ятия. ДИС. – № 5, 2002. – С. 6-16.
5. *Астахов А.* Разработка эффективных политик информационной безопасности // IT Директор – №1. – 2005.

Дагаев Александр Владимирович
Технологический институт федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге
E-mail: fin_val_iv@tsure.ru
347928, Таганрог, ГСП 17А, Некрасовский, 44. Тел: 88634-371-689

Лебедев Владимир Борисович
E-mail: fin_val_iv@tsure.ru
Тел: 88634-311-310

Бородянский Юрий Михайлович
E-mail: fin_val_iv@tsure.ru
Тел: 88634-311-310

Ивахненко Александр Александрович
E-mail: fin_val_iv@tsure.ru
Тел: 88634-311-310

Проскуриков Александр Викторови
E-mail: fin_val_iv@tsure.ru
Тел: 88634-311-310

Dagaev Alexandr Vladimirovich
Taganrog Institute of Technology - Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University
E-mail: fin_val_iv@tsure.ru
44, Nekrasovsky, Taganrog, 347928. Phone: 88634-371-689

Lebedev Vladimir Borisovich
E-mail: fin_val_iv@tsure.ru
Phone: 88634-311-310

Borodiansky Ury Mikhailovich
E-mail: fin_val_iv@tsure.ru
Phone: 88634-311-310

Ikhvanenko Alexandr Alexandrovich
E-mail: fin_val_iv@tsure.ru
Phone: 88634-311-310

УДК 681.3.06

Н.В. Драгныш

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДОВ И МОДЕЛЕЙ ПОСТРОЕНИЯ КОМПЛЕКСОВ ПРОГРАММ

Рассмотрены базовые модели и принципы построения комплексов программ, основанные на разработках в процессе проектирования, анализа и син-

теза сложных систем, использование которых позволяет создать среду конструирования комплексов программ, удовлетворяющую рассмотренным требованиям. Предложен конструктор моделей системы, отличающийся универсальностью, общностью подхода к реализации различных видов моделей системы. Конструкторы моделей позволяют специалистам конкретных предметных областей разрабатывать модели программных систем в понятиях самих предметных областей, а не с помощью конструкций языков программирования.

Модель; системы; программирования.

N.V. Dragnysh

RESEARCH AND DEVELOPMENT OF THE METHODS AND MODELS OF PROGRAMS COMPLEXES CONSTRUCTION

Base models and principles of programs complexes construction, based on developments in the process of designing, the analysis and synthesis of the difficult systems which usage allows to create the environment of programs complexes designing, meeting the considered requirements are considered. The designer of system's models, different by universality, an approach generality to implementation of various sorts of system's models is offered. Models designers allow experts of concrete subject domains to develop program systems models in subject domains concepts, instead of by means of programming languages constructions.

Model; system; programming

Постановка задачи. Все более ускоряющаяся компьютеризация всех областей человеческой деятельности приводит к огромному росту потребности в программных продуктах, которые будут удовлетворять потребностям все новых пользователей. При этом, будущие или настоящие пользователи компьютеров не в состоянии написать программный продукт из-за непреодолимой для них сложности языков программирования, а профессиональные программисты, которых не хватает, вынуждены каждый раз тратить значительное время на изучение предметной области, в которой будет использоваться их программный продукт.

Выходом из сложившейся ситуации может служить создание среды разработки сложных комплексов программ. Эта среда должна быть реализована таким образом, чтобы для создания программного продукта не требовались глубокие познания в области программирования и чтобы качественный программный продукт мог быть написан представителем конкретной предметной области, конкретного производства, который, в свою очередь, владеет темой намного лучше даже самого подготовленного программиста.

Анализ современной сферы программирования и развития компьютерных технологий позволяет зафиксировать очередной текущий кризис программирования. Возникновение всех кризисов программирования являлось следствием отставания повышения эффективности разработки программных систем от запросов пользователей и развития аппаратных решений.

Математические основы и базовые принципы построения среды разработки комплексов программ. Сложные системы в программировании ничем не отличаются от других сложных систем. Следовательно, подходить к разработке программных систем надо с помощью наработок, полученных при проектировании, анализе и синтезе сложных или больших систем [1].

