Методология сбора и оценки данных, формирующих характеристики научных проектов*

Д. С. Богданов, В. А. Кац, А. С. Корягин, Е. Л. Плискин

Аннотация. Информационные системы крупных научных агентств, таких как РФФИ и РГНФ, ежегодно обрабатывают десятки тысяч заявок на гранты и отчётов о выполненной работе. Документы заполняются заявителями через веб-интерфейс информационной системы (ИС) в виде форм, в совокупности образующих информационную модель (ИМ) проекта. В статье рассматриваются особенности автоматизации научных агентств, связанные с динамикой ИМ проектов. Особое внимание уделяется вопросам перенастройки ИС на этапе эксплуатации для адаптации к изменениям ИМ, модернизации файлов модели, генерации кода в системах с открытым исходным кодом, каталогам форм, хранящих структуры документов. Обсуждаются различные программные реализации MVC-технологии (model-view-controller). Описывается технология инкрементального сохранения проектных форм, для минимизации трафика при редактировании.

Ключевые слова: *веб-интерфейс, хранимые объекты, каталог форм, генерация кода, model-view-controller, РФФИ, РГНФ, MVC, JavaScript.*

Введение

Составной частью общегосударственной системы управления научной деятельностью являются научные агентства, осуществляющие распределение грантов на проведение научных исследований на конкурсной основе. Крупные агентства, такие как Российский фонд фундаментальных исследований (РФФИ), Российский гуманитарный научный фонд (РГНФ), Российский научный фонд (РНФ), ежегодно рассматривают десятки тысяч заявок на проекты. Для максимально широкой доступности информационные системы (ИС) крупных фондов снабжены веб-интерфейсом. Заявки на проекты и отчёты о выполненной работе заполняются грантозаявителями в браузере через веб-интерфейс ИС в виде структурированных документов, т. е. форм. Набор показателей, предусмотренных в формах заявок и отчётов, в совокупности образует информационную модель (ИМ) проекта.

В данной статье рассматриваются особенности разработки и эксплуатации систем автоматизации научных агентств, связанные с динамикой информационной модели проектов. Динамика ИМ проектов имеет выраженную годовую составляющую вследствие годового цикла бюджетного финансирования. Год от года могут уточняться показатели, по которым оцениваются заявки и отчёты для различных типов проектов. Кроме того, модель проекта может адаптироваться в соответствии с условиями определённого конкурса. В силу этих факторов информационные модели проектов могут меняться в ходе эксплуатации, но изменения носят дискретный характер и могут сопровождаться человеко-машинными процедурами для перенастройки ИС.

Роли участников процесса перенастройки ИС следующие. Изначально человеко-машинная процедура перенастройки информационной системы для адаптации к изменениям информационной модели проектов может проектироваться **разработчиками** программного обеспечения (ПО) ИС. Изменения инициируются **специалистами отделов** Фонда, которые могут курировать определённые области науки или определённые виды конкурсов. Заказ на внесение изменений поступает к **IT-специалистам** Фонда, которые выполняют процедуру перенастройки ИС. Результаты изменений становятся видны **грантозаявителям** и **экспертам** в виде новых форм заявок и отчётов.

Если информационная система разработана на основе программного обеспечения с открытым исход-

* Работа выполнена при частичной финансовой поддержке грантов РФФИ (проект № 14–29–05040 и № 14–29–05049).

ным кодом, то процедура перенастройки ИС может включать регламентированные операции, приводящие к изменению исходного кода, с последующей компиляцией программы. Хотя ручная правка кода IT-специалистами нежелательна, но может применяться автоматическая генерация кода на основе формализованной модели. Другими словами, инструкция по перенастройке ИС может включать описание ручных изменений, которые могут вноситься IT-специалистами в определённые файлы модели, и последующих автоматических шагов для генерации кода и компиляции программы.

Под генерацией кода мы будем понимать выполнение любой программы, выходом которой является код на языке разработки ИС. Входными данными для такой программы могут служить файлы любого формата и содержания: XML, SQL, текстовые файлы свойств и т. п. Генератор кода может быть разработан без больших трудозатрат специально для данной ИС, с учётом всех особенностей ИМ проекта. Наш подход отличается от использования готовых инструментов программной инженерии (CASE, Computer-Aided Software Engineering), предполагающих автоматизацию процесса проектирования и обслуживания программы с использованием модели высокого уровня.

