МОДЕЛИРОВАНИЕ И РАЗРАБОТКА СОВРЕМЕННЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ ИССЛЕДОВАНИЙ ЭНЕРГЕТИКИ

Л. В. Массель, Е. А. Болдырев
Институт систем энергетики им. Л. А. Мелентьева СО РАН
Иркутск, Россия
e-mail: massel@isem.sei.irk.ru

The problem of design and implementation of modern software for power engineering research is investigated by the example of "INTEK-2" software, which is used for the energy security problem research. The authors suggest the approach to design and implementation of modern software using open technologies Java, RMI and others. The presented approach has been developed as a generalization of authors' and their colleagues' experience and it can be useful for IT specialists engaged in scientific software development.

Введение

Разработка программного обеспечения для научных исследований, и в частности исследований энергетики, имеет свою специфику. Как правило, оно создается силами небольших исследовательских групп и ими же используется. Для большинства программных систем характерны монолитность, закрытость (сложность переноса на другие платформы), трудоемкость внесения изменения и дорогостоящая поддержка. Усугубляемые плохой документацией эти свойства сокращают жизненный цикл систем в условиях быстро меняющихся информационных технологий. С другой стороны, программное обеспечение научных исследований должно обладать высокими способностями модификации, учитывать появление новых информационных технологий, возможности применения новых математических методов, моделей, подходов, а также возможные изменения в объекте исследований (в нашем случае — в топливно-энергетическом комплексе (ТЭК)).

Возможности современных информационных технологий позволяют создавать программные комплексы (ПК) нового поколения, обладающие гибкостью, расширяемостью, возможностью переноса на другие платформы. Такие ПК должны отвечать определенным требованиям, разрабатываться в соответствующей архитектуре и технологиях. Большое внимание при создании таких ПК должно уделяться их предварительному моделированию на этапе проектирования. Широкий спектр базовых инструментальных средств и технологий предъявляет повышенные требования и к квалификации разработчиков таких ПК. Распространение идеологии Internet и концепции web-служб диктуют переход от

"монолитных" систем — к компонентным, от закрытых — к вычислительным серверам научных приложений. Авторам представляется, что подобный подход к созданию программного обеспечения научных исследований позволил бы существенно увеличить срок жизни программных систем, расширить круг пользователей такого программного обеспечения и создать предпосылки для его коммерческого применения. В то же время разработка программных комплексов нового поколения, безусловно, является сложной проблемой.

Авторы надеются, что выполненное ими обобщение опыта работы группы специалистов — разработчиков программного обеспечения для исследований энергетики[1] — будет полезно проектировщикам ПО и программистам, работающим в смежных областях.

1. Специфика исследований энергетики

Рассмотрим специфику изучения энергетики на примере исследований проблемы энергетической безопасности. Основными инструментами исследований являются экономико-математическое моделирование и вычислительный эксперимент. Проблема состоит в сложности экономико-математических моделей, как количественной (несколько тысяч переменных и неравенств), так и структурной. Эта сложность определяется большим количеством моделируемых объектов, распределенностью и протяженностью энергетических сетей, необходимостью совмещать как агрегированное, так и детальное рассмотрение объектов и процессов.

Под *энергетической безопасностью* страны понимается состояние защищенности ее граждан, общества, государства, экономики от угроз дефицита в обеспечении их потребностей в энергоносителях экономически доступными энергетическими ресурсами приемлемого качества, а также от угроз нарушения бесперебойности энергоснабжения. В обычных условиях подобное состояние защищенности соответствует обеспечению в полном объеме обоснованных потребностей в энергоресурсах, в экстремальных условиях — гарантированному обеспечению минимально необходимого объема потребностей [1]. Эти исследования включают решение следующих задач [2]:

— определение состава потенциально возможных угроз и формирование на этой основе сценариев возмущений;

— оценка состояния ТЭК после реализации различных сценариев возмущений и выявление "узких мест" в топливо- и энергоснабжении потребителей;

— оценка эффективности мероприятий по устранению "узких мест" при реализации угроз;

— отбор инвариантных и наиболее эффективных мероприятий по обеспечению энергетической безопасности.

Особое место среди перечисленных задач занимает оценка состояния ТЭК в условиях возможных сценариев возмущений, так как на их основе производится обоснование мероприятий и рекомендаций по обеспечению энергетической безопасности.

Для проведения исследований состояния систем энергетики при возможных возмущениях и их влияния на условия топливо- и энергоснабжения потребителей разработаны модель текущего состояния ТЭК в нормальных и критических ситуациях и модель оптимизации территориально-производственной структуры ТЭК (на основе принятых стратегий развития ТЭК) [3]. Математической основой этих исследований является решение классической общей задачи линейного программирования.

