УДК 658.512.011.56:004.42

М.П. Силич

ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО ПОСТРОЕНИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ МОДЕЛЕЙ ДЛЯ ОПТИМАЛЬНОГО ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

Предлагается инструментальный комплекс, позволяющий формировать объектно-ориентированную модель сложной системы в виде совокупности взаимосвязанных компонент (подсистем и элементов), каждой из которых сопоставляется множество вариантов ее реализации, формируемых на основе соответствующего класса описания компоненты. Для оценки и выбора оптимальных вариантов используются отношения функциональной зависимости, установленные на множестве атрибутов. Комплекс включает ряд модулей, предназначенных для создания отдельных подмоделей – модели классов, модели вариантов реализации компонент, модели функциональных отношений. Открытая архитектура комплекса позволяет расширять его функциональные возможности.

Объектно-ориентированные представления, в частности унифицированный язык моделирования UML (Unified Modeling Language) [1], широко используются для построения моделей сложных систем – как правило, на этапах концептуального и логического проектирования информационных систем для структурированного описания автоматизируемой предметной области. Чаще всего модели предназначены для визуализации и документирования существующего состояния системы (модель «As-is») или желаемого состояния (модель «To-be») как предварительного шага в процессе разработки программного обеспечения. Однако возможности объектных языков моделирования не исчерпываются только описательными средствами. Механизм включения в объектное описание процедур (так называемых методов), способных изменять состояния объектов, позволяет осуществлять поиск решений на модели. Множество атрибутов, описывающих класс некоторой компоненты системы, при этом рассматривается как пространство поиска экземпляра класса, соответствующего оптимальному варианту реализации компоненты.

В данной статье рассматривается инструментальный комплекс, предназначенный для построения объектно-ориентированных моделей сложных систем и выработки оптимальных проектных решений на моделях. Данный комплекс является автоматизированным средством поддержки информационной технологии проектирования социально-экономических систем для разрешения сложных многофакторных проблемных ситуаций [2–5]. В основе данной технологии лежит объектно-ориентированная методология моделирования, главной особенностью которой является возможность объединять различные методики системного анализа и инженерии знаний на базе декларативной модели, формируемой с использованием экспертных знаний, описывающих типовые свойства, структуры и закономерности отдельных классов систем. Объектно-ориентированный язык моделирования предоставляет следующие важные преимущества: возможность комплексирования декларативного представления знаний с процедурным; возможность сборки модели из готовых повторно используемых компонент; возможность накапливания теоретических и опытных знаний в виде библиотек классов.

Предлагаемое инструментальное средство моделирования сложных систем Engine 2.0 является развитием пакета Engine [6]. Оно реализовано на базе COM-технологий, что является актуальным на современном этапе развития программных продуктов.

МОДЕЛЬ СЛОЖНОЙ СИСТЕМЫ, ФОРМИРУЕМАЯ С ПОМОЩЬЮ ENGINE 2.0

Объектно-ориентированная модель сложной системы включает в себя следующие основные виды моделей (рис.1):

– модель классов (ClassModel), содержащую иерархию классов (структурированных наборов информационных единиц – атрибутов и методов) для описания компонент системы, а также методы для работы с классами;

– модель вариантов компонент системы (SubsystemObjectModel), содержащую дерево компонент (подсистем и присоединенных элементов) системы, каждой из которых сопоставлено множество вариантов ее реализации в виде мультиобъекта (набора экземпляров класса), а также методы для работы с мультиобъектами;

– модель зависимостей атрибутов (AttributeNetModel), содержащую множество отношений функциональной зависимости между атрибутами, описывающими компоненты системы, каждому из которых сопоставлена закономерность в виде набора правил-продукций, а также методы вывода на основе закономерностей.