Исходя из анализа развития методов и моделей построения комплексов программ, текущей ситуации в проблемной области и проблем сложных систем

можно выработать требования, предъявляемые к среде разработки комплексов программ, выполнение которых необходимо для выхода из текущего кризиса программирования:
- Ориентированность на пользователя-непрограммиста.
- Разработка средств управления сложностью.
- Поддержка не только синтеза системы, но и анализа, хотя бы на описательном уровне. Также должна существовать возможность перехода от задачи реализации системы к задаче поиска (исследования) ее вида.
- Гибкое управление уровнем абстрагирования.
- Возможность гибкого редактирования проектируемой системы в любой момент времени, в том числе и при эксплуатации.
- Повторное использование конструкций как на уровне любой работающей подсистемы, так и на уровне модели предметной области.
- Контроль ошибок. Пользователь должен быть поставлен в такие условия работы, при которых ему не только будет легче обнаруживать ошибки, но и гораздо труднее их совершать.
- Автоматическая реализация параллельности.
- Совместимость как можно с большим количеством языков программирования.

Любую систему можно представлять в виде совокупности элементов, соединенных между собой различными связями. При этом элементы сами могут являться системами и в свою очередь состоять из подсистем, соединенных связями. Каждый элемент удобно рассматривать как "черный ящик", т.е. он имеет входы и выходы для связи с другими элементами, а также внутренность, скрытую от разработчика на данном уровне иерархии.

В качестве основных “кирпичиков” для построения сложных программных систем целесообразно использовать понятия: система, связи и модели.
Формально под программной системой будем понимать объект [2,3]: $s = [X, Y, G, M]$, где X – множество входов; Y – множество выходов; G – множество сигналов управления; M – множество моделей. Множества X,Y разных систем образуют связи между системами. Все связи между двумя подсистемами s_i и s_j образуют множество связей $(C_{i,j} : s_i \to s_j)$. Поток управления G включает в себя управление системой (например, изменение входов, выходов, выбор текущей модели, изменения числа моделей) и управление выделенной моделью (например, настройка универсальной модели предметной области для решения частной задачи).

Каждая система содержит множество моделей. Причем одна из моделей является выделенной. Именно она и определяет текущее преобразование входов в выходы. $M = \{m_1, \ldots, m_j^*, \ldots, m_N\}$. Под моделью системы будем понимать упрощенное отображение существенных сторон реальной системы, выраженное в некоторой форме и описывающее правило (оператор) преобразования входных X сигналов в выходные Y. ($Y=M(X)$, где M - модель системы).

Оператор M представляет собой совокупность любых операций и действий (математических, логических, алгоритмических и т.д.), позволяющих установить соответствие между входными и выходными сигналами.

Совокупность доступных операций и действий, а также правила и ограничения их применения для разработки конкретной модели системы определяет язык описания моделей.

Язык описания моделей определим как упрощенное отображение существенных сторон реальной проблемной области, позволяющее в понятиях этой области описать модель системы.

Таким образом, осуществляется запись постановки задачи и методов ее решения в терминах только самой задачи. Проектировщик должен выбрать язык описания моделей из некоторого банка языков и описать с помощью него модель для решения конкретной прикладной задачи. При этом в общем случае пользователь занимается исследованием, а не программированием.

Заметим, что пользователь не обязательно имеет дело напрямую с некоторым языком, в общем случае построение модели может происходить через интерфейс, в том числе возможно и интеллектуальный. Способ взаимодействия разработчика системы (модели системы) и модели проблемной области определяет разработчик языка описания моделей.

Интерпретатор языка описания моделей преобразует пользовательское описание модели либо в язык, понимаемый компьютером, либо в запись модели на другом языке описания моделей.

Некоторые понятия и отношения, определяющие модель системы трудно или нецелесообразно определять внутри только одной предметной области. Поэтому в каждом языке описания моделей должен существовать инструмент, позволяющий использование других языков для доопределения некоторых частей рассматриваемой модели. Чтобы унифицировать этот процесс, необходимо выбрать способ такого взаимодействия и представить его как требование для языка описания моделей. Так как мы создаем модель для системы, то для доопределения ее частей логично использовать подсистемы.

