Инструментарий для визуализации программного обеспечения в трехмерном пространстве

Романов В.Ю., Шульга И.В.

Аннотация. **В статье рассматриваются методы визуализации архитектуры программной системы в составе инструмента обратного проектирования и восстановления архитектуры программной системы. Рассматриваются методы визуализации и анализа структуры программы в трехмерном пространстве, основанные на метафоре представления программной системы как города со зданиями и районами, а также визуализации объектно-ориентированных метрик программы в такой программе-городе. Описанные в статье возможности инструмента предоставляют инфраструктуру для последующего обнаружения и исправления ошибок проектирования программных систем, а также для рефакторинга программной системы.**

Ключевые слова **— software visualization, reengineering, reverse engineering, architecture recovery.**

I. Введение

Задача обратного проектирования (reverse engineering) программной системы является весьма важной при разработке программной системы с использованием библиотек с исходными кодами. Построение и визуализация UML-модели [1] для вновь разрабатываемой программной системы, так и для используемых ею библиотек существенно упрощает понимание их структуры и функциональности, выбор необходимой версии и разработчика библиотеки. Объем информации получаемой при решении этих задач может быть слишком велик для их восприятия человеком, а получение и визуализация всех связей может требовать чрезмерно большого времени. Поэтому визуализация программной системы необходима лишь для наиболее существенной ее части - архитектуры. В предыдущих работах автора особо рассматривались способы визуализации архитектуры системы [2] и визуализации результатов измерения качества программной системы с помощью объектно-ориентированных метрик [3]. В работе [4] сделан обзор и анализ объектно-ориентированных метрик. Рассмотрены простейшие объектно-ориентированные метрики для анализа проектирования отдельных классов. Затем рассмотрены метрики связанности класса, позволяющие оценить качество проектирования структуры класса. В работе [5] рассмотрены способы визуализации программного обеспечения в трехмерном пространстве.

В данной работе рассматривается возможности инструмента для визуализации и анализа архитектуры программного обеспечения в инструменте восстановления архитектуры [6] на основе визуализации объектно-ориентированных метрик в трехмерном пространстве. Данная статья написана на основе дипломной работы, выполненной одним из авторов на отделении дополнительного образования факультета ВМК МГУ им. Ломоносова.

II. Визуализация программного обеспечения в трехмерном пространстве

Все предложенные решения, предлагаемые инструментами визуализации программного обеспечения, на сегодняшний день могут быть условно разбиты на группы по следующим способам представления программы в трехмерном пространстве [7]:

- графы;
- деревья;
- геометрические формы.

Далее рассматриваются инструменты, использующие каждую из перечисленных абстракций.

A. Представление программы в виде графа в трехмерном пространстве

Граф представляет собой набор узлов (вершин) и ребер, где вершинами представлены классификаторы (классы, интерфейсы), а ребрами связи между ними [7]. В простейшем случаем отображаемые элементы в трехмерном пространстве могут быть представлены диаграммой классов языка UML [8].

Рассмотри два инструмента для этого способа визуализации:

- Walrus - Graph Visualization Tool [9];
- GEF3D GEF [10];

Walrus - Graph Visualization Tool представляет собой инструмент для отображения информации в виде графа в трехмерном пространстве [9]. Позволяет отображать графы, содержащие миллионы узлов. При большом количестве узлов и связей между ними изображения узлов могут перекрываться, поэтому рекомендуется использовать решения для количества узлов до тысячи с небольшим количеством связей.

Владимир Юрьевич Романов, факультет ВМК МГУ им. М.В. Ломоносова, vromanov@cs.msu.su, vladimir.romanov@gmail.com. И.В. Шульга – выпускник программы РКТ факультета ВМК МГУ имени М.В.Ломоносова. (email: shulganew@gmail.com)

Реализация сделана на языке Java и использует библиотеку-связку графического представления Java3D, поддерживающую библиотеку трехмерной графики OpenGL[11]. Walrus является самостоятельной программой, не поддерживает API и не может быть встроена как самостоятельный модуль в интегрированные среды разработки.

На рисунке 1 приведен пример отображения программы в виде трехмерного графа программой Walrus.

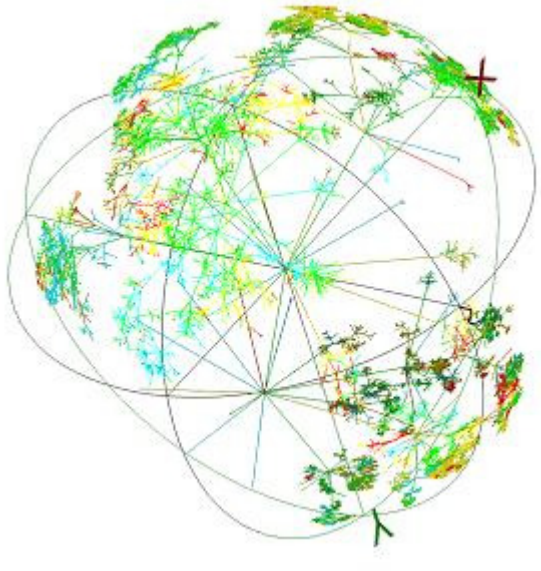

Рис.1. Отображение программы в виде графа.