[1]Реализация первой версии ИНТЭК выполнялась коллективом под руководством Л. В. Массель в составе: Е. А. Болдырев, А. Р. Ершов, Н. Н. Макагонова, В. В. Трипутина.

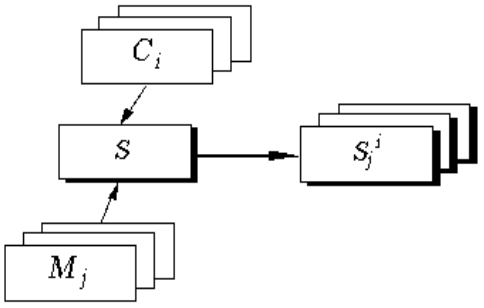

Рис. 1. Схема проведения вычислительного эксперимента для оценки состояния топливно-энергетического комплекса в условиях возможных сценариев возмущений с учетом энергетической безопасности.

Сложность вычислительного эксперимента (ВЭ) определяется многовариантным характером исследований, который иллюстрируется рис. 1. Здесь S — начальное состояние системы; C_i — i-й сценарий чрезвычайной ситуации; M_j — набор мероприятий, нейтрализующих или смягчающих последствия чрезвычайной ситуации; S_j^i — состояния системы после чрезвычайной ситуации C_i с учетом реализации набора мероприятий M_j. Основная цель — определение инвариантного набора мероприятий M_j, выполнение которых в условиях нормального функционирования позволит минимизировать последствия наибольшего числа возможных чрезвычайных ситуаций. Сложность проводимых исследований усугубляется еще и тем, что в зависимости от времени чрезвычайной ситуации, степени готовности системы S, уровня и возможности реализации наборов мероприятий число состояний S_j^i системы может быть достаточно большим, что определяет многовариантный характер расчетов, связанных с исследованиями направлений развития ТЭК с учетом энергетической безопасности.

Кроме того, следует отметить существенное влияние на исследования ТЭК внешних факторов, которое выражается в изменении условий функционирования ТЭК и, соответственно, в изменениях в самом ТЭК как реакции на изменение внешних условий. Это, в свою очередь, обусловливает изменения в методах и моделях исследований ТЭК, порождает новые задачи и новые направления исследований. Изменения в процессе исследований требуют адекватной инструментальной поддержки и изменений в инструментальных средствах, т. е. легкость адаптации инструментальных средств фактически является залогом успешного проведения исследований.

Таким образом, специфика исследований энергетики предъявляет особые требования к программному обеспечению исследований. Учитывая, что задача оценки состояния ТЭК и прогнозирования направлений его развития является актуальной для всех регионов России, создание ПК нового поколения, доступного через Internet, было бы своевременным. Кроме того, предполагается, что подход, разработанный при участии авторов, может быть применен при создании системы мониторинга энергетической безопасности [4]. Рассмотрим далее предпосылки предлагаемого подхода.

2. Изменение технологии создания программного обеспечения

За последние 5–10 лет существенно изменились условия и технология разработки программного обеспечения. Появились не только новые языки программирования и техноло-

гии, но изменились и сама парадигма программирования, концептуальная база и типовые архитектуры программного обеспечения. Основной парадигмой программирования стала объектно-ориентированная, заменившая структурную. Это явилось следствием того, что большинство современных активно использующихся языков программирования — объектно-ориентированные (Java, C#, C++, Object Pascal). Переход к распределенным объектным приложениям определил смену концептуальной базы, что, в свою очередь, обусловило изменение архитектуры ПО: от "монолитной" к двухуровневой архитектуре "клиент — сервер" и распределенной многоуровневой архитектуре.

Основным фактором, обусловившим этот процесс, является появление и развитие Internet, обостряющего и усиливающего существующие тенденции в области информационных технологий, а именно: развитие и применение на практике объектного подхода, распространение концепции открытых систем и переход к работе в локальных и глобальных вычислительных сетях.

Авторы считают, что в настоящее время происходит слияние web-технологий и технологий разработки прикладных программных систем [5, 6]. Эта тенденция вызвана, с одной стороны, требованиями пользователей к расширению функциональности, предоставляемой web-технологиями, а с другой — увеличением степени распределенности прикладного программного обеспечения.

Стремительное развитие информационных технологий обусловливает быструю смену системного программного обеспечения, в первую очередь операционных систем (ОС), что приводит к изменениям в пользовательском интерфейсе. Интерфейс новых операционных систем развивается и улучшается с каждой новой ОС, что, в свою очередь, приводит к необходимости переписывания прикладного программного обеспечения, даже если логика и вычислительные процедуры программы остаются неизменными. Дополнительная сложность возникает, если программа написана для динамично изменяющейся предметной области, когда необходимо изменять и логику, и вычислительные процедуры.

