УДК 004.4.242

Гагарин А.П.

Московский авиационный институт (национальный исследовательский университет), Москва, к.т.н., профессор по кафедре информатики

ИНТЕГРАЦИЯ ПРОЦЕССОВ ПРОЕКТИРОВАНИЯ И КОНСТРУИРОВАНИЯ ПРОГРАММ В ГИПЕРТЕКСТОВОЙ СРЕДЕ

АННОТАЦИЯ

Предлагается подход к преодолению технологического разрыва между программированием как конструированием программ на языке программирования и предварительным проектированием путём использования единой гипертекстовой среды, предоставляющей необходимые средства как для проектирования, так и для конструирования.

КЛЮЧЕВЫЕ СЛОВА

Проектирование программ; конструирование программ; язык UML; доменная спецификация; класс; диаграмма; атрибут класса; операция класса; ассоциация; гипертекст.

Gagarin A.P.

Moscow Aviation Institute (National Research University), Moscow, Russia

INTEGRATION OF SOFTWARE DESIGN AND CONSTRUCTION PROCESSES IN THE HYPERTEXT ENVIRONMENT

ABSTRACT

An approach is outlined how to overcome the technological gap between constructing of software products in programming language and its preceding design by using a common hypertext environment that provides all means needed for design as well as for constructing.

KEYWORDS

Software design; software construction; UML; domain specification; class; diagram; class attribute; class operation; association; hypertext.

Стандарты и авторитетные руководства в области программной инженерии декларируют, что составлению (“конструированию”) исходного текста программного продукта на языке программирования должны предшествовать разработка требований к нему и соответствующее проектирование. Как разработка требований (спецификаций), так и проектирование сопровождается моделированием среды, в которой должен найти применение разрабатываемый продукт, за которым следует моделирование самого продукта. Для этой цели применяются в разной степени формализованные изобразительные средства, такие как блок-схемы, ER-диаграммы, и приобретающий всё большее распространение язык UML, предложенный Консорциумом OMG -- международной «Рабочей группой по созданию и продвижению объектно-ориентированных технологий и стандартов» [1].

Универсальность языка UML выражается, в частности, в том, что модель объекта может быть представлена в разных “видах” на специализированных диаграммах. Диаграммы классов и компонентов отображают структуру самой программы и обрабатываемых данных. Примером использования диаграмм классов могут служить доменные спецификации, разрабатываемые в OMG. При следовании этим спецификациям в процессе создания программных систем конфигурации классов, показанные в диаграммах, могут быть легко переписаны на объектно-ориентированных языках программирования C++, C#, Java и включены в соответствующие программные продукты.

Развитые продукты для моделирования на языке UML, такие, как StarUML [2] и Software Architect Designer [3], обеспечивают автоматическое преобразование диаграмм классов в модули на языках программирования, основным содержанием которых являются определения классов.

Автоматизированное преобразование диаграмм классов в запросы языка SQL, формирующие модель данных в базе данных, обеспечено в Visual Studio корпорации Microsoft.

Язык UML позволяет выражать не только структурные, но и алгоритмические, в терминах OMG – "поведенческие" особенности программ. Для этого, в основном, служат диаграммы активности, состояний и последовательностей.

Для представления условий и некоторых простых действий в диаграммах языка UML допускается использование выражений на языке OCL - Object Constraint Language [4]. Эти выражения размещаются в пространстве диаграммы. Явно, посредством идентификаторов, определённых в модели UML, или неявно, по умолчанию, могут быть заданы ссылки на сущности модели.

Любые тексты, в частности, фрагменты программ на языках программирования, могут быть размещены в комментариях, каждый из которых связывается специальной связью с некоторой сущностью диаграммы. Но ссылки из текстов комментариев к сущностям диаграммы или свойствам модели не обеспечиваются.

Известные средства моделирования на UML не предоставляют готовых инструментов для преобразования алгоритмов, выраженных на этом языке, в тексты на языках программирования. Они лишь открыты для подключения программных компонентов, выполняющие указанные преобразования по некоторому шаблону. Обеспечить шаблон и реализующие его программные компоненты предоставляется пользователю. В этих условиях объективно образуется разрыв между процессом проектирования и процессом конструирования программного продукта: структурные и процедурные решения, зафиксированные в проекте программного продукта средствами проектирования должны быть «вручную» воспроизведены на языке программирования. В результате разработчик программного продукта оказывается не заинтересован в углублении проекта, несмотря на то, что решения, представленные на языке проектирования обладают, как правило, большей наглядностью, чем на языке программирования.