Такой способ визуализации программы хорошо подходит для обзора структуры программы при задании необходимых фильтров, позволяющих визуализацию программы по частям. Однако для визуализации метрик программы возможно использовать лишь цвет узлов и ребер, чего явно недостаточно. Также затруднена навигация в трехмерном пространстве при большом количестве узлов. Поскольку данная реализация не поддерживает программных интерфейсов для использования внешними программами, а также не может быть встроено в среду разработки Eclipse.

Eclipse Graphical Editing Framework 3D - представляет собой набор инструментов [10] для работы с диаграммами языка UML. Разработан организацией Eclipse Foundation для среды разработки Eclipse. Инструмент устанавливается как плагин в среду Eclipse и позволяет выполнять прямое и обратное проектирование. Визуализация выполняется для UML модели, которая разворачивается в трехмерном пространстве в виде множества взаимосвязанных графов-проекций на плоскость. На рисунке 2 представлен пример работы с плагином GEF3D для среды Eclipse.

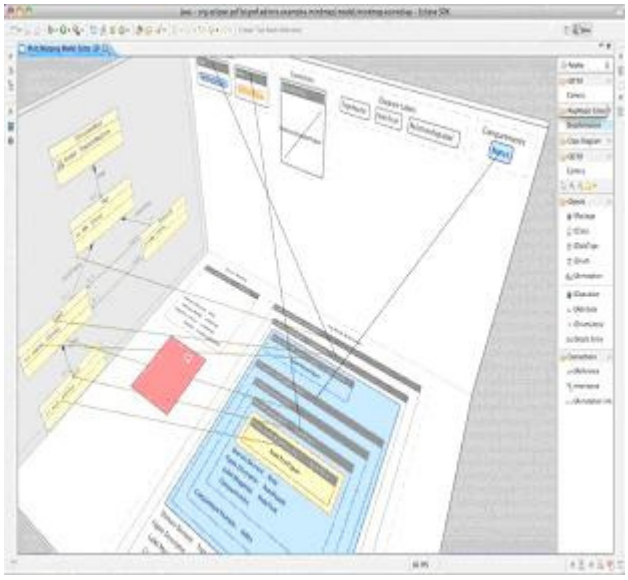

Рис.2. Визуализация программы инструментом GEF3D.

В трехмерном пространстве располагаются связи между отдельными элементами диаграмм, расположенных на одной из трех проекций.

Плагин использует библиотеки JOGL [12, 13] и LWJGL [14] для представления диаграмм в трехмерном пространстве. Отображаемая диаграмма визуально дает дополнительные возможности для анализа и проектирования программы, чем стандартное двумерное преставления большинства CASE инструментов. Важно отметать, что и обычное двумерное представления UML-диаграммы также поддерживается инструментом.

Объем данных, представленных на UML диаграмме остается по-прежнему велик для анализа. Изображаемые в трехмерном пространстве связи между элементами диаграммам препятствуют визуализации метрик.

B. Представление программы в виде дерева в трехмерном пространстве

В виде дерева может быть представлены множество классификаторов объектно-ориентированного языка связанных отношениями между собой. Важной особенность дерева является отсутствие циклов, в отличие от графов, что упрощает расположение узлов в пространстве и делает наглядным представление наследования.

Представления в виде дерева может быть выполнено следующими способами:

- соединенные между собой узлы дерева;
- отображение содержимого;
- отображения связей;
- комбинированное отображение.

Xerox PARC's Information Visualizer Cone Tree [15] является примером реализации отображения соединенных между собой узлов. При этом способе отображения не раскрывается структура самого узла. Визуализация программы как трехмерного дерева может быть удобна для оценки дерева наследования классификаторов. Пример отображения дерева наследования приведен на рисунке 3.

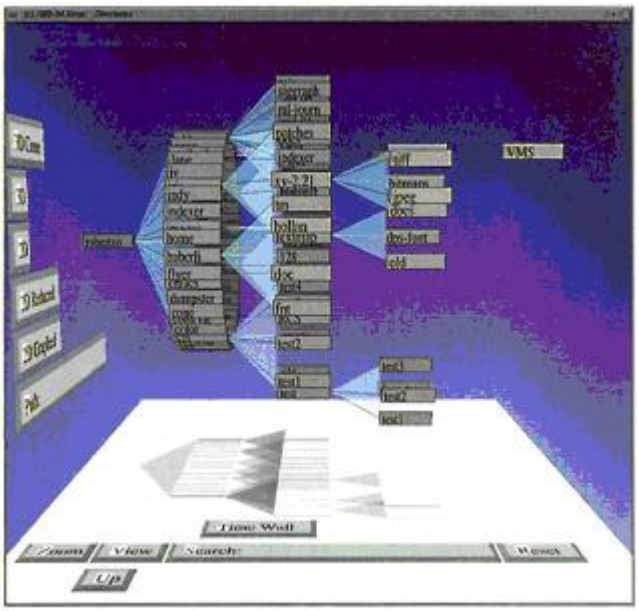

Рис.3. Трехмерное дерево наследования классификаторов.

