Технология синтеза каркаса информационной системы*

В. В. Парамонов, Е. А. Черкашин, Р. К. Федоров,
И. В. Бычков, Г. М. Ружников
Институт динамики систем и теории управления СО РАН, Иркутск, Россия
e-mail: {slv, eugeneai, fedorov, idstu, rugnikov}@icc.ru

Рассматривается оригинальная технология для синтеза каркаса информационной системы по ее формализованному представлению. Для трансформации модели в программный код используются логические методы. Приводится пример применения технологии для создания медицинской информационной системы "Популяционный раковый регистр".

Ключевые слова: информационная модель, CASE, MDA, трансформация, UML, генерирующее программирование, разработка.

Введение

В настоящее время популярна разработка информационных систем (ИС), предназначенных для сбора, хранения, редактирования и представления информации в реляционных базах данных (БД). Процесс разработки таких ИС, как правило, включает следующие этапы:

1) проектирование логической и физической моделей БД — определение набора таблиц и взаимосвязи между таблицами;

2) разработка интерфейса пользователя к информации БД — для большинства сущностей, представленных в таблицах БД, необходимо создавать новые формы ввода и редактирования или вносить изменения в уже существующие;

3) разработка документации — процесс разработки ИС необходимо сопровождать описанием структур и функций всех составляющих подсистем, например, объектов или компонент, пользовательских интерфейсов, таких как формы ввода, интерфейсы прикладного программирования и обменные форматы данных.

Изменение требований к системе, как правило, приводит к повторному выполнению всех перечисленных этапов. Поэтому актуальна разработка технологии, позволяющей автоматизировать процесс конструирования ИС и проводить их модификацию с сохранением функций, структуры и семантики. В случае изменения программно-аппаратной платформы, на которой реализована ИС, как правило, меняется лишь способ кодирования алгоритмов, а логические структуры объектов ИС и БД остаются неизменными. С увеличением функциональных возможностей ИС возрастает количество компонент, составляющих информационную систему, а также число взаимосвязей между ними, что затрудняет поддержку программного кода подсистем ИС в актуальном состоянии.

*Работа выполнена при финансовой поддержке РФФИ (гранты № 08-07-00163-а, 09-07-12017-офи_м, 08-07-98005-р_сибирь_а) и президентской программы "Ведущие научные школы РФ" (грант № НШ-1676.2008.1).

Как правило, модель рассматриваемых информационных систем состоит из трех основных подсистем, которые формируют их каркас [3]:

— база данных, обеспечивающая хранение информации в реляционном формате;

— приложение — системы взаимодействующих компонент, представляющие собой модель предметной области, выраженную средствами конкретной системы программирования;

—интерфейс пользователя, представляющий информацию из БД.

Для сохранения контроля над сложностью ИС в процессе ее проектирования предлагается оригинальная технология, основанная на визуальном моделировании [1], в которой формальное описание модели, представленное на унифицированном языке моделирования UML (Unified Modeling Language) [2], автоматически преобразуется в часть программного кода каркаса ИС. Программный код каркаса генерируется в результате преобразования исходной модели при помощи набора взаимодействующих модулей трансформации [3].

1. Логический подход к трансформации моделей

Рассматриваемая технология автоматической генерации программного кода подсистем каркаса базируется на идеях подхода Model Driven Architecture (MDA) к проектированию программного обеспечения [4], но в качестве формализмов представления процесса трансформации моделей использует логические языки первого порядка. Процесс трансформации исходной модели осуществляется в соответствии с логической моделью программно-аппаратной платформы (ПАП) и системой программирования, на которых ведется разработка и предполагается функционирование информационной системы. Эта модель включает также формализованные знания дизайнеров и программистов.

Преимущество технологии MDA состоит в замещении кодирования формальными преобразованиями исходной модели в программный код, что позволяет перенести акцент от кодирования ИС к ее проектированию (моделированию). Проект ИС представляется в виде UML-диаграммы, называемой в рамках MDA платформонезависимой моделью — PIM (Platform Independent Model). Следует отметить, что PIM описывает ИС на достаточно абстрактном уровне, что позволяет рассматривать разрабатываемую ИС независимо от конкретных свойств программно-аппаратной платформы.