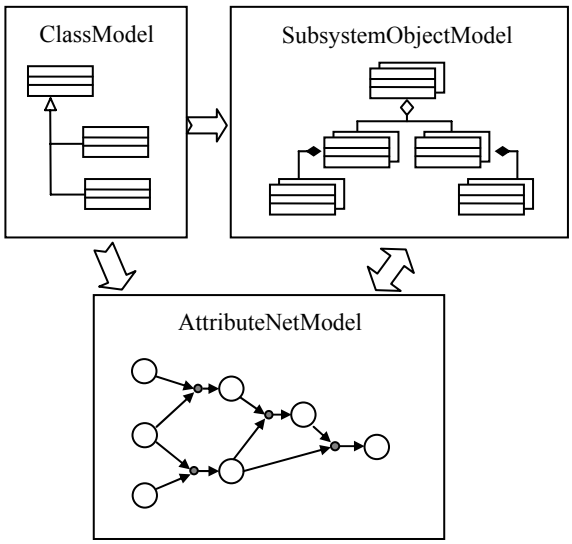

Рис. 1. Основные виды моделей сложной системы

Все модели взаимосвязаны. На базе модели классов формируется модель вариантов реализации компонент. Обе эти модели используются для создания модели зависимостей атрибутов. В процессе поиска решений на модели зависимостей модель вариантов используется как «доска объявлений», с которой считываются текущие значения атрибутов и в которую записываются найденные значения.

Рассмотрим подробнее каждую из перечисленных моделей.

Модель классов предназначена для задания структур описания компонент системы. Термином «компонента» будем обозначать любую часть предметной области, которая может быть выделена и описана как некоторая самостоятельная сущность. Это и система в целом, рассматриваемая как «черный ящик» на верхнем уровне абстрагирования, и любая часть системы – под-

система, элемент, комплексное свойство. Будем полагать, что описание компоненты вариативно, т.е. может отражать различные состояния или варианты реализации компоненты, однако структура описания (шаблон) инвариантна. Структура, определяющая состав атрибутов и методов, задается с помощью класса, сопоставляемого компоненте. На основе класса формируются экземпляры (объекты), содержащие конкретные значения атрибутов.

Модель классов включает множество классов $C = \{c_i\}$, на котором определено отношение наследования (inheritance) $R^{in} \in C \times C$. Класс представляет собой кортеж

$$c_i = \langle n(c_i), \{ n(c_j) \mid c_i R^{in} c_j \}, A(c_i), P(c_i) \rangle,$$

где $n(c_i)$ – имя класса; $\{ n(c_i) \}$ – множество имен классов, от которых наследуется данный класс; $A(c_i) = \{ a_m \}$ – множество атрибутов класса, включающее подмножества наследуемых и новых атрибутов; $P(c_i)$ – множество методов класса, включающее подмножества наследуемых, перекрытых и новых методов.

Каждый из атрибутов задается тройкой

$$a_m = \langle n(a_m), t(a_m), D(a_m) \rangle,$$

где $n(a_m)$ – имя атрибута; $t(a_m)$ – тип атрибута (<String>, <Real>, <Integer>, <Object>, …); $D(a_m) = \{ d_k \}$ – множество значений (домен) атрибута.

Домен задается в зависимости от типа атрибута. Например, для атрибута типа <Real> или <Integer> – в виде интервала, для атрибута типа <String> – в виде списка строк, для атрибута типа <Object> – в виде имени объекта. Посредством атрибутов типа <Object> в описание объекта может включаться описание другого объекта. Например, атрибуты подсистемы могут содержать ссылки на объекты, описывающие элементы данной подсистемы или некоторые комплексные свойства, которые сами описываются множеством собственных атрибутов. Множество методов класса содержит обязательное подмножество методов доступа к значениям атрибутов, которые могут быть двух типов: get – метод получения значения и set – метод задания значения.

Для работы с классами модель классов должна содержать собственные методы (присоединенные процедуры), которые можно разделить на две основных группы: методы наследования, позволяющие создавать новые классы на базе имеющихся классов; методы обобщения, позволяющие автоматически или в диалоговом режиме формировать иерархию наследования классов.

Основу модели вариантов реализации компонент составляет дерево компонент сложной системы. Обозначим множество компонент через $K = \{ k_n \}$. Из множества компонент выделим подмножество подсистем $S \subseteq K$, на котором определено отношение агрегации (aggregation) $R^{ag} \in S \times S$. Это отношение типа «часть–целое» («Part-of»). Множество подсистем, связанных отношениями агрегации, представляет собой дерево подсистем. Дерево, как правило, формируется путем последовательной декомпозиции подсистем.