1. Типы проектов и виды форм

Разветвлённая организационная структура агентства находит своё отражение во множестве типов проектов. Различные отделы могут специализироваться как на научных направлениях, так и на видах конкурсов, например на международных конкурсах, или проводимых совместно с другими организациями, на региональных конкурсах и т. п. Если отделы слабо взаимодействуют между собой, то каждый отдел может вносить коррективы в информационные формы для «своих» проектов. При этом в ИС образуются различные **типы проектов**. Для различных типов проектов могут отличаться состав и порядок показателей в форме, а также словесные формулировки подсказок по заполнению форм грантозаявителями. Разумеется, помимо организационных причин, есть и объективные причины для различий в формах между, например, многолетними исследованиями, проводимыми научными коллективами, и такими локальными проектами, как издание рукописи или командировка учёного на конференцию.

Стремление агентства унифицировать условия проведения различных конкурсов приводит к образованию **видов форм**, которые могут использоваться в различных типах проектов. Примеры видов форм: форма «общие сведения о проекте», форма «научное содержание проекта», форма «смета расходов». Если сравнить между собой одинаковые виды форм для различных типов проектов, то можно найти как сходство общей структуры, так и разночтения в некоторых показателях.

Можно рассматривать «тип проекта» как группировку «видов форм»: в каждой такой группе близкие по смыслу формы могут несколько различаться. Кроме того, для различных типов проектов могут отличаться некоторые параметры. Например, может отличаться максимальный процент расходов на организационно-финансовое и техническое сопровождение проекта. Если настраивать такие параметры на более высоком уровне «типа проектов», то один и тот же параметризованный шаблон формы можно использовать для различных типов проектов, что упрощает сопровождение.

С точки зрения динамики информационной модели указанные особенности означают, что при проектировании ИС должны быть предусмотрены возможности на этапе эксплуатации создавать новые типы проектов и новые формы на основе старых; реже приходится создавать совершенно новые виды форм.

2. Типы проектов и типы конкурсов, многолетние проекты

В условиях конкурса перечисляются типы проектов, по которым могут подаваться заявки. Один и тот же тип проекта может использоваться в различных конкурсах. Конкурс может налагать лимиты на сумму финансирования одного проекта и на количество лет выполнения проекта. Конкурс также ограничивает количество поддержанных проектов и общую сумму финансирования всех проектов. Конкурс привязывается к календарному году, в котором будет выделяться финансирование («год конкурса»), хотя заявки могут заблаговременно готовиться в предыдущем году.

Если конкурс предусматривает однократное финансирование и выполнение проекта длительностью не более одного года, то цикл документооборота такого проекта может включать следующие этапы:

- подготовка заявки грантозаявителем в информационной системе;
- проверка и регистрация заявки сотрудником Фонда в электронном виде;
- печать и отправка бумажного варианта заявки грантозаявителем;
- проверка и регистрация бумажного варианта заявки сотрудником Фонда;
- экспертиза проекта и обсуждение проектов в Фонде;
- утверждение финансирования и платёж;
- отчёт грантополучателя о выполненной работе и о расходовании средств.

Если же конкурс допускает проекты продолжительностью более одного года, то финансирование выделяется не сразу на весь проект, а на каждый год отдельно. Начальная сумма финансирования утверждается только на первый год проекта. В конце каждого года проекта грантополучатель представляет промежуточный годовой отчёт о выполненной работе и о расходовании средств, а также заявку на финансирование в следующем году. По окончании проекта представляется итоговый отчёт.

Многолетние проекты вносят в информационную модель такое понятие, как «год оборота» (от термина «документооборот»). Год оборота может отличаться от календарного года. К примеру, осенью 2015 года могут готовиться документы на финансирование в 2016 году, по формам, утверждённым на 2016 год. В таком случае мы говорим, что в заявке фигурирует 2016-й «год оборота». В общем случае в документах на первый год проекта «год оборота» совпадает с «годом конкурса», в заявке на следующий год проекта «год оборота» на 1 больше, а в последнем году проекта «год оборота» формально может быть на 1 больше года окончания проекта. Годом оборота определяется комплект форм, которые заполняются грантозаявителем. В комплект года N входят формы отчётов за год (N – 1) и формы заявок на год N. Пакет документов грантозаявителя в любой год проекта всегда включает формы из одного годового комплекта. Такое решение практично, если комплект форм на год N проходит корректировку и утверждение в Фонде в одном и том же календарном году: (N – 1) или N.