Частично эту проблему решает многоуровневая распределенная архитектура программного обеспечения, когда логика и вычислительные процедуры предметной области отделяются от интерфейса пользователя и постоянно хранимых данных. Нерешенной остается часть проблемы, связанная с изменениями в логике и вычислительных процедурах предметной области. Эта задача может быть решена, если ПО обладает свойством расширяемости, в его классической трактовке, предложенной Н. Виртом, где под *расширяемым программированием* подразумевается, что добавление нового модуля возможно без необходимости вносить какие-либо изменения в существующие модули — не должно быть необходимости даже их перекомпилировать, при этом новые модули могут добавлять также новые типы данных [7]. Таким образом, *расширяемой программой* называется только такая программа, которая может быть адаптирована к решению новой задачи без редактирования существующего исходного кода [8]. Программа называется *независимо расширяемой (independently extensible)*, если она может взаимодействовать с расширениями, добавленными позже, без необходимости общей проверки на целостность [9]. Далее под расширяемой программой будет подразумеваться именно такая программа.

3. Повышение требований к квалификации программистов

Вышеперечисленные тенденции в области ИТ привели к повышению требований к квалификации и знаниям программиста. Десять лет назад разработчику ПО достаточно было

хорошо знать один из языков (например, C или Фортран). Пять лет назад разработчику уже необходимо было осваивать специальные языки программирования четвертого поколения и среды разработки (Visual Basic, Delphi, PowerBuilder), знать SQL и быть в курсе характеристик определенной СУБД. Помимо этого разработчик должен был представлять себе, как работает архитектура "клиент — сервер".

В настоящее время идеальный разработчик ПО должен владеть специальными языками презентационного уровня (HTML, XML, JavaScript, JSP, ASP), уровня логики предметной области (Java, C++ или любой другой подобный язык программирования), уровня данных — SQL, а также владеть технологиями создания распределенных объектных систем (одной обязательно), которые позволяют объединять различные уровни в единую систему, такими как DCOM (Distributed Component Object Model), CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation). Разработчик должен также понимать принципы работы и архитектуру распределенных программных систем, а также типичные проблемы, которые могут возникать при их работе (проблемы безопасности, проблемы, связанные с гетерогенным и распределенным характером среды функционирования, и т. д.). Таким образом, сложность разработки современного ПО за последние годы существенно возросла [10].

Одним из способов борьбы со сложностью разработки ПО является предварительное проектирование программного обеспечения. Фактически предварительное объектное моделирование ПО является необходимым этапом разработки программного обеспечения при использовании объектно-ориентированных языков программирования. Одним из факторов, затруднявших использование объектного моделирования, являлось отсутствие единого стандарта на язык моделирования ПО. Эта проблема была решена с появлением в 1998 г. разработанного Г. Бучем, И. Якобсоном и Д. Рамбо языка UML (Unified Modeling Language), который принят в настоящее время в качестве стандартного языка моделирования ПО [11]. С появлением UML программисты получили общий для всего программистского сообщества язык моделирования ПО, который отражает все фазы жизненного цикла ПО: от анализа и проектирования до размещения ПО на машинах пользователей. Для облегчения построения моделей и обеспечения взаимосвязи между ними могут применяться CASE-средства (Computer Aided Software Engineering), например Rational Rose. Владение новым средством требует от программиста дополнительных усилий, но зато они компенсируются получением готового кода программы из объектных моделей.

Далее на основе этих предпосылок авторы предлагают подход к созданию современных программных комплексов, который является обобщением опыта, полученного при разработке программных комплексов для исследований проблемы энергетической безопасности ИНТЭК [12] и ИНТЭК-2 [5].

4. Требования к современным программным комплексам и системно-концептуальные соглашения

Представляются целесообразными следующие требования к современным программным комплексам:

1) современные ПК разрабатываются, как правило, с использованием объектно-ориентированных языков программирования, при этом необходимым предварительным этапом является объектно-ориентированное проектирование, или объектное моделирование будущего ПК;

2) необходимо стремиться к отделению интерфейса пользователя от логики предметной области и вычислительного ядра. Это означает, что вычислительное ядро выполняется в виде отдельного модуля, который получает данные либо со стандартного ввода, либо из файла и отправляет результаты на стандартный вывод или в файл. В качестве источника исходных данных и приемника результатов также можно использовать базу данных, но это менее универсальное решение, которое может стать источником проблем при переносе ПК с одной операционной системы на другую;

3) использование стандартных возможностей языков программирования при написании вычислительного ядра для облегчения внесения изменений в приложение для переноса его на другую платформу, например, при работе с языком C желательно использовать только процедуры и функции, определенные стандартом ANSI C (без каких-либо расширений);

4) обеспечение возможности распределения приложения по нескольким компьютерам;

5) обеспечение возможности удаленного доступа к ПК;

6) использование какой-либо *стандартной* технологии создания распределенных приложений (CORBA, RMI, COM+, EJB) в зависимости от сложности задачи и других ограничений.

