МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ВОЛЖСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА «ИНФОРМАТИКА И ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ»

О. Ф. Абрамова, Д. Н. Лясин

ВВЕДЕНИЕ В ПРОГРАММНУЮ ИНЖЕНЕРИЮ:

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

на тему:

«Основные сведения о UML и BOUML.

Диаграммы вариантов использования»

Методические указания

Волгоград

2013

УДК

Р е ц е н з е н т :

Доцент кафедры ВАЭ, к.т.н., Капля Виктор Иванович

Издается по решению редакционно-издательского совета

Волгоградского государственного технического университета

Абрамова О. Ф., Д. Н. Лясин **Введение в программную инженерию: методические указания к лабораторной работе на тему «Основные сведения о UML и BOUML. Диаграммы вариантов использования [**Электронный ресурс**]**: методические указания / О. Ф. Абрамова// Систем.требования:Windows 95 и выше; ПК с процессором 486+; CD-ROM.

Рассмотрены основные принципы визуального моделирования, основные правила и элементы языка UML; принципы построения диаграмм use case. Дан общий обзор case-средства BOUML 4.0, описание его основных элементов, команд и действий. Приведен пример выполнения лабораторной работы по дисциплине «Введение в программную инженерию». Приведены варианты заданий к лабораторной работе.

Предназначены для студентов бакалавриата, обучающихся по направлению подготовки 231000.62 «Программная инженерия». CD-ROM

Цель работы

Знакомство с UML, основными правилами и элементами языка. Изучение правил и основных подходов формирования диаграммы прецедентов. Знакомство с основными возможностями и инструментами case-средства BOUML. Анализ предметной области, написание сценариев вариантов использования с помощью case-средства BOUML.

Введение

Визуальное моделирование в настоящее время используется повсеместно при проектировании различных программных систем. Поэтому знание основ построения различных моделей и компьютерных средств, используемых для их визуализации, является обязательным для современного специалиста в области разработки программных систем.

Методика выполнения лабораторной работы

Для успешного выполнения лабораторной работы необходимо внимательно изучить теоретическую информацию и ответить на контрольные вопросы. Затем, для задания, вариант которого будет выдан преподавателем на занятии, необходимо:

1) выделить основные варианты использования проектируемой системы;
2) создать главную диаграмму прецедентов, задав на ней все варианты использования и актеров;
3) создать диаграммы прецедентов для каждого варианта использования;
4) для каждого варианта использования описать поток событий в виде отдельного файла и прикрепить его к варианту использования
5) выполнить описание каждого варианта использования и составить глоссарий проекта.

Варианты заданий

Для каждого из нижеприведенных вариантов необходимо разработать модель системы, поддерживающей указанные в варианте сценарии использования. Самостоятельно выполнить составление спецификации требований к системе, выбор действующих лиц и описание вариантов использования. С помощью case-средства BOUML сформировать диаграмму прецедентов для одного из описанных сценариев. Сформировать диаграмму use case для всей системы в целом (смоделировать работу всей систему с указанием всех возможных действующих лиц, прецедентов и связей между ними).

Результатом выполнения лабораторной работы должен стать отчет (в печатном и электронном вариантах), состоящий из следующих пунктов:

1) спецификацию требований к ПО;
2) глоссарий проекта;
3) список действующих лиц (актеров) проекта;
4) список вариантов использования со спецификациями (описаниями);
5) диаграммы вариантов использования (BOUML).

Варианты:

1. **Интернет-магазин.** Смоделировать ПС, в которой должны быть реализованы сценарии: покупка товара, поиск товара, добавление нового товара в базу данных магазина, просмотр и обработка заказов покупателей, регистрация нового покупателя.
2. **Книжный каталог.** В системе необходимо реализовать следующие сценарии: добавление новой книги, поиск книги по нескольким полям, бронирование книги, списание старых книг, регистрация пользователей каталога. Доступ к системе могут иметь как читатель, так и администратор, но возможности их четко разграничены. Читатель может выполнить только поиск книги и бронирование, а администратор выполняет все действия с каталогом книг (списание, подтверждение бронирования и т.д.).
3. **Адресная книга.** Смоделировать ПС, в которой должны быть реализованы сценарии: добавление нового абонента, добавление категорий абонентов, поиск абонентов по нескольким полям, добавления администраторе каталога (пользователей, которые имеют право редактировать данные адресной книги), редактирование данных абонента.
4. **Расписание занятий.** Система должна поддерживать выполнение следующих вариантов использования: добавление новой группы, добавление занятий (с указанием названия предмета, времени, аудитории, группы, недели, преподавателя, типа занятия), просмотр списка занятий на выбранную дату, добавление списка преподавателей, поиск занятий по нескольким полям (предмету, преподавателю, группе, времени, типу занятия).
5. **База студентов.** Смоделировать ПС, в которой должны быть реализованы сценарии: добавление новой группы, добавление нового студента, поиск студента по различным полям, добавления информации об оценках по различным предметам, отчисление студента.
6. **Прайс-лист фирмы.** Смоделировать ПС, в которой должны быть реализованы сценарии: добавление новой категории товаров, добавление нового товара, поиск товара по различным полям, добавление администратора прайс-листа (пользователей, которые имеют право редактировать прайс-лист), перемещение товара из одной категории в другую.

7. **База склада фирмы.** Смоделировать ПС, в которой должны быть реализованы сценарии: добавление нового товара на склад, списание товара, выдача товара, поиск товара по различным полям, изменение месторасположения товара на складе.

8. **Аптечная база.** Смоделировать ПС, в которой должны быть реализованы сценарии: прием заказа от клиента на изготовление раствора, продажа лекарства, списание просроченных лекарств, добавление новые лекарств в базу данных, поиск заказов по различным полям.

9. **Система автоматизации для пункта проката видеокассет.** Смоделировать ПС, в которой должны быть реализованы сценарии: формирование каталога видеокассет, имеющихся в наличии (добавление новых данных о видеокассетах, удаление данных о кассетах, редактирование данных) – для администратора системы. Так же система должна рассчитывать стоимость проката (основываясь на стоимости за сутки и сроке проката). Для клиента возможны варианты использования: выбрать кассету по каталогу, внести залог.

10. **Каталог Интернет- ресурсов.** Каталог должен содержать следующие данные о ресурсах: название, уникальный локатор (URL), краткое описание, контактную информацию, дату последнего обновления, список ключевых слов. Данные о ресурсах должны быть классифицированы по разделам. Пользователи каталога могут добавлять информацию о новых ресурсах, осуществлять поиск ресурса по ключевым словам или списку ресурсов из определенного раздела, сортировать список ресурсов по нескольким полям. Система должна следить за обновлениями, периодически опрашивая сайты, URL которых записаны в каталоге.

Содержание отчета

Результатом выполнения лабораторной работы должен стать отчет (в печатном и электронном вариантах), состоящий из следующих пунктов:

1) спецификация требований к ПО;
2) глоссарий проекта;
3) список действующих лиц (актеров) проекта;
4) список вариантов использования со спецификациями (описаниями);
5) диаграммы вариантов использования.

Отчет лабораторной работы должен быть произведен студентом преподавателю в срок до начала следующего лабораторного занятия. Во время занятия преподавателю необходимо предоставить:

- отчет о лабораторной работе (в печатном виде);
- электронную версию отчета.

Оценка (в баллах) выставляется за **устный отчет** студента по предоставленным преподавателю материалам.

Рекомендации по выполнению лабораторной работы

Визуальное моделирование

Современный подход к разработке сложных программных систем основывается на использовании моделей этих систем. Это позволяет спланировать, рассмотреть все связи, возможные ситуации и сбои в работе, а также достоинства и недостатки системы еще до ее создания. Такой подход дает возможность сэкономить серьезные средства, что важно для заказчика. И в то же время, что еще более важно, дает и разработчику, и заказчику уверенность в том, что такая система будет выполнена с учетом всех требований, и кардинальных изменений на последнем этапе отладки и тестирования - не потребует.

Процесс, включающий:

– предварительный анализ и изучение всех заявленных заказчиком требований к системе;

– преобразование требований в понятную для команды разработчиков форму;

– построение предварительного каркаса будущей программной системы с использованием какой-либо определенной техники (словесного описания, визуального моделирования с помощью программных средств или отрисовки общей блок-схемы системы вручную)

называется ***моделированием***.

Естественно, создание модели будущей программной системы в графическом виде (блок-схема, диаграмма) с помощью современных case-средств, так называемое *визуальное* (графическое) *моделирование* системы, это наиболее современный подход к созданию моделей как сложных программных систем, так и простых программных продуктов. Такое моделирование подразумевает использование некоторого набора стандартных графических элементов, количество которых ограниченно, а смысл – однозначен и понятен даже неспециалисту. Это позволит реализовать одно из главных преимуществ визуального моделирования – коммуникацию между всеми заинтересованными в данном проекте сторонами: пользователями, заказчиками, разработчиками, аналитиками и т.п.

Человек – это зрительно-ориентированное существо, поэтому сложную и новую информацию ему гораздо проще усвоить в том случае, если она представлена в виде схем, рисунков, диаграмм, а не в виде объемного текстового описания. Визуальная модель, состоящая из сравнительно не-

большого количества стандартных элементов и таких же стандартизированных связей между ними, может быть достаточно легко прочитана любым из участников проекта:

- пользователи могут определить варианты своего взаимодействия с системой,
- разработчики четко увидят каркас системы и ее составные части (не забывайте, что разработчиков сложной программной системы может быть от 1 до 100 и более человек),
- аналитикам такая модель предоставить ясную картину связей между объектами системы и т.д.

Введение в унифицированный процесс моделирования

Унифицированный язык моделирования (UML - Unified Modeling Language) в настоящее время можно рассматривать как стандартный инструмент для создания графического описания (диаграмм) разрабатываемого программного обеспечения. Используя средства UML можно выполнять визуализацию процесса разработки различных программных систем, а так же конструировать, специфицировать и документировать этот процесс. UML с одинаково успешным результатом можно применять как для разработки простых информационных систем, например, информационных систем масштаба предприятия, так и достаточно сложных, таких как распределенные Web-приложений и даже встроенные системы реального времени. Этот язык достаточно выразителен, для того, чтобы позволить рассмотреть разрабатываемую систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию, но, с другой стороны, достаточно прост для понимания, изучения и использования.

Моделирование, т.е. предварительное планирование, особенно в графическом виде, чрезвычайно важно как для понимания системы в целом, так и для усвоения ее работы при решении отдельных, конкретных задач. Поэтому обычно наличия единственной модели, разработанной на самом раннем этапе анализа, никогда не бывает достаточно. Наоборот, для понимания, как разработчиками, так и заказчиками практически любой программной системы приходится разрабатывать большое количество взаимосвязанных моделей. И на этом этапе возникает множество проблем. Потому как характерной особенностью мышления большинства программистов является то, что размышления о том, как реализовать проект, для них практически всегда подразумевают банальное создание программного кода для этого проекта. Конечно, не будем спорить, некоторые вещи проще и лучше выразить именно в коде на каком-нибудь языке программирования, но так происходит далеко не всегда. Такой подход, когда разработчик программной системы пытается сразу написать программу, не представив предварительно графически свои мысли по поводу ее работы и вариантов использования, чреват рядом неприятностей:

- во-первых, вести диалог по обсуждению разработанной модели можно только в том случае, когда участники понимают друг друга, т.е. говорят на одном языке и понимают смысл каждого слова (или элемента диаграммы, как в нашем случае);

- во-вторых, наличие модели системы просто необходимо в том случае, когда система выходит за рамки текстового языка программирования, и проиллюстрировать связи и взаимодействия ее компонентов, пользуясь только программным кодом, очень и очень затруднительно;

- в-третьих, если все-таки разработчик (системный аналитик) разрабатывал модели системы, но делал это от руки, «на коленке», как говорится, а не официально, документируя все построенные диаграммы с помощью case-средств, то будет очень сложно восстановить эти модели в случае неожиданного отсутствия доступа к этому человеку (например, при его уходе на длительный больничный или, еще того хуже, к конкурентам).

Использование UML позволяет решить эти проблемы. Этот язык моделирования - не просто набор графических символов, за каждым этих символов стоит определенная семантика, смысл, которые подразумевают, что модель, написанная одним разработчиком, может быть однозначно интерпретирована другим. Причем на месте второго разработчика может выступать не только человек, но и некоторое инструментальное средство. Это решение первой проблемы.

Некоторые особенности системы лучше всего моделировать в виде текста, другие - графически. Практика свидетельствует, что во всех интересных системах существуют структуры, которые очень сложно, а иногда и попросту невозможно представить с помощью одного языка программирования. А UML – это графический язык, что позволяет решить нашу вторую проблему.

Ну и, наконец, явная графическая модель, состоящая из сравнительно небольшого количества унифицированных элементов, намного облегчает общение как между отдельными разработчиками, так и между разработчиком и заказчиком, что позволяет решить и третью, озвученную выше, проблему.

Надеюсь, мы вас убедили в важности предварительного моделирования программных систем. А теперь подробнее рассмотрим, как это просто сделать с помощью case-средств.

Основные конструкции UML

Словарь UML включает ***три вида основных конструкций*** (рис.1):

- **сущности** - абстракции, являющиеся основными элементами модели;
- **отношения** - связи между сущностями;
- **диаграммы**, группирующие представляющие интерес множества сущностей и отношений.