Структура PIM задается при помощи стандартных редакторов UML-моделей. Созданная модель ИС сохраняется в виде XMI-файла [5], который имеет строгую XML-структуру, что дает возможность использовать его для обработки стандартными трансляторами XML.

Исходный XMI-файл передается в подсистему трансляции XML-документов, где XML преобразуется в древовидный терм с узлами следующей структуры: `xml_node(<имя тега>, <тип вершины>, <пространство имени XML>, <список атрибутов>, <список дочерних тегов>)`, где списки атрибутов и дочерних тегов могут быть пустыми. В процессе преобразования из содержимого XMI-файла экстрагируются OCL-выражения [6] (Object Constrain Language), если таковые существуют. Анализ полученной древовидной структуры осуществляется набором правил языка Prolog. Правила распознают в дереве поддеревья с заданными свойствами и экстрагируют из них необходимые данные о структуре диаграмм UML. В результате обработки данных PIM

представляется в виде списков фактов, задающих диаграмму UML, а OCL-выражения — в виде деревьев синтаксического разбора.

Приведем пример распознавания элемента "класс" диаграммы классов в древовидном представлении XMI-документа (`Node`).

```
xmi_parse_node(Node):-
% Фаза распознавания структуры поддерева.
% Выделение класса в Диаграмме классов UML.
    Node=xml_node('packagedElement', _, _, Attributes, Children),
    xml_attribute_value(Attributes, 'type', xml_ns('xmi', _),
        'uml:Class'),
    xml_attribute_value(Attributes, 'name', ClassName),
    (xml_attribute_value(Attributes, 'visibility', Visibility);
        Visibility='public'),
    (xml_attribute_value(Attributes, 'isAbstract', Abstract);
        Abstract='false'),
    xmi_id_from_attributes(Attributes, ID),
% Конец фазы распознавания.
    get_context(OwnerRef), store_object('xmi_classes',
        xmi_class(ClassName, ID, Visibility, Abstract,
        OwnerRef), ClassRef),
    push_context('class', ClassRef),
    xmi_parse_children_list(Children),
    pop_context('class', ClassRef).
```

Здесь необходимы некоторые пояснения. `Attributes` — список атрибутов тега `packagedElement`; атрибут `type` должен быть равен "`Class`"; если атрибуты `visibility` и `isAbstract` не включаются в XMI-файл, то заменяются значениями по умолчанию. Все элементы в UML-диаграмме помечены индивидуальными идентификаторами при помощи атрибута `id`, правило `xmi_id_from_attributes (Attributes, ID)` получает этот идентификатор для распознаваемого класса. Правило `get_context(OwnerRef)` получает идентификатор структуры, в которую входит распознаваемый класс; такой структурой для класса выступают "модель", "диаграмма" и "пакет". Предикат `push_context/2` создает новый контекст, контекст данного класса, в котором будут порождаться факты об атрибутах и методах класса из списка дочерних элементов `Children`. Предикат `store_object 'xmi_classes', xmi_class(ClassName, ID, Visibility, Abstract, OwnerRef), ClassRef` создает в рабочей памяти факт о существовании данного класса в исходной UML-модели. Предикат `store_object/3` в третьем аргументе возвращает ссылку на созданный факт.

Следующий шаг в синтезе программного кода — обогащение PIM информацией о среде реализации. На этом шаге в PIM добавляется информация (в виде Prolog-фактов), касающаяся реализации элементов PIM на конкретных программно-аппаратной платформе и среде программирования. В результате создается платформозависимая модель ИС (PSM, Platform Specific Model). Трансформация из PIM в PSM производится разработанным транслятором в результате анализа PIM на основе логических правил (базы знаний (БЗ) на языке Prolog) [7], соответствующих модели ПАП.

