г.