Сущности UML

В качестве основных объектно-ориентированных элементов языка UML выступают так называемые сущности, с помощью которых можно создавать корректные модели.

В UML поддерживает сущности четырех типов:

- структурные;
- поведенческие;
- группирующие;
- аннотационные.

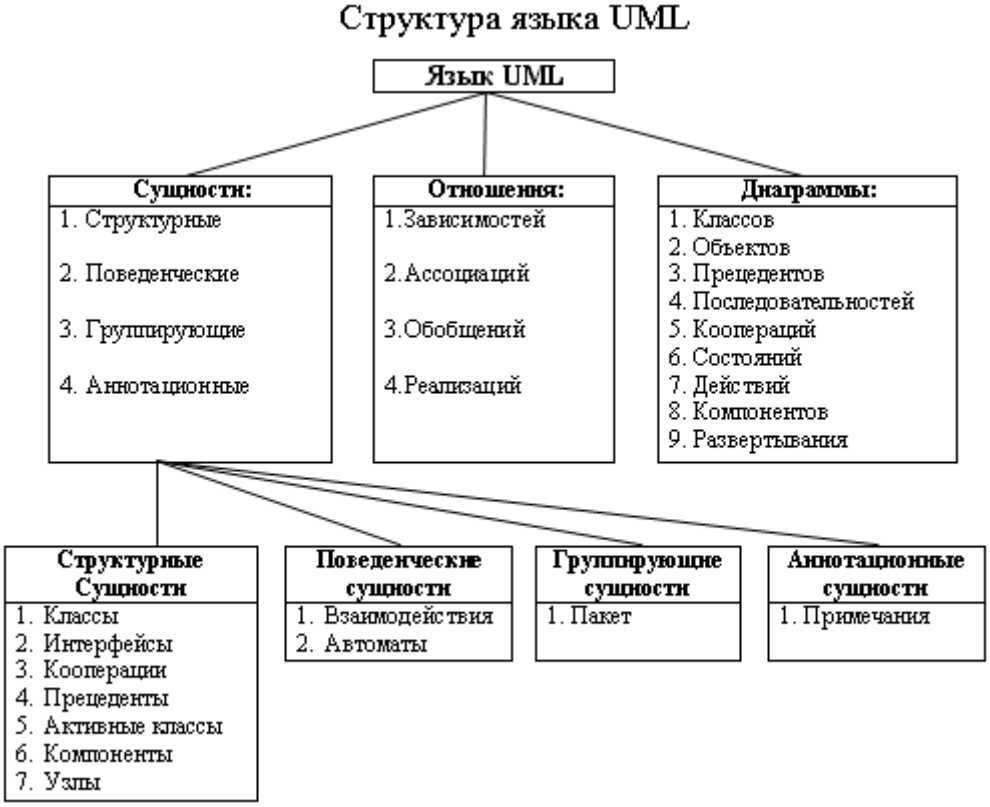


Рис. 1 Основные конструкции UML

Структурные сущности - *это имена существительные в моделях на языке UML.* Такие элементы, как правило, представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. В UML реализовано достаточно большое количество разновидностей структурных сущностей. В качестве базовых элементов

выделяют следующие структурные сущности: класс, активный класс, интерфейс, кооперация, прецедент, компонент, узел. Рассмотрим их более подробно.

Класс (class) - описание общих атрибутов, операций, отношений и семантики некоторой совокупности объектов. На диаграмме класс визуализируется в виде прямоугольника, разделенного на три секции. В этих секциях указываются:

- его имя,
- атрибуты, т.е. общие свойства этого класса,
- операции, т.е. действия, которые могут выполнять объекты, принадлежащие данному классу (рис. 2).

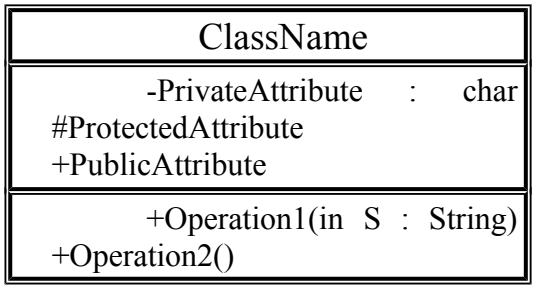


Рис. 2 Пиктограмма класса

Интерфейс (interface) - некая совокупность операций, определяющих конкретную службу (например, сервис, набор услуг), которые предоставляет класс или компонент. Как правило, интерфейс присоединяется к реализующему его классу или компоненту. Графически интерфейс изображается в виде круга, под которым указывают его имя (рис. 3).

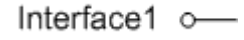


Рис. 3 Пиктограмма интерфейса

Кооперация (collaboration) – используется для определения взаимодействия ролей и других элементов, которые, работая вместе, производят некий кооперативный эффект, но не сводящийся к обычной сумме слагаемых. На диаграмме кооперация изображается в виде эллипса, который ограничивается пунктиром, внутри которого указывается только имя (рис. 4).

Рис. 4 Пиктограмма кооперации

Прецедент (use case) - описание последовательности действий, вы-

полняемых системой, и приводящих к значимому для определенного актера, наблюдаемому результату. На диаграмме прецедент так же визуализируется в виде эллипса, но ограниченного непрерывной линией, с указанием его имени внутри (рис. 5).

Рис. 5 Пиктограмма прецедента

Активным классом (active class) называется класс, объекты которого могут инициировать управляющее воздействие, т.к. вовлечены в один или несколько процессов, или нитей (threads). Графически активный класс представляется также как и простой класс, включает имя, атрибуты и операции, но ограничивается прямоугольником, отрисованным жирной линией (рис. 5).

ClassName
-PrivateAttribute : char #ProtectedAttribute +PublicAttribute
+Operation1(in S : String) +Operation2()

Рис. 6 Пиктограмма активного класса

Компонент (component) – представляет собой физическую заменяемую часть системы, соответствующую некоторому набору интерфейсов и обеспечивающую его реализацию. На диаграмме компонент отрисовывается в виде прямоугольника с вкладками, внутри которого указывается его имя (рис. 7).

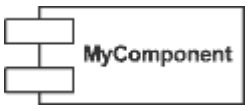


Рис. 7 Пиктограмма компонента

Узел (node) - это также элемент реальной (физической) системы, он существует во время функционирования программного продукта и представляет собой, как правило, некий вычислительный ресурс, обладающий, например, некоторым объемом памяти, а часто еще и возможностью обработки. Графически узел представляется в виде куба, внутри которого указывается его имя (рис. 8).

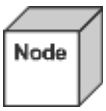


Рис. 8 Пиктограмма узла

К структурным сущностям можно также отнести и *актеров(actor)* – тех, кто взаимодействует с данной системой. Это может быть человек, а может быть и другая система, но расположенная извне, относительно данной. Актеры на диаграммах обозначаются пиктограммой в виде человечка (рис. 9).

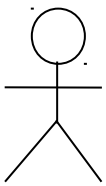


Рис. 9 Пиктограмма актера

Применяют также и другие разновидности сущностей:

- актеры, сигналы, утилиты (виды классов),
- процессы и нити (виды активных классов),
- приложения, документы, файлы, библиотеки, страницы и таблицы (виды компонентов).

К поведенческим сущностям (behavioral things) языка UML, являющимися динамическими составляющими (иначе говоря, глаголами) модели UML, и описывающими поведение модели в пространстве и во времени, относят два основных типа сущностей:

Взаимодействие (interaction) – т.е. обмен сообщениями (messages) между объектами в рамках некоторого контекста для достижения четко определенной цели. Поэтому взаимодействие предполагает наличие других элементов: сообщений, последовательности действий (описывающих поведение, инициированное сообщениями), связей (между объектами). На диаграмме сообщение отображается в виде стрелки, над которой в большинстве случаев указывается имя соответствующей операции (рис. 10).

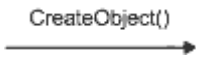


Рис. 10 Пиктограмма сообщения

Автомат (state machine) – некий алгоритм поведения объекта (последовательность его состояний) в ответ на определенные события, а также реакции на эти события. С помощью автоматов описываются поведение отдельного класса или кооперации классов. Графически состояние отображается в виде прямоугольника с закругленными углами (рис. 11), содержащего имя и, в некоторых случаях, промежуточные состояния.

Рис. 11 Пиктограмма состояния

Группирующие сущности - это организующие части модели UML, т.е. блоки, на которые эту модель можно разложить. К такому типу сущностей относят *пакеты* (packages) - универсальные механизмы объединения элементов модели в группы. В такие пакеты можно поместить сущности любого типа, но, в отличие от компонентов, реально существующих во время работы программы, пакеты существуют только в процессе разработки, т.е. носят чисто концептуальный характер. Для отображения пакета на диаграмме используется пиктограмма папки с закладкой, содержащей имя пакета, а иногда - и его содержимое (рис. 12).

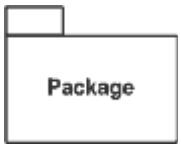


Рис. 12 Пиктограмма пакета

Аннотационные сущности – пояснительные, разъясняющие части модели UML, которые можно применить к любому элементу модели. В качестве базового типа аннотационных элементов выступает *примечание* (note) – символ, используемый для изображения комментариев или ограничений, присоединяемый к элементу или группе элементов. На диаграмме примечание изображается в виде прямоугольника с загнутым краем, внутрь которого помещается текстовый или графический комментарий (рис. 13).

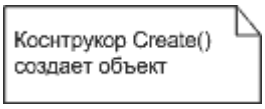


Рис. 13 Пиктограмма примечания

Отношения UML

В унифицированном языке моделирования определены четыре основных типа отношений, которые являются основными связующими конструкциями и применяются для построения корректных моделей. К этим типам относят: зависимость, ассоциацию, обобщение и реализацию.

Рассмотрим типы отношений подробнее.

Зависимость (dependency) представляет собой семантическое отношение между двумя сущностями, одна из которых является зависимой от другой. при использовании такого типа отношения изменение независимой сущности может повлиять на семантику зависимой. Для графического

отображения зависимости применяют пунктирную линию со стрелкой (рис. 14); возможно также наличие метки.

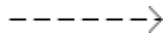


Рис. 14 Пиктограмма зависимости

Ассоциация (association) представляет собой структурное отношение, описывающее совокупность смысловых связей между объектами. Разновидностью ассоциации является агрегирование (aggregation) - отношение между целым и его частями. Графически ассоциация изображается в виде сплошной линии (иногда завершающейся стрелкой или содержащей метку), рядом с которой могут присутствовать дополнительные обозначения, например кратность и имена ролей (рис. 15).

Рис. 15 Пиктограмма ассоциации

Обобщение (generalization) описывает взаимодействие объектов-потомков (child) и объектов-родителей (parent), возможность их взаимозаменяемости. При этом, как и положено в объектно-ориентированном программировании, потомок наследует структуру и поведение своего предка. Графически отношение обобщения представляется в виде линии с незакрашенной стрелкой, указывающей на предка (рис.16).

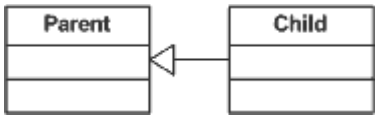


Рис. 16 Пиктограмма обобщения

Реализация (realization) представляет собой семантическое отношение между двумя классификаторами, при этом один классификатор определяет некоторое обязательство, а другой гарантирует выполнение этого обязательства. Такой тип отношений может применяться в двух случаях:

- между интерфейсами и реализующими их классами или компонентами,
- между прецедентами и реализующими их кооперациями.

На диаграмме реализация отображается пунктирной линией с незакрашенной стрелкой (рис. 17).

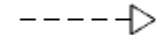


Рис. 17 Пиктограмма реализации

Диаграммы UML

Диаграмма в UML - это графическое представление набора элементов, представляющее собой некий связанный граф с вершинами (сущностями) и ребрами (отношениями). Основная цель составления диаграмм – это визуализация разрабатываемой системы с разных точек зрения и в разных обстоятельствах.

Теоретически диаграммы программных систем могут содержать любые комбинации сущностей, однако на практике применяется сравнительно небольшое количество типовых комбинаций, соответствующих одному из пяти наиболее необходимых видов архитектуры программной системы. Поэтому в UML определены **девять типов диаграмм**:

1) диаграмма вариантов использования или диаграмма прецедентов (*use case diagram*);
2) диаграмма объектов (*object diagram*);
3) диаграммы взаимодействия:
    a) диаграмма последовательностей (*sequence diagram*);
    b) диаграмма кооперации (*collaboration diagram*);
4) диаграмма классов (*class diagram*);
5) диаграмма состояний (*statechart diagram*);
6) диаграмма деятельности (*activity diagram*);
7) диаграмма компонентов (*component diagram*);
8) диаграмма развертывания (*deployment diagram*).

При моделировании программных систем важно представить их с различных точек зрения, иначе говоря, выполнять конструирование сразу в нескольких измерениях, используя весь предлагаемый case-средствами арсенал средств. Иначе вы рискуете не заметить или забыть подробно рассмотреть какие-то важные вопросы, что вполне возможно подставить под угрозу всю вашу работу.