Например, если в ИС предполагается реализация объектно-реляционного уровня, то для некоторых классов PIM необходимо определить реляционные таблицы базы дан-

ных, представленных в PSM. При этом в таблице сохраняются только атрибуты неабстрактных классов. Если неабстракный класс унаследован от абстрактного класса, то атрибуты последнего необходимо сохранить в этой же таблице:

```
generating_attribute(Cls, [Attr]) :- % атрибут в явном виде
    g_attribute(Cls, Attr).
generating_attribute(Cls, [Attr]) :-
    abstract_parent(Cls, Parent), % унаследованный
% атрибут из абстрактного предка
g_attribute(Parent, Attr).
g_attribute(Cls, Attr) :- % атрибут класса Cls - элемент диаграммы классов.
    pim_attr(_, Cls, _, _, _, Attr1),
pim_name(Attr1, Attr).
a_parent(Cls, Parent):-
% Parent - абстрактный ''родитель'' Cls
    pim_parent(Cls, Parent),
    pim_stereotype(Parent, 'abstract').
```

PSM представляет каркас ИС на несколько более абстрактном уровне, чем непосредственно программный код. Система анализа PIM и синтеза PSM основана на иерархии связанных модулей анализа и генерирования программного кода. Программный код синтезируется из PSM на основе шаблонов, содержащихся в базе шаблонов, которая представляется в виде набора подпрограмм языка программирования Python. Управляющий скрипт выбирает необходимый шаблон в зависимости от результата запроса к БЗ и настроек сценария трансформации PSM в программный код каркаса. Приведем пример подпрограммы, создающей текстовые шаблоны ZOPE для среды The Object Publishing Environment (ZOPE) [8], предназначенные для организации обмена данными между распределенными подсистемами:

```
def gen_class(self, cls, table):
    answer=[]
    name=cls.getName()
    answer.append('''<?xml version="1.0" encoding="UTF-8"?>
        <%s
            xmlns="http://iood.ru/CIS/HPI/ExchangeFormat-1.4"
            xmlns:tal="http://xml.zope.org/namespaces/tal"
            xmlns:metal="http://xml.zope.org/namespaces/metal"
            meta_type="class" table="%s">''' \
        % (name,table)) # Шаблон заголовка
    # Вызов функции для создания списка возможных атрибутов
    schema=self.gen_schema(cls)
    answer.append(schema)
    answer.append("</%s>" % name) # Добавление значений
    return answer
```

В результате работы скрипта автоматически синтезируются исходные коды частей каркаса ИС. Необходимый для синтеза программного кода шаблон также выбирается по результатам запроса к БЗ о структуре идентифицированной сущности. Структура генератора каркаса ИС представлена на рис. 1.

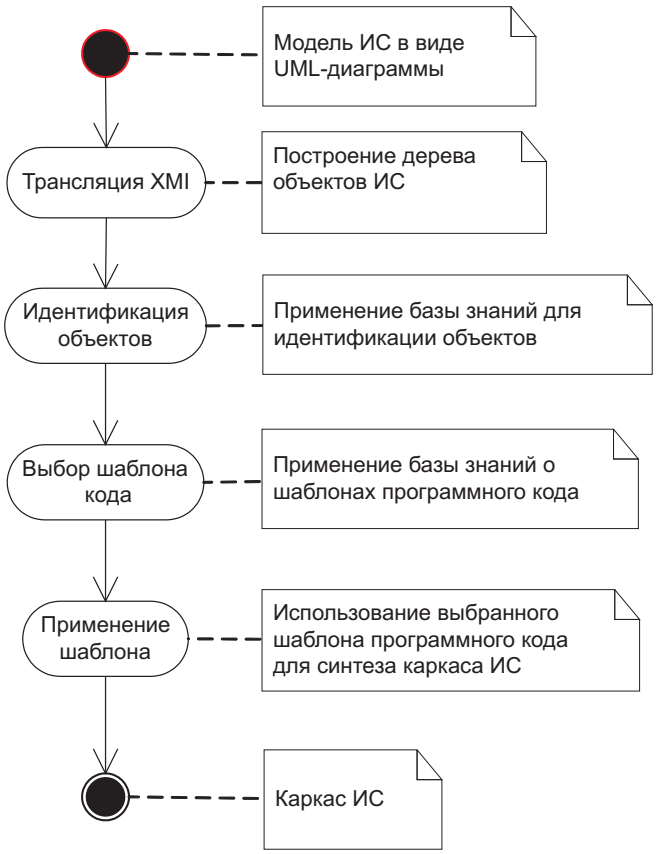

Рис. 1. Структура генератора каркаса ИС

Отличие предлагаемой технологии от модулей генерирования исходного текста подсистем существующих средств MDA, например BOLD for Delphi [9], заключается в том, что генерация кода происходит не на одном фиксированном языке программирования (в случае BOLD for Delphi — Object Pascal), а в принципе на любом языке программирования, для которого разработаны шаблоны генерирования исходного кода, и для любой ПАП, описанной в БЗ.