Для преодоления разрыва между проектированием и конструированием программных продуктов предлагается использовать единую среду, обладающую всеми основными возможностями как сред проектирования, так и сред программирования. Модель такой среды реализована на базе специального гипертекстового редактора [5]. Редактор обеспечивает создание, развёртывание и архивирование "мелкогранулированного" гипертекста.

Узел этого гипертекста представляет одну или несколько строк текста в рамке, не видимых, пока узел не открыт в результате "вызова" ссылки.

Ссылка вызывается или создается, когда "курсор" находится на слове текста, путём нажатия правой клавиши мыши. Если ссылка от этого слова уже создана, она вызывается. В противном случае создаётся новый узел гипертекста, на который будет ссылаться данное слово. Ссылка может быть создана от любого слова, но не более, чем одна. Только что созданный узел открывается как пустое строчное окно, и в него устанавливается "каретка", то есть "текстовый курсор", что позволяет сразу же после создания ссылки продолжать заносить текст в новый узел. С помощью мыши можно совместить только что порожденный узел с другим существующим узлом. Узел, содержащий ссылку, и узел, открытый по этой ссылке, соединяются пунктирной, в потенциале коленчатой линией – "ребром" гипертекста.

Узел может перемещаться мышью при нажатой левой клавише. Установив в окне узла каретку, узел можно погасить, нажав клавишу ESCAPE или уничтожить узел вместе со ссылкой на него, нажав клавишу DELETE при нажатой клавише ALT. Погашенный узел может быть открыт снова.

Путем двух последовательных вызовов ссылок можно обеспечить условный вызов ссылки. Для этого текстовое содержание ссылки, вызываемой первой, составляется из "ключевых" слов, для каждого из которых создается ссылка. Эти ссылки второго уровня могут быть вызваны в зависимости от выбора ключевого слова в тексте первой ссылки. Эти ключевые слова образуют условия вызова ссылок второго уровня, называемые аспектами. В результате из одного и того же слова гипертекста могут исходить несколько ссылочных связей, относящихся к различным аспектам. Аспекты произвольно задаются программистом и назначаются связям. Созданная ссылка остается на экране, и от неё можно построить новые ссылки.

Указанные возможности рассматриваемой гипертекстовой среды достаточны для визуального представления диаграмм классов и компонентов UML:

- классы и компоненты представляются узлами;
- атрибуты, операции и сигналы – текстами в узлах, представляющих классы;
- отношения между классами – агрегатами узлов; причём наименование и другие атрибуты ассоциаций представляются специальными узлами внутри агрегата, которые вызываются от центрального узла агрегата, представляющего ассоциацию в целом, как показано на Рис.1.

Для точного представления отношений в гипертекстовом редакторе нетрудно обеспечить узлы с прозрачными рамками, ограничивающими текст, и различным исполнением рёбер гипертекста (сплошная линия, мелкий или крупный пунктир, линия с конечной стрелкой или ромбом).

Привязка узла гипертекста к слову внутри узла позволяет визуально представить связь между именами и их значениями для фрагментов программ, помещённых в узел.