В течение лабораторных работ мы с вами познакомимся с основными видами диаграмм, что позволит нам рассмотреть проектируемую систему с различных точек зрения и позволит вам разобраться в разных подходах к отображению модели системы. А также научиться выбирать необходимый вид диаграмм для отображения того или иного варианта деятельности системы, для контроля за разработкой системы, для анализа результатов деятельности системы.

Моделирование системы или подсистемы осуществляется следующим образом:

- Идентифицируйте основные функциональные составляющие системы, которые можно разрабатывать, выпускать и развертывать до не-

которой степени независимо. На результаты этого разбиения системы часто влияют технические, юридические и организационные факторы;

- Для каждой подсистемы специфицируйте ее контекст, так же как это делается для системы в целом при этом число актеров, окружающих систему включаются все соседние подсистемы, поэтому необходимо проектировать их совместную работу;
- Смоделируйте архитектуру каждой подсистемы так же, как это делается для всей системы.

Подробнее каждый из типов диаграмм мы рассмотрим чуть позже, а пока остановимся на самой первой из них – диаграмме вариантов использования - *use case diagram.*

Диаграммы вариантов использования

Впервые модели use case были применены в процессе разработки программного обеспечения А. Якобсоном в 1992г. Изначально такие модели были разработаны для проектирования объектно-ориентированного ПО, но успех их оказался столь значительным, что впоследствии произошла интеграция таких моделей во все основные методы и подходы проектирования программных средств.

Основная идея создания диаграммы вариантов использования состоит в детализированном описании некоторой задачи, достаточно упрощенном, завершенном и не зависящем от технологии реализации, выраженном на языке данной прикладной области и пользователей системы. Каждый элемент use case в повествовательной форме описывает, в основном, определенное взаимодействие рассматриваемой системы и пользователя, обеспечивая функциональность проектируемой системы и понятное пользователю описание применения этой системы.

Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом).

Действующее лицо (Actor) – представляет собой *роль* (а не конкретных людей), которую пользователь играет по отношению к разрабатываемой программной системе. В некоторых случаях в качестве действующего лица может выступать даже некая внешняя система, которая вступает во взаимосвязь с моделируемой системой на предмет получения информации.

Так как варианты использования создаются разработчиками совместно с заказчиком, индивидуально для каждого случая, то стиль написания во многом зависит от сложности проекта, количества и профессионального уровня участников. Но можно рекомендовать несколько общих правил:

- наименования разрабатываемых вариантов использования должны быть понятны заказчику, т.е. должны содержать минимальное количество специализированных, технических терминов;

- каждый вариант использования должен описывать завершенную ситуацию, причем эта ситуация должна иметь определенную ценность для пользователя системы;
- вариант использования должен быть прост, легко читаем и доступен в изучении как для разработчиков, так и для заказчика.

Например, на рисунке 18 приведена основная диаграмма прецедентов для виртуального книжного магазина. По ней легко можно отследить внешних сущностей по отношению к программной системе – актеров (администратор, покупатель, пользователь), и их возможные действия в данной системе.

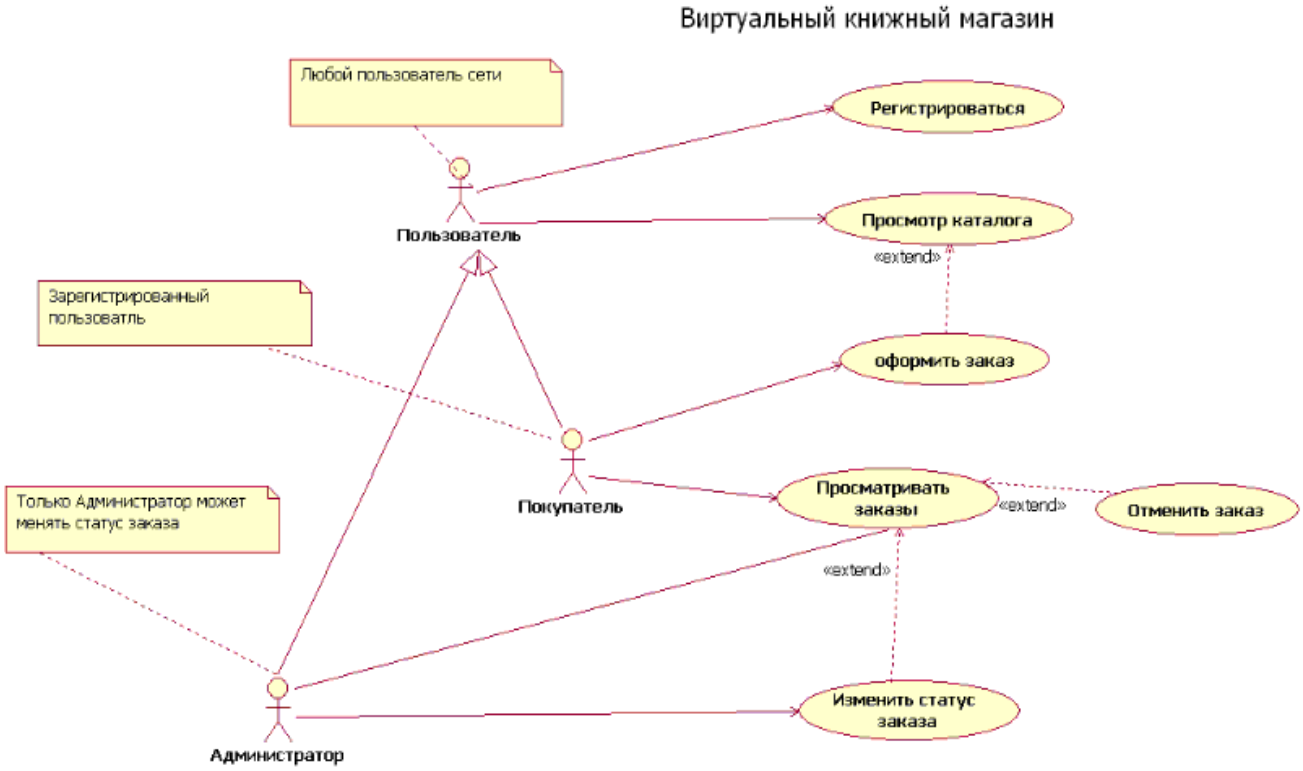


Рис. 18 диаграмма прецедентов виртуального книжного магазина

Формат описания варианта использования (по Коберну):

1. Имя – цель в виде краткой активной глагольной фразы.
2. Контекст использования – более подробное описание цели.
3. Область действия.
4. Уровень точности.
5. Основное действующее лицо.
6. Другие участники и их интересы.
7. Предусловие (определяет, выполнение какого условия гарантирует система перед тем, как разрешить запуск варианта использования).

8. Минимальные гарантии (наименьшие обещания системы участникам, в частности, когда цель основного действующего лица не может быть достигнута).

9. Гарантии успеха (или постусловие – postcondition – устанавливает, что интересы участников удовлетворяются по успешном завершении варианта использования в конце основного сценария).

10. Триггер (событие, которое запускает вариант использования).

11. Основной сценарий или поток (простой для понимания типичный сценарий, в котором достигается цель основного действующего лица и удовлетворяются интересы всех участников). Каждый шаг основного сценария описывает:

- взаимодействие двух действующих лиц ("Клиент вводит адрес");
- шаг подтверждения для защиты интереса участника ("Система подтверждает PIN-код");
- внутреннее изменение для удовлетворения интереса участника ("Система выводит сумму из баланса").

12. Расширения (запускаются при возникновении определенного условия, содержат последовательность шагов, описывающих, что происходит при этом условии, и заканчивается достижением цели или отказом от неё).

13. Список изменений в технологии и данных.

14. Вспомогательная информация.

Сценарии использования

Прежде чем переходить к программному составлению диаграмм с помощью любого доступного case-средства, необходимо определиться с основными элементами (актерами и прецедентами) и, образно говоря, сюжетом. Для успешного проектирования программного средства необходимо ясно и четко представлять себе все возможные ситуации, которые могут возникнуть в процессе его работы. Описание этих ситуаций, реалистичное, детализированное и достаточно правдоподобное и называют *сценарием* или вариантом использования.

Каждый такой вариант использования должен быть сначала в общих чертах обдуман разработчиком, а потом обязательно реализован графически. Графическая визуализация сценариев выполняется в виде диаграмм вариантов использования (use-case диаграмм).

С другой стороны, такие модели предполагают совместную работу разработчика и заказчика по постановке задачи и обычно обладают большим количеством характеристик. Для достижения высокой реалистичности возможно (и, зачастую, необходимо) некоторое количество возвратов для уточнения.

При разработке пользовательского интерфейса сценарии описывают взаимодействие между проектируемой системой и пользователем (или типом пользователей). При этом обыкновенные сценарии при попытке использовать их для проектирования пользовательского интерфейса обладают некоторыми серьезными ограничениями. В них делается основной упор на реалистичность и детали, а вот на серьезные проблемы и общую организацию обращается внимания не достаточно.

Поэтому очень важно еще на этапе анализа будущей программной системы четко представить и согласовать с заказчиком максимальное количество вариантов использования этой системы, а так же продумать их состав и последовательность действий. Это необходимо для того, чтобы свести к минимуму количество возвратов к началу уже на этапе построения диаграмм, а то и на этапе представления отчетности заказчику.

BOUML – просто и надежно

Основные сведения о BOUML

BOUML – это CASE-средство, предназначенное для автоматизации этапов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и выпуска проектной документации. Проще говоря, это UML инструмент, позволяющий строить различные диаграммы и затем по ним определять и генерировать код на C + +, Java, PHP, Python и IDL. Кроме того, BOUML содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах. В основе работы BOUML лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты.

Основными достоинствами BOUML являются:

- работает под Linux, MacOS X и Windows, что позволяет программировать одновременно в C + +, Java, PHP, Python и IDL;
- бесплатное распространение в сети Интернет;
- является расширяемым, а внешние инструменты (плагины, потому что они выполняются за пределами BOUML) могут быть разработаны в C + + или Java;
- обладает высоким быстродействием и не требует много памяти для управления несколькими тысячами классов.

В результате разработки проекта с помощью CASE-средства BOUML формируются следующие документы:

- диаграммы UML, в совокупности представляющие собой модель разрабатываемой программной системы;
- спецификации классов, объектов, атрибутов и операций;
- заготовки текстов программ.

Тексты программ являются заготовками для последующей работы программистов. Они формируются в рабочем каталоге в виде файлов ти-

пов .h (заголовки, содержащие описания классов) и .cpp (заготовки программ для методов). Система включает в программные файлы собственные комментарии. Состав информации, записываемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

Лицензия

Релизы BOUML / bouml.exe от 5,0 не являются свободными, чтобы использовать их нужно купить лицензию, и запускать их на хост на дату авторизирования этой лицензии. Версии BOUML и связанные с ними инструменты до 5.0, **bomlViewer**, генераторы кода **projectControl** и **projectSynchro** свободны в использовании, т.е. являются бесплатными, вы можете распространять и/или изменять их в соответствии с условиями GNU General Public License, опубликованной в Free Software Foundation. Копирование и распространение документации или части документации и скриншотов разрешается на любом носителе, при условии упоминания BOUML или http://www.bouml.fr.

BOUML распространяется в Интернете по адресу http://www.bouml.fr/.

Запуск

При запуске BOUML без лицензии, на экране появится сообщение (рис. 19):

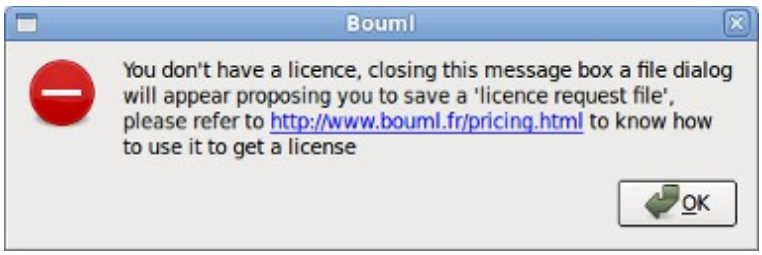


Рис. 19 Окно допуска BOUML

Затем файл запроса лицензии по созданию предлагает вам создать и отправить файл запроса с лицензией. Диалоговое окно необходимо закрыть. После этого выводится диалог (рис. 20), где необходимо ввести свой идентификатор в первом поле.