Процесс формирования БЗ является достаточно трудоемким. Благодаря использованию языка Prolog можно создавать БЗ, настраиваемую при помощи сценариев генерации на решение различных задач синтеза ИС. Данный подход дает возможность проводить автоматический синтез объектов ИС в соответствии с точкой зрения программиста, реализованной посредством логических правил. Последовательная трансляция позволяет на каждой стадии определить методы и способ представления каждого из объектов ИС, представленных в PIM. Накопленные знания могут быть использованы повторно. В то же время стандартные средства разработки, применяющие CASE, как правило, генерируют программный код без учета специфики реализации объектов ИС.

Использование стандартных CASE-средств, реализующих MDA, сводится к следующей схеме: Класс $\rightarrow$ Таблица $\rightarrow$ Форма $\rightarrow$ Стандартный набор компонент.

Часто возникает необходимость, например, для увеличения производительности представить разные классы в одной таблице или с целью повышения удобства для пользователя расположить несколько сущностей на одной форме. Применение гибкой легконастраиваемой БЗ позволяет решить данные проблемы, исключая жесткую привязку к данным.

Предлагаемая технология обеспечивает накопление формализованных знаний о программно-аппаратных платформах реализации ИС и позволяет строить формализованное описание процесса программирования. Подход дает возможность продвинуться в решении задачи понижения себестоимости внесения изменений на ранних стадиях процесса разработки ИС, а также на этапе сопровождения уже функционирующей системы. Предложенная технология использована для создания информационной системы "Популяционный раковый регистр" (ПРР) [10].

2. Практическое применение технологии

Разработка интегрирующей информационной системы "Популяционный раковый регистр" (ППР) (рис. 2), обеспечивающей формирование, обработку и анализ данных регистра, а также осуществляющей информационное сопровождение и поддержку курсов лечения больных (регистрация, обследование, диагностика, лечение, автоматическая генерация отчетных и статистических форм), актуальна и ее необходимость обусловлена Концепцией информатизации здравоохранения [11] и приказами № 135 от 19.04.1999 [12] и № 420 от 23.12.1996 [13] Министерства здравоохранения РФ. Кроме того, данная ИС обладает механизмами, позволяющими осуществлять обмен данными с субъектами, участвующими в информационных процессах ведения "Популяционного рakового регистра" (органы местного самоуправления, отделы записи актов гражданского состояния (ЗАГС), территориальные фонды обязательного медицинского страхования (ТФОМС)).