Для того чтобы избежать морального устаревания программного обеспечения, которое часто наступает уже к моменту завершения разработки, необходимо правильно выбирать инструменты и технологии, используемые при разработке ПО.

На основании результатов анализа тенденций развития современных информационных технологий и сформулированных на его основе требований при разработке современных ПК авторами предлагаются следующие системно-концептуальные соглашения:

1. Применение при проектировании и реализации ПК объектно-ориентированного подхода, а именно языка моделирования UML и CASE-средств как для проектирования ПК (например, Rational Rose), так и для проектирования базы данных (например, ErWin), а также паттернов проектирования и объектно-ориентированного языка программирования Java для реализации как серверной, так и клиентской частей ПК.

2. Использование в качестве основной архитектуры современных ПК трехуровневой распределенной архитектуры "клиент — сервер" с сервером приложений и сервером баз данных.

3. Разработка архитектуры отдельных компонентов в соответствии с моделью MVC (Model/View/Controller) [13].

На основе анализа тенденций развития ИТ и в соответствии с принятыми системно-концептуальными соглашениями предлагается использовать следующие технологии и программные продукты для их реализации (по принципиальным соображениям и ввиду ограниченности финансовых ресурсов научных учреждений выбрана ориентация на ПО с открытым кодом (open source) и свободно распространяемое ПО (free software)):

1. В качестве базовой операционной системы для уровня ресурсов и уровня логики предметной области предлагается использовать Linux (хотя может быть использована любая операционная система), так же как и для уровня представления, для которой существует реализация виртуальной машины Java.

2. В качестве сервера баз данных предлагается использовать свободно распространяемый SQL-сервер Interbase. Существуют версии этого сервера для различных операционных систем: Windows'95, NT 2000, Linux и Solaris.

3. Для разработки и эксплуатации программного комплекса предлагается использовать виртуальную машину Java фирмы Sun Microsystems версии 1.3.0 (как для сервера, так и для клиентов) как наиболее доступную.

4. В качестве интеграционной технологии для организации взаимодействия между клиентами и серверами предлагается использовать технологию RMI (Remote Method Invocation).

Предложенный подход, системно-концептуальные соглашения и инструментальные средства были применены при создании ПК ИНТЭК-2 для исследования направлений развития ТЭК с учетом требований энергетической безопасности.

5. Описание реализации ПК ИНТЭК-2

Первая версия ПК ИНТЭК была реализована в двухуровневой архитектуре "клиент — сервер" [12]. Тогда были заложены основные концептуальные решения, такие как использование объектно-ориентированного подхода, языка Java, ориентация на программное обеспечение с открытым кодом и свободно распространяемое ПО. ИНТЭК-2 является развитием первой версии ПК.

В соответствии с системно-концептуальными соглашениями ПК ИНТЭК-2 спроектирован и реализован в трехуровневой архитектуре "клиент — сервер". Первый уровень — это уровень интерфейса пользователя, он отвечает за представление информации пользователю и отображает действия пользователя в вызовы функций, фактически он соответствует компонентам "Вид" и "Контроллер" в архитектуре "Модель/Вид/Контроллер". Второй уровень — это уровень логики предметной области, реализующий основную функциональность приложения, фактически он соответствует компоненту "Модель" в архитектуре "Модель/Вид/Контроллер". Данный уровень выполняет функции сервера для интерфейса пользователя и клиента для уровня ресурсов. Третий уровень — это уровень ресурсов. Обычно на этом уровне функционируют SQL-сервер, унаследованные системы и другие необходимые приложения. В данном случае на этом уровне функционирует SQL-сервер Interbase. Взаимодействие между первым и вторым уровнями организовано с помощью технологии RMI, между вторым и третьим — с помощью языка SQL. Общая архитектура и схема взаимодействия между отдельными компонентами программного комплекса изображены на рис. 2. Основными достоинствами этой архитектуры являются масштабируемость, высокая степень повторного использования, гибкость, переносимость и настраиваемость.