Таким образом, в любом языке описания моделей должны существовать встроенные конструкции, описывающие подсистемы. Описание ограничивается только входами и выходами. То есть модель относится к подсистеме как к черному ящику, ее не интересует реализация этой подсистемы. Соответственно входы и выходы участвуют в описании модели, в задании понятий и отношений. Подсистема на новом уровне абстрагирования становится самостоятельной системой полностью независимой от модели, впервые описавшей ее. Для реализации моделей этой новой системы соответственно не налагается никаких ограничений на языки описания модели. Ограничения накладываются только на имя новой системы, названия и типы входов и выходов. Менять их можно соответственно только в модели базовой системы.

Исследователь (проектировщик) должен уметь собирать различные точки зрения, коллекционировать полноту представления. Этому способствует множество моделей, которое поддерживает циклическую модель жизненного цикла системы, отражает эволюцию моделей. Переключение между моделями может осуществляться во время работы системы, через поток управления.

Модель должна позволять рассматривать проблему с различной степенью полноты (настраиваемый инструмент). С помощью потока управления возможна адаптация модели прямо во время работы системы.

Для реализации конкретной модели системы с помощью понятий некоторой предметной области пользователю надо выбрать конкретный язык описания модели. Чтобы был возможен выбор из некоторого банка языков, к каждому ЯОМ надо прикрепить некоторое описание. Для того чтобы компьютер понимал модель, описанную на ЯОМ, необходим интерпретатор ЯОМ. И, наконец, необходим некоторый интерфейс, с помощью которого пользователь сможет работать с языком описания модели и его интерпретатором. Введем конструктор мо-

дели, который можно описать следующим образом [3]: *КМ=[ЯОМ, ИЯОМ, ИКМ, ОКМ]*, где *ЯОМ* – язык описания моделей, *ИЯОМ* – интерпретатор языка описания моделей, *ИКМ* – интерфейс конструктора модели, *ОКМ* – описание конструктора модели.

ИЯОМ преобразует описание модели системы в вид, понимаемый компьютером или другим конструктором модели. *ИКМ* в простейшем случае – редактор языка описания моделей, в более сложном – интеллектуальный интерфейс, упрощающий работу пользователя с *ЯОМ* в рамках конкретной предметной области. Вид и содержание интерфейса полностью зависит от производителя конструктора модели. *ОКМ* включает в себя как подробное, так и краткое описание всего конструктора модели, понятия и конструкции, используемые в *ЯОМ*, место в классификации конструкторов моделей и т.д.

Представим иерархию конструкторов модели и их групп в виде выходящего дерева [4, 5] (орграф с источником, не имеющий полуконтуров): $G = (X, U)$. Под ориентированными дугами будем понимать отношение включения, т.е. запись вида $< x, y >$ означает, что группа x включает элемент y. Множество вершин разделим на три непересекающихся подмножества:

$$\begin{cases} X = x_{нач} \cup X_{гр} \cup X_{КМ} \\ x_{нач} \cap X_{гр} = X_{гр} \cap X_{КМ} = \varnothing \end{cases},$$

где $x_{нач}$ – начальная группа (источник исходящего дерева), $X_{гр}$ – множество групп, $X_{КМ}$ – множество конструкторов модели.

Пользователь, для того чтобы реализовать конкретную систему, должен выбрать некоторый язык описания моделей, т.е. конструктор модели. Выбор можно осуществить через иерархию, на каждом уровне отметая или выбирая группу, пока, в конечном счете, не придем к конкретному конструктору модели, т.е. к конкретному языку описания моделей. Данный подход учитывает родственность, взаимосвязь конструкторов моделей, но может быть достаточно громоздким, так как приходится осуществлять перебор ветвей иерархии. Как альтернатива запрос конструктора моделей с искомыми характеристиками [6]. Пользователь вводит запрос, в котором содержатся ключевые слова о типе конструктора модели, языке описания моделей, искомой предметной области. Среда разработки выводит ему ссылки на подходящие конструкторы модели, используя информацию запроса и *ОКМ* всех конструкторов моделей и их групп.