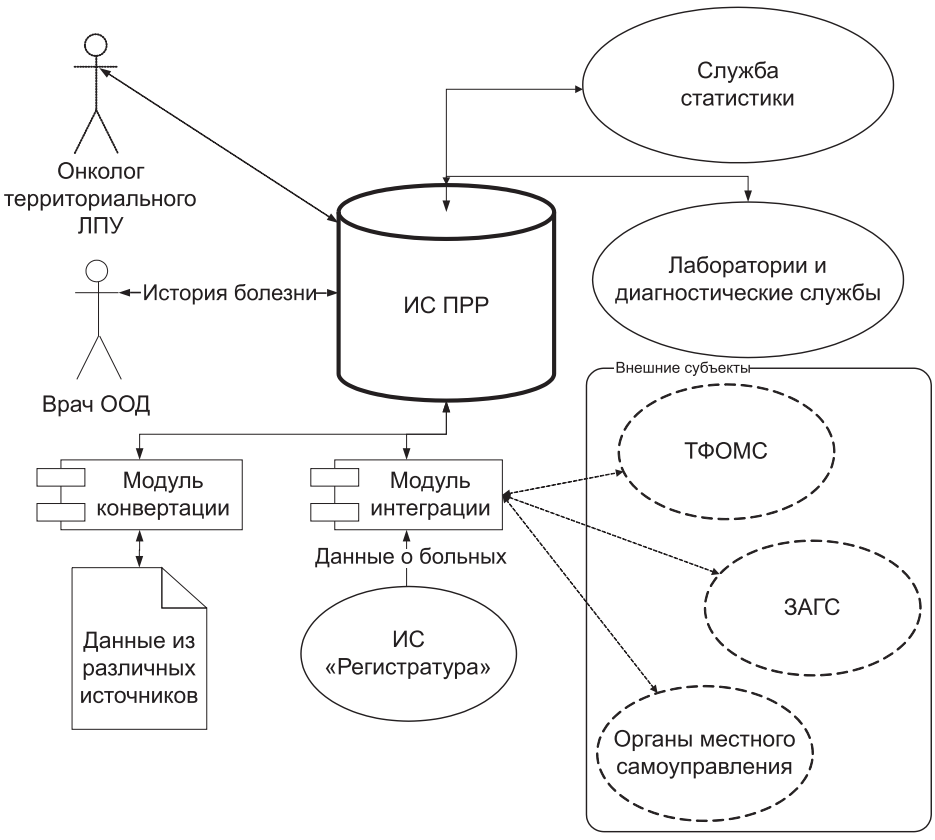

Рис. 2. Субъекты, взаимодействующие с ИС "Популяционный раковый регистр"

Применение рассмотренной технологии для разработки ИС ПРР обеспечивает адаптацию процесса проектирования и эксплуатации информационной системы в условиях динамично развивающихся методов диагностики и подтверждения диагноза, внедрения новых технологий лечения больных, изучения и выявления зависимостей между различными факторами, влияющими на развитие и распространение заболеваемости. Для ИС ПРР требуется постоянная модификация, а в ряде случаев и перепроектирование некоторых программных объектов.

В традиционном подходе к созданию программных продуктов выделяют такие этапы как разработка требований к информационной системе, определение архитектуры ИС и структуры ее БД, создание документации для программиста, кодирование, тестирование и эксплуатация. При внесении изменений необходимо заново пройти всю технологическую цепочку. Использование предлагаемого в статье подхода позволяет значительно сократить затраты на создание и модификацию ИС. UML-модель описывает архитектуру ИС, структуру БД, является основой для синтеза программного кода, а также служит средством документации для разработчика. Программный код блоков каркаса ИС синтезируется на основе PIM автоматически. При модификации ИС требуется лишь провести изменение PIM и, возможно, адаптировать взаимодействие с новыми программными библиотеками (рис. 3).

