Автоматизация ввода и редактирования документа на основе графовой модели

Р. К. Федоров
Институт динамики систем и теории управления СО РАН, Иркутск, Россия
e-mail: fedorov@icc.ru

An approach to automation of the input and editing of documents, whose data should be stored in a structural form is considered. The approach is based on a description of the document model for generation of the user interface.

Как правило, деятельность любой организации сопровождается большим количеством бумажных документов, поэтому разрабатываются информационные системы, автоматизирующие деятельность организаций. Такие информационные системы можно разделить на два класса. Минимальной единицей обработки в первом классе является документ [1]. Примерами таких систем являются Кодекс, ГрандДок и т. д. Системы из второго класса представляют собой структурированную совокупность данных. Отдельные элементы данных документа могут влиять на содержание других документов или использоваться в формировании отчетов. Обычно данные таких документов хранятся в реляционных базах данных (БД).

Создание таких информационных систем является трудоемкой задачей, которая решается в основном с помощью процедурного программирования, т. е. программируются все формы ввода документов, методы доступа к базам данных и т. д. При анализе программного кода форм ввода и редактирования двух разных документов можно увидеть, что он отличается главным образом структурой и правилами верификации. Поэтому представляют интерес развивающиеся дескриптивные подходы [2], создающие пользовательские интерфейсы на основе описания структуры документа.

Рассмотрим структуру документа в соответствии с объектно-ориентированным подходом [3]. Сформированный документ содержит информацию о множестве различных объектов. Каждый объект представляется с помощью набора элементарных свойств и свойств объектов. Элементарное свойство — это типизированное значение, к которому привязана некоторая семантика. Можно сказать, что объект является некоторым отношением его свойств.

Таким образом, формально структура документа в общем виде может быть представлена в виде конечных множеств и связей элементов этих множеств между собой. Математическая нотация может быть представлена в виде двойки $D = \{P, R\}$, где D — формальная модель документа; P — множество объектов; R — множество отношений между объектами.

Эта нотация означает следующее: “Документ — это множество объектов и отношений между ними”. Множество P определяется как конечное множество объектов. В существующих языках программирования элементарные свойства объектов можно

представить простыми типами данных: число, строка, дата, время и т. п. R определяется как конечное множество отношений между объектами.

В объектно-ориентированном подходе выделяют три типа наиболее общих отношений между объектами: зависимости, обобщения и ассоциации. Отношение зависимости — это отношение использования, согласно которому изменение в спецификации одного объекта может повлиять на другой объект. Данное отношение чаще всего используется, чтобы отразить динамическое взаимодействие между объектами. Документ не содержит динамического взаимодействия. Скорее, это некоторый статический срез состояния текущего мира на момент подписания документа. Поэтому данное отношение не используется в документах. Отношение обобщения связывает общие классы со специализированными, оно необходимо на этапе сохранения документа в БД, так как проектирование БД ведут с использованием этого отношения. Отношение ассоциации — структурные взаимосвязи между объектами. Данное отношение часто используется между объектами документа. Например, в свидетельстве о рождении отношение между родителем и ребенком является ассоциацией.

Логическую модель документа наглядно можно представить в виде геометрического графа (рис. 1). При построении графовой модели документа предлагается использовать следующий способ отображения: множество возможных объектов используется для обозначения вершин графа, а множество отношений — для обозначения ребер графа. Используя нотацию, принятую для математического представления графа, можно сказать, что $v(i) = P(i)$ и $e(i) = R(i)$.

Во всех документах присутствует метаинформация: дата документа, регистрационный номер, автор и т. д. Эту метаинформацию можно представить как один объект. Можно сказать, что с этого объекта начинаются создание документа и генерация пользовательского интерфейса для его редактирования. Можно также сказать, что все остальные объекты связаны с данным объектом. Обозначим его как $P(0)$.