В отличие от предыдущего метода визуализации, в трехмерном пространстве располагаются не только связи между элементами дерева, но и сами элементы. Однако для анализа и визуализации метрик такое трехмерное представление не подходит. Дополнительное третье измерение также уже занято.

Circle Data Packing - система, позволяющая визуализировать содержимое узлов дерева [16]. Система формирует ландшафт, на котором располагаются граничные узлы, прорисовывается их содержимое. Пример такого ландшафта приведен на рисунке 4.

Отображаемая в виде ландшафта информация является достаточно наглядной. У пользователя программы возникают знакомые ассоциации с тем, что он уже видел в реальной жизни. Однако сложность в отслеживании связей между узлами существенно уменьшает объем анализируемых метрик. Инструмент визуализации программы в виде природного ландшафта не получил большого распространения.

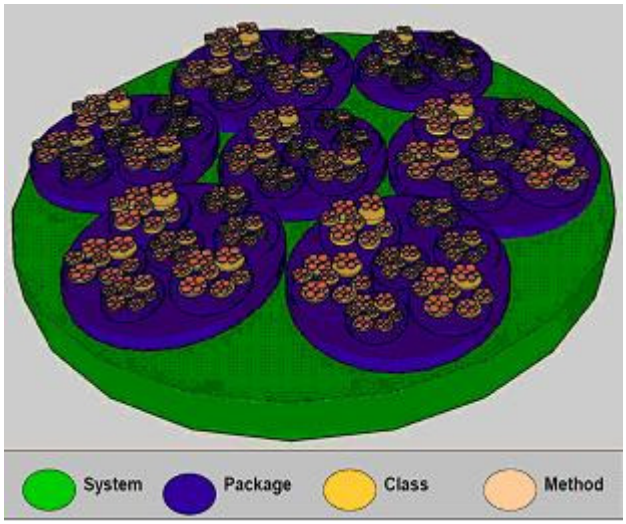

Рис.4. Визуализация программы в виде природного ландшафта.

Vizz3d. Этот инструмент [14] предоставляет дополнительные возможности для визуализации и анализа связей между элементами программы. Дополнительные возможности навигации пользователя инструмента в трехмерном пространстве позволяют детально рассмотреть отношения связи между узлами. Пример визуализации отношений между элементами программы приведен на рисунке 5.

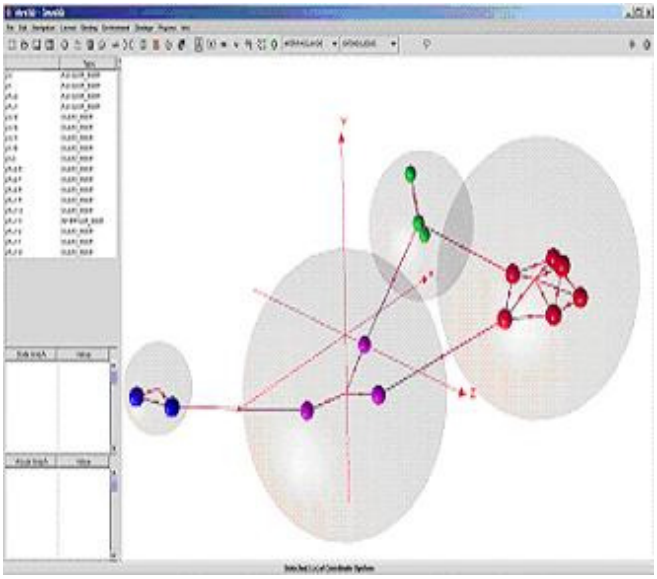

Рис.5. Визуализация отношений в трехмерном пространстве.

Hierarchical Net 3D - является примером комбинированного подхода отображения элментов программы и их связей как деревьев ландшафта в трехмерном пространстве [17]. Такой подход позволяет отображать, как наследственные связи узлов дерева - классов программы, так и раскрывать их содержимое. Пример такого отображения приведен на рисунке 6.