Среди множества компонент будем выделять также подмножество элементов $E \subseteq K$. При этом под элементом будем понимать активные и пассивные сущности, участвующие в деятельности подсистемы (например, конечные продукты, предметы деятельности, средства деятельности, исполнители) или комплексные свойства, описываемые набором характеристик (например, технологические параметры, экономические результаты, технические условия и т.д.). Элемент может «присоединяться» к подсистеме или к другому элементу посредством указания ссылки на объект, сопоставленный присоединяемому элементу, в качестве значения одного из атрибутов объекта присоединяющей компоненты. Таким образом, объектное описание присоединяемой компоненты дополняет описание присоединяющей компоненты, т.е. является его частью. Обозначим отношение присоединения (connection) через $R^{cn} \in K \times K$. На рис. 2 представлено дерево компонент сложной системы.

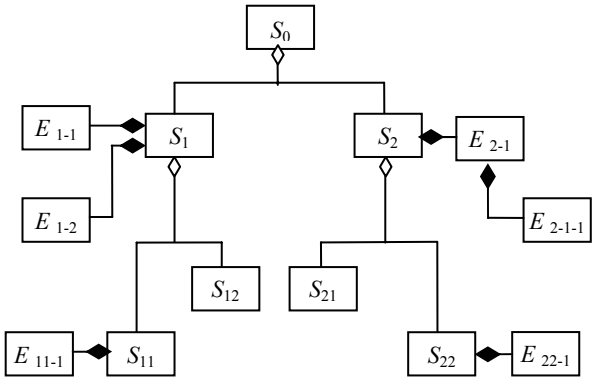

Рис. 2. Дерево компонент сложной системы

Каждой компоненте сопоставляется класс, на базе которого формируется один или несколько экземпляров класса (объектов), содержащих конкретные значения атрибутов, дополненные факторами уверенности:

$$o_k = \langle n(o_k), c(o_k), \{ d_k(a_m) / cf(d_k(a_m)) \} \rangle,$$

где $n(o_k)$ – имя объекта; $c(o_k)$ – указатель на класс, на базе которого реализован объект; $d_k(a_m) \in D(a_m)$ – значение атрибута; $cf(d_k(a_m)) \in [0;1]$ – фактор уверенности в значении атрибута, принимающий значение в интервале от 0 (полная недостоверность) до 1 (абсолютная достоверность). Если фактор уверенности не указан, по умолчанию он считается равным единице.

Для отражения множества вариантов или состояний компоненты (например, в различные моменты времени, в различных точках пространства, в различных условиях) используется мультиобъект – набор экземпляров, выделенных в соответствии с некоторым признаком p:

$$O^p(c_i) = \{ o_k \mid o_k = o(p), c(o_k) = c_i \} \subset O(c_i).$$

В качестве признака выступает заданный ключевой атрибут или комбинация нескольких ключевых атрибутов. Каждому значению признака соответствует свой экземпляр. Различают три основных типа базовых признаков – время, пространство и группа (population). От них могут наследоваться конкретные признаки.

Поскольку объект однозначно идентифицируется значениями ключевых атрибутов, обращение к нему осуществляется указанием после имени, общего для всех

объектов, в скобках этих значений. Отдельный объект можно рассматривать как вырожденный мультиобъект, мощность множества которого равна единице. На рис. 3 представлен пример мультиобъекта.

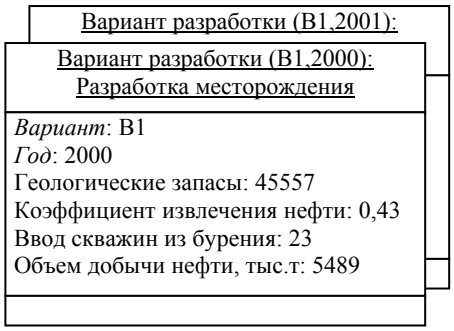

Рис. 3. Пример мультиобъекта

В состав множества методов модели вариантов реализации компонент могут быть включены следующие группы методов: методы создания и редактирования дерева подсистем и присоединенных элементов; методы создания и редактирования мультиобъектов; переборные методы генерации мультиобъектов (в частности метод морфологического анализа); методы поиска объектов по заданному шаблону поиска; методы вывода закономерностей на множестве объектов и т.д.