Упомянутые особенности годовой динамики ИС располагают к тому, чтобы структурировать каталог форм по годам и предусмотреть в формах параметры «год оборота» для заявок и «год оборота минус 1» для отчётов.

3. Уровни информационной модели проекта

В информационной модели проекта можно выделить нижний и верхний уровни:

- **Модель информационных объектов**. На нижнем уровне проект представляется набором хранимых в базе данных информационных объектов (DAO — data access object). Классы хранимых объектов могут генерироваться исходя из таблиц и представлений БД. На этом уровне могут решаться задачи проверки прав доступа, универсального поиска объектов и отображения объектов в виде текста. Также на этом уровне могут генерироваться веб-формы для администраторского редактора базы данных. Подробно модель нижнего уровня описана в п. 4 ниже.
- **Каталог форм**. На верхнем уровне проект представляется комплектом содержательных форм. На этом уровне могут генерироваться веб-формы для ввода данных, а также выходные документы для просмотра и печати, в форматах HTML и PDF. Каталог форм описан в п. 6 ниже.

Различные виды изменений могут вноситься как на верхнем, так и на нижнем уровне информационной модели проекта. Так, при добавлении нового показателя в форму может потребоваться добавление столбца в таблицу БД и корректировка модели нижнего уровня. Другие изменения могут касаться только верхнего уровня модели, например, изменение словесной формулировки показателя или перестановка показателей в форме не затрагивают модель нижнего уровня.

4. Генерация кода хранимых объектов

Генерация кода хранимых и транспортных объектов (DAO — data access object и DTO – data transfer object) традиционно применяется в том слое информационной системы, который располагается между нижним уровнем программного интерфейса (API) СУБД и более высоким уровнем прикладной логики. Некоторые разработки такого рода поддерживают сразу несколько платформ, см. например, [2]. Исходными данными для генерации кода могут служить как метаданные БД, так и файлы кода на макроязыке, которые обрабатываются препроцессором [3].

Генерация кода облегчает труд разработчика, устраняя необходимость ручного перечисления в коде многочисленных свойств информационных объектов, и позволяет создавать программное обеспечение (ПО), способное подключаться к различным источникам данных. Генерация кода является характерным признаком CASE-технологий.

В информационных системах *с открытым исходным кодом* генерация кода может выполняться не только на этапе разработки, но и на этапе эксплуатации и сопровождения, как часть процедуры адаптации ИС к изменениям информационной модели.

Использование готовых CASE-средств имеет как достоинства, так и недостатки [5]. Распространено мнение, что CASE-средства могут быть дороги и трудны в изучении, а шлифовка их для нужд конкретной АС может быть затруднена. В тоже время генератор кода для данной конкретной информационной системы зачастую может быть разработан без больших затрат и принести немалую пользу. Мы считаем, что проектировщикам крупных информационных систем с динамической информационной моделью всегда следует оценивать целесообразность разработки собственного специализированного генератора кода.

5. Модель информационных объектов

Модель информационных объектов приближена к структуре базы данных. Для каждой таблицы БД модель предусматривает класс хранимых информационных объектов. Свойства данного класса объектов отвечают, во-первых, столбцам таблицы БД, а во-вторых, столбцам представлений БД (views), которые могут служить для получения краткой или подробной информации о данном типе объектов.

При запуске сервера модель загружается в память из метаданных БД, а также из следующих файлов:

- файлы на языке SQL, описывающие таблицы и представления базы данных, со специальными комментариями. О комментариях в SQL-файлах см. п. 5.2 ниже;
- текстовые файлы свойств, содержащие отображаемые названия столбцов. Например, для столбца «рег_номер» таблицы «проекты» может быть задано отображаемое название «Регистрационный номер»;
- файлы формата XML с дополнительными сведениями о модели.

Модель используется сервером для формирования запросов к БД, а также при проверке и разборе информации, поступающей от клиентской части.

Клиентская часть запрашивает от сервера модели информационных объектов не все сразу, а по мере потребности, для динамической генерации вебстраниц просмотра и редактирования информационных объектов.

