Научная визуализация, 2018, том 10, номер 5, страницы 123 - 139, DOI: 10.26583/sv.10.5.08

3D визуализация архитектуры и метрик программного обеспечения

Д.Е. Намиот[1], В.Ю. Романов[2]

Московский Государственный Университет имени М.В. Ломоносова, Россия
[1] ORCID: 0000-0002-4463-1678, dnamiot@gmail.com
[2] ORCID: 0000-0001-5140-9576, vladimir.romanov@gmail.com

Аннотация

В статье приводится обзор методов трехмерной визуализации метрик программного обеспечения. Метрики для программ (пакетов, классов, репозиториев) есть одно из наиболее активных в использовании направлений в программной инженерии. Это направление исследований, которое относится к анализу программного обеспечения, и визуальный анализ здесь является одним из наиболее часто используемых инструментов. Такого рода визуализация является, обычно, частью процесса анализа качества программного обеспечения, может использоваться при обучении, при рефакторинге программ, а также при интегрировании (объединении) отдельных компонент (пакетов) в сложные программные комплексы. Очевидно, что визуализация облегчает и ускоряет процесс понимания структуры программных компонент. Это становится все более и более актуальным, поскольку сейчас многие программные компоненты с открытым кодом (то, что наиболее часто интегрируется в другие системы), например, представляют собой большие и достаточно сложные программные комплексы. Соответственно, их интеграция в новый проект представляет собой весьма непростую задачу. Заметим, что задача интеграции становится еще более сложной, если нет доступа к исходным текстам интегрируемых компонент. В этом случае визуальное представление метрик есть, по сути, основной элемент анализа. Анализ сторонних компонент является не единственной областью применения. Точно такие же проблемы возникают и в корпоративной разработке, когда над большим проектом работают отдельные группы, которые, к тому же, могут часто меняться по составу исполнителей. В данной работе рассматриваются, например, методы визуализации и анализа структуры программы в 3D пространстве, которые основаны на метафоре представления программной компоненты как города, который состоит из отдельных зданий, объединяющихся в районы и т.д. Также мы рассматриваем вопросы использования виртуальной реальности для представления метрик программного обеспечения.

Ключевые слова: программная инженерия, метрики программного обеспечения, 3D визуализация, анализ программного обеспечения.

1. Введение

Данная статья является продолжением нашей работы [1], в которой рассматривались разные модели визуализации для программных метрик. Мы отмечали, что трехмерные модели визуализации, в силу их объема, требуют отдельного рассмотрения. К тому же, в последнее время такое направление, как виртуальная реальность, находит свое применение и в анализе программного обеспечения [2]. Это также может быть отнесено к трехмерной визуализации.

Метрики для программных систем стандартизованные (в данном случае – это де-факто стандарты, оговоренные и признаваемые участниками) измерения для программных компонент. Эти измерения есть количественные показатели для программных модулей, которые мо-

гут использоваться для сравнительного анализа [3]. Именно количественные показатели служат базой для сравнительного анализа. Таких показателей может быть много, они могут быть связаны довольно сложными отношениями. Соответственно, визуальный анализ выступает здесь как инструмент, который призван упростить понимание системы показателей и их взаимосвязей [4]. Идеи использования количественных метрик (оценок) для программного обеспечения активно продвигаются многими производителями. Метрики используются для планирования, оценки стоимости и производительности. Нормированные значения метрик используются как показатели качества.

В данной статье приводится обзор методов визуализации программных метрик (методов визуального анализа архитектуры программных компонент) в трехмерном пространстве (рис. 1).

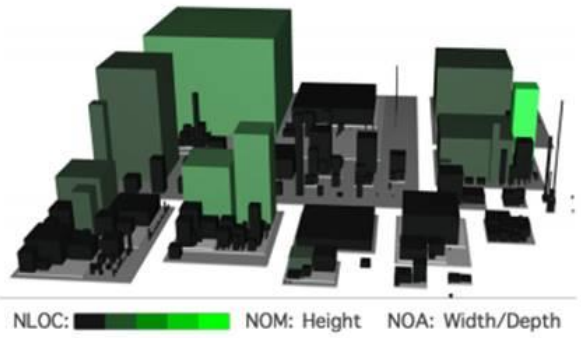

Рис. 1. Строки кода и методы в трехмерном представлении [5]

Тема трехмерной визуализации программных метрик сравнительно мало представлена в отечественной литературе по программной инженерии. Мы можем указать на две работы, выполненные в Лаборатории Открытых Информационных технологий факультета ВМК МГУ имени М.В. Ломоносова [6,7]. Можно также упомянуть обзор [8] и монографию [9], которые описывают визуализацию графовых моделей. Здесь вопросы трехмерной визуализации будут рассмотрены более подробно.

2. 3D визуализация для программного обеспечения: графы и деревья

Исходя из анализа имеющихся работ [7,10], способы представления трехмерной визуализации метрик (атрибутов) программных компонент могут быть классифицированы по следующим группам: графы (деревья) и геометрические формы.