При разработке ИНТЭК-2 было проведено предварительное объектное моделирование

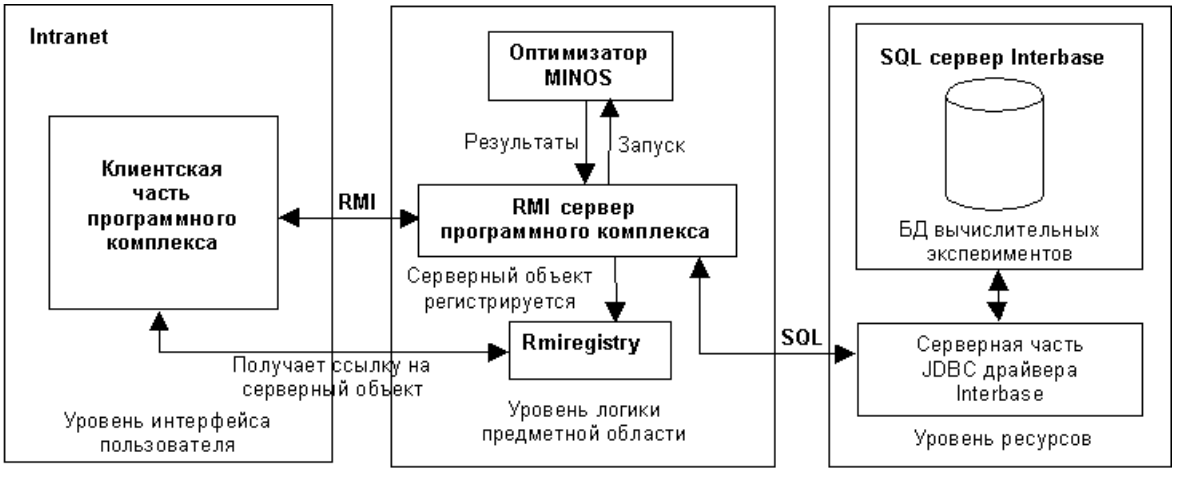

Рис. 2. Архитектура программного комплекса ИНТЭК-2.

с использованием CASE-средства Rational Rose 2000 на языке UML. Объектная модель серверной части ПК приведена на рис. 3.

Серверная часть программного комплекса реализует его основную функциональность. В основе архитектуры серверной части лежит паттерн Abstract Factory [14, с. 93]. Это порождающий паттерн. В данном случае он используется для создания экземпляров класса, реализующего интерфейс isem.server.dbserver.IntecServer, который собственно и определяет сигнатуры методов для реализации необходимой функциональности. Абстрактной фабрикой является интерфейс IntecServerListener, конкретной — класс IntecServerListenerImpl.