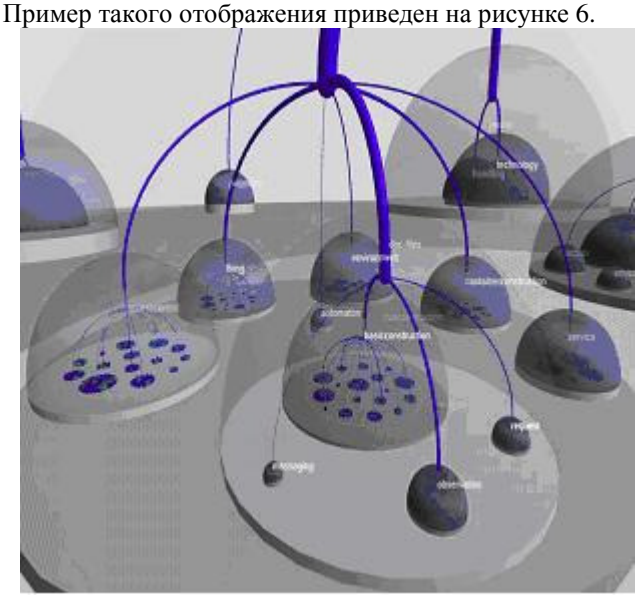

Рис.6. Программа - ландшафт с отображением отношений.

Инструмент позволяет изменять уровень детализации отображаемых узлов программы, а с помощью элементов навигации позволяет перемещаться в трехмерном пространстве во всей отображаемой структуре программы. Недостатком данного решения

является отсутствие интеграции с какой либо средой разработки программ.

Все способы отображения дерева, как и отображения графа, имеют зачастую непонятную для пользователя ориентацию в пространстве. Фактически ориентация дерева или графа в трехмерном пространстве определена только алгоритмом его отображения. Поэтому для анализа метрик программы в трехмерном пространстве следует рассмотреть другие подходы.

C. Представление программы в виде города в трехмерном пространстве

Способ визуализации в виде города позволяет пользователю более привычным образом ориентироваться в трехмерном пространстве [7]. Код программы представлен в виде аналогии с реальным городом. Классы и интерфейсы языка Java в соответствии с этой аналогией представлены в виде домов, при этом выделенные районы этого города представляют собой пакеты. При таком подходе отображаемая информация и процесс навигации по городу лучше и привычнее воспринимается пользователем.

Кроме того, важно отметать, что при отображении программы наглядно отображается не только структура классов и пакетом, но также могут быть визуализированы различные метрики, что дает еще более достоверный анализ отображаемой программы. В большинстве случаев для представления в виде города используются метрики CS, NOM и NOA. Как правило, отображаемая метрика влияет на ширину и/или высоту отображаемого здания, что позволяет визуально оценить высокие или широкие здания - возможный результат неправильного проектирования пакетов или классов (избыточная ответственность).

Наиболее распространенным инструментом для отображения объектно-ориентированных метрик на сегодняшний день является CodeCity [18], разработанный компанией REVEAL group. На рисунке 7 представлен пример визуализации программы CodeCity.

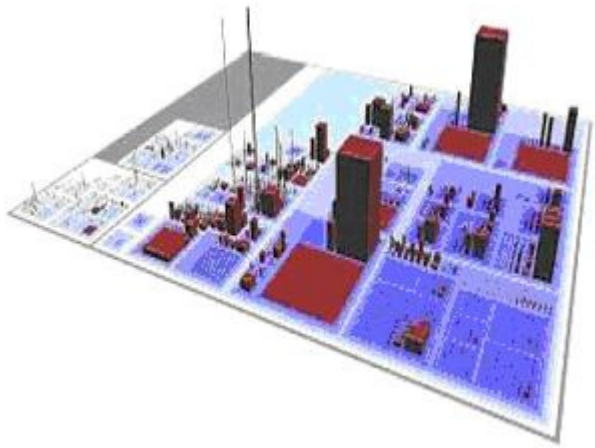

Рис.7. Визуализация программы в виде города.

Классы в программе отображаются красным цветом, размер класса определяется метриками NOM и NOA, которые определяют ширину и длину здания. Классы размещаются в пакетах, которые отображаются синим цветом.

Интерфейсы отображаются пиками темно-синего цвета, высота может определяться количеством методов, заданных интерфейсом.

Для определения вложенности элементов используются различные уровни рельефа, что наглядно позволяет визуализировать вложенность пакетов.

На рисунке 8 представлен пример такой визуализации.

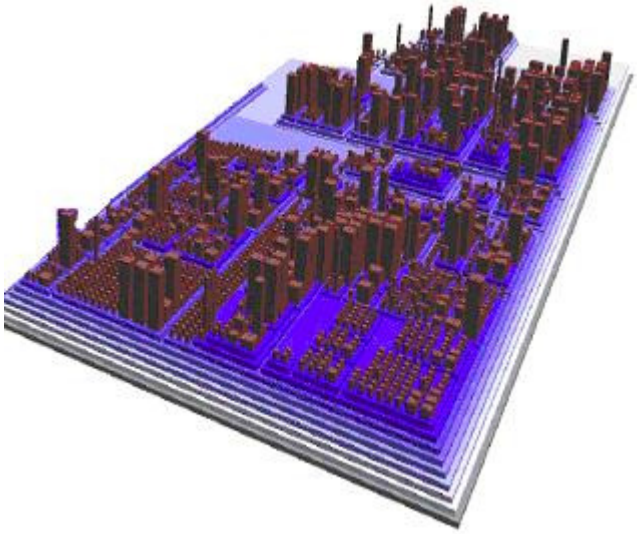

Рис.8. Визуализация вложенности пакетов как рельефа.