Модель зависимостей атрибутов включает множество отношений функциональной зависимости (functional dependence) между атрибутами, описывающими компоненты системы: $R^{fd} \in A \times ... \times A \left(A = \bigcup_i A(c_i) \right)$. Каждое такое отношение связывает некоторый атрибут-функцию a_0 с множеством атрибутов-аргументов a_1, …, $a_n : (a_1,..., a_n) R^{fd} a_0$. Закономерность, показывающая, как именно значение атрибута-функции определяется значениями атрибутов-аргументов, представляется совокупностью правил-продукций Rl_k вида:

$$Rl_k : (d(a_1) \in D_k(a_1)) \& ... \& (d(a_n) \in D_k(a_n)) \rightarrow$$
$$\rightarrow (d(a_0) = f_k(a_1,..., a_n)) / cf(Rl_k),$$

где $d(a_m)$ – текущее значение атрибута a_m; $D_k(a_m) \subseteq$ $\subseteq D(a_m)$ – допустимая область атрибута a_m; $f_k(a_1,..., a_n)$ – функция, представленная либо в виде конкретного значения, либо в виде формулы, либо в виде некоторой процедуры-функции; $cf(Rl_k) \in [0,1]$ – фактор уверенности правила. Правило имеет смысл: «ЕСЛИ атрибут a_1 принимает значение из области $D_k(a_1)$ … И атрибут a_n принимает значение из области $D_k(a_n)$, ТО атрибут a_0 принимает значение функции $f_k(a_1,..., a_n)$ с уверенностью $cf(RL_k)$».

Допустимые области значений в условной части правил для атрибутов с текстовыми значениями задаются перечислением значений, для количественных параметров – в виде интервалов. Если на значение атрибута не накладывается никаких ограничений (атрибут может принимать любое значение), то вместо допустимой области указывается пробел. Формула в заключении правила задается только для количественных атрибутов (типа Integer, Real) и может включать текущие значения атрибутов-аргументов, константы, знаки арифметических операций и математических функций и т.д. Процедура-функция может задаваться для атрибутов любого типа. Она может иметь в составе входных параметров текущие значения атрибутов-аргументов и должна возвращать значение определяемого атрибута.

Графически совокупность функциональных отношений можно представить в виде направленной сети без циклов, вершинами которой являются атрибуты, а дугами – функциональные зависимости между атрибутами (рис.4). В истоках сети располагаются независимые атрибуты, значения которых разработчик может непосредственно задавать, в стоках – целевые атрибуты, к которым относятся показатели эффективности и качества моделируемой компоненты системы.

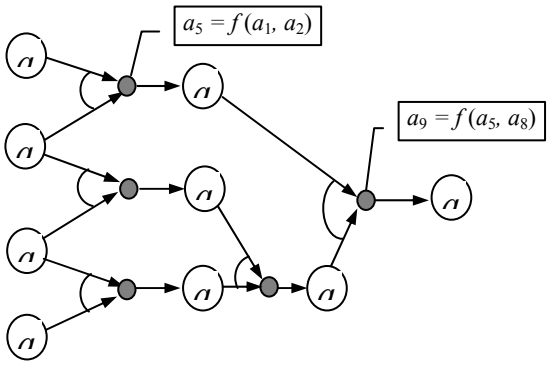

Рис. 4. Сеть отношений функциональной зависимости

В модели зависимостей атрибутов отношение функциональной зависимости задается в виде объекта стандартного класса «FuncDependence», содержащего имя атрибута-функции, список имен атрибутов-аргументов, а ссылку на мультиобъект – с правилами-продукциями, описывающими закономерность.

Множество методов модели зависимостей атрибутов включает две основные группы: методы формирования сети зависимостей и задания закономерностей; методы поиска решений, в том числе методы прямого и обратного вывода на модели функциональных отношений [7].

СТРУКТУРА ИНСТРУМЕНТАЛЬНОГО СРЕДСТВА ENGINE 2.0

Основными принципами реализации комплекса Engine 2.0. являются: языковая независимость реализации отдельных компонент; открытая архитектура, позволяющая интегрировать различные приложения, удобство пользовательского интерфейса.

Программный комплекс включает в себя три базовых модуля (рис. 5): «Project (Проект)», «Register (Менеджер регистрации)» и «Work space manager (Менеджер рабочей области)».