Методологии построения комплексов программ. Общая последовательность действий, выполняемая при разработке программных систем:

1. Создание новой программной системы.
2. Определение входов-выходов в текущей системе.
3. Определение связей текущей программной системы с другими программными системами, в том числе возможно с операционной системой.
4. Реализация одной или нескольких моделей системы.
5. Выбор текущей (активной) модели.
6. Выполнение программной системы (выполняется средой разработки без активного вмешательства разработчика).

Определенные выше действия можно выполнять параллельно, осуществлять возвраты на любой шаг в любой момент времени: изменять входы-выходы, переопределять связи с другими системами, расширять набор моделей системы, редактировать любую модель. Причем для редактирования программной систе-

мы не нужно приостанавливать ее выполнение. Разработка системы является процессом исследования и поиска оптимального или удовлетворяющего решения, в отличие от классического программирования, представляющего собой процесс кодирования жесткого алгоритма (причем последовательность операций алгоритма определяет, по сути, управление ресурсами ЭВМ).

Переход на абстракции уровня подсистем позволяет использовать такие положения как автоматическая инкапсуляция, агрегация, виртуализация, классификация [7]. Следование этим принципам в рамках рассматриваемой методологии разработки может существенно облегчить работу пользователя.

Инкапсуляция подразумевает объединение и защиту кода и данных в некоторой сущности. Инкапсуляция необходима, если к создаваемому проекту предъявляются требования надёжности и модернизируемости. Главным следствием инкапсуляции можно назвать необычайно высокую гибкость и настраиваемость системы. Автоматическая инкапсуляция позволит безбоязненную смену модели подсистемы не только на этапе проектирования, но и при работе программной системы.

Вполне естественным является использование агрегации нескольких систем в новую систему-контейнер. Каждая подсистема обладает некоторым декларированным интерфейсом, с помощью которого с ней могут взаимодействовать другие подсистемы. Контейнеры образуют собственную иерархию – иерархию вложений. Контейнеры могут иметь неограниченную, по крайней мере, теоретически, глубину вложенности, вплоть до элементарных составляющих. Важное качество контейнера заключается в том, что его можно конструировать, а не кодировать.

Простота и удобство механизма виртуальности подсистем в контейнере, предоставляемых агрегацией, позволяют существенно облегчить разработку сложных систем за счёт перехода от упрощенных опытных моделей к промышленным образцам. С другой стороны, на основе данного вида виртуализации можно моделировать и реальную эволюцию сложных систем.

Готовые системы хранятся в хранилище, для удобства работы с ними используется классификация систем. Функционально родственные системы объединяются в классы, которые, в свою очередь, объединяются в классы следующего уровня иерархии. Классы не могут прямо использоваться как системы. Они только несут информацию об общих функциональных свойствах систем, входящих в этот класс.

На основе рассматриваемой технологии становится проще разрабатывать комплексы программ с высоким уровнем параллелизма. Систему можно представить как некоторую обособленную сущность, взаимодействие с которой осуществляется с помощью объявленного интерфейса. Автоматическая инкапсуляция защищает внутренние подсистемы от случайного или умышленного изменения. Как следствие, несколько систем могут работать параллельно, избегая негативного влияния друг на друга.

Для разработки модели системы пользователь должен выбрать конкретный конструктор модели. То есть разработкой программной системы, необходимой для выполнения задач конкретного пользователя, занимается он сам, используя для этого знакомые ему понятия и конструкции (выбирая соответствующий конструктор модели). А создание конструкторов модели выполняют профессионалы-программисты, скрывающие особенности работы с компьютером за интерфейсом конструктора модели и языком описания модели.

Общая последовательность действий, выполняемая при разработке модели системы:

1. Добавление новой модели во множество моделей текущей системы.
2. Выбор конструктора модели наиболее подходящего для текущей системы и конкретного пользователя.
3. Реализация модели системы выбранным конструктором модели.
4. Разработка подсистем, выделенных разработчиком при конструировании модели системы средствами конструктора.