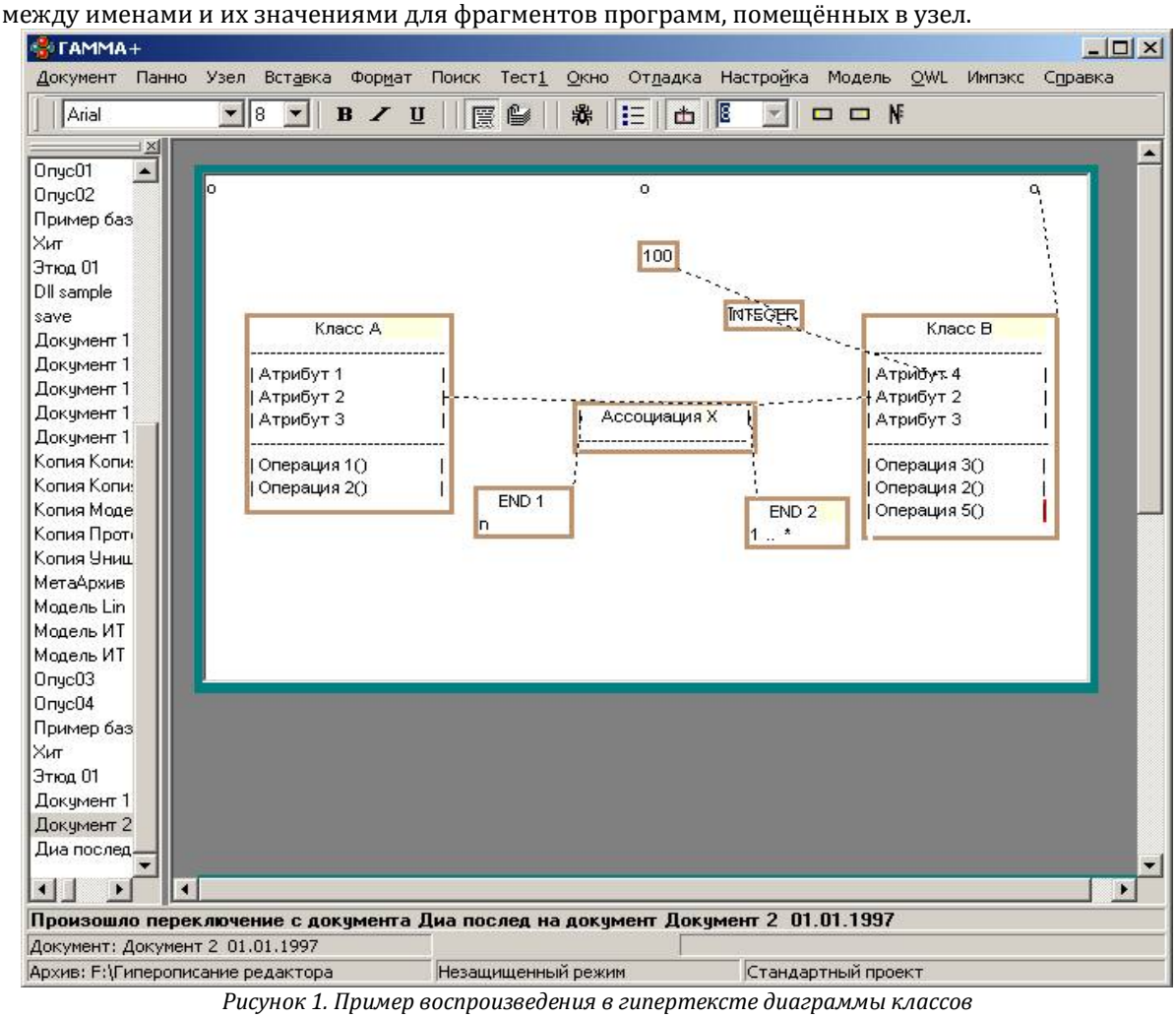

Рисунок 1. Пример воспроизведения в гипертексте диаграммы классов

Подобным образом в гипертексте можно представить диаграмму последовательностей. Объект класса изображается узлом, а “линии жизни” и “линки” (связки) между событиями на этих линиях – рёбрами гипертекста, как на рис.2.

Совмещение в одном пространстве структурного и алгоритмического описания программы позволяет при проектировании переходить от общих решений к детальным, от проектирования к конструированию без потери мотивирующей информации, обеспечивая прослеживание этих решений.

При достижении определённой детальности проекта оказывается возможным организовать моделирование работы проектируемой программы в динамическом режиме (режиме имитации). Универсальные возможности гипертекстового редактора дополняются следующими специализированными:

Вводятся два новых типовых узла (дескриптора) гипертекста:

- П-дескриптор;
- М-дескриптор – драйвер ансамбля П-узлов.

Оба дескриптора подключаются к любому узлу гипертекста как дополнительное свойство.

П-дескриптор наделяет узел свойством быть “программируемым”. Такому узлу по ссылке от П-дескриптора сопоставляется программа-обработчик, которая задаёт преобразование значений свойств узла. Программируемый узел входит в состав некоторого ансамбля программируемых узлов (не более чем в один ансамбль). Драйвер ансамбля программируемых узлов координирует выполнение обработчиков узлов, запуская их в определенной последовательности и передавая между ними данные в случае, если такая передача предусмотрена.