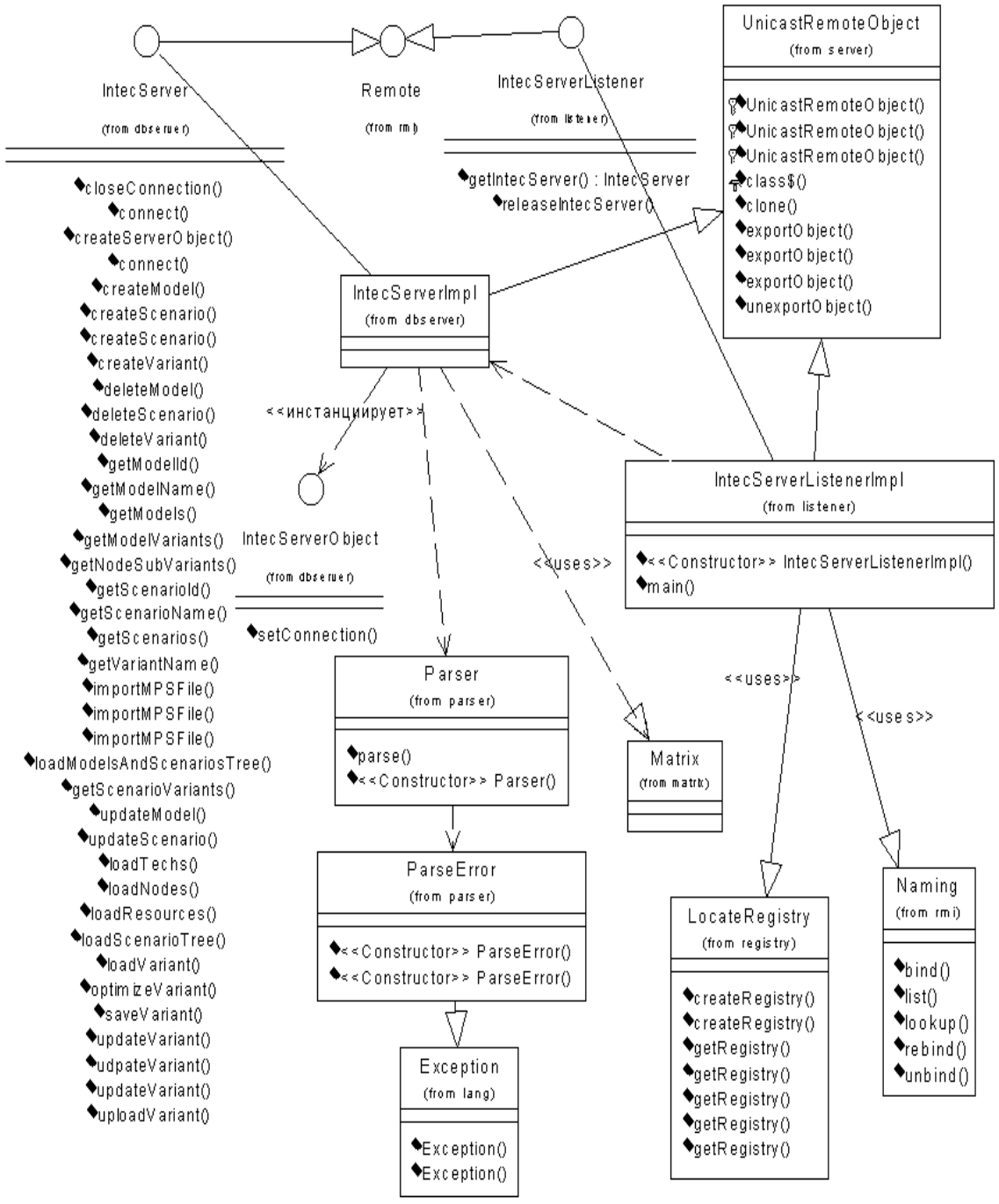

Рис. 3. Диаграмма классов серверной части программного комплекса.

В данном случае применение этого паттерна позволило избежать проблемы с организацией одновременного доступа нескольких клиентов к серверной части программного комплекса. Каждый клиент работает со своим серверным объектом, после завершения работы клиент отсоединяется от серверного объекта и память, которую он занимал, освобождается процессом сборки мусора виртуальной машины Java.

Комплекс имеет MDI (Multiple Document Interface) интерфейс, где каждый компонент, обеспечивающий интерфейс с пользователем, представлен отдельным окном внутри интеграционной среды. Каждый из компонентов при добавлении в интеграционную среду добавляет в панель инструментов собственную панель инструментов, а в меню среды — собственное меню. Основными компонентами комплекса являются компонент управления вычислительным экспериментом и сценариями исследований, редактор моделей и окно сообщений системы.

Многооконный интерфейс программного комплекса приведен на рис. 4. Овалами и стрелками выделены окна разных компонентов. Вверху показаны окна управления вычислительным экспериментом и сценариями исследований и редактора информационных моделей (матриц условий). В окне слева приведены представление дерева вариантов с разными сценариями и информация об их заполнении и счете.

В соответствии со спецификой предметной области одним из основных требований, которым должно соответствовать программное обеспечение для исследований ТЭК, является возможность расширения его функциональных возможностей и, таким образом,