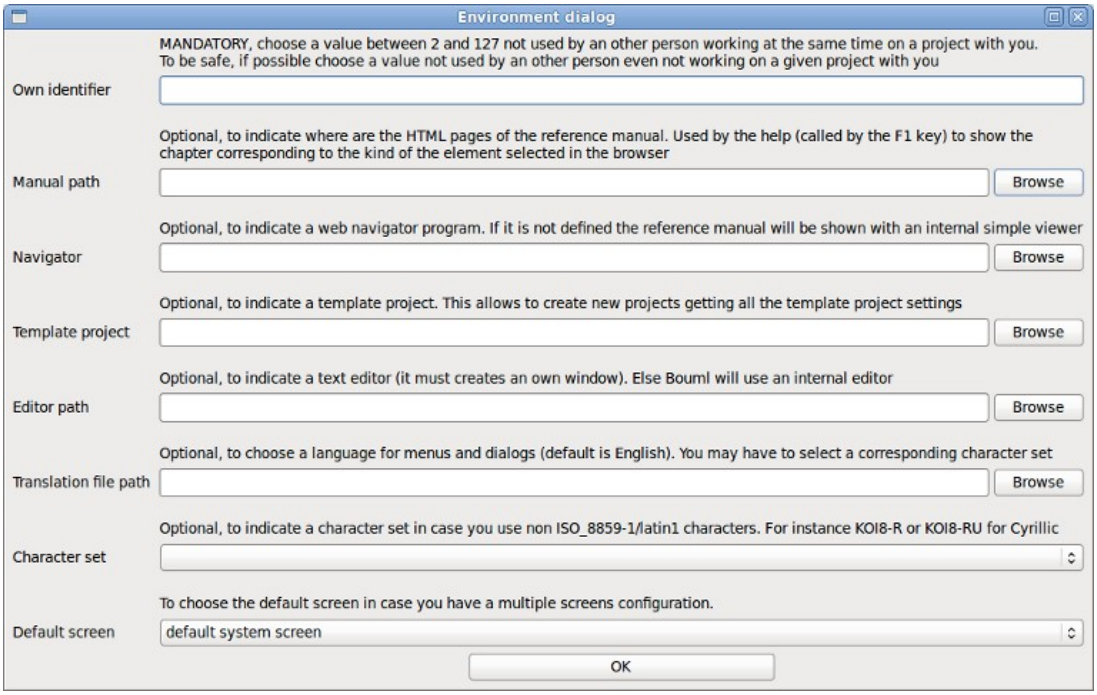


Рис. 20 Диалоговое окно для ввода идентификатора пользователя

Другая информация не является обязательной. Обратите внимание, вы можете задать экран по умолчанию в случае, если у вас есть конфигурации с несколькими экран, и набор символов, если вы хотите использовать не кодировку latin1.

После этого Вы можете использовать BOUML: открыть уже имеющийся проект или создать новый.

Общий вид BOUML

BOUML- окно состоит из трех частей (рис.21).

Левое подокно отображает элементы вашего проекта, навигация по которому может быть выполнена с помощью мыши или стрелок клавиатуры. Жирный шрифт используется, если элемент является изменяемым, пункт доступен только для чтения, когда вы не имеете права на запись для файла (ов) поддерживающие его (см. файлов проекта). Система элементов поддержки API плагин-Out (например UmlBaseActor) также доступны только для чтения.

Нижнее правое подокно используется для отображения и изменения комментариев, связанных с выбранным элементом.

Верхняя правая часть рабочего окна используется для отображения и изменения диаграмм. Здесь можно развернуть как все диаграммы, так и свести отображение к минимуму.

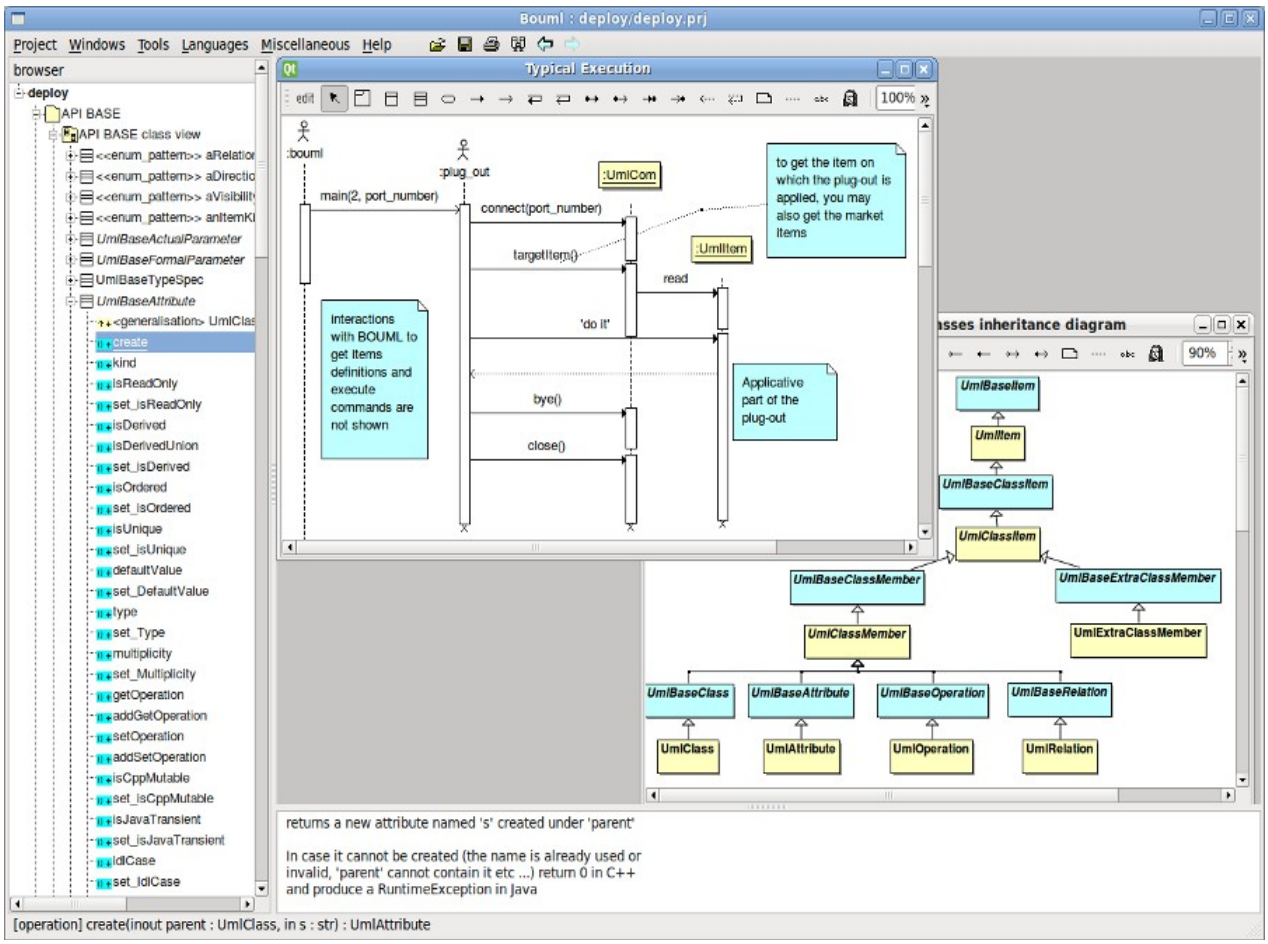


Рис. 21 . Общий вид рабочего окна BOUML

Меню верхнего уровня

Меню рабочего окна BOUML содержит следующие элементы:

- **Project** – меню команд для работы с проектом;
- **Windows** - команды для управления окнами;
- **Tools** – сервис;
- **Languages** – выбор языка программирования;
- **Miscellaneous** – остальные команды;
- **Help** – помощь.

Конечно, навигация по управляющим командам BOUML осложняется для большинства студентов отсутствием русскоязычного перевода. Но хочется отметить, что большинство названий настолько явно или интуитивно понятны, что можно с уверенность констатировать: изучение BOUML, даже на самостоятельной основе, довольно простое и выполнимое занятие. А чтобы еще более облегчить нелегкий студенческий путь, рассмотрим основные элементы меню более подробно.

Меню Project

Команды, расположенные в меню Project (Проект) позволяют (рис. 22): создать новый проект, загрузить проект, сохранить его на месте или в новом месте, распечатать текущую схему, закрыть проект или совсем по-

кинуть BOUML. В нижней части меню содержит историю событий, верхняя линия соответствует последнему из открытых проектов.

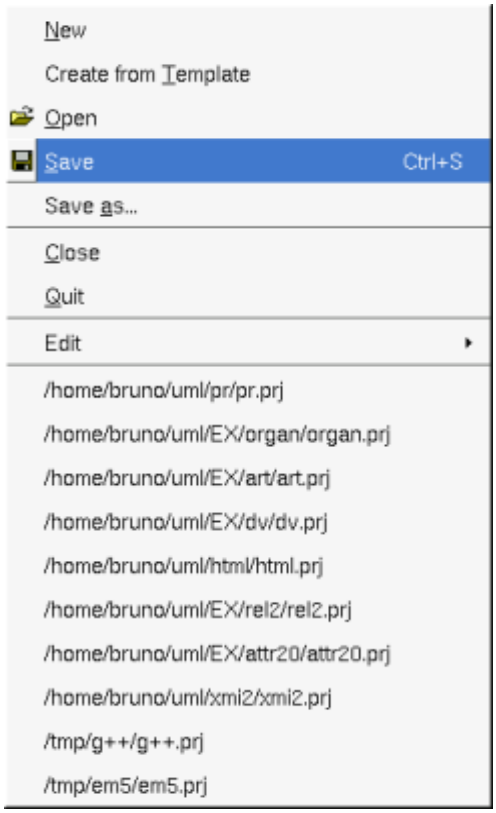


Рис. 22 Меню Project

Возможно создание проекта из шаблона, в этом случае настройки и установки проекта-родителя сохраняются без изменений.

Меню Windows

Как понятно из названия, меню Windows (рис. 23) содержит набор команд для управления окнами в среде BOUML.

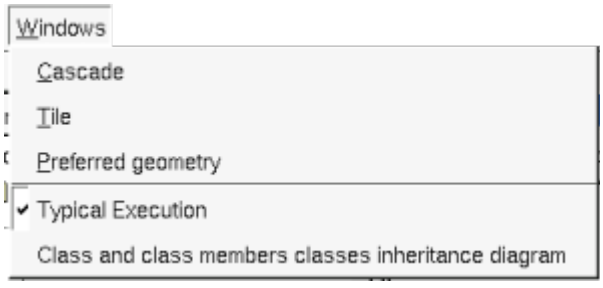


Рис. 23 Меню Windows

Позволяет задавать взаимное расположение при открытии нескольких окон, управлять их выводом.

Меню Tools

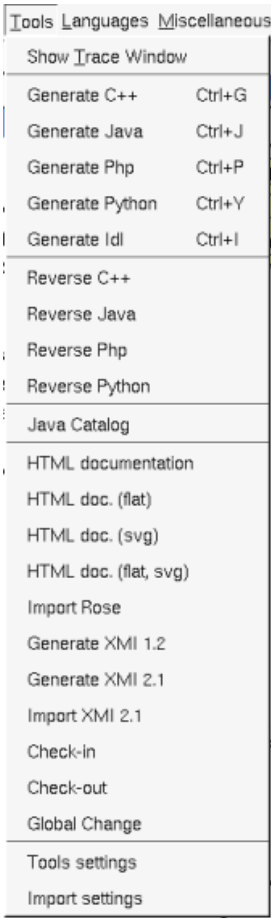


Рис. 24 Меню Tools

Команды меню Tools (Сервис), общий вид которого представлен на рисунке 24, позволяют открыть окно трассировки, а так же окно для записи сообщений (примечаний к элементам диаграмм). Команды этого окна также позволяют добавить/удалить/изменить плагины для генерации шаблонов программ по представленным шаблонам, а также выполнять обратное действие (reverse) к существующему программному коду (при наличии соответствующих плагинов). Возможно также импортировать в окно диаграмм дополнения из других программ, таких как Rational Rose, HTML.

Меню Languages

Меню Languages (рис. 25) позволяет выбирать: производить или нет по умолчанию преобразование разрабатываемых диаграмм на заданные пользователем языки (C++ generation, Java generation, Php generation, Python generation и Idl generation).

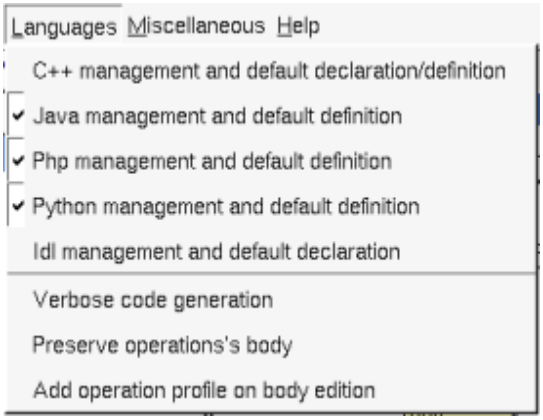


Рис. 25 Меню Languages

При этом необходимо учитывать, что полностью программный код BOUML производить не может. Он создает описания классов/артефактов/операций/отношений и др.

Меню Miscellaneous

В меню Miscellaneous (Разное) собраны команды для работы с видом и стилем элементов создаваемых диаграмм и самих диаграмм в целом (рис. 26). Флажок Show stereotypes in browser позволяет управлять выводом стереотипов, флажок Completion in dialog – регулирует поиск в списках выбора в диалогах. Команда Style позволяет управлять стилем позволяет управлять стилем вывода; команда Front size – устанавливает размер шрифта (рис. 27). Остальные команды этой группы управляют форматом вывода диаграмм: можно задавать разрешение, стиль и форму диаграмм, а также выводить сетку на диаграмме (рис 28).

.