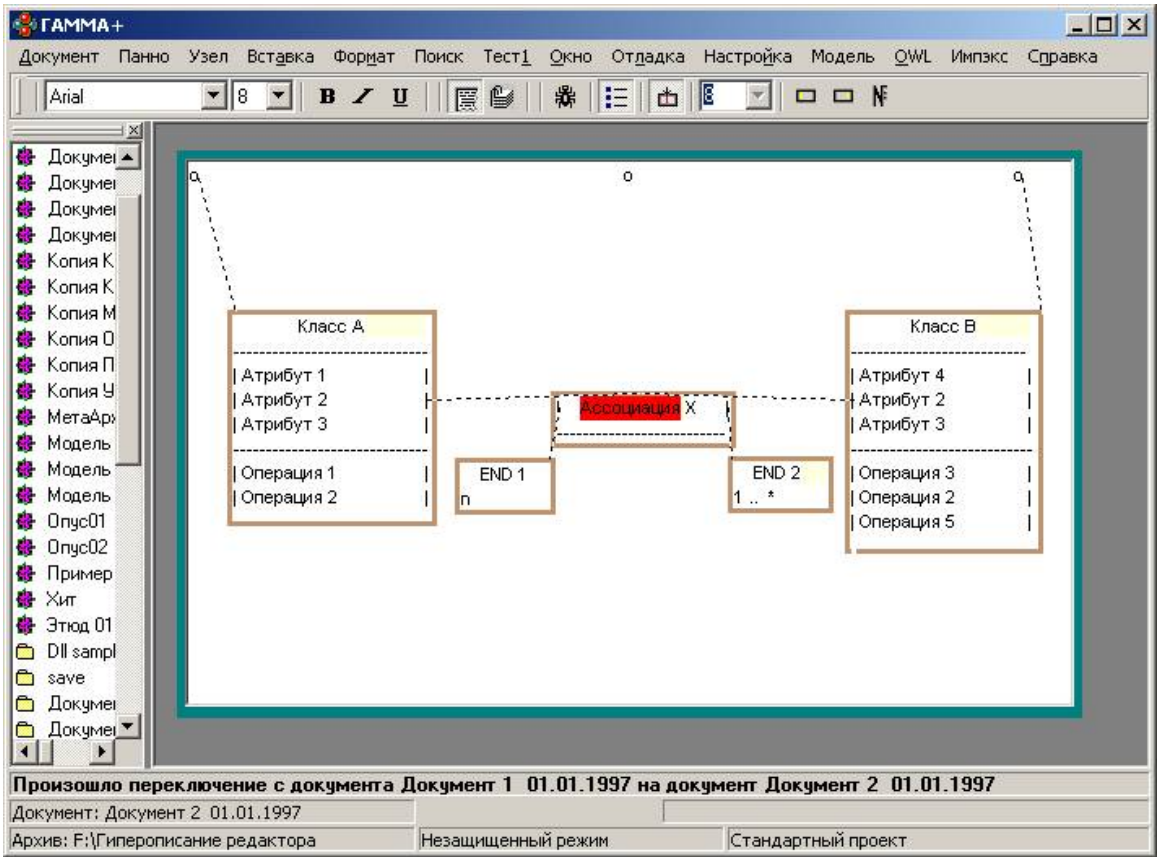

Рисунок 2. Пример воспроизведения в гипертексте диаграммы последовательностей

П-дескриптор содержит:

- ‘$’ в левом верхнем углу, как и любой дескриптор;
- на первой строке через один пробел слово “П-узел” или “П-node” – для опознавания типа сегмента узла;
- слово “Driver” или “Драйвер” во второй строке, используется для прикрепления текста программы –обработчика узла.

М-дескриптор прикрепляется к узлу, представляющему некоторый процесс трансформации узлов гипертекста. В частности, такой узел может представлять имитационную модель, объекты которой изображаются ансамблем программируемых узлов. Структура узла, к которому прикрепляется М-дескриптор в данных предложениях более детально не специфицируется.

Концептуально в гипертексте может быть задано несколько ансамблей программируемых узлов, но в рассматриваемой реализации такая возможность не обеспечена.