5.1. Хранимые объекты и части объектов

В этом разделе статьи мы предлагаем возможный подход к проектированию базы данных научного агентства с использованием объектно- реляционного отображения (ORM, object-relational mapping), общие принципы которого описаны, например, в [1]. Наш подход характеризуется следующими особенностями.

- **Автоматическая нумерация записей**. Во всех таблицах БД первичным ключом является целочисленный столбец «ид», который автоматически формируется базой данных при добавлении каждой новой записи.
- **Части объектов**. Информационный объект может быть либо «одномерным», т. е. храниться в одной записи базы данных, либо «двумерным», т. е. включать запись в «главной» таблице и множество дополнительных записей в одной или нескольких таблицах.
- Пример одномерного информационного объекта: запись справочника должностей.
- Пример двумерного информационного объекта: проект. Включает одну запись в таблице «проекты», а также: список участников проекта, список заполненных грантозаявителем форм, список публикаций, созданных в результате выполнения проекта и т. п.
- **Виды частей**. Дополнительные или «дочерние» записи будем называть «частями информационного объекта». Дочерние записи из одной таблицы БД относятся к одному «виду частей». Список частей одного вида может быть упорядочен по одному или нескольким столбцам, а порядок сортировки фиксирован в описании модели информационного объекта. В каждой части имеется столбец с кодом «родительского» объекта, а имя этого столбца фиксировано в модели.
- **Краткое и подробное представления БД**. С каждой таблицей БД могут быть (но не обязательно) связаны два дополнительных представления для загрузки краткой и подробной информации об объекте. Краткое представление применяется при загрузке из БД нескольких объектов «порциями», например при просмотре списка объектов пользователем. Подробное представление применяется при загрузке индивидуального объекта, для просмотра или для редактирования.
- **Краткая и подробная загрузка объекта в память**. При краткой загрузке объект загружается с использованием краткого представления, но без частей. При подробной загрузке объект загружается с использованием подробного представления для основной таблицы БД, а части загружаются с использованием краткого представления. Для оптимизации при подробной загрузке может быть задан список видов частей, которые следует загрузить в память из БД. Например, если выполняемая операция деловой логики касается участников проекта, то ей могут не понадобиться связанные с проектом публикации. Все части одного вида загружаются или не загружаются в память одновременно: все участники, или все публикации и т. п.
- **Количество частей объекта невелико**. Таблицы и объектная модель проектируются так, чтобы количество частей одного вида было невелико. К примеру, для объекта «персона» можно предусмотреть части вида «организации», в которых будут записаны должности сотрудника в одной или нескольких организациях. Однако для объекта «организация» не стоит вводить часть «сотрудники», если количество сотрудников в одной организации может быть большим.
- **Количество данных в каждой части невелико**. Например, такой вид части проекта, как список заполненных форм проекта, не должен включать подробного содержания каждой формы. Для содержания форм следует предусмотреть отдельные таблицы, которые будут отдельными от проекта информационными объектами.

- **Краткое представление содержит достаточно данных для текстового отображения объекта.** К примеру, если частью объекта «персона» является связь с организацией, то в соответствующей таблице БД, назовём ее «люди_организации», могут храниться код персоны, код должности по справочнику должностей и код организации. Этих данных недостаточно для текстового отображения персоны, т. к. пользователю следует показывать не коды, а названия должностей и организаций. Однако *краткое представление* для данной таблицы может включать столбец с расшифровкой названия должности по справочнику должностей и столбец с названием организации, из таблицы организаций.
- **Отложенная загрузка (lazy load) не применяется**. Объект однократно загружается в память, кратко или подробно, со всеми частями, или только с теми видами частей, которые нужны для выполняемой операции деловой логики. При сохранении объекта в БД вместе с ним сохраняются изменения во всех загруженных в память частях.

При описанных ограничениях загрузка объектов из БД и сохранение изменений в БД могут быть реализованы при помощи относительно несложного «менеджера хранения». Наша модель частей объектов представляет собой ограничение универсального объектно-реляционного подхода [1] к отображению произвольной структуры БД в классы языков программирования. Подобные ограничения могут быть оправданными, если «двумерных» информационных объектов с частями различных видов оказывается достаточно для представления всех сущностей и связей в данной предметной области.