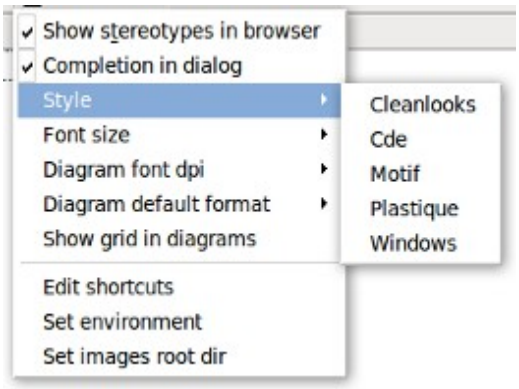


Рис. 26 Меню Miscellaneous, вкладка Style

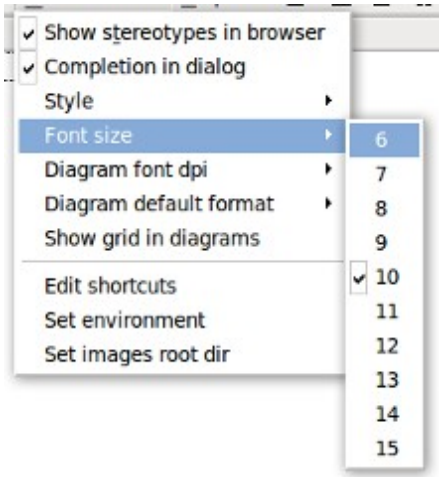


Рис. 27 Меню Miscellaneous, вкладка Front size

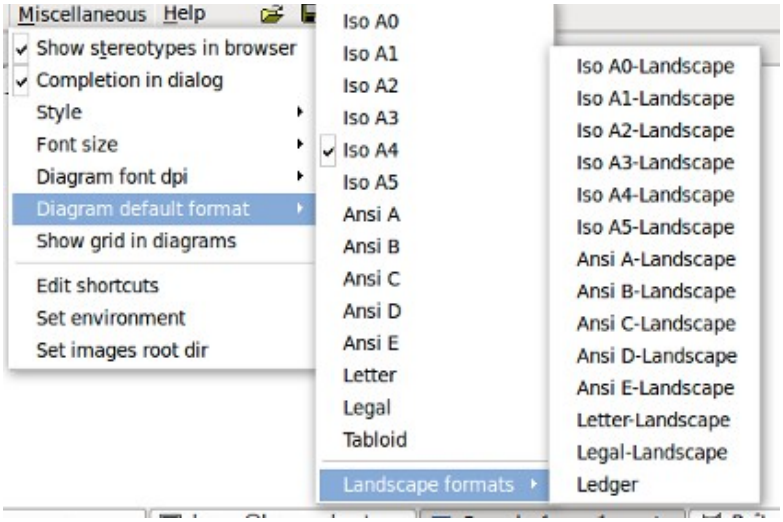


Рис. 28 Меню Miscellaneous

Команда Edit shortcuts содержит возможности для управления ярлыками (горячими клавишами) в проекте. При выборе этой команды появляется диалоговое окно (рис. 29):

Окно содержит две вкладки. Первая позволяет назначить горячие клавиши для команд (на рисунке вы видите установки по умолчанию), а вторая позволяет определить горячие клавиши для команд дополнительных плагинов. Назначения выполняются при помощи щелчка мыши на соответствующих ячейках для выбранных клавиш. Последняя колонка *do* позволяет копировать/вставлять/ вырезать/изменять в той строке, где производится щелчок мыши.

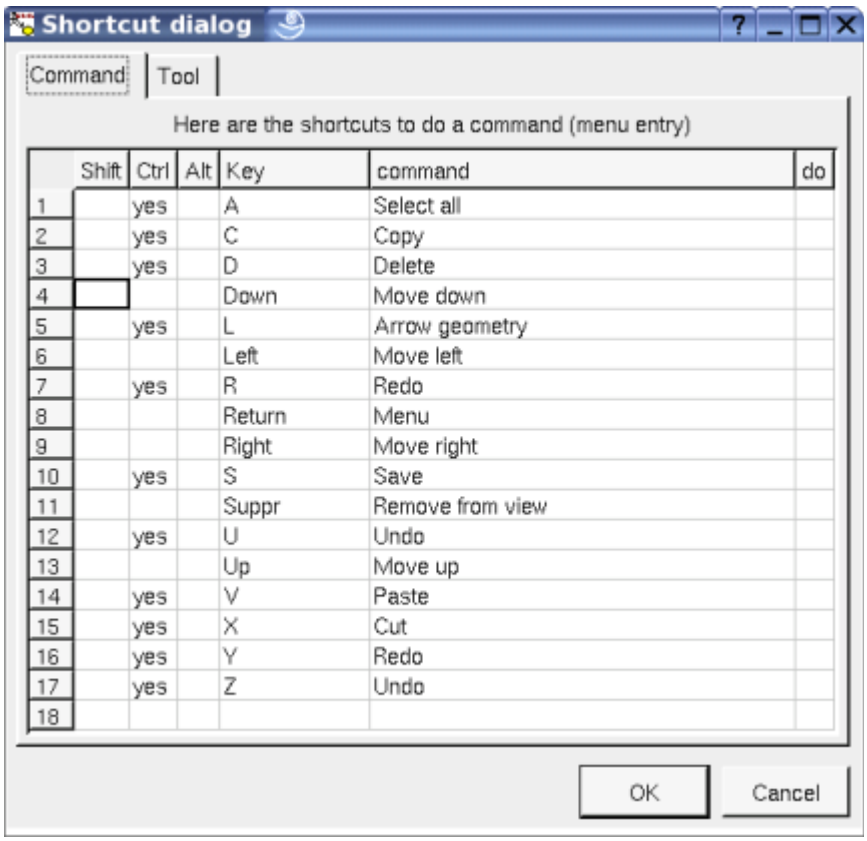


Рис. 29 Диалоговое окно Edit shortcuts

Следующая команда Environment выводит диалоговое окно, которое вы видели еще в самом начале, при установке BOUML (рис. 20), которое позволяет управлять окружающей средой. Изменить идентификатор пользователя здесь уже нельзя, но управлять языком вывода, и устанавливать различные дополнительные опции для проекта – можно.

Панель инструментов

Панель инструментов BOUML расположена в верхней части окна и содержит следующие кнопки (рис. 30):


Рис. 30 Панель инструментов BOUML

Эти кнопки позволяют:

• открыть проект;

• сохранить текущий проект;

• выполнить печать активной диаграммы;

• выполнить поиск элементов через их имена или стереотипы (род), или выполнить поиск заданной строки в описаниях / определениях, которые можно задать с помощью появляющегося диалогового окна (рис. 31):

Рис. 31 Диалоговое окно поиска элементов

• ориентироваться в истории последних выбранных элементов в браузере

• получать некоторые справочные данные.

Browser items(меню проекта)

Общий вид окна Browser items (меню проекта) представлен на рисунке 32. В данном окне указываются все элементы создаваемой пользователем диаграммы (виды диаграмм, классы, актеры, стереотипы) с их описанием (возможно даже без особого смысла для BOUML). Описание элементов может добавляться автоматически, используя установки по умолчанию, а могут быть составлены разработчиком вручную.

Жирный шрифт используется в том случае, если элемент является изменяемым. Некоторые элементы доступны только для чтения, например, система элементов поддержания APIплагин-Out (UmlBaseClass), поэтому визуальное оформление поможет вам не изменять их, чтобы иметь шанс сохранить совместимость с будущими версиями.

BOUML предлагает 4 специальных вида просмотра на многих уровнях. Это необходимо для того, чтобы обеспечить необходимую структуру и организацию для диаграмм и элементов моделей. При наведении курсора на большинство из элементов всплывают контекстные меню, позволяющие создавать только те элементы диаграмм, которые соответствуют выбранному способу отображения.

Для создания нового элемента используется щелчок левой кнопки мыши по названию элемента в контекстном меню, вызываемом в браузере проекта. При этом элемент отображается последним в списке соответствующих данной диаграмме (группе) элементов.

После создания элемент можно перетаскивать, удерживая нажатой левую кнопку мыши, но успешное выполнение этого действия зависит от возможностей самого элемента (например, отношения переместить нельзя). Множественный выбор в браузере в BOUML невозможен.

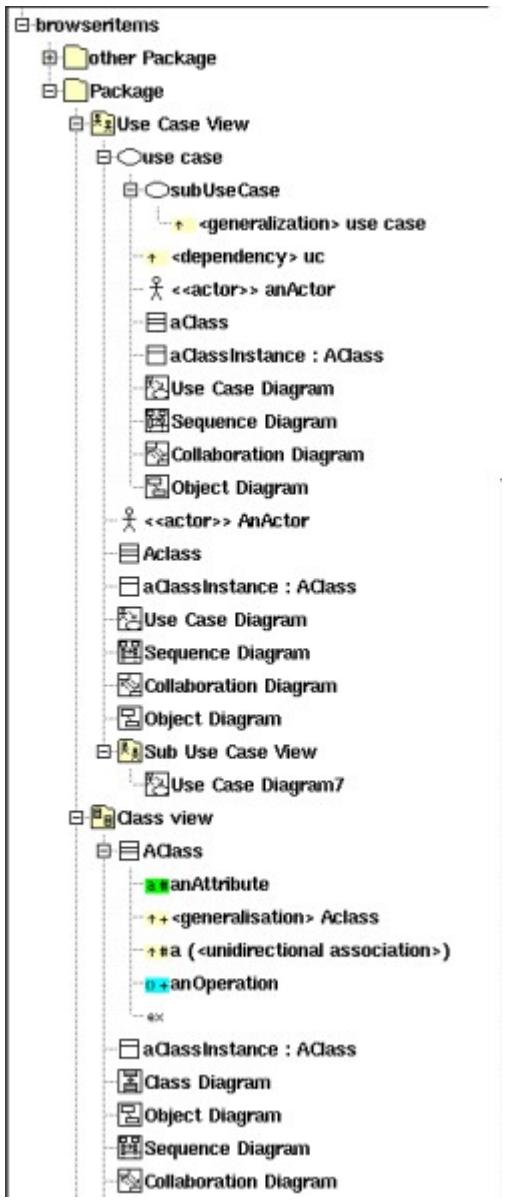


Рис. 32 Окно Browser items(меню проекта)

При удалении элемента из диаграммы рядом отображается значок красного креста (☒). При этом нужно помнить, что удаленные элементы не сохраняются, но их можно восстановить, пока проект не закрыт.

Двойной щелчок мыши по элементу диаграммы в браузере вызывает диалоговое окно для редактирования свойств этого элемента: можно изменить название, установить формат доступа, добавить текстовое описание или изменить цвет.

Создание нового проекта

Для создания нового проекта необходимо воспользоваться командой New из меню Project, расположенного в верхней части окна BOUML. При создании нового проекта сначала появится диалоговое окно (рис. 33), с помощью которого вы можете выбрать каталог, куда будут сохранен ваш

проект, и имя проекта. На рисунке выбран каталог для размещения проекта / TMP и имя проекта - Foo.

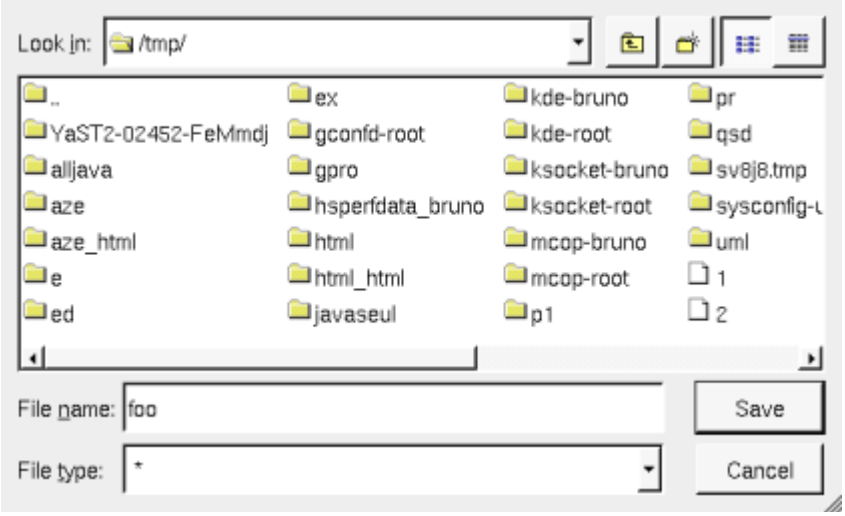


Рис. 33 Сохранение нового проекта

В этом случае BOUML создаст каталог Foo в / TMP и поместит туда некоторые файлы, в том числе foo.prj, который и содержит сам проект (рис. 34):

Рис. 34 Файлы проекта

Переименовать или удалять файлы, созданные BOUML, нельзя!

Для открытия уже созданного проекта необходимо воспользоваться командой Open, которая находится в меню Project, расположенным вверху рабочего окна BOUML.

Создание диаграммы use case в BOUML

Диаграммы прецедентов могут содержать описание этих же прецедентов, но с другой точки зрения или в других вариантах использования,

актеров, классы, экземпляры классов, виды деятельности и различные диаграммы (диаграммы прецедентов, диаграммы объектов, диаграммы последовательности, сотрудничества схемы и диаграммы классов) в любом порядке. Список всех используемых элементов диаграммы можно просматривать в браузере BOUML (рис. 35).