В объектно-ориентированном анализе указывают мощность ассоциации. Мощность — это количество объектов, которое может быть связано посредством одного экземпляра ассоциации. Например, в счет-фактуре мощность отношения между поставляющей организацией и товарными ценностями следующая: “один ко многим”, т. е. одна организация и много различных товаров. Рассмотрим все возможные варианты мощности ассоциации. Мощность “один ко многим” означает отношение между одним объектом

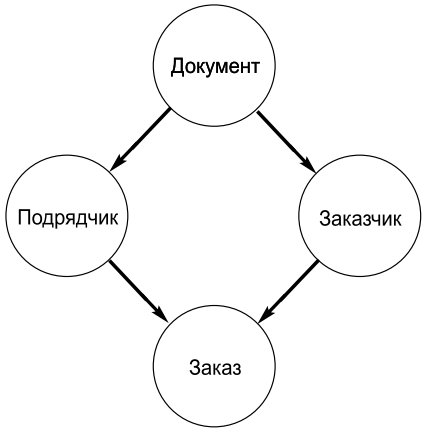

Рис. 1. Графовая модель документа

$P(k)$ и объектами $P(i_1), P(i_2), \ldots, P(i_n)$, где n может принимать конечное значение от 1. Если ассоциация носит обязательный характер и не допускаются не до конца определенные ассоциации, то это означает, что объект $P(k)$ должен существовать до создания любого $P(i_j)$ объекта, где $j = \overline{1,n}$.

Мощность ассоциации "один к одному" означает, что объекты, участвующие в этой ассоциации, должны быть созданы одновременно.

Мощность ассоциации "многие ко многим" не накладывает ограничений на порядок создания объектов.

Кроме ограничений на порядок создания объектов, накладываемых мощностью ассоциации, порядок создания может определяться в зависимости от семантики конкретного документа.

На основе рассмотренных случаев мощности ассоциации можно сказать, что среди объектов существует некоторая частичная упорядоченность их создания, т. е. $P(i)$ создается на основе $P(i_1)$, $P(i_2)$, ..., $P(i_n)$ объектов. Обозначим этот факт $P(i) = P(i_1)$, $P(i_2)$, ..., $P(i_n)$. На основе этой частичной упорядоченности можно определить некоторую ориентированность графа документа, определив направление дуг в соответствии с порядком создания, т. е. объект, соответствующий вершине, из которого исходит дуга, создается раньше, чем объект, соответствующий конечной вершине дуги.

Выделим два состояния документа: состояние редактирования документа и состояние хранения документа. Переход из состояния редактирования в состояние хранения означает отображение документа в таблицы реляционной БД. В состоянии редактирования документ не должен влиять на работу информационной системы. При отображении документа в БД производится создание новых, редактирование и удаление существующих объектов в БД. Отображение может быть сложным и потребовать программирования. Тем не менее существует довольно много документов, в которых изменение состояния БД можно разделить на шаги, где для каждого объекта документа производится одна из следующих операций: добавление нового, изменение свойств и удаление существующего объекта в БД.

Объекты определяются набором свойств. Перечислим возможные источники значений свойств объектов документа:

— ввод пользователем;

— использование значений свойств объектов, введенных ранее;

— использование заранее определенных констант;

— вычисление некоторых специфичных для документа функций.

С помощью вычисления функций можно получить данные из всех других источников, не перечисленных выше.

С целью апробации приведенной графовой модели документа создан прототип. Прототип работает с подмножеством документов, описываемых графовой моделью, а именно с документами, имеющими иерархическую структуру. Данное ограничение работы прототипа появилось из-за сложности отображения неиерархического графа на планарный граф. Необходимость данного отображения возникает при представлении документа на форме. Схема работы данного прототипа представлена на рис. 2. На вход прототипа поступает графовая модель документа. На основе этой модели прототипом создается пользовательский интерфейс для ввода и редактирования документа. Пользователь вносит данные в документ, которые сразу сохраняются в реляционной БД.

Архитектура прототипа разработана в соответствии с шаблоном проектирования Model View Controller (MVC) — это архитектура программного обеспечения, в которой

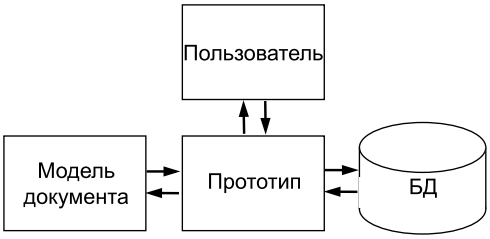

Рис. 2. Схема работы прототипа

модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента так, что модификация одного из них оказывает минимальное воздействие на другие компоненты.