5.2. Комментарии в SQL-файлах

При загрузке информационной модели в память часть информации может черпаться из комментариев в SQL-файлах. Менеджер модели, загружающий модель в память, распознаёт следующие виды комментариев в SQL-файлах:

- Ссылка на таблицу (внешний ключ). Комментарий используется в предложении «`create table`» и имеет вид: REF "имя таблицы". Например, в файле «проекты.sql» может быть такая строчка:

```
руковод_ид int, – – REF "люди" –
руководитель проекта
```

В данном примере столбец «руковод_ид» таблицы «проекты» является внешним ключом и ссылается на таблицу «люди».

- Текстовое представление внешнего ключа. Комментарий используется в предложении «`create view`» и имеет вид T(имя ссылочного столбца). Например, в файле представления «`вид_проекты_кратко.sql`» могут быть такие строчки:

```
– – Руководитель
п.руковод_ид, – – код персоны –
руководителя проекта
л1.отобр_текст as руко-
вод_отобр_текст, – –
T(руковод_ид)
л1.фамилия as руковод_фамилия,
л1.имя as руковод_имя,
л1.отчество as руковод_отчество
```

В данном примере столбец «`руковод_отобр_текст`» представления содержит отображаемое имя руководителя проекта, т. е. текстовое представление ссылочного столбца «`руковод_ид`». Такая информация в модели может использоваться администраторским редактором базы данных, для отображения ссылок на объекты.

5.3. Текстовое представление объектов и простой поиск

В информационной системе с веб – интерфейсом, такой как ИС научного агентства, объекты могут извлекаться из базы данных и отправляться клиенту для просмотра или для редактирования. В графическом интерфейсе браузера объекты отображаются в виде текста. Текст объекта может складываться из его атрибутов. Например, для персоны отображаемый текст может складываться из фамилии, имени и отчества, а заявка на проект может отображаться в виде регистрационного номера заявки.

В общем случае объект может извлекаться из базы данных либо подробно, либо в кратком виде. При подробном извлечении все атрибуты объекта могут быть отправлены клиенту. При кратком извлечении объекта из БД желательно заранее определить один или несколько атрибутов, которые могут «определять» объект. Мы предлагаем включать в краткие представления базы данных атрибуты для текстового отображения объектов и частей, а в модели информационного объекта указывать способ формирования отображаемого текста объекта и различных видов его частей из атрибутов. На основе таких атрибутов можно реализовать простой поиск различных типов объектов. Например, поиск проектов по регистрационному номеру, или по части названия проекта.

5.4. Ролевая модель доступа

Можно выделить следующие роли пользователей системы автоматизации научного агентства:

- роль «администратор» — обладает всеми правами доступа;

- роль «внешний пользователь» — роль для доступа к публичной информации о проектах;
- роль «сотрудник» — для внутренних сотрудников агентства;
- роль «сам» — роль любого пользователя для редактирования своих персональных данных;
- роль «эксперт» — для проведения экспертизы проектов;
- роли «руководитель проекта», «участник с правом редактирования», «участник без права редактирования» — роли для заполнения проектных форм;
- роль «координатор организации» — для заполнения карточки внешней организации;
- роль «автор публикации» — для корректировки информации о публикациях.

Роли могут учитываться при экспорте объектов для отправки клиентам, и при импорте информации от клиентов, для записи в БД. В модели информационных объектов могут быть указаны атрибуты, доступные различным ролям для экспорта и для импорта. Помимо доступа к атрибутам объектов, роли могут определять доступ к функциям информационных объектов и к глобальным функциям ИС. Перечисление указанных в модели функций может включаться в генерируемый код класса информационного объекта.

6. Каталог форм

Теперь мы переходим от модели информационных объектов нижнего уровня к более высокоуровневой модели структурированных документов — проектных форм. В частности, нас будут интересовать процедуры перенастройки ИС для адаптации к динамике проектных форм.

Формализация структуры документов вводится в системах автоматизации для упорядочения сбора и обработки информации. Разбивка документов на значимые части позволяет обрабатывать части по отдельности, в том числе вводить, проверять, хранить и отображать части документов, а также вносить локальные изменения отдельных частей, не меняя всех частей одновременно.