М-дескриптор содержит:

- ‘$’ в левом верхнем углу, как и любой дескриптор;
- на первой строке через один пробел слово “М-ансамбль” или “M-assembly” – для опознавания типа сегмента узла;
- слово “I-params” во второй строке;
- слово “O-params” в третьей строке;
- “Тип_среды” или “Environment” в четвертой строке;
- слово “Driver” или “Драйвер” в пятой строке;
- словосочетание “Тип_времени” или “Runtime_type” в шестой строке;
- словосочетание “ОТ ммм ДО” или “FROM v TILL” в седьмой строке.

Для использования существенно, чтобы оба дескриптора были в свернутой (многострочной) форме. Расположение слов в строке, количество пробелов между ними не существенно. Все рассмотренные слова и словосочетания должны быть слогами в смысле гипертекстового редактора.

Слова “I-params” и “O-params” предназначаются для привязки входных и выходных параметров моделирования. В данной реализации возможность их использования не обеспечена.

“Тип_среды” или “Environment” определяют тип среды, в которой выполняются драйвер ансамбля и обработчики узлов. Ссылка от этих слов дает узел, содержащий следующий список идентификаторов среды:

- WinPascal;
- WinC++;
- LinC++;
- LinMPI.

“Тип времени” или “Runtime type” определяют тип исполнения ансамбля. Ссылка от этих слов дает узел, содержащий следующий список типов исполнения:

- Increment;
- Real;
- Step.

Перед элементами этих списков как отдельный слог стоит литера ‘>’ или 'v'. Литера ‘v’указывает действующий элемент списка. В первом списке она может встречаться не более одного раза. Все рассмотренные элементы должны быть слогами в смысле гипертекстового редактора.

Если элемент “Increment” действует (перед ним стоит ‘v’), то ссылка от этого элемента содержит узел с числовой константой – приращением модельного времени.

Активность элемента “Real” означает, что ансамбль преобразуется в реальном времени вычислительной системы под управлением трассы событий.

Активность элемента “Step” означает, что ансамбль выполняет программы-обработчики узлов по шагам, на каждом шаге все обработчики выполняются по одному разу и обмениваются данными. Пошаговое выполнение может сочетаться с активностью элемента “Increment”.

Ссылка от слов “ОТ” или “FROM” указывает начальный момент исполнения ансамбля в модельном времени, а ссылка от слов ”ДО” или “TILL” – конечный момент.

Ссылка от слова “Driver” или “Драйвер” дает наименование программы-драйвера ансамбля. За ним может по ссылке следовать текст этой программы. Отсутствие ссылки означает, что (по умолчанию) будет действовать встроенный драйвер.

П-дескрипторы ансамбля связаны с его М-дескриптором ссылками от слова “П-узел” или “П-node” к М-дескриптору.

Обработчик узла должен быть поименован на латинице. Его имя указывается по ссылке от слова “Driver” или “Драйвер” в П-дескрипторе. Имя должно быть первым словом (слогом) в этой ссылке. За ссылкой через пробел должен следовать “#”, к которому по ссылке прикрепляется узел, содержащий текст обработчика на исходном языке программирования.

Если задана среда winPascal, то обработчик должен быть представлен на языке Pascal Delphi с соблюдением следующих правил:

- обработчик является телом процедуры без наружных begin и end (они подставляются автоматически при подготовке ансамбля к функционированию);
- заголовок процедуры не нужен, он формируется автоматически: в качестве имени используется указанное выше имя обработчика, а сигнатура принимается нулевой;
- локальные функции и процедуры не допускаются;
- объявления локальных переменных и констант не нужны: каждая переменная или константа объявляется ссылкой от любого вхождения её имени в тексте программы к специальному узлу гипертекста, описывающему переменную;
- указанный выше описатель в общем случае является трехзвенным, но второе звено может отсутствовать;
- первый узел звена содержит признак, характеризующий переменную как локальную (local), входную (input) и выходную (output);
- к этим словам прикрепляется по ссылке второй узел звена (или третий, если второй опущен);
- второй узел звена именует элемент данных на уровне гипертекста (это имя может не совпадать с именем переменной в тесте программы); за именем должен следовать “#”, к которому по ссылке прикрепляется третий узел звена;
- третий узел звена содержит ‘$’, тип значения переменной (в текущей реализации INTEGER, FLOAT, STRING, DATETIME)
- третий узел звена по ссылке от слога-типа значения переменной указывает её слог-резидент;
- в слоге резиденте значение должно быть первым слогом узла-резидента;

- ‘$’ третьего звена для входных параметров содержит указатель по ссылке на описатель роли выходного параметра (другого узла) – источника значения.