Модель (Model). На вход прототипа поступает графовая модель документа. Кроме структуры документа в модели определяется ряд свойств, необходимых для генерации пользовательского интерфейса. Каждому свойству объекта ставится в соответствие название, отображаемое пользователю, и элемент управления, который отвечает за отображение и редактирование этого свойства. Определен порядок отображения свойств объектов. Задаются ограничения на значения элементарных свойств: запрет на NULL-значения, максимальный размер строки.

Представление (View) отвечает за отображение информации (пользовательский интерфейс). Иерархическая структура документа отображается линейно на форму с помощью элементов управления. Обход графовой модели документа производится рекурсивно. На каждом узле графа выполняется запрос к БД с целью получения объектов, соответствующих данному узлу. Производится анализ мощности отношений с вышестоящими узлами дерева документа. Если анализ показал, что в текущем узле недостаточно объектов, то они автоматически создаются. Затем производится отображение свойств объектов с помощью элементов управления, определенных в модели. Структура прототипа спроектирована таким образом, что новый элемент управления можно легко встроить в систему. Это позволяет гибко настроить пользовательский интерфейс на ввод специфичных данных или на выполнение функций. Элементы управления могут не отображаться на форме и возвращать значения, полученные выполнением вложенных в них алгоритмов.

Контролер (Controller) осуществляет сохранение данных документа в реляционной БД. Документ состоит из множества объектов. Следовательно, необходимо предусмотреть объектное хранение данных в реляционной БД. Существует несколько подходов к хранению объектов в реляционной БД. В программной системе выбран подход, где каждому классу объектов соответствует таблица с теми же атрибутами [5]. Связи между объектами сохраняются в виде ссылочных ключей (reference key). Сохранение состояния документа в БД производится пошагово, где на каждом шаге осуществляется одна из следующих операций: добавление нового, изменение свойств и пометка на удаление существующего объекта. Получение и интерпретация данных осуществляются с помощью элементов управления. При сохранении данных производится верификация значений в соответствии с ограничениями, указанными в модели. В таблице, соответствующей классу объектов документа, содержатся объекты, которые могут принадлежать как различным узлам одного и того же документа, так и разным документам. При обходе графовой модели необходимо уметь определять объекты, принадлежащие узлу.

Поэтому в структуре БД имеется вспомогательная таблица. Во время создания новых объектов документа в эту таблицу вносится информация о принадлежности объекта к документу и узлу.

С помощью прототипа были созданы пользовательские интерфейсы к входным документам ракового регистра. В ходе применения выявлены следующие недостатки:

— использование только древовидного графа;

— негибкий пользовательский интерфейс;

— отсутствие удобного механизма наложения ограничений на вводимые данные;

— возможность сохранять документы, состояние объектов которых отображается в состоянии соответствующих объектов БД.

Несмотря на недостатки, применение прототипа показало практическую значимость предложенного подхода для определенного класса документов. По сравнению с традиционным программированием определение графовой модели документа занимает значительно меньше времени. Следовательно, при изменении пользовательских требований к структуре документов упрощается модификация интерфейса ввода и редактирования документов. Предложенный дескриптивный подход при построении модели документа учитывает мощность отношений между объектами. Определяются частичная упорядоченность создания объектов документа и способ отображения в реляционную БД. Планируется проводить дальнейшую работу с целью уточнения модели, расширения класса документов, описываемых моделью, развивать программную систему, интерпретирующую модель документа. Также необходимо внести в модель ограничения на возможные значения, количество объектов и т. д., чтобы более полно верифицировать данные документа.

Список литературы

[1] Круковский М.Ю. Концепция построения моделей композитного документооборота. `http://www.viaduk.net/viaduk/web5.nsf/0/712825EEB282D241C225715400528746`

[2] Романов Б.Л. Представление структурированных информационных объектов в виде электронных форм // Сб. тр. Ин-та системного анализа РАН, 2002.

[3] Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++: Пер. с англ. 2-е изд. М.: Бином; СПб.: Невский диалект, 2000. 560 с.

[4] Дейт К.Дж. Введение в системы баз данных: Пер. с англ. 6-е изд. К.: Диалектика, 1998. 784 с.

[5] Котляревский В. ООП в РСУБД. `http://www.ibase.ru/devinfo/oop_rdbms.htm`

Поступила в редакцию 25 января 2008 г.