Программа поддерживает возможность выбора определенных элементов для дальнейшей работы с пакетом или классом. Это осуществляется как непосредственным выбором элементов, так и возможностью использовать различные фильтры для поиска определенных элементов. На рисунке 9 светло-зеленым цветом показан пример выделения в городе квартала-пакета.

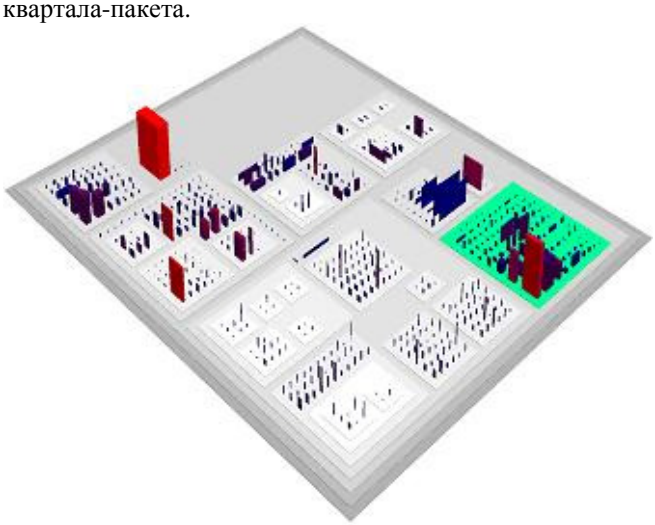

Рис.9. Пример выделения пакета в городе-программе.

Визуализация программы в виде города позволяет визуализировать дефекты проектирования. Класс, обладающий определенным сочетанием значений метрика, обладает дефектом и показывается на карте города определенным цветом. На рисунке 10 показан пример такой раскраски дефектов проектирования в Java JDK.

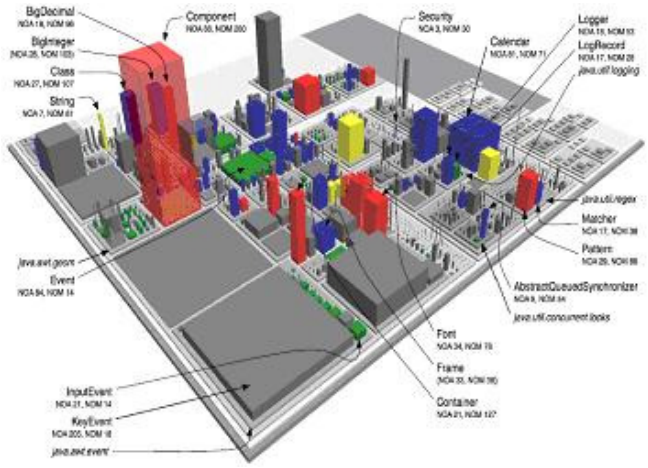

Рис.10. Пример выделения пакета

III. Реализация Инструмента визуализации структуры и метрик в трехмерном пространстве

A. Анализируемые метрики

В рассматриваемом примере визуализации в качестве объектно-ориентированных метрик[4] используются:

- KLOC - Lines of Class in Kilobyte
- NOA - Number of Attributes
- NOM - Number Of Methods

В зависимости от максимального значения метрики в рассматриваемом пакете определяется максимальная высота здания. Остальные отображаемые элементы визуализируются относительно отображаемого элемента.

Для определения метрики KLOC используется класс JarEntry, входящий в пакет java.util. Для данного класса доступны методы getSize(), возвращающий размер входящих в архив несжатых данных. С помощью этого метода вычисляется размер данных выбранного пакета. Метрики NOA и NOM определяются стандартными методами пакета org.eclipse.uml2 - getAllOperations().size()и getAllOperations().size().

B. Визуализация структуры программы

Для визуализации объектно-ориентированных метрик в трехмерном пространстве было выбрано представление программы в виде города. Данное представление наиболее привычно для пользователя для ориентации в трехмерном пространстве.

Инструмент визуализации программы в трехмерном пространстве является одним из видов в инструменте восстановления архитектуры программ написанных на языке Java [6]. Инструмент восстановления архитектуры реализован как расширение среды Eclipse. В качестве исходных данных для визуализации архитектуры могут использоваться jar-файлы используемых в проекте Eclipse библиотек, двоичные файлы проекта, получаемые в результате трансляции, или абстрактное синтаксическое дерево программы проекта, получаемое в результате трансляции. В результате распознавания исходных данных строится UML-программы, которая затем визуализируется как множество проекций (видов) этой модели в окнах (views) среды Eclipse. Среди множества таких видов [6] существует вид - взгляд на трехмерный город-программу сверху ("вид сверху"). На рисунке 11 изображен такой "вид сверху" на пакет JFace, широко используемый разработчиками в среде Eclipse для построения интерфейса пользователя.