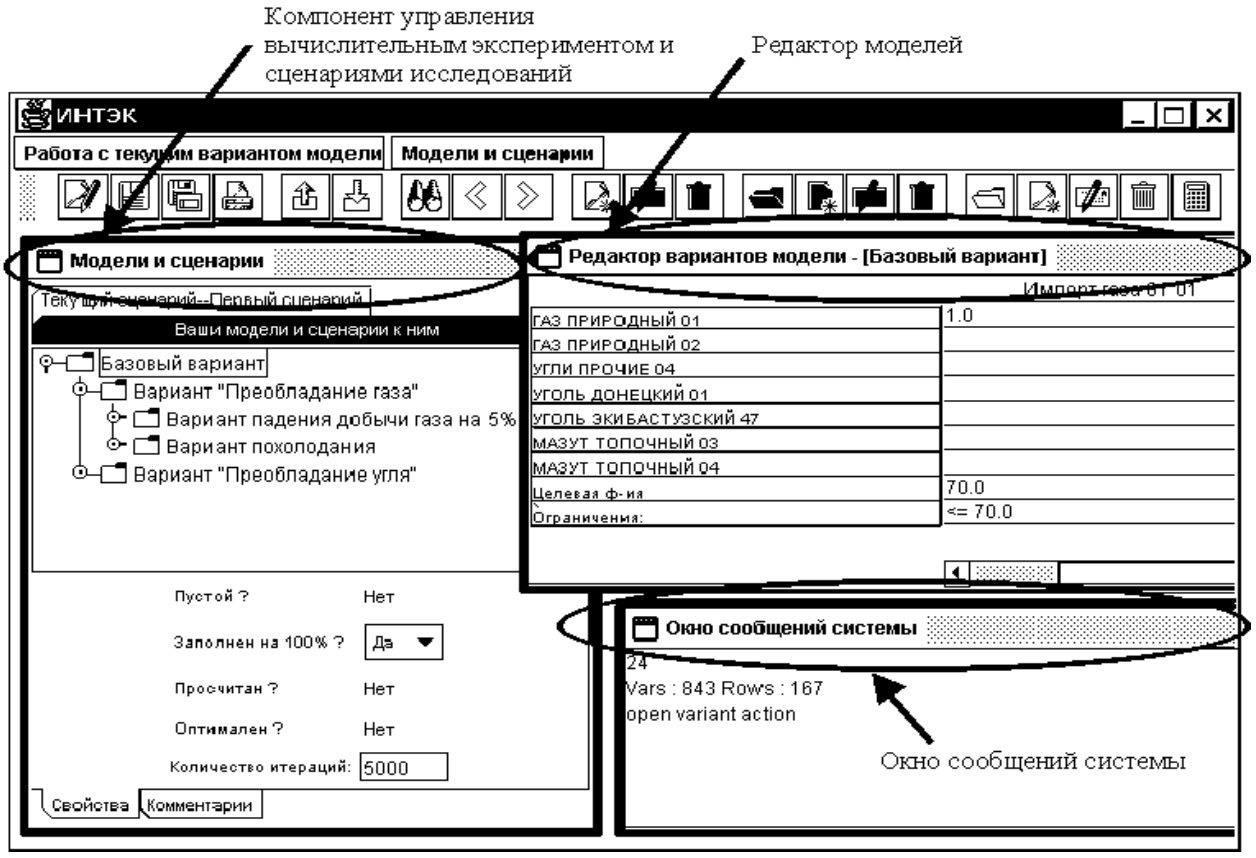

Рис. 4. Интерфейс программного комплекса ИНТЭК-2.

включения новых методов и моделей в функциональную схему исследований как в ответ на изменения, происходящие в объекте исследований, так и в ответ на появление новых математических методов и моделей и новых информационных технологий.

В соответствии с разработанной архитектурой компонент расширения, как и основные компоненты комплекса, состоит из двух частей. Одна часть соответствует "Виду" и "Контроллеру" в архитектуре "Модель/Вид/Контроллер" и функционирует на уровне логики представления, будем называть ее клиентской частью, а другая часть соответствует "Модели" в вышеупомянутой архитектуре и функционирует на уровне логики предметной области, будем называть ее серверной частью. Программный комплекс спроектирован таким образом, что для подключения компонентов расширения нет необходимости в перекомпиляции всего комплекса. Новый компонент несет в себе только то, что необходимо, полностью используя существующие функциональные возможности программного комплекса. Это обеспечивается разработанной архитектурой и возможностями Java платформы, а именно моделью компоновки программы Java (linking model) [15], которая позволяет создавать программы, динамически загружающие во время выполнения необходимые классы и интерфейсы. Это означает, что при компиляции программы нет необходимости знать обо всех классах и интерфейсах, которые будут использоваться в программе во время выполнения. Таким образом, используемые во время исполнения классы и интерфейсы могут даже не существовать во время компиляции программы. Для того чтобы загрузить определенный класс, программе необходимо указать строку с полностью определенным именем требуемого класса (например, чтобы загрузить класс JFrame, который находится в пакете javax.swing, программе нужно указать полное имя класса javax.swing.JFrame). Java предоставляет также механизмы для создания объектов загруженного класса и вызова их методов. Этот механизм используется для загрузки как клиентской части компонента расширения, так и серверной.

Для подключения клиентской части компонента расширения в файл Intec.conf, который расположен в той же директории, где и JAR-файл с клиентской частью программного комплекса, должна быть добавлена строка следующего содержания:

$$\text{EXTENSION.N} = \text{extPackage.extClassName},$$

где EXTENSION (можно строчными буквами) — зарезервированное слово; N — номер расширения (каждый компонент расширения должен иметь свой уникальный номер, выбирается произвольно); extPackage.extClassName — полностью определенное имя основного класса компонента расширения.

Клиентская часть компонента расширения может состоять из произвольного количества классов, но есть один класс, который представляет интерфейс пользователя и обеспечивает доступ к функциональности данного компонента, имя этого класса и указывается в строке настроечного файла Intec.conf. Классы компонента расширения могут как использоваться в открытом виде, так и быть упакованы в JAR-файл.

Для того чтобы клиентская часть программного комплекса смогла загрузить клиентскую часть компонента расширения, необходимо при запуске клиентской части ПК указать к нему путь в списке директорий, в которых JVM должна осуществлять поиск классов, необходимых для работы программы, если классы компонента расширения находятся в открытом виде, или указать полный путь к JAR-файлу, если классы компонента расширения упакованы в JAR-файл.

Для подключения серверной части компонента расширения к серверной части программного комплекса необходимо при запуске серверной части ПК указать к нему путь в списке директорий, в которых JVM должна осуществлять поиск классов, необходимых для

работы серверной части, если классы серверной части компонента расширения находятся в открытом виде, или указать полный путь к JAR-файлу, если классы серверной части компонента расширения упакованы в JAR-файл.