Такие операции с частями документов, как ввод, хранение, проверка и отображение, могут быть реализованы в различных частях программы. Однако информацию о структуре документов желательно сосредоточить в одном месте, и таким местом может быть каталог форм. Каталог форм понимается как нейтральное по отношению к функциям хранилище информации о структуре проектных документов.

Каталог форм может быть структурирован по годам и по типам проектов. Мы предполагаем, что каталог представляет собой папку на диске, в которой созданы подпапки по годам, а ниже подпапки по типам проектов. Например:

```
Папка «Каталог форм»
  Папка «2016»
    Папка «Тип проекта а»
      Файл «типро.а.xml»
      … файлы форм
    Папка «Тип проекта б»
      Файл «типро.б.xml»
      … файлы форм
```

Каждому типу проектов отведена папка в каталоге. В этой папке находится файл «типро.XX.xml», где XX — условное обозначение типа проектов. В этом файле содержится список форм для данного типа проектов. Сами файлы форм могут располагаться как в этой же папке, так в папках для других типов проектов. Некоторые формы могут совпадать в различных типах проектов, а другие быть специфичными для определённого типа проектов.

Файл оглавления типа проектов «типро.XX.xml» содержит следующие элементы и атрибуты.

- корневой элемент «типро», с атрибутами «год» — год комплекта форм, и «типро» — условное обозначение типа проектов;
- элементы «форма», подчинённые корневому элементу «типро», со следующими атрибутами:
  - атрибут «вид_формы» — обозначение вида формы. Одинаковые виды форм в различных типах проектов могут совпадать или отличаться;
  - атрибут «файл» — либо имя файла описания формы в папке данного типа проектов, либо относительный путь до файла описания формы в папке другого типа проектов, например: «..\YY\ф_Z.2016.xml», где YY — другой тип проектов, а Z — вид формы;
  - атрибут «ш1п» — имя шаблона для просмотра формы. О шаблонах форм см. в п. 7 ниже;
  - атрибут «ш2пр» — имя шаблона для редактирования формы;
- элемент «параметры», подчинённый корневому элементу «типро», группирует параметры данного типа проектов.

Элемент «парам», подчинённый элементу «параметры». Атрибуты «имя» и «знч» описывают имя и значение данного параметра данного типа проектов.

6.1. Файл описания формы

Файл описания формы располагается в папке для типа проектов и включает следующие элементы и атрибуты:

- корневой элемент «форма» с атрибутами «год» — год комплекта форм, «вид_формы» — условное обозначение вида форм, «заг» — отображаемый

заголовок формы. Атрибуты «модель», «модель2» и «модель3» определяют имя таблицы БД, которая служит первой, второй или третьей моделью формы. Привязка полей формы к модели описана в п. 6.2 ниже. Под корневым элементом «форма» могут располагаться элементы «текст», «поле» и «файл»;

- элемент «текст» описывает статический текст, которые следует отображать при просмотре или при редактировании данной формы. Данный элемент может иметь атрибут «ид» — программный идентификатор элемента формы;
- элемент «поле» описывает одно информационное поле формы. Поле может иметь следующие атрибуты:
  - атрибуты «ид» — программный идентификатор элемента формы; «н» — отображаемый составной номер поля в форме, например «2.1»; «т» — отображаемый заголовок поля;
  - привязки поля к модели формы, см. п. 6.2 ниже. Поле формы может складываться из нескольких элементов модели;
  - дополнительные тексты: пояснения, единицы измерения и т. п.;
  - атрибуты, указывающие, может ли поле редактироваться и обязательно ли для заполнения.
- элемент «файл» описывает один приложенный к форме файл. Может иметь атрибуты «ид», «т» и привязку к модели.

При загрузке каталога форм из XML-файлов в память сервера, в текстах могут выполняться замены: `{ГОД}` заменяется годом формы, а `{ГОД - 1}` заменяется годом на единицу меньшим. Это упрощает формирование каталога форм на следующий год.