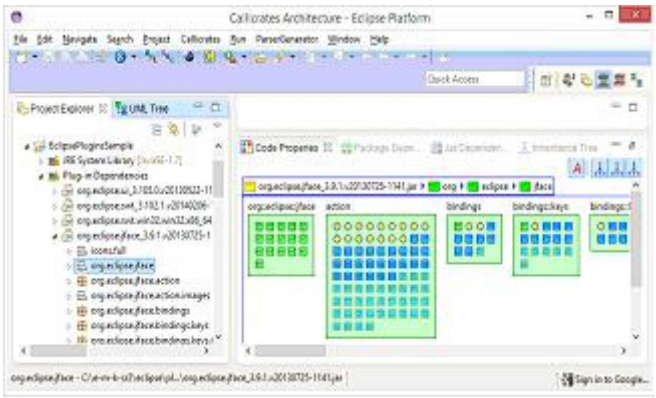

Рис.11. Вид сверху на пакет JFace среды Eclipse.

В виде зеленых конвертов показываются пакеты языка Java, в виде желтых окружностей показываются интерфейсы, синими квадратами классы. Публичные классы имеют более темную окраску. С помощью инструментов в панели этого вида можно расцвечивать классы и интерфейсы в корне/листьях/середине иерархии наследования, или обладающие каким либо свойством, например, абстрактностью.

"Вид Сверху" используется в инструменте восстановления архитектуры как самостоятельно, так и в сочетании с трехмерным изображением для ускоренного перемещения по большим пакетам со сложной структурой (навигации). Этот вид используется как фундамент для построения трехмерного изображения добавлением третьего измерения.

На рисунке 12 показано изображение трехмерного вида со зданиями-классами программы. На крышах этих зданий показаны изображения аналогичные изображениям, показанным на "виде сверху".

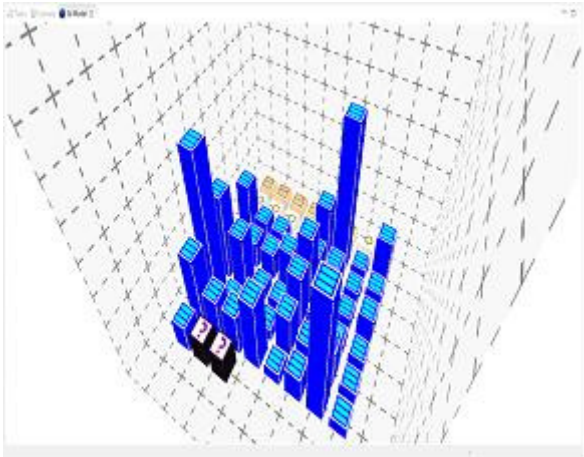

Рис.12. Трехмерное изображение пакетов, интерфейсов и классов языка Java.

Трехмерное изображение строится из прямоугольных примитивов, из которых конструируются стены зданий

города. Размер здания города определяется через одну из выбранных пользователем метрик класса и других элементов программы.

C. Навигация пользователя в программе-городе

Для реализации навигации пользователя в построенном изображении города используются методы библиотеки Lightweight Java Game Library (LWJGL)[14]. Для навигации используются два метода библиотеки:

- void glRotated(double angle, double x, double y, double z)
- void glTranslatef(float x, float y, float z)

Метод glRotated предназначен для поворота по вокруг вектора x, y, z на угол angle, а метод glTranslatef смещает начало координат в указанную точку.

Таким образом, используя эти два метода, можно отобразить из любой точки пространства (метод glTranslatef) проекцию под любым углом (метод glRotated). Для расчета смещения координат и вычисления угла используются формулы тригонометрии, общая схема приведена на рисунке 13.

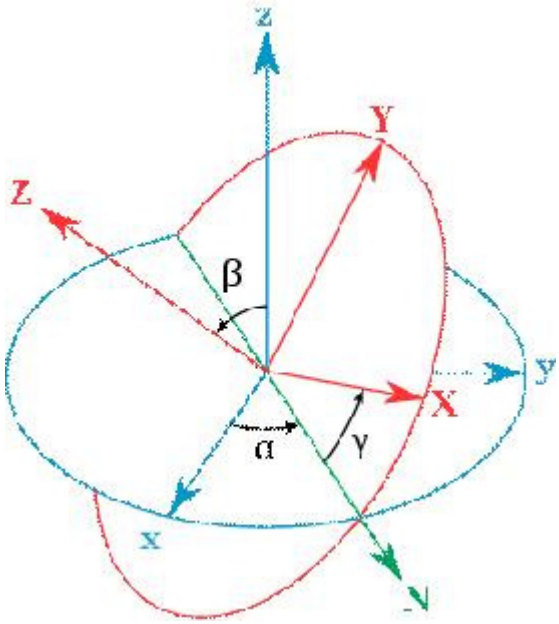

Рис.13. Навигация пользователя в трехмерном пространстве.