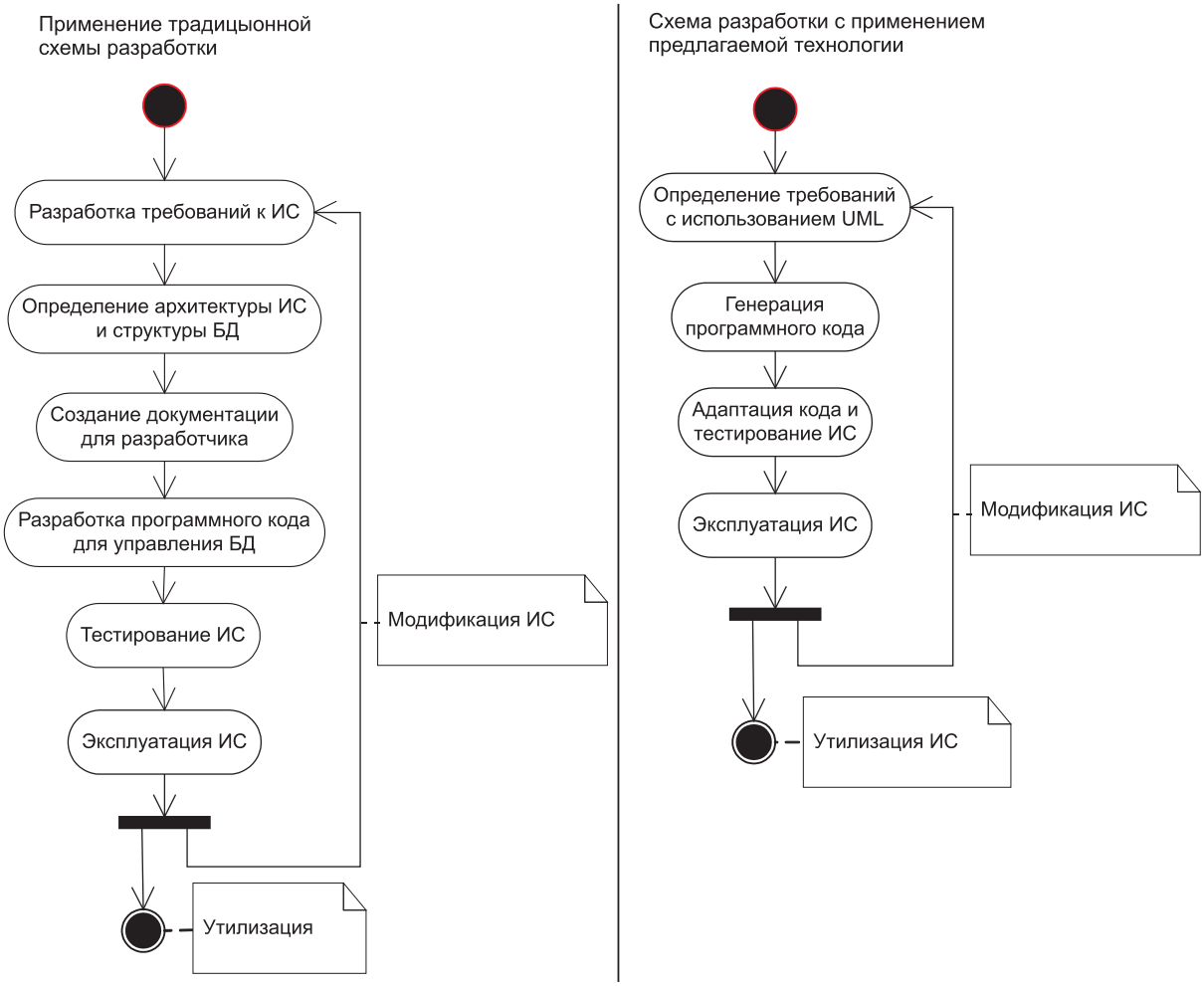

Рис. 3. Сравнение традиционного и нового подходов к разработке информационных систем

В соответствии с предложенной технологией разработана модель ИС ПРР, представленная в виде UML-диаграммы. В модели приводится формализованное описание предметной области, задана структура ИС.

Использование нового подхода для разработки ИС позволяет автоматически генерировать следующие объекты ИС:

— структуру БД на диалекте MySQL [14];

— иерархию классов и метаинформацию об этих классах, доступную во время исполнения ИС;

— XML-шаблоны уровня представления данных для обеспечения сетевого взаимодействия ИС с другими программными системами;

— шаблон функций языка программирования C++ для модуля импорта данных;

— функции для модуля обмена данными на языке Pascal;

— шаблоны элементов и каркас пользовательского интерфейса, в котором создается минимально необходимый набор функций для управления данными системы.

Синтезированный каркас представляет интернет-приложение доступа к БД и объектно-ориентированную библиотеку компонент и интерфейсов, при помощи которых реализуются процедуры порождения отчетов и осуществляется взаимосвязь со сторонним программным обеспечением. При необходимости проводится доработка синтезированных элементов и их взаимосвязь.

Показателями сложности структуры ИС ПРР являются число программируемых сущностей (объектов модели), описывающих объектные структуры ИС уровня приложений, отображаемые в таблицах базы данных, и количество взаимодействий между этими объектными структурами — их более 100. В результате генерации программного кода сформированы 91 таблица баз данных, порядка 8000 строк кода методов классов. Информационная система реализована как трехуровневое приложение для среды ZOPE. Объекты ИС выполнены на языке программирования Python [15], запросы создания базы данных — на языке MySQL. Элементы пользовательского интерфейса синтезированы частично на языке HTML.

Информационная система выполнена в виде WEB-ресурса и обеспечивает весь спектр учета, обследования и лечения больных злокачественными новообразованиями. Так, система позволяет проводить поиск среди пациентов, реализована возможность загрузки информации из БД ИС регистратуры. На рис. 4 представлена архитектура информационной среды ИС ПРР.

Разработанная ИС реализует следующий комплексный подход к ведению регистра:

— первичные данные в БД ИС может вносить непосредственно врач онкокабинета на территориальном уровне. Если отсутствует возможность телекоммуникационного доступа к ИС ПРР, то данные вводятся в систему в электронном виде через обменный формат с переносимого носителя;

— подтверждение или опровержение диагноза вводится непосредственно в лаборатории;

— лечащий врач самостоятельно вносит данные о проведенном обследовании и лечении, что повышает оперативность и корректность информации;

— статистическая и аналитическая информация о больных, представленная в ПРР, доступна для служб статистики в режиме реального времени;

— масштабируемость и открытость ИС позволяет реализовать функции обмена данными с органами местного самоуправления, отделами ЗАГС, а также связать информа-

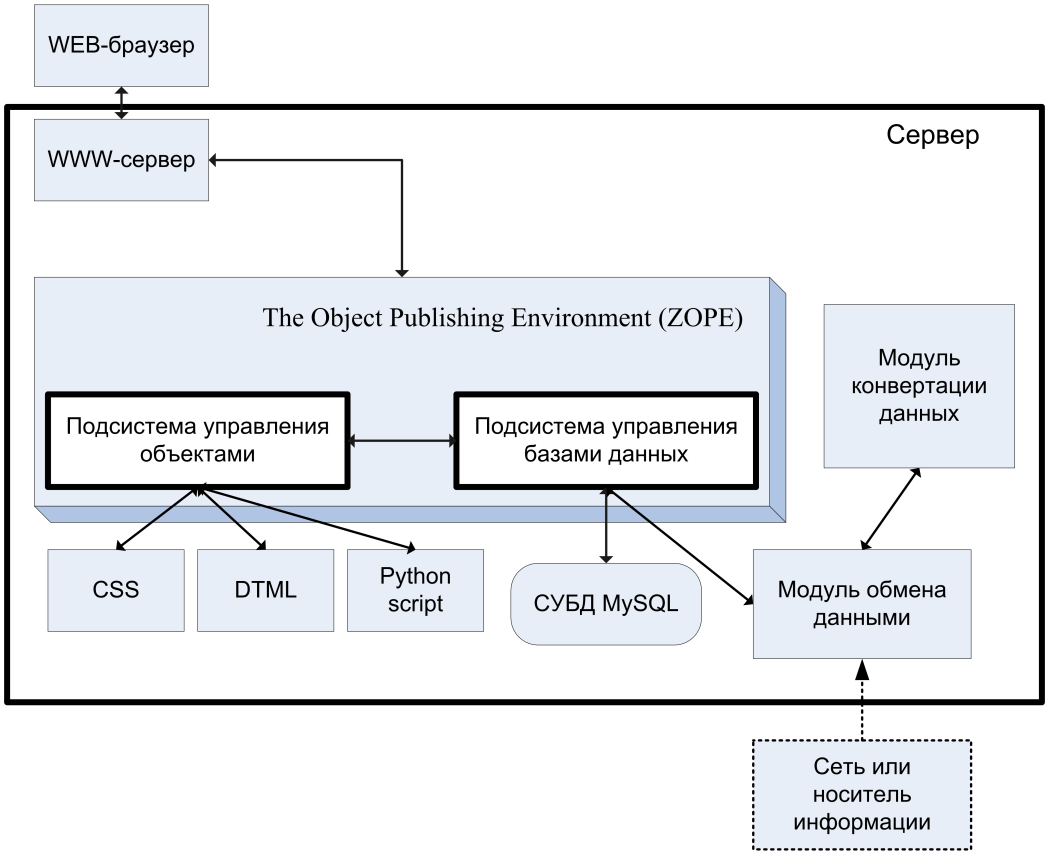

Рис. 4. Архитектура информационной системы “Популяционный раковый регистр”

ционную систему с OLAP-технологией и с геоинформационными системами для многомерного анализа данных и их привязки к геоинформационным ресурсам.