Граф представляет собой набор узлов (вершин) и ребер, где вершинами представлены классификаторы (классы, интерфейсы), а ребрами связи между ними [7]. То же самое относится и к деревьям, за исключением того, что в классическом определении, дерево не имеет циклов (связный граф без циклов). Для программной компоненты графом может быть, например, UML-диаграмма.

Хороший обзор средств 3D визуализации графов представлен, например, в работе [11]. В числе конкретных инструментов, использовавшихся нами в ряде проектов, можно назвать CGV [12], Walrus [11], GEF 3D [14]

CGV представляет собой интерактивный инструмент для визуализации графов, включая 3D модели (рис. 2).

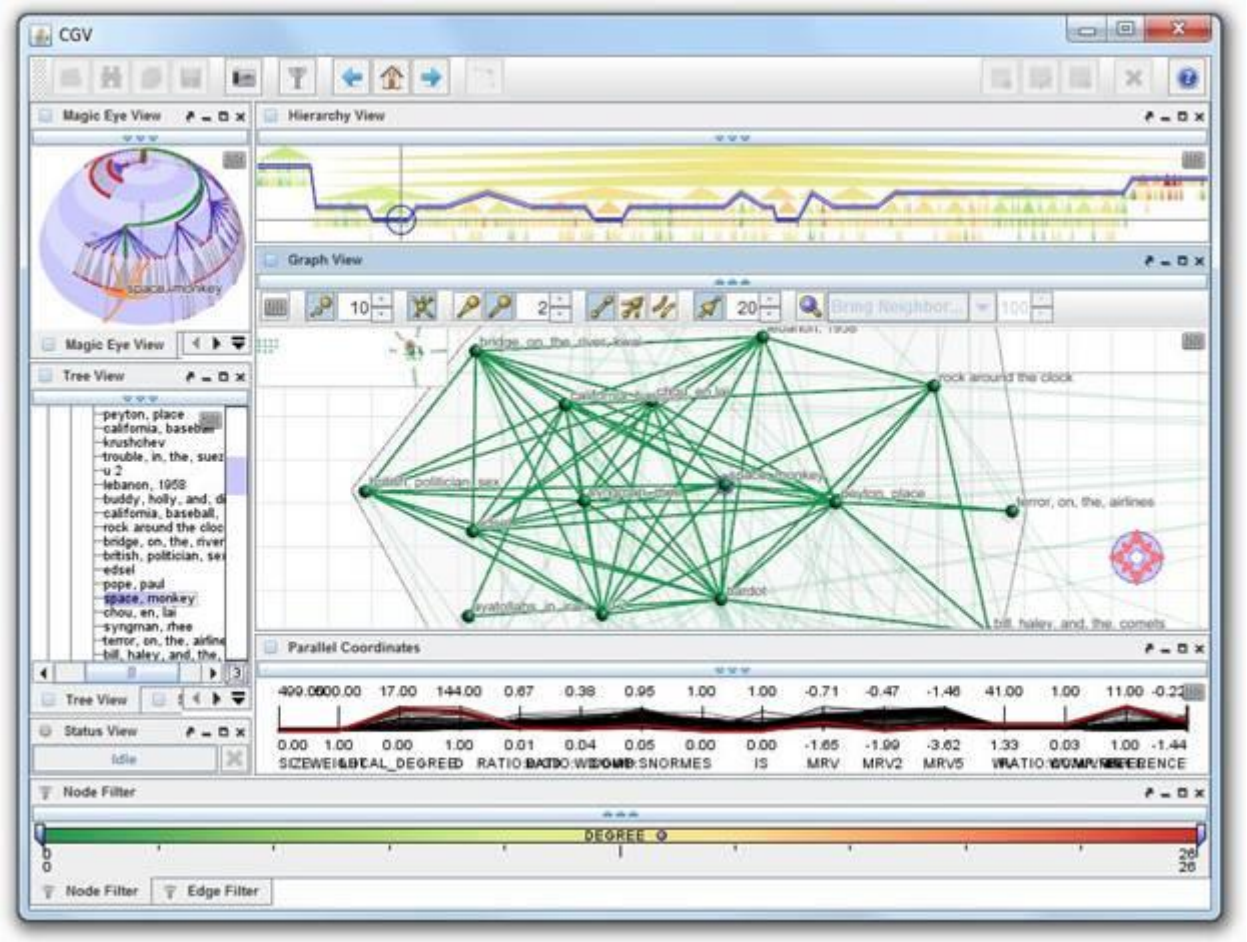

Рис. 2. CGV studio [15].

Walrus позволяет отображать графы, содержащие миллионы узлов. Реализован на языке Java, использует библиотеку-связку графического представления Java3D, поддерживающую библиотеку трехмерной графики OpenGL [16]. Это обеспечивает хорошую переносимость системы. Но необходимо отметить, что Walrus является самостоятельной программой, не поддерживает API и, соответственно, система не может быть встроена как самостоятельный модуль в интегрированные среды разработки. Для визуализации метрик программы можно использовать разный цвет узлов и ребер, хотя для больших систем этого может быть недостаточно. Имеющаяся система фильтров позволяет визуализировать граф по частям.