Модуль «Project» является основным, отражающим модель (совокупность моделей) предметной области, создаваемую пользователем. Он состоит из множества, так называемых узлов (Project node), соответствующим отдельным видам моделей. Стандартными узлами проекта являются: узел классов (classes), узел деревьев мультиобъектов подсистем (subsystems objects), узел зависимостей атрибутов (attributes net). Кроме того, пользователь может определить собственные виды узлов проекта.

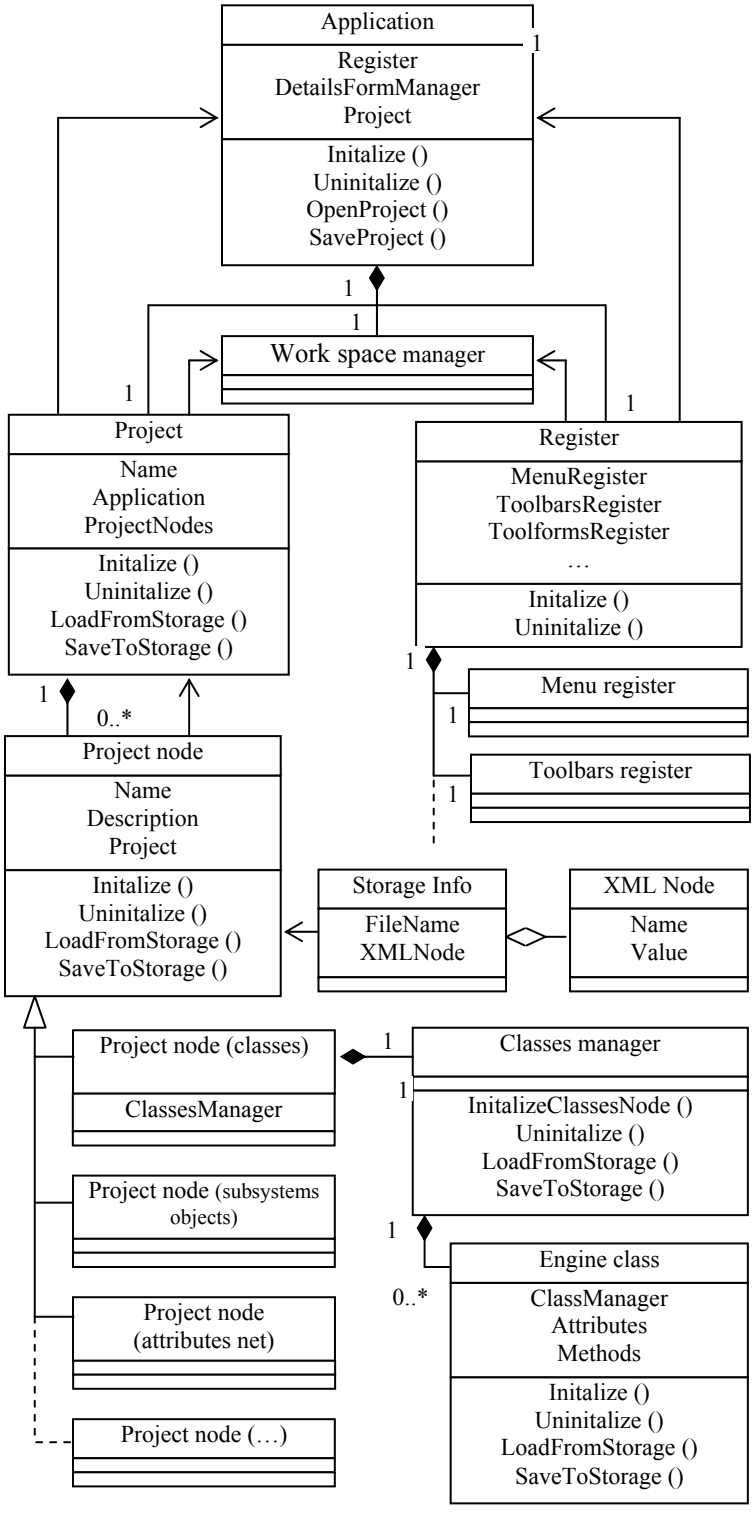

Рис. 5. Диаграмма классов Engine 2.0

Узел проекта включает множество элементов формируемой с его помощью модели и менеджер узла, позволяющий создавать соответствующую модель, записывать ее в хранилище, загружать сохраненную ранее модель из хранилища и т.д. Так, узел классов содержит множество элементов типа «Engine class», описывающих классы компонент моделируемой сложной системы. С помощью менеджера классов пользователь может создавать новые классы, редактировать существующие и т.д. Менеджеры узлов обеспечивают возможность визуального представления проектируемой системы в виде различных диаграмм (иерархии классов, дерева подсистем и элементов, сети зависимостей атрибутов). Все стандартные узлы имеют однотипный пользовательский интерфейс, что значительно упрощает и ускоряет процесс обучения пользователя работе с системой.

Результаты проектирования системы могут быть сохранены в хранилище в виде файлов формата XML.

Каждый из узлов для обмена информацией с хранилищем использует модуль «Storage Info», содержащий множество узлов «XML Node» для записи (чтения) информации об отдельных элементах модели.

Любой узел реализуется в виде COM-библиотеки, содержащей некоторый набор обязательных объектов с определенными интерфейсами (меню, панелью инструментов, компонентами, формами и т.д.). Пользователю предоставляется возможность замены любого узла на альтернативный, а также возможность добавления новых программ. Для того чтобы интегрировать новую библиотеку с Engine 2.0, ее необходимо зарегистрировать. Для регистрации используется менеджер регистрации «Register». В его состав входят модули, осуществляющие регистрацию отдельных элементов интерфейса, таких, как меню, панели инструментов и т.д.

Модуль «Work space manager» обеспечивает взаимодействие всех модулей системы.

ЗАКЛЮЧЕНИЕ