Описатели переменных могут быть определены только в контексте обработчика, но также могут использоваться в ином контексте: например, как элементы содержания операционной среды.

Кроме того, в состав собственных параметров П-узла могут входить указатели на текущее состояние и интегральные результаты функционирования ансамбля П-узлов.

Пример применения этих правил показан на рис.3. В этом примере моделируется взаимодействие двух процессов: поставщика и потребителя. Поставщик представлен П-узлом с драйвером PRODUCT, а потребитель – П-узлом с драйвером CONSUM. Объекты, которыми обмениваются процессы, представлены входными и выходными параметрами П-узлов. Все параметры являются числовыми величинами, тип которых показан в соответствующих узлах гипертекста. Переменные в текстах драйверов ссылаются на параметры. Эти связи на рис.3 не показаны, чтобы избыточно не усложнять представление, но они могут быть прослежены нажатием мышью на имена переменных.

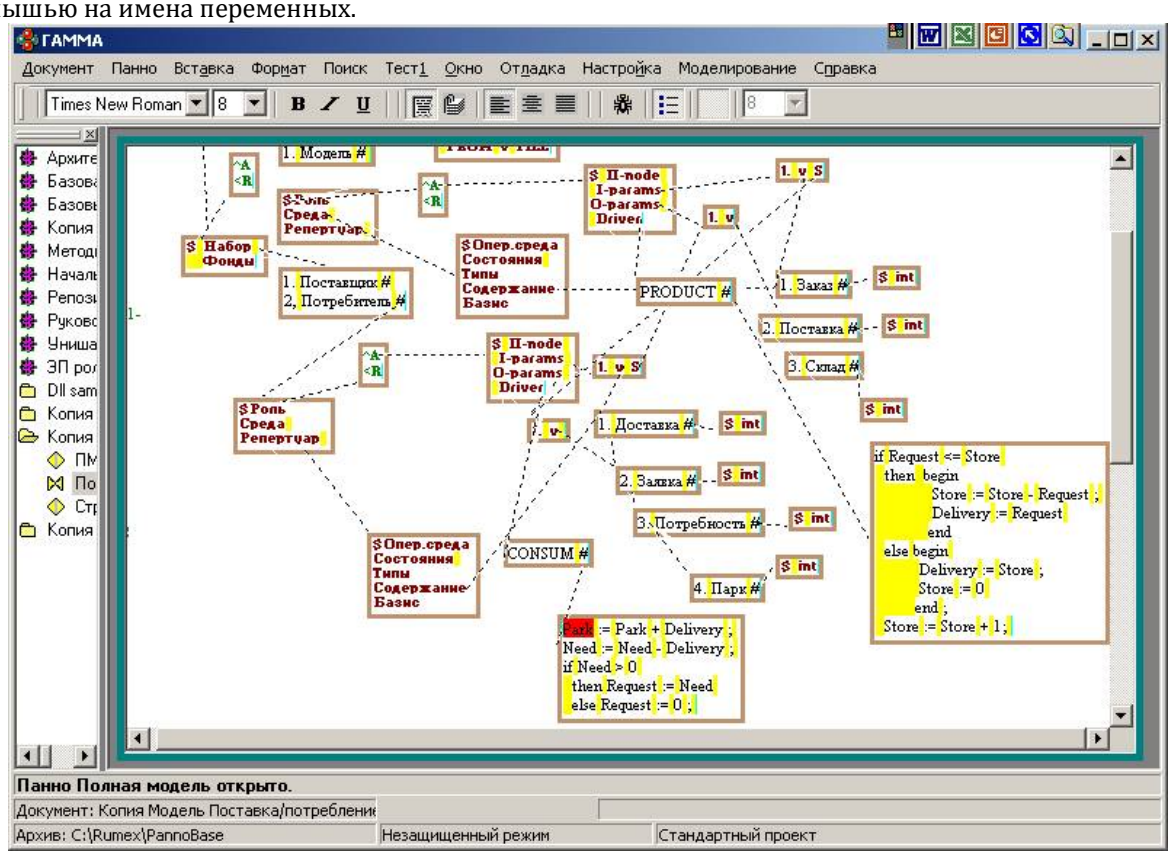

Рисунок 3. Пример в гипертексте интеграции программы и её проекта

Для хранения моделей вне базы данных, которая обеспечивает работу гипертекстового редактора, каждый узел гипертекста может быть представлен элементом XML-документа. Все эти элементы находятся на одном, верхнем, уровне документа. Каждый слог также представляется элементом нижнего уровня документа. Ссылка указывается атрибутом элемента слога. Значением атрибута является некий уникальный признак элемента-узла. В качестве такого уникального признака может выступать номер “ссылки” (узла), некоторый пересчет узлов (вне реализации гипертекстового документа). Этот номер виден в XML-документе также в качестве атрибута элемента-узла.

Пример:

```
<?xml version = “1.1”?>
<RHytxt xmlns:Fhn=’http:/www … ‘>
  <Rhynode>
    <Rhsyl>  …. </Rhsyl>
    <Rhsyl>  …. </Rhsyl>
  </Rhynode>
  <Rhynode  Rn=1>
    <Rhsyl>  …. </Rhsyl>
```

```
        ….
    </Rhynode>
    <Rhynode>
      <Rhsyl>  …. </Rhsyl>
           ….
      <Rhsyl  call=1>  …. </Rhsyl>
    </Rhynode>
  </RHytxt>
```

Многострочные ссылки представляются дополнительным уровнем элементов, например:

```
  <RHynode>
    <Rhline>
      <Rhsyl>  …. </Rhsyl>
           …
      <Rhsyl>  …. </Rhsyl>
    </Rhline>
  </RHynode>
```