Для управления навигацией в городе инструмент содержит специальный вид среды Eclipse, в котором, помимо панели для "вида сверху", присутствуют кнопки навигации пользователя инструмента в трехмерном пространстве. Это показано на рисунке 14.

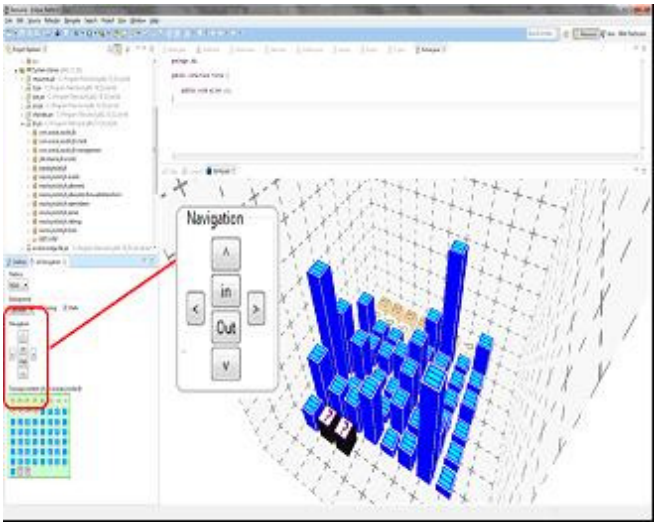

Рис.14. Кнопки панели навигации пользователя в программе-городе.

Ниже приводятся условные обозначения для зданий города.

Условное обозначение	Значение
	**артефакт (файл)**
	**класс**
	**элемент расположен во внешнем jar-файле, а не в том jar-файле который использовался при построении UML-модели.**
	**вид элемента (класс, интерфейс, перечисление) неизвестен**
	**интерфейс**
	**пакет**
	**примитивный тип данных программы**
	**не определен**

Рис.15. Условные обозначения для зданий города-программы

В трехмерной визуализации достаточно сложно показать названия отображаемых классов, поскольку представление полного квалифицированного имени в трехмерном пространстве для каждого класса достаточно громоздко. Поэтому для просмотра полного имени класса удобнее использовать двумерное представление "вид сверху", как показано на рисунке 16.

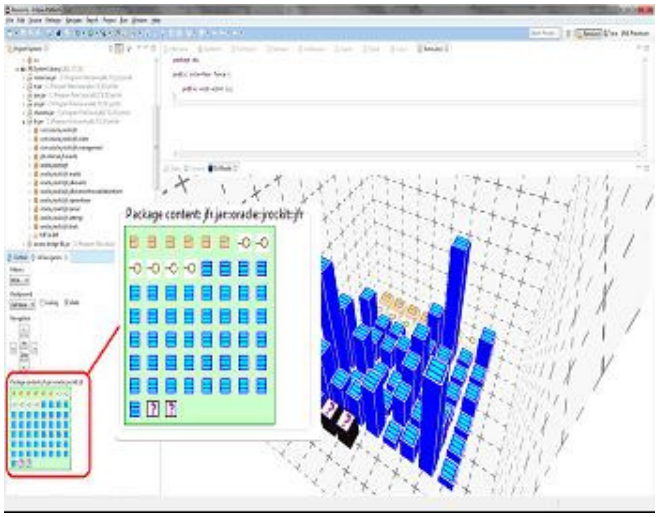

Рис.16. "Вид сверху" в панели управления.

Имя элемента программы показывается на этом виде как всплывающая подсказка. Это показано на рисунке 17.

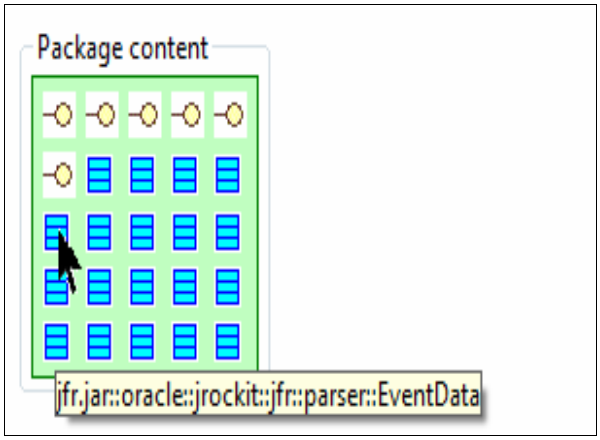

Рис.17. Полное квалифицированное имя элемента программы.

Отображение полного квалифицированного имени пакета в трехмерном пространстве выбранного для рассмотрения пакета языка Java показывается всплывающей подсказкой при наведении на него указателя мыши.