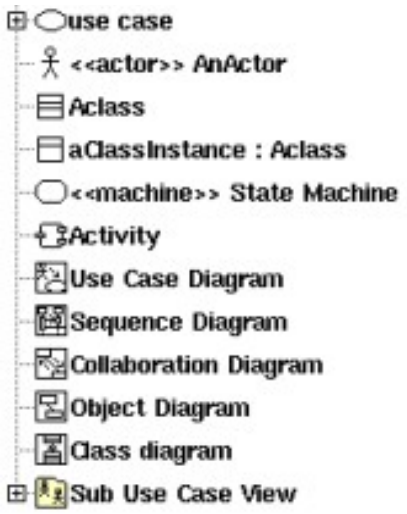


Рис. 35 Вид списка элементов в браузере BOUML

Сама диаграмма прецедентов представляет собой схему из определенных (возможных) объектов (актеров, прецедентов, пакетов, описаний) и связей между ними. Пример окна, содержащего диаграмму прецедентов, приведен на рисунке 36.

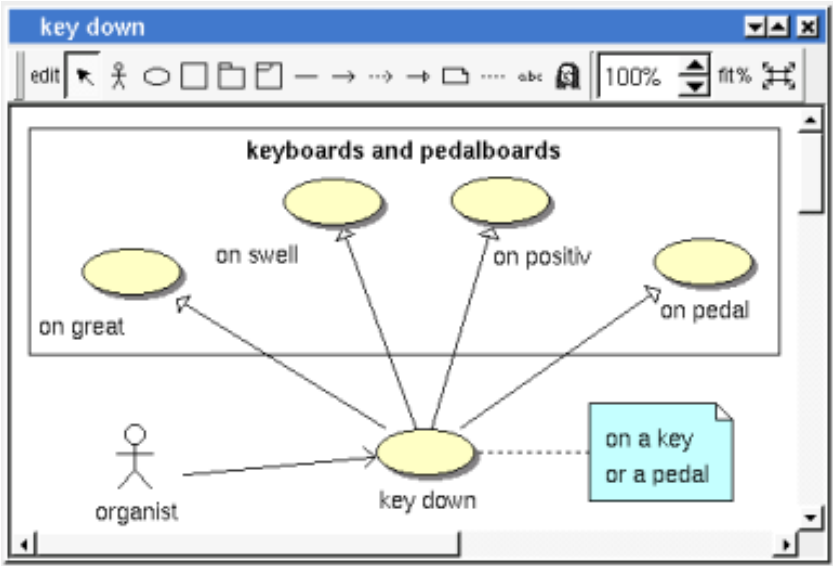


Рис. 36 Пример диаграммы

Диаграмма прецедентов создается с помощью контекстного меню, которое можно вызвать из браузера. Узел меню появляется по нажатию правой кнопки мыши на имя диаграммы в браузере (рис. 37):

Рис. 37 Контекстное меню создания диаграммы

Первая команда этого меню – Show – выводит диаграмму в окно BOUML. Меню, содержащее команды, необходимые для создания всевозможных элементов диаграммы прецедентов, появляется при нажатии правой кнопки мыши на представление вариантов использования в браузере, и выглядит следующим образом (рис.38):

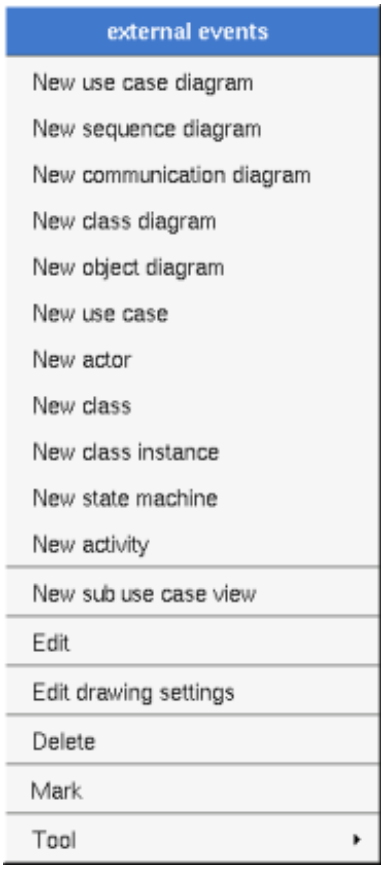


Рис. 38 Контекстное меню для создания диаграммы use case

В верхней части меню перечисляются команды для создания новых диаграмм по их типам, названия которых говорят сами за себя:

New use case diagram – создание диаграммы use case;

New sequence diagram – создание диаграммы последовательностей;

New communication diagram создание диаграммы взаимодействий;

New class diagram – создание диаграммы классов;

New object diagram – создание объектной диаграммы.

Затем следуют команды для создания новых элементов на диаграмме:

New use case – создание прецедента;

New actor – создание актера;

New class – создание класса;

New class instance; New state machine; New activity – создание различных видов классов.

Далее в меню расположены команды для управления уже созданными элементами. Рассмотрим их действие подробнее.

Edit - выводит диалоговое окно (рис. 39), в котором предоставляется возможность редактирования свойств и описаний созданного прецедента, выбирая для него имя, стереотип и составляя описание.

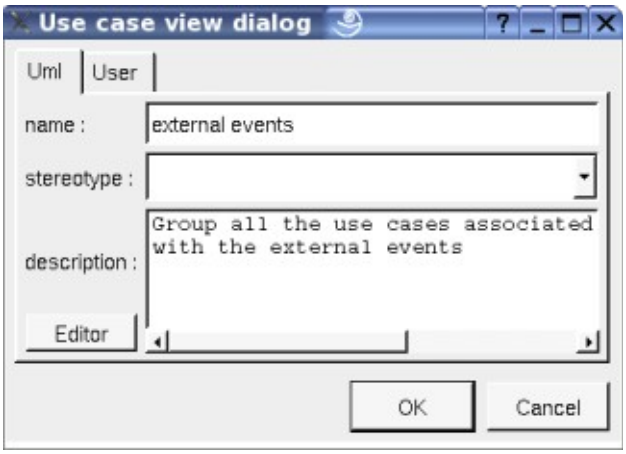


Рис. 39 Диалоговое окно Edit

Причем, задания стереотипа (в соответствующем поле этого окна) не имеет особого значения для BOUML.

Edit drawing settings - этот диалог (рис. 40) позволяет изменить свойства самой диаграммы, используемые по умолчанию (цвет, вид, тень и т.п.).

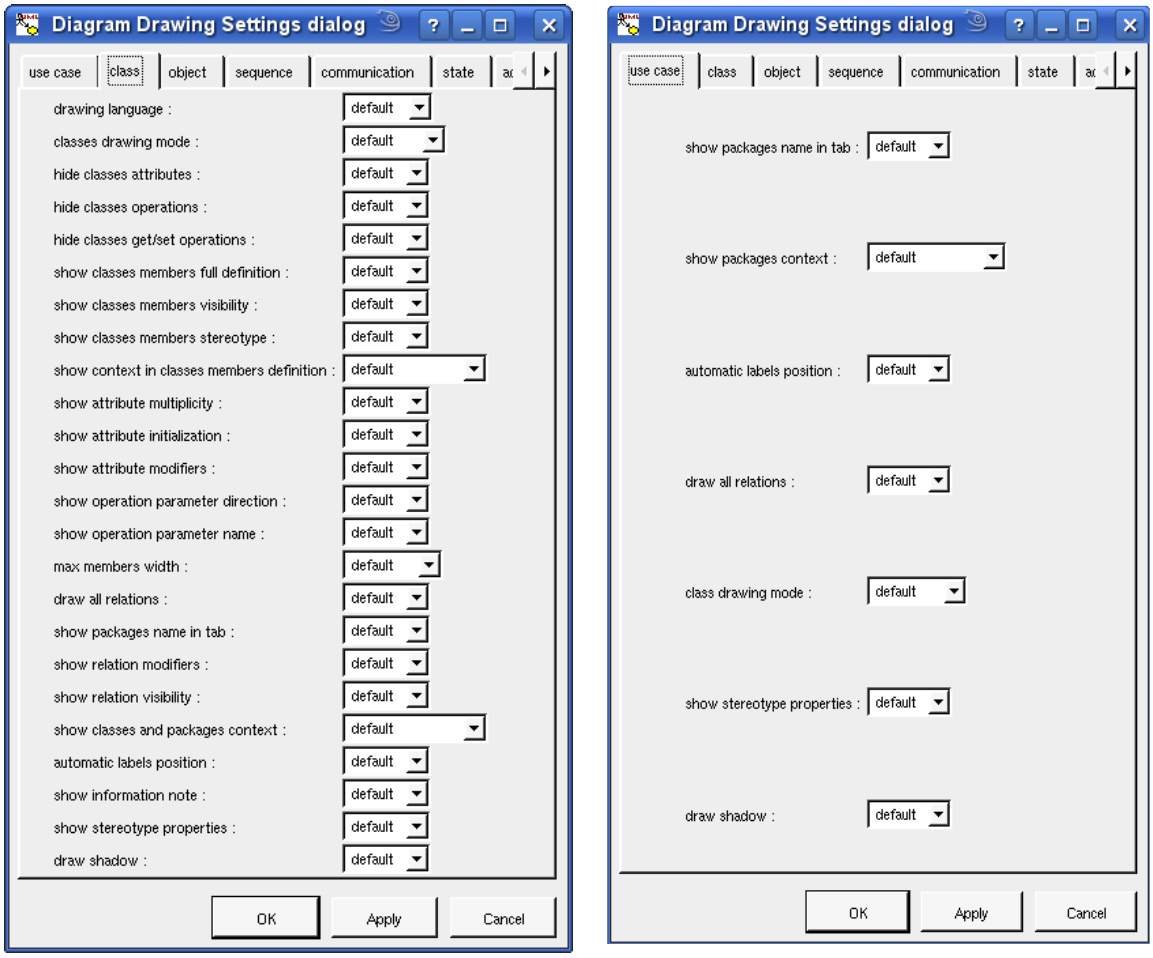
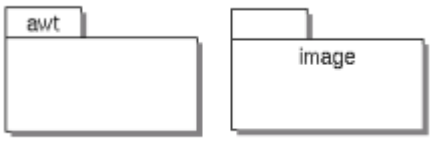


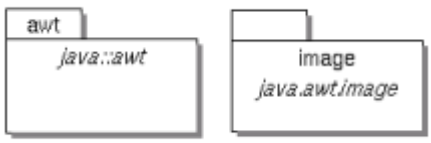
Рис. 40 Диалоговое окно редактирования свойств диаграмм

Например:

package name in tab : местоположение имени пакета в закладке: имя пакета должно быть написано на верхнем прямоугольнике значка пакета или на нижнем:



show packages context : показать пакеты контекста. Контекст, т.е. путь к пакету, может быть указан как “UML context” и тогда будет показан путь к пакету в браузере, или C++ / Php.



show stereotype properties: показывать или нет свойства элемента. По умолчанию свойства скрыты.

Delete - возможность удаления доступна только в том случае, если пакет установлен не только для чтения. При удалении элемента в браузере появляется значок с красным крестом (). Удаленные прецеденты и всех их наследников можно восстановить из браузера, пока проект не закрыт.

Если элемент удалить не удается, значит, либо он сам, либо его наследники имеют свойство «только для чтения».

Удаленные элементы после закрытия проекта не сохраняются.

Mark -команда для маркировки необходимых элементов диаграммы с возможностью, например, их последующего копирования.

Tools - пункт меню *tool* присутствует в том случае, если необходимо использовать дополнительные плагины в диаграмме прецедентов.

Важные дополнения

Диаграмма вариантов использования может содержать актеров, классы, варианты использования, пакеты, фрагмент, нотации, тексты, диаграммы ярлыков, предметов и отношений. Для отображения на диаграмме этих элементов можно воспользоваться контекстными меню Browser items, о которых было написано выше. Но можно использовать и другой подход, применяя для создания элементов диаграмм кнопки панели инструментов, расположенной в верхней части окна диаграмм (рис. 41).

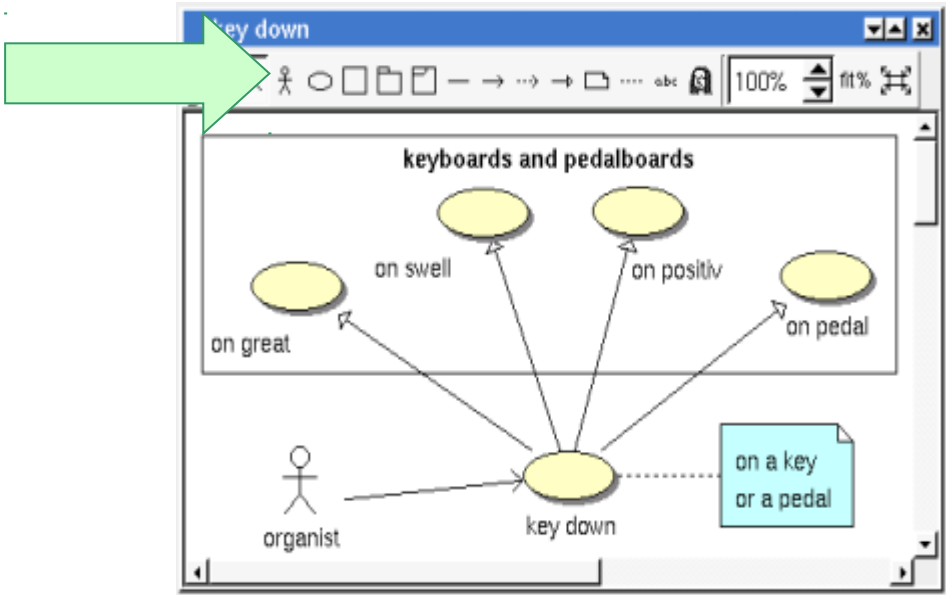


Рис. 41 Пример диаграммы вариантов использования

На этой панели расположены значки (ярлыки, иконки) основных элементов, вывод которых на диаграмму осуществляется щелчком левой кнопки мыши. Значение каждого ярлыка рассматривались в разделе «Основные конструкции UML».