Важным аспектом методологии разработки программных систем является реализация множества моделей системы с помощью различных конструкторов моделей. Эффективность моделей одной системы разработчик может исследовать и “переосмысливать” на любом этапе разработки текущей системы и в любой момент переключить текущую модель системы.

Проектирование среды разработки сложных программных систем. Общая структура среды разработки изображена на рис. 1.

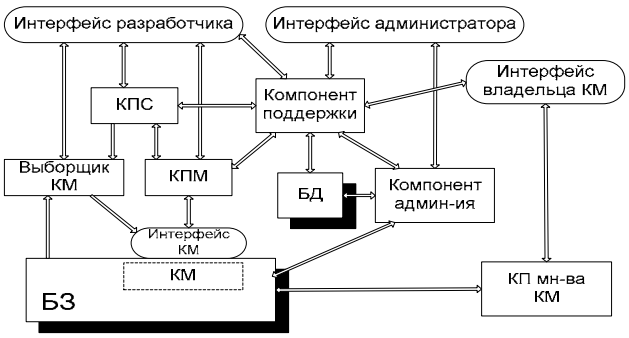

Рис. 1. Архитектура среды разработки сложных программных систем

Интерфейс разработчика предоставляет графические средства пользователю для работы со средой разработки [8].

КПС – компонент поддержки системы. КПМ – компонент поддержки модели. Выборщик КМ – выборщик конструктора моделей. БЗ – база знаний. Интерфейс КМ – интерфейс конструктора моделей (ИКМ). Компонент поддержки предназначен для выполнения вспомогательных функций, обеспечивает поддержку разработки программных систем и их моделей, разграничивает данные и процессы в многопользовательской среде, предоставляет доступ к необходимым ресурсам. БД – база данных. КП мн-ва КМ – компонент поддержки множества конструкторов моделей. Интерфейс владельца КМ – интерфейс владельца конструктора моделей.

Показатели эффективности разработки, рассчитанные для реализаций конкретных систем с помощью разработанной среды построения комплексов программ значительно лучше аналогичных реализаций на объектно-ориентированном языке.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Молчанов А. А.* Моделирование и проектирование сложных систем – Киев: Выща школа, 1988. – 360с.

2. *Астанин С.В., Драгныш Н.В.* Базовые принципы построения среды проектирования сложных систем / Труды XI Всероссийской науч.-мет. конф. «Телематика». Т.1. – СПб. 2004. – С. 202-203.
3. *Астанин С.В., Драгныш Н.В.* Математические основы построения среды проектирования сложных программных систем Труды Международных научно-технических конференций “Интеллектуальные системы”(AIS’05) и “Интеллектуальные САПР”(CAD-2005). Научное издание в 4-х томах. – М.: ФИЗМАТЛИТ, Т.4. 2005.
4. *Харари Ф.* Теория графов /пер. с англ. – 2-е изд. – М.: УРСС, 2003. – 300 с.
5. *Татт У.* Теория графов /пер. с англ. – М.: Мир, 1988. – 424 с.
6. *Драгныш Н.В.* Язык запросов среды разработки сложных программных систем Сборник трудов по итогам XI Международной открытой научной конференции “Современные проблемы информатизации в моделировании и программировании”. – Воронеж, 2006. – С.227-228.
7. *Драгныш Н.В.* Концептуальные положения на уровне систем для новой среды проектирования/ Информационные технологии моделирования и управления. – Воронеж: Научная книга, Выпуск 1(26), 2006.
8. *Астанин С.В., Драгныш Н.В.* Архитектура среды разработки сложных программных систем / Известия ТРТУ. Темат. выпуск «Интеллектуальные САПР». – Таганрог: Изд-во ТРТУ, 2006. – № 8(63).

Драгныш Николай Васильевич
Таганрогский государственный педагогический институт
E-mail: nicktrtu@yandex.ru
347936, г. Таганрог, ул. Инициативная, д. 48. Тел: 88634 60-18-99

Dragnysh Nikolay Vasilievich
Taganrog State Pedagogical Institute
E-mail: nicktrtu@yandex.ru
48, Initsiativnaia, Taganrog, 347936. Phone: 88634 60-18-99