II. Заключение

В статье рассмотрены вопросы анализа и визуализации зависимостей между классами в пакетах программных систем написанных на языке Java. Данная задача весьма актуальна для восстановления архитектуры программной системы при решении задачи обратного проектирования. В статье рассмотрены методы визуализации таких зависимостей с помощью матричного представление графа, описывающего связи между классами внутри анализируемого пакета, а также связи между классами разных пакетов. Показано использование такого метода визуализации зависимостей между пакетами в инструменте обратного проектирования и восстановления архитектуры. Данный инструмент основан на языке моделирования UML и реализован как расширение среды Eclipse. Статья является продолжением цикла публикаций по программной инженерии и применению языка моделирования UML, начатой в журнале INJOIT работами [2, 3, 4, 5, 6, 19, 20, 21, 22, 23]. Эта работа относится к числу одного из направлений исследований в Лаборатории ОИТ факультета ВМК МГУ [24, 25].

Библиография

[1] Object Management Group, UML 2.4 Superstructure Specification, OMG document. http://www.omg.org/spec/UML/2.4.1/
[2] Романов В. Ю. Визуализация для измерения и рефакторинга программного обеспечения //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 9. – С. 1-10.
[3] Романов В.Ю. Визуализация программных метрик при описании архитектуры программного обеспечения //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 2. – С. 21-28.
[4] Романов В.Ю. Анализ объектно-ориентированных метрик для проектирования архитектуры программного обеспечения//International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 3. – С. 1117.
[5] Романов В. Ю. Визуализация и анализ больших программных систем с помощью их трехмерного представления //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 5. – С. 1-9.
[6] Романов В.Ю. Анализ и визуализация зависимостей между пакетами программных систем //International Journal of Open Information Technologies. – 2015. – Т. 3. – №. 1. – С. 23-29.
[7] Alfredo T., Marcelo C. An Overview of 3D Software Visualization // Visualization and Computer Graphics, IEEE Transactions on Volume:15, Issue: 1
[8] T. Dwyer, "Three dimensional uml using force directed layout," in APV is '01: Proceedings of the 2001 Asia-Pacific Symposium on Information visualisation. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2001, pp. 77–85.
[9] Walrus - Graph Visualization Tool // http://www.caida.org/
[10] GEF3D http://eclipse.org/gef3d/
[11] Mark Segal, Kurt Akeley. The OpenGLR Graphics System: A Specification Version 3.3 (Core Profile) - March 11, 2010
[12] Java™ Binding for the OpenGL® API. http://jogamp.org/jogl/www/
[13] OpenGL // https://ru.wikipedia.org/wiki/OpenGL
[14] Lightweight Java Game Library. http://www.lwjgl.org/
[15] Steven A. Information Visualization at the Turn of the Century // AALL Spectrum October 2000
[16] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of large hierarchical data by circle packing," in CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM Press, 2006, pp. 517–520.
[17] Balzer and O. Deussen, "Hierarchy based 3d visualization of largesoftwarestructures,"inVIS'04:ProceedingsoftheConference on Visualization '04. Washington, DC, USA: IEEE Computer Society, 2004, p. 598.4.
[18] R.Wettel, M.Lanza, R.Robbes. Software Systems as Cities: A Controlled Experiment. In Proceedings of ICSE 2011 (33rd International Conference on Software Engineering), pp. 551 - 560, ACM Press, 2011.
[19] Романов В.Ю. Инструмент обратного проектирования и рефакторинга программного обеспечения написанного на языке Java //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 8. – С. 1-6.
[20] Романов В.Ю. Моделирование свободно-распространяемого программного обеспечения с помощью языка UML //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 7. – С. 11-15.
[21] Романов В.Ю. Моделирование и верификация архитектуры программного обеспечения разработанного на языке Java. Сб. трудов VIII Международной конференции «Современные информационные технологии и ИТ-образование», Москва, 2013, с. 343-348
[22] Романов В.Ю. Использование шаблонов пакетов для анализа архитектуры программной системы//International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 4. – С. 18-24.
[23] Романов В.Ю. Визуализация внутренней структуры и зависимостей пакетов в программной системе //International Journal of Open Information Technologies. – 2015. – Т. 3. – №. 4. – С. 18-26.
[24] Намиот Д., Сухомлин В. О проектах лаборатории ОИТ //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 5. – С. 18-21.

[25] Гурьев Д. Е., Намиот Д. Е., Шнепс М. А. О телекоммуникационных сервисах //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 4. – С. 13-17.

The case tool for software structure and metrics visualization in 3D space

Romanov V.Y., Shulga I.V.

Abstract — **This article discusses software code visualization systems using 3D space. We consider the software visualization in 3D space as part of architecture recovery tool. The metaphor a software system as a code city with buildings and districts are used to visualize software structure and object oriented metrics.**

Keywords — **software visualization, reengineering, reverse engineering, architecture recovery.**