Рассмотрим каждую из частей компонента расширения подробнее и сформулируем требования, которым должны отвечать клиентская и серверная части компонента расширения данного программного комплекса. Требования к основному классу (имя этого класса указывается в настроечном файле Intec.conf) клиентской части компонента расширения следующие. Основной класс должен:

— порождаться от класса javax.swing.JInternalFrame;

— реализовывать интерфейс isem.client.env.ClientExtension (разработан самостоятельно);

— реализовывать интерфейс isem.client.env.IntecComponent (разработан самостоятельно);

— иметь конструктор без параметров;

Основной класс серверной части компонента расширения должен:

— порождаться от класса java.rmi.UnicastRemoteObject;

— реализовывать интерфейс isem.server.dbserver.IntecServerObject (разработан самостоятельно);

— иметь конструктор без параметров.

Выводы

За последние годы существенно изменились условия и технологии разработки программного обеспечения. Переход к работе в локальных и глобальных сетях предъявляет новые требования к современным программным комплексам. Широкий спектр новых инструментальных средств и технологий повышает требования к разработчикам программного обеспечения научных исследований, касающиеся как квалификации, так и требуемого объема знаний.

Программное обеспечение научных исследований, безусловно, имеет свою специфику. Тем не менее авторы полагают, что разработка программного обеспечения научных исследований в виде программных комплексов нового поколения позволила бы организовать доступ к ним через Internet, расширить круг потенциальных пользователей, организовать консультирование пользователей через Internet, создать предпосылки для создания вычислительных серверов научных приложений и, возможно, организации web-служб.

Представляется, что изложенный подход, являющийся обобщением опыта работы в области создания ПО исследований энергетики, может быть полезен специалистам, работающим в области создания ПО для научных исследований.

Список литературы

[1] Славин Г. Б., Чельцов М. Б. Энергетическая безопасность. Термины и определения // Иркутск, 1999 (Препр. ИСЭМ СО РАН; №4).

[2] Воропай Н. И., Клименко С. М., Криворуцкий Л. Д. и др. Энергетическая безопасность России (введение в проблему). Иркутск: СЭИ СО РАН, 1997. 57 с.

[3] Надежность систем энергетики: достижения, проблемы, перспективы / Г. Ф, Ковалев, Е. В. Сеннова, М. Б. Чельцов и др. / Под ред. Н. И. Воропая. Новосибирск: Наука. Сиб. предприятие, 1999. 434 с.

[4] Массель Л. В., Макагонова Н. Н., Трипутина В. В. и др. Система поддержки принятия решений по обеспечению энергетической безопасности // Изв. РАН: Энергетика. 2000. №6. С. 40–48.

[5] Болдырев Е. А. Современные архитектуры и технологии построения программных комплексов / Под ред. Л. В. Массель // Иркутск, 2001 (Препр. ИСЭМ СО РАН; №10).

[6] Болдырев Е. А. Современная архитектура программного обеспечения и пример ее реализации // Современные подходы к интеграции информационных технологий. Иркутск: ИСЭМ СО РАН, 2001. С. 20–32.

[7] Пешио К. Никлаус Вирт о культуре разработки ПО // Открытые системы. 1998. №1. С. 41–44.

[8] Krishnamurthi Sh., Felleisen M. Toward a Formal Theory of Extensible Software // Proc. ACM Conf. on the Foundation of Software Eng. 1998. P. 88–98.

[9] Szipersky C. Independently extensible systems — software engineering potential and challenges // Proc. 19th Australian Computer Sci. Conf., Melbourne, Australia, Jan. 31 – Feb. 2, 1996. P. 19–32.

[10] Циметьере Ж.-К. О практике разработки современных приложений // Открытые системы. 2001. №11. С. 41–42.

[11] Буч Г., Рамбо Д., Джекобсон А. UML-руководство пользователя. М.: ДМК, 2000. 432 с.

[12] Массель Л. В., Макагонова Н. Н., Ершов А. Р. и др. Применение современных информационных технологий для исследований ТЭК с позиций энергетической безопасности // Методические вопросы исследования надежности больших систем энергетики. Вып. 50: Некоторые вопросы надежности систем энергетики. Новосибирск: Изд-во СО РАН, 1999. С. 60–69.

[13] Krasner G. E., Pope St. T. A cookbbook for using the model-view-controller user interface paradigm in Smalltalk-80 // J. of Object-Oriented Programming. 1998. Vol. 1(3): 26–49, Aug./Sept.

[14] Гамма Э., Хели Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. 368 с.

[15] Venners B. Inside the Java Virtual Machine. McGraw-Hill Computing, 1999.

Поступила в редакцию 25 декабря 2001 г.