6.2. Привязка полей формы к модели

Атрибуты «м», «м2» и «м3» поля формы могут иметь следующую структуру:

- «`имя_свойства`» — привязка к указанному свойству главной модели формы. Главной моделью формы считается таблица, указанная в атрибуте «модель» элемента «форма»;
- «`имя_таблицы.имя_свойства`» — привязка к заданной модели формы. Здесь имя таблицы может выбираться из атрибутов «модель», «модель2» и «модель3» формы;
- «`вид_части[N].свойство`» — привязка вычисляется так: 1) взять информационный объект для главной модели формы, 2) взять список частей заданного вида, 3) найти в списке часть с индексом N и 4) взять заданное свойство части;
- «`имя таблицы.вид_части[N]. свойство`» — то же, для заданной модели формы. На первом шаге имя таблицы может выбираться из атрибутов «модель», «модель2» и «модель3» формы, а следующие шаги 2–4 такие же, как в предыдущем случае;
- «`вид_части[атрибут = значение].свойство`» — привязка вычисляется так: 1) взять информационный объект для главной модели формы, 2) взять список частей заданного вида, 3) найти в списке часть с заданным значением атрибута и 4) взять заданное свойство части;
- «`имя таблицы.вид_части [атрибут = значение].свойство`» — то же, для заданной модели формы.
- «`вид_части[all].свойство`» — специальный случай, когда в форме последовательно перечисляются все экземпляры частей данного вида. В данном случае «`all`» означает привязку к очередному экземпляру части. Например, привязка «`люди[all].фамилия`» может использоваться в случае перечисления в форме всех участников проекта.

7. Шаблоны для просмотра и редактирования формы

Шаблоны форм играют роль «представлений» в технологии «модель — представление — контроллер» (model-view-controller, MVC) [4]. Для просмотра и для редактирования форм могут использоваться различные программные реализации MVC, соответственно на сервере [7] и в клиентской части [6]. Разделение технологий просмотра и редактирования между сервером и клиентской частью может быть оправдано следующими соображениями:

- на сервере — однажды сформированный пакет проектных документов может запоминаться в базе данных в формате HTML для многократного просмотра различными пользователями.;
- на сервере — для печати пакета документов желательно формировать файл формата PDF, а этот сложный формат требует мощного инструментария [8];
- на клиенте — для интерактивных веб-форм желательно максимально использовать вычислительные ресурсы клиентских компьютеров;
- на клиенте — графический интерфейс на основе прикладной платформы AngularJS [6] легче отлаживать без лишних обращений к серверу.

В силу этих соображений шаблоны для просмотра форм и шаблоны для редактирования форм могут создаваться на различных языках программирования. Можно отметить следующие особенности использования MVC-технологии в рассматриваемой предметной области:

- шаблон для каждого вида формы желательно сделать универсальным по отношению к различным типам проектов. Специфичные для различных типов проектов детали, такие как статические тексты, заголовки полей, порядок элементов формы и т. п., желательно не кодировать в шаблонах, а черпать из каталога форм. Шаблон простой формы может включать стандартный цикл по элементам формы и выбор представления элемента в зависимости от типа элемента: текст, число, дата, выбор из справочника или поиск объекта в БД;
- контроллер может использоваться для подготовки перечисления элементов формы: всех для просмотра, или только некоторых для редактирования. Специального внимания требует перечисление частей, например, участников проекта. Подготовленный для данной формы массив элементов вливается в модель;
- на контроллер ложится задача добавления в модель информации из каталога форм, см. п. 6 выше. В частности, из каталога форм могут извлекаться статические тексты и заголовки полей. После того, как контроллер дополняет модель текстами из каталога, запускается процесс наполнения шаблона данными из модели.

Разработчикам ИС следует обобщать и упрощать код шаблонов и код контроллеров с тем, чтобы, во-первых, минимизировать изменения кода для адаптации к изменениям информационной модели проекта, а во-вторых, сделать код шаблонов и код контроллеров доступными IT-специалистам на этапе эксплуатации. Сложные и общие куски кода следует выносить в общие службы или библиотеки, куда IT-специалистам не придётся заглядывать.

8. Инкрементальное редактирование проектных форм

Проектные формы заполняются грантозаявителями в браузере через веб-интерфейс ИС. Каждая форма может редактироваться длительное время и неоднократно. Для удобства пользователей желательно регулярно сохранять изменения на сервере. При этом возникает задача минимизации трафика, с тем, чтобы не отправлять на сервер всю форму целиком, включая длинные, а то и многостраничные тексты полей, в которых не сделано никаких изменений пользователем.