Кнопка <abc> позволяет напечатать текст на фоне (например, заголовок диаграммы), а кнопка с изображением головы используется для вставки на диаграмму сторонних объектов(рисунков и т.п.). Там же расположе-

но окошко для изменения масштаба диаграммы. Масштаб может быть изменен от 30 до 200%, чтобы позволить увидеть соотношение видов даже с небольшим масштабом, при этом размеры стрелок и другие стандартные выводы не меняются.

Последние две кнопки позволяют регулировать размеры диаграммы до установленных по умолчанию и управлять размерами окна вывода диаграммы.

Выбор элемента выполняется левой кнопкой мыши. Правая кнопка мыши используется для вывода контекстного меню. Контекстные меню содержат установки по умолчанию для выбранного элемента диаграммы и команды для редактирования этих установок. Почти для каждого элемента можно задать имя, описание, формат вывода (цвет, стиль, толщина линий) и др.

Когда выбраны несколько элементов на диаграмме, то контекстное меню, появляющиеся по щелчку правой кнопки мыши (рис. 42), позволяет управлять их взаимным расположением или изменять размеры выбранных элементов:

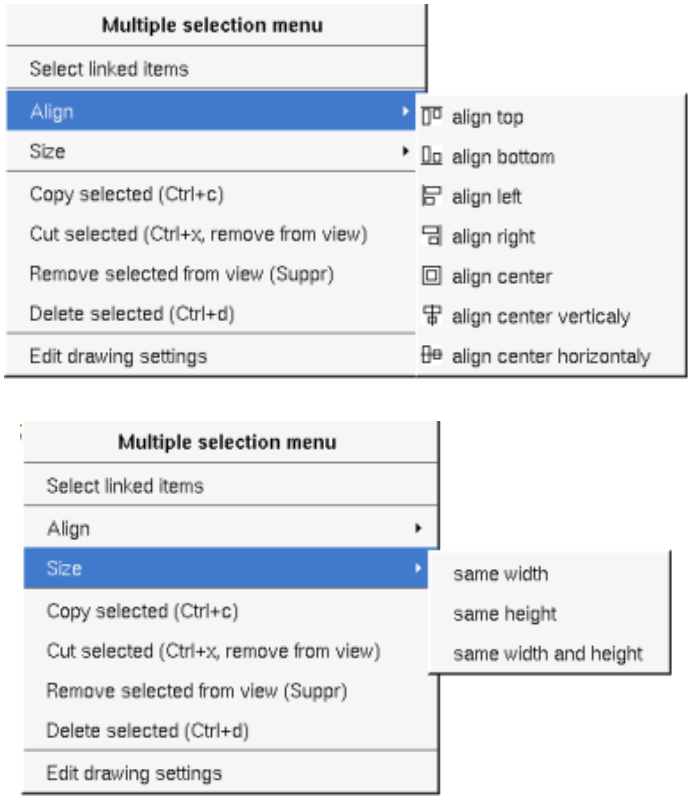


Рис. 42 Контекстные меню для управления выводом нескольких элементов диаграммы

Если все выбранные элементы имеют один и тот же вид (например, все прецеденты), то можно редактировать их параметры рисования (рис. 43):

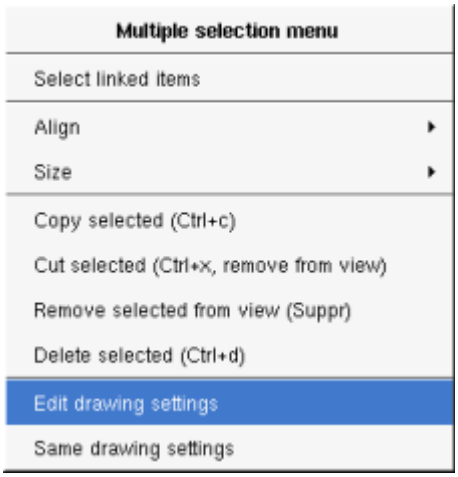


Рис. 43 Контекстное меню для редактирования рисования

При этом выводится диалоговое окно, в котором параметр *<unchanged>* указывает на то, что данное изменение не применимо для выбранных элементов. В противном случае, изменение устанавливается для всех элементов.

Пример выполнения лабораторной работы

Постановка задачи:

Разработать клиент-серверную систему регистрации. Система должна позволять студентам регистрироваться на курсы и просматривать свои табели успеваемости с персональных компьютеров, подключенных к локальной сети университета. Профессора должны иметь доступ к системе, чтобы указать курсы, которые они будут читать, и проставить оценки за курсы. Причем, база данных, содержащая всю информацию о курсах (каталог курсов), остается функционировать в прежнем виде. Новая система будет работать с существующей БД в режиме доступа, без обновления.

Регистрация на курсы происходит следующим образом: в начале каждого семестра студенты могут запросить у регистратора каталог курсов, содержащий список курсов, предлагаемых в данном семестре. Информация о каждом курсе должна включать имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Студент может выбрать 4 курса в предстоящем семестре.

После регистрации некоторого студента, регистратор направляет информацию в расчетную систему, чтобы студент мог внести плату за семестр.

В конце семестра студенты могут просмотреть свои табели успеваемости. Поскольку эта информация конфиденциальная, система должна обеспечивать ее защиту от несанкционированного доступа.

Профессора должны иметь доступ к онлайновой системе, чтобы указать курсы, которые они будут читать, и просмотреть список студентов, записавшихся на их курсы. Кроме этого, профессора должны иметь возможность проставить оценки за курсы.

Ход выполнения:

Построение диаграммы прецедентов (диаграммы вариантов использования – Use Case Diagram)

Use Case Diagram представляет собой графическое описание всех или части актеров, прецедентов и варианты их взаимодействия в системе. В каждой разрабатываемой программной системе обычно есть главная диаграмма прецедентов, которая отображает границы системы (актеров) и ее основное функциональное поведение (прецеденты). Другие же диаграммы прецедентов могут создаваться при необходимости, например:

1) диаграмма, отражающая все прецеденты для определенного актера;
2) диаграмма, отображающая все прецеденты для данного варианта использования системы;
3) диаграмма, отображающая определенный прецедент и все его отношения в системе.

Для решения поставленной задачи и составления диаграмм вариантов использования для проектируемой системы необходимо выполнить следующие шаги.

1. Спецификация требований

Основываясь на задании, можно выделить следующие функциональные требования к ПС:

Система должна позволять студенту регистрироваться на курсы.

Система должна позволять студенту просматривать свои табели успеваемости с персонального компьютера, подключенного к локальной сети университета.

Профессора должны иметь доступ к системе для указания курсов, которые они будут читать.

Профессора должны иметь доступ к системе для проставления оценок за курсы.

Система должна формировать каталог курсов, который может предоставляться зарегистрированному студенту по требованию.

Информация о каждом курсе должна включать имя профессора, наименование кафедры и требования к предварительному уровню подготовки (прослушанным курсам).

Система должна быть связана с расчетной частью для выставления счета студенту.

Система должна предоставлять возможность студенту выбрать 4 курса.

Система должна обеспечивать защиту информации от несанкционированного доступа.

2. Выделение действующих лиц

В результате анализа требований к разрабатываемой программной системе (см. «Спецификация требований») были выделены следующие действующие лица, а также определены их основные возможные действия:

1. **Расчетная система** – получает и обрабатывает информацию об оплате некоторого курса.
2. **Регистратор** (registrator) – хранит все данные о всех субъектах системы (профессорах, студентах, курсах) и успеваемости; составляет учебный план; составляет каталог курсов для текущего семестра.
3. **Каталог курсов** – содержит всю информацию о предлагаемых университетом курсах (база данных).
4. **Профессор** (professor) – регистрируется в системе; может выбирать курсы для ведения и выставлять оценки за выбранный курс студенту.
5. **Студент** – регистрируется в системе; выбирает и записывается на курс; просматривает свой табель успеваемости в текущем семестре.
6. **Polzovatel** – человек, еще не получивший доступ к системе.

3. Описание вариантов использования

Основываясь на требованиях к системе (см. «Спецификацию требований») и потребностях основных действующих лиц (см. «Выделение действующих лиц») были выделены следующие сценарии:

1. Вход в систему.
2. Выбор курсов для ведения.
3. Выставление оценок.
4. Регистрация на курсы.
5. Просмотр табеля успеваемости.
6. Хранение и обработка информации о профессорах.
7. Хранение и обработка информации о студентах.
8. Закрытие регистрации.

4. Спецификации по вариантам использования

Сценарий 1: « Вход в систему»

Описание: данный сценарий описывает осуществление начального доступа пользователя к системе.

Основной поток событий:

1. Система запрашивает у пользователя регистрационные данные (логин и пароль).
2. Пользователь вводит регистрационные данные (логин и пароль).
3. Система обрабатывает регистрационные данные (логин и пароль) и открывает доступ.

Альтернативные потоки:

1) Неправильный ввод регистрационных данных: при обнаружении системой неправильного ввода пользователем регистрационных данных, выводится сообщении об ошибке. Для пользователя предусмотрены две возможных реакции:
   a. повторить ввод (использование этого сценария заново);
   b. отказаться от входа.

При успешном выполнении данного варианта использования пользователь получает доступ к системе. В противном случае состояние системы неизменно.

Предусловия: нет.

Постусловия: при успешном выполнении сценария – пользователь получает доступ к ресурсам системы, в противном случае – состояние системы не меняется.

Сценарий 2: «Выбор курсов для ведения»

Описание: данный сценарий позволяет профессору выбрать дисциплину для преподавания из каталога курсов.

Основной поток событий:

1) система запрашивает требуемое действие:
   a) просмотреть каталог курсов;
   b) выбрать курс;
   c) сохранить список курсов;
   d) обновить список курсов;
   e) посмотреть список выбранных курсов;
   f) удалить список курсов.
2) После выбора действия выполняется один из подчиненных потоков:
   a) просмотреть каталог курсов:
      1. система запрашивает номер семестра;
      2. система выводит каталог всех существующих курсов для текущего семестра;
   b) *выбрать курс*:
      1. система запрашивает номер семестра;

2. система выводит каталог всех существующих курсов для текущего семестра;

3. профессор выбирает из каталога курс для ведения;

4. система добавляет выбранный курс в список курсов;

5. выполняется подчиненный поток «Сохранить список курсов»;

c) обновить список курсов:

1. система выводит текущий список курсов для преподавателя;

2. система запрашивает номер семестра;

3. система выводит каталог всех существующих курсов для текущего семестра;

4. профессор выбирает из каталога курс для ведения;

5. система добавляет выбранный курс в список курсов;

6. выполняется подчиненный поток «Сохранить список курсов» (SOHRANit spisok kursov);

d) посмотреть список выбранных курсов:

1. система выводит текущий список курсов для преподавателя;

2. система предлагает пользователю выбор из двух подчиненных потоков: «Обновить список курсов» или «Удалить список курсов»;

e) сохранить список курсов:

1. для каждого выбранного пользователем курса проверяется его наличие в списке курсов и, при отсутствии такового, курс добавляется в список и список сохраняется в системе;

f) *удалить список курсов*:

1. система выводит текущий список курсов для преподавателя;

2. система запрашивает подтверждение на удаление списка;

3. профессор подтверждает удаление;

4. система удаляет список курсов данного профессора.

Альтернативные потоки:

1) Не выполнены предварительные требования: курс уже выбран другим преподавателем – система выдает сообщение об ошибке и предла-

гает пользователю выбор из подчиненных потоков: «Обновить список курсов» или «Удалить список курсов»;

2) Список курсов не найден – при отсутствии списка курсов для данного преподавателя выдается сообщение об ошибке и пользователю предлагается выполнить основной поток событий сначала;

3) Каталог курсов недоступен – при отсутствии связи с базой данных система выдает сообщение об ошибке и завершает вариант использования;

4) Отмена удаления – при выборе пользователем во время подчиненного потока «Удалить список курсов» отказа от удаления, система предлагает пользователю выполнения основного потока событий сначала.

Предусловия: успешно выполнен вход в систему.

Постусловия: при успешном завершении варианта использования – список курсов создан, обновлен или удален, при неуспешном – состояние системы неизменно.

И так далее. В данных методических указаниях мы не будем приводить описание всех заявленных сценариев использования системы, так как объем издания ограничен. Но рассмотренные примеры описаний являются, по мнению авторов, достаточными для понимания и использования их как образца при выполнении студентом собственной лабораторной работы.

5. Составление глоссария проекта

Глоссарий предназначен для описания терминологии предметной области. Он может быть использован как неформальный *словарь данных* системы.

Курс – некая дисциплина, предлагаемая университетом для изучения.

Профессор – личность, выполняющая обучение в университете, т.е. преподаватель университета.

Студент - личность, проходящая обучение в университете.

Регистратор (registrator) - некая система, формирующая каталог курсов и учебный план, а так же записывает студентов на курсы и содержит все необходимые данные о курсах, профессорах, студентах и успеваемости.