Созданная по предлагаемой технологии ИС ПРР удовлетворяет критериям эффективности, позволяет взаимодействовать со всеми диагностическими и лечебными службами медицинского учреждения онкологического профиля, что обеспечивает ввод данных лицами, напрямую осуществляющими диагностирование или лечение больного. В ряде случаев возможно получение информации напрямую с диагностических приборов, что позволяет избежать ошибок, связанных с повторным вводом информации, и, как следствие, уменьшить число возможных врачебных ошибок. Таким образом, применение данной ИС приводит к повышению качества лечения и клинической эффективности.

Автоматизация ведения ПРР, достигаемая при использовании ИС, уменьшает временные, а значит финансовые затраты на ввод и обработку данных популяционного ракового регистра. Генерация базовых отчетных документов происходит автоматически. Разработка ИС ППР в соответствии со стандартами открытых систем, наличие механизмов интеграции и адаптации при смене программной платформы снижают финансовые затраты, связанные с ее модернизацией и интеграцией с другими медицинскими ИС, что, в свою очередь, повышает показатели организационной и экономической эффективности.

Таким образом, предложенная оригинальная технология, основанная на последовательной трансформации исходной модели информационной системы, позволяет суще-

ственно ускорить и упростить процесс разработки и модификации ИС одного класса. Процесс трансформации описывается при помощи логических правил, формирующих базу знаний. Знания о способе программной реализации объектов модели позволяют учитывать особенности их интерпретации в программный код. Разработанная технология использована для реализации информационной системы "Популяционный раковый регистр".

Список литературы

[1] Кознов Д.В. Основы визуального моделирования. М.: Бином, 2008. 248 с.

[2] Буч Г., Рамбо Дж., Джекобсон А. UML. Руководство пользователя. М.: ДМК, 2001. 423 с.

[3] Парамонов В.В. Многоступенчатая система синтеза программного кода на основе платформо-независимой модели // Тр. XII Байкальской Всероссийской конф. "Информационные и математические технологии в науке и управлении". Иркутск: ИСЭМ СО РАН, 2007. С. 177–183.

[4] Frankel D., Parodi J. The MDA Journal: Model Driven Architecture Straight From the Masters. Advanced Business-Technology Books for Competitive Advantage. Tampa: Meghan-Kiffer Press, 2004. 219 p.

[5] Yang H. Software Evolution with UML and XML. Hershey: Idea Group Publ., 2005. 405 p.

[6] Clark T. Object Modeling with the OCL. N.Y.: Springer, 2002. 276 p.

[7] Cherkashin E.A., Paramonov V.V. An intelligent programming system for information system generation // MIPRO 2005. XXVIII. Internat. Conv. Opatija, Croatia, 2005. P. 140–143.

[8] Spicklemire S., Friedly K. ZOPE: Web Application Development and Content Management. Indianapolis: New Riders, 2002. 454 p.

[9] Грибачев К. BOLD — инструмент реализации MDA в Delphi. Часть 1. MDA — технология будущего // Компьютер пресс. 2003. № 2 / http://www.compress.ru/

[10] Парамонов В.В., Федоров Р.К., Черкашин Е.А. и др. Популяционный раковый регистр Иркутского областного онкологического диспансера // Сибирский онколог. журн. 2009. Приложение № 1. С. 153–154.

[11] Концепция информатизации Федерального агентства по здравоохранению и социальному развитию / Приказ Министерства здравоохранения РФ № 240 от 30.12.2004.

[12] О совершенствовании системы Государственного ракового регистра / Приказ Министерства здравоохранения Российской Федерации № 135 от 19.04.1999.

[13] О создании государственного ракового регистра / Приказ Министерства здравоохранения Российской Федерации № 420 от 23.12.1996.

[14] Дюбуа П. MySQL. Справочник. М.: Вильямс, 2004. 1056 с.

[15] Lutz M., Lewin L., Willison F. Programming Python. N.Y.: O'Reilly, 2001. 652 p.

Поступила в редакцию 29 сентября 2009 г.,
с доработки — 20 апреля 2010 г.