Для редактирования формы сервер загружает из БД и отправляет в браузер подробные сведения о каждом из объектов, указанных в качестве модели в описании формы, см. п. 6.1 выше. Обратно на сервер клиентом присылаются, однако, не целые изменённые объекты, а только списки изменений. Технология инкрементального сохранения изменений может включать следующие элементы клиентской части:

- **кэш объектов** содержит рабочие копии объектов, которые меняются по мере редактирования пользователем. Ключом объекта в кэше является текст вида «имя_таблицы.код объекта». Заметим, что изменения могут вноситься в рабочую копию объекта либо сразу при вводе значений пользователем, либо с некоторой задержкой. Задержка обновления модели имеет смысл для длинных текстов, чтобы не создавать видимого дрожания или замирания графического интерфейса при каждом нажатии клавиши;
- **кэш оригиналов** хранит оригиналы объектов, полученных от сервера. Оригиналы используются для вычисления изменений, которые следует отправить на сервер для сохранения;
- **списки изменений**. Для отправки на сервер формируются три списка данных: 1) список изменённых свойств объектов и частей, 2) список удалённых частей и 3) список новых частей;
- **путь до свойства**. Элемент списка изменений для отправки на сервер включает «путь» до свойства и новое значение. Путь может состоять либо из трёх элементов, либо из пяти. Короткий путь характеризует свойство объекта: 1) имя таблицы БД, 2) код объекта и 3) имя свойства объекта. Длинный путь характеризует свойство части: 1) имя таблицы БД, 2) код объекта, 3) вид части, 4) код части и 5) имя свойства части;
- **создание новых частей**. Новые части создаются в памяти JavaScript как пустые объекты, со свойствами «имя_таблицы_бд» и «ид». В качестве временного идентификатора может присваиваться очередное отрицательное значение: (–1), (–2) и т. д. После сохранения в БД сервер возвращает список новых кодов записей, которые заносятся в свойство «ид» части;
- **удаление частей**. Если удаляется новая часть с временным отрицательным кодом, которая ещё не сохранялась на сервере, то она просто вычёркивается в кэше объектов. Если же удаляется часть с постоянным кодом, ранее сохранённая в БД, то для удаления из БД на сервер отправляется «путь» до свойства «ид» такой части;
- **автосохранение**. Изменения могут отправляться на сервер либо по кнопке «Сохранить», либо автоматически по таймеру. Интервал таймера автосохранения может задаваться в параметрах сервера, в минутах. Кроме того, изменения могут автоматически сохраняться при попытке пользователя закрыть вкладку браузера или перейти в другой раздел сайта.

Литература

1. *Ambler S. W.* Mapping Objects to Relational Databases: O/R Mapping In Detail. http://www.agiledata.org/essays/mappingObjects.html
2. Генератор классов DTO и DAO «SQL DAL Maker». http://sqldalmaker.sourceforge.net
3. Code Generation and T4 Text Templates. https://msdn.microsoft.com/en- us/library/bb126445.aspx
4. *Fowler M.* GUI Architectures. http://martinfowler.com/eaaDev/uiArchs.html
5. Advantages and Limitations of CASE Tools. http://www.petruska.com/xoops1/modules/AMS/print.php?storyid = 15
6. Библиотека AngularJS by Google на языке JavaScript. https://www.angularjs.org
7. Язык шаблонов Apache Velocity. http://velocity.apache.org
8. Библиотека «iText» для формирования PDF-файлов. http://itextpdf.com/product/itext

Богданов Дмитрий Степанович. Зав. лабораторией ИСА ФИЦ ИУ РАН К. т. н. Окончил в 1980 г. МГУ имени М. В. Ломоносова. Количество печатных работ: 18. Область научных интересов: автоматизированные информационные системы, интеллектуальные автоматизированные системы. E-mail: bogdanov@cs.isa.ru

Кац Владислав Анатольевич. Студент НИТУ «МИСиС». Область научных интересов: автоматизированные информационные системы. E-mail: vladk_94@mail.ru

Корягин Андрей Сергеевич. Студент МФТИ (ГУ). Область научных интересов: автоматизированные информационные системы. E-mail: zjunk58@gmail.com

Плискин Евгений Львович. Вед. н. с. ИСА ФИЦ ИУ РАН. К. т. н. Окончил в 1982 г. МФТИ. Количество печатных работ: 15. Область научных интересов: автоматизированные информационные системы. E-mail: eugene.pliskin@gmail.com