Расчетная система – некая **с**истема, выполняющая обработку получаемой от студента информации об оплате за курсы.

Предлагаемый курс (Course Offering) - дисциплина, которая будет читаться в определенном семестре, содержит точные дни недели и время.

Каталог курсов - полный список всех предлагаемых университетом дисциплин.

Список курсов – список тех дисциплин, которые выбрал профессор для ведения в текущем семестре.

Список курса (Roster) – полный список студентов, зарегистрировавшихся на предлагаемый курс.

Учебный график (Schedule) – список дисциплин, выбранных студентом в текущем семестре.

Оценка - оценка, выставленная профессором студенту за конкретный курс.

Табель успеваемости (Report Card) – список всех оценок за все курсы, изученные студентом в текущем семестре.

Polzovatel – человек, еще не получивший доступ к системе.

Регистрационные данные – логин и пароль, необходимые для доступа к системе. Пользователи получают их заранее, в зависимости от своего статуса (профессор или студент).

6. Составление диаграмм вариантов использования в среде BOUML

Для составления диаграммы выбранного варианта использования в среде BOUML необходимо воспользоваться рекомендациями, изложенными в разделе «Создание диаграмм use case в BOUML» настоящих методических указаний. Здесь мы приведем результат наших действий по предложенным алгоритмам.

Диаграмма для варианта использования «Вход в систему»(рис. 44):

На диаграмме отображены два актера, участвующие в данном варианте использования системы: polzovatel и registrator. Представлены все возможные действия этих актеров в рамках данного варианта использования системы (описание варианта использования см. выше).

Описание прецедентов диаграммы:

Zapros parolz - запрос регистрационных данных (логина и пароля).

Vvod parolz – ввод пользователем регистрационных данных.

Dostup – разрешение системы на вход.

Oshibka – ошибка ввода регистрационных данных (альтернативный поток).

Зеленым цветом обозначены прецеденты основного потока, красным – альтернативного.

Совет: не забывайте про возможности управлять внешним видом элементов на диаграмме (цвет, форма, положение надписи, размеры). Использование этих дополнительных возможностей сделает вашу диаграмму более информативной, читабельной и интересной, в конце концов.

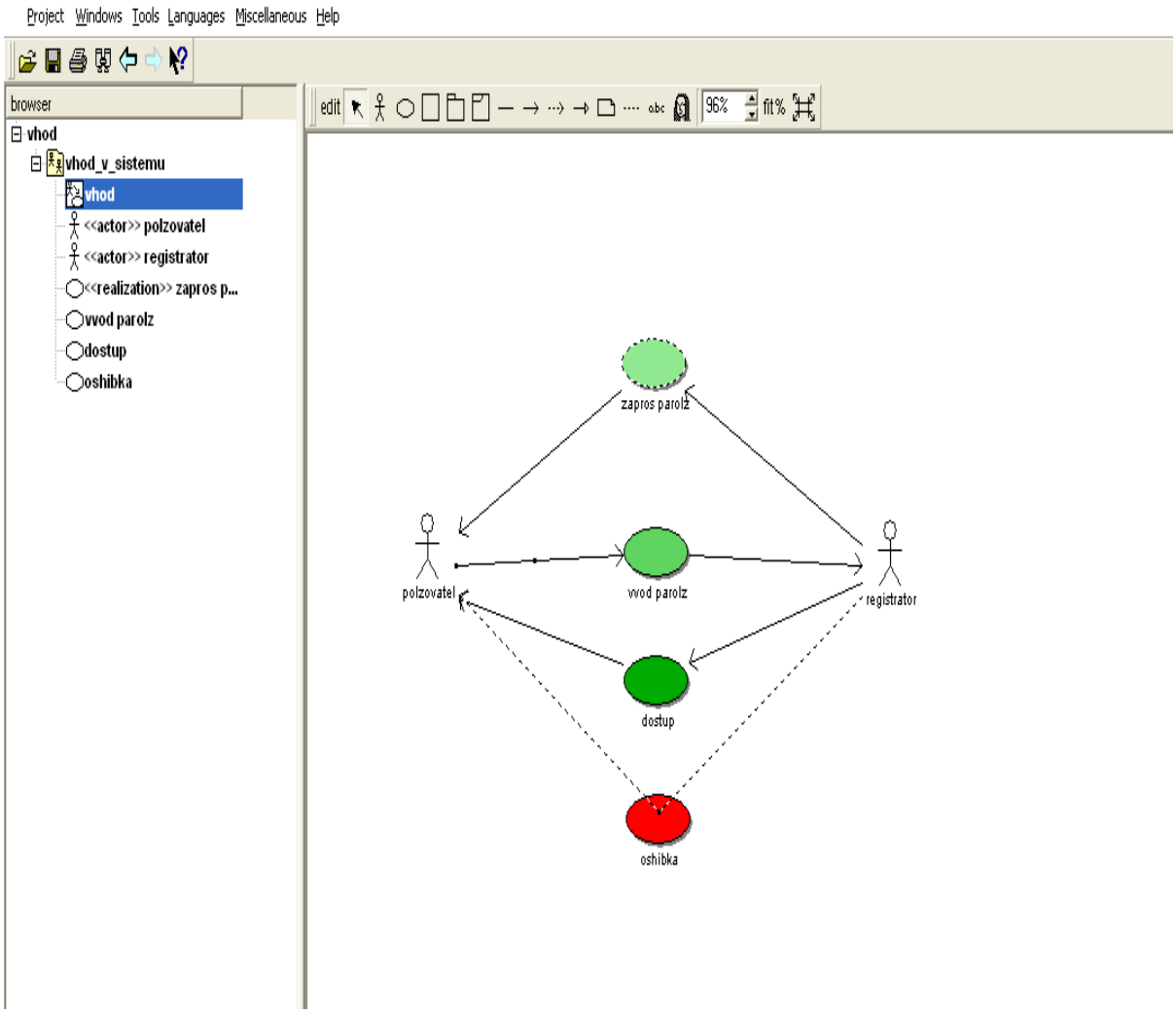


Рис. 44 Диаграмма для варианта использования «Вход в систему»

Диаграмма для варианта использования «Выбор курсов для ведения»(рис. 45):

Диаграмма показывает все возможные действия двух сущностей: professor (внешняя сущность) и registrator (управляющий класс системы). На диаграмме они отображены в виде актеров. Синим цветом показаны возможные в рамках данного варианта использования действия системы, оранжевым – действия актера professor, зеленым – взаимодействие актера registrator со списком выбранных профессором курсов, представляющую собой базу данных системы (статический класс).

Совет: Последовательность действий на данной диаграмме не отображается, для этого предназначены другие виды диаграмм (например, диаграмма последовательностей).

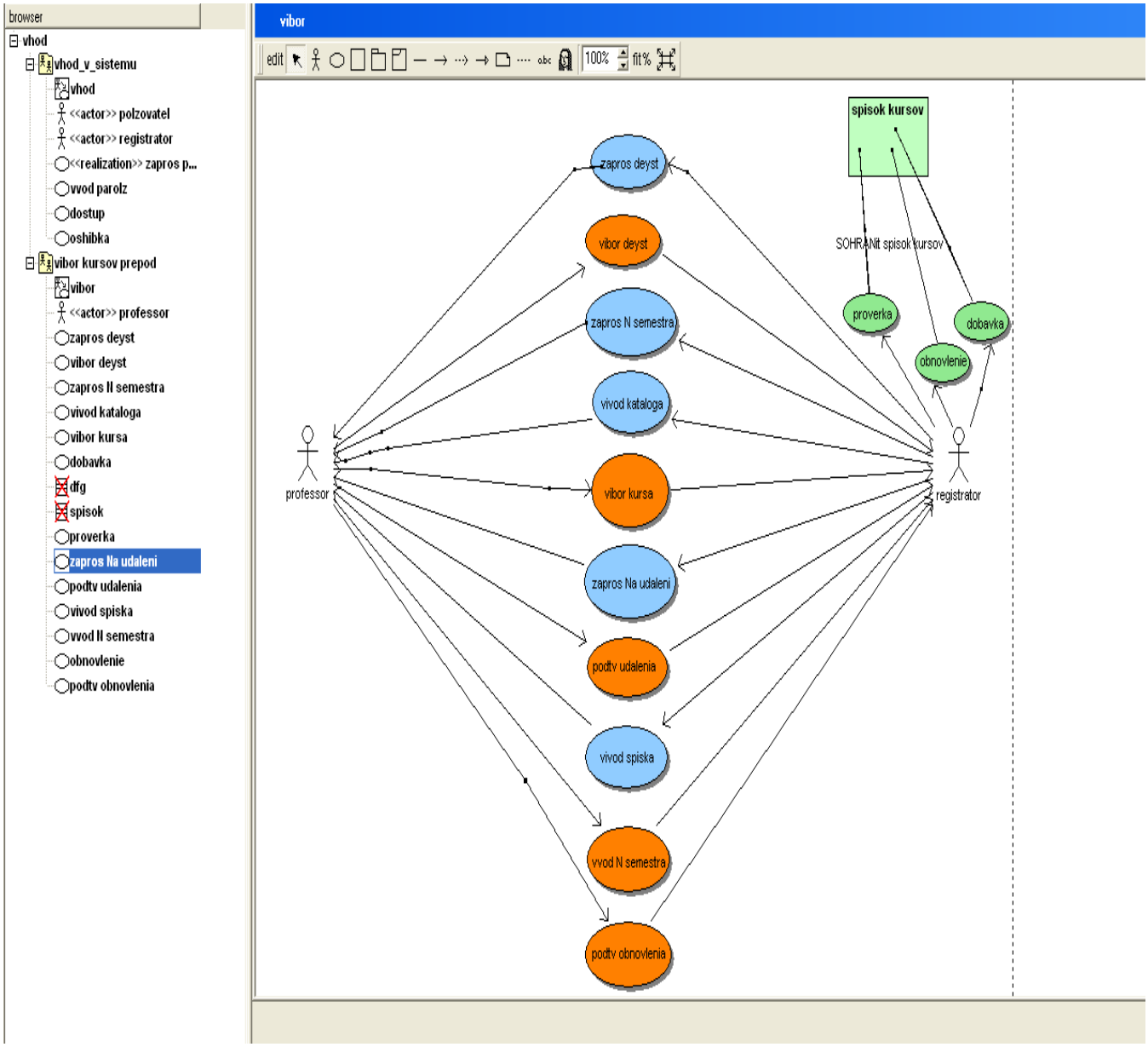


Рис. 45 Диаграмма для варианта использования "Выбор курсов для ведения"

Контрольные вопросы

1. Объясните основные преимущества предварительного моделирования программных систем.
2. Каким образом производится моделирование программных систем?
3. Опишите преимущества использования визуального моделирования.
4. Что такое сценарий использования?
5. Что такое элемент use case?
6. Какие основные конструкции поддерживает UML?
7. Что такое сущностные элементы use case?
8. Виды сущностей в UML?
9. Виды отношений в UML?
10. Какие типы сущностей относят к поведенческим?
11. Что такое структурные сущности? Примеры.
12. Перечислите и опишите виды семантических отношений в UML.
13. Перечислите основные типы диаграмм.
14. Опишите подробно назначение диаграмм прецедентов.
15. Опишите общий алгоритм моделирования системы.
16. Чем отличаются сценарии использования от модели use case?
17. Каким образом можно описать варианты использования?
18. Приведите пример описания варианта использования по Коберну?
19. Основной смысл создания диаграммы вариантов использования?
20. Опишите общий вид рабочего окна BOUML.
21. Перечислите этапы, которые необходимо выполнить для создания диаграммы вариантов использования в BOUML.
22. Что такое глоссарий проекта?
23. Какими методами и правилами необходимо воспользоваться для выделения актеров при формировании диаграммы?
24. Опишите возможные варианты добавления элементов на диаграмму в BOUML.
25. Опишите способы изменения вида элементов диаграмм (цвет, имя, тень) и всей диаграммы в целом.

Литература

1. А. Коберн, Современные методы описания функциональных требований, Лори, 2002.

2. Л. Константайн, Л. Локвуд, Разработка программного обеспечения, Классика Computer Science, Питер,2004.

3. О. Г. Казанцева, Основы моделирования систем с помощью UML и пакета Rational Rose. Знакомство с UML и пакетом Rational Rose, учебно-методическое пособие, Витебск, УО «ВГУ им. И. П. Машерова», 2006.

4. Материалы сайта http://www.bouml.fr

5. Буч Г., Рамбо Дж., Джекобсон А., Язык UML. Руководство пользователя. – С-П.: Издательство «Питер», 2003.

6. Материалы сайта www. uml.ru

7. Материалы сайта http://pvti.ru/lect1-lecture3.htm

8. Материалы сайта http://www.caseclub.ru/articles/use_case.html

Оглавление

Учебное издание

Оксана Федоровна **Абрамова**

Дмитрий Николаевич **Лясин**

Методические указания к лабораторной работе

на тему:

«Основные сведения о UML и BOUML.

Диаграммы вариантов использования»

План электронных изданий 2013 г. Поз. № 16В

Подписано на « Выпуск в свет» . Уч-изд. л. .

На магнитоносителе.

Волгоградский государственный технический университет.

400131, г. Волгоград, пр. Ленина, 28, корп. 1.