В заключение следует отметить, что полная реализация графических особенностей диаграмм активностей и состояний UML требует введения в гипертекстовый редактор соответствующих специализированных графем, но эти изменения локальны и не затрагивают редактор в целом. Точное графическое воспроизведение диаграмм последовательностей подводит к специальной интерпретации межузлового пространства, что противоречит основным принципам построения гипертекста. Представляется более целесообразным реализовать диаграмму последовательностей как частный случай протокола изменения состояний объектов класса по мере совершения актов их взаимодействия.

Рассмотренные усовершенствованные подходы к проектированию и конструированию программ могут оказаться полезными для обработки больших данных, требующих разнообразных и сложных программных инструментов.

Литература

1. OMG Unified Modelling Language (OMG UML). Version 2.5. URL: http://www.omg.org/spec/UML/2.5/PDF
2. StarUML is a UML tool by MKLab. URL: http://staruml.io/
3. IBM® Rational® Software Architect Designer (RSAD and formerly RSA) is a comprehensive design, modeling and development tool for end-to-end software delivery URL: http://www-03.ibm.com/software/products/en/ratsadesigner
4. Object Constraint Language. Version 2.4 URL http://www.omg.org/spec/OCL/2.4/PDF/
5. А.П.Гагарин. Управление формированием знаний в процессе разработки компьютерных программ//Труды международной конференции "Идентификация систем и задачи управления", М: Изд-во Института проблем управления им. В.А. Трапезникова, 2005.

References

1. OMG Unified Modelling Language (OMG UML). Version 2.5. URL: http://www.omg.org/spec/UML/2.5/PDF
2. StarUML is a UML tool by MKLab. URL: http://staruml.io/
3. IBM® Rational® Software Architect Designer (RSAD and formerly RSA) is a comprehensive design, modeling and development tool for end-to-end software delivery URL: http://www-03.ibm.com/software/products/en/ratsadesigner
4. Object Constraint Language. Version 2.4 URL http://www.omg.org/spec/OCL/2.4/PDF/
5. A.P.Gagarin. Upravlenie formirovaniem znaniy v protsesse razrabotki komp'yuternykh programm//Trudy mezhdunarodnoy konferentsii "Identifikatsiya sistem i zadachi upravleniya", M: Izd-vo Instituta problem upravleniya im. V.A. Trapeznikova, 2005.

Поступила: 15.09.2016

Об авторах:

Гагарин Андрей Петрович, кандидат технических наук, профессор Московского авиационного института (национального исследовательского университета), gagarin_ay@outlook.com**.**