Инструментальный комплекс Engine 2.0. предоставляет непрограммирующему пользователю удобные средства для создания моделей проектируемых сложных систем и поиска решений на моделях. Для генерирования вариантов компонент системы, их оценки и выбора могут быть использованы традиционные методы системного анализа и теории принятия решений, такие, как морфологический анализ, методы одно- и многокритериального выбора и др., адаптированные к объектно-ориентированному описанию системы, а также методы инженерии знаний, в частности методы нечеткого логического вывода на множестве правил-продукций.

Разработчик модели может использовать типовые классы для описания свойств и характеристик системы, а также экспертные знания о зависимостях между характеристиками. В то же время он легко может пополнять, корректировать состав атрибутов, изменять правила определения их значений. Декларативное задание зависимостей позволяет легко изменять вид зависимостей, не меняя методы, их обрабатывающие.

Комплекс был использован для формирования региональной программы повышения энергетической эффективности на территории Томской области, а также для создания системы выбора инвестиционных проектов разработки нефтегазовых месторождений.

ЛИТЕРАТУРА

1. *Буч Г, Рамбо Д., Джекобсон А.* Язык UML. Руководство пользователя / Пер. с англ. М.: ДМК, 2000. 432 с.
2. *Силич М.П.* Системная технология: объектно-ориентированный подход. Томск: Том. гос. ун-т систем управления и радиоэлектроники, 2002. 224 с.
3. *Силич В.А., Силич М.П.* Проектирование сложной системы на основе объектно-ориентированного подхода // Известия Томского политехнического университета. 2003. Т. 306. № 2. С. 99–103.
4. *Силич В.А., Силич М.П.* Метод объектного моделирования для проектирования сложных систем // Автоматизация и современные технологии. 2003. № 4. С. 14–21.
5. *Силич М.П.* Объектно-ориентированная модель сложной системы // Ползуновский вестник. 2004. № 3. С. 93–98.
6. *Безвершенко С.А., Силич В.А., Сорокин В.А.* Инструментальное средство поддержки проведения системного анализа «Engine» // Интеллектуальные автоматизированные системы проектирования, управления и обучения: Сб. статей. Томск: Изд-во НТЛ, 2000. С. 185–192.
7. *Силич М.П., Хабибулина Н.Ю.* Поиск решений на модели функциональных отношений // Информационные технологии. 2004. № 9. С. 27–33.

Статья представлена кафедрой автоматизации обработки информации факультета систем управления Томского государственного университета систем управления и радиоэлектроники, поступила в научную редакцию «Кибернетика» 15 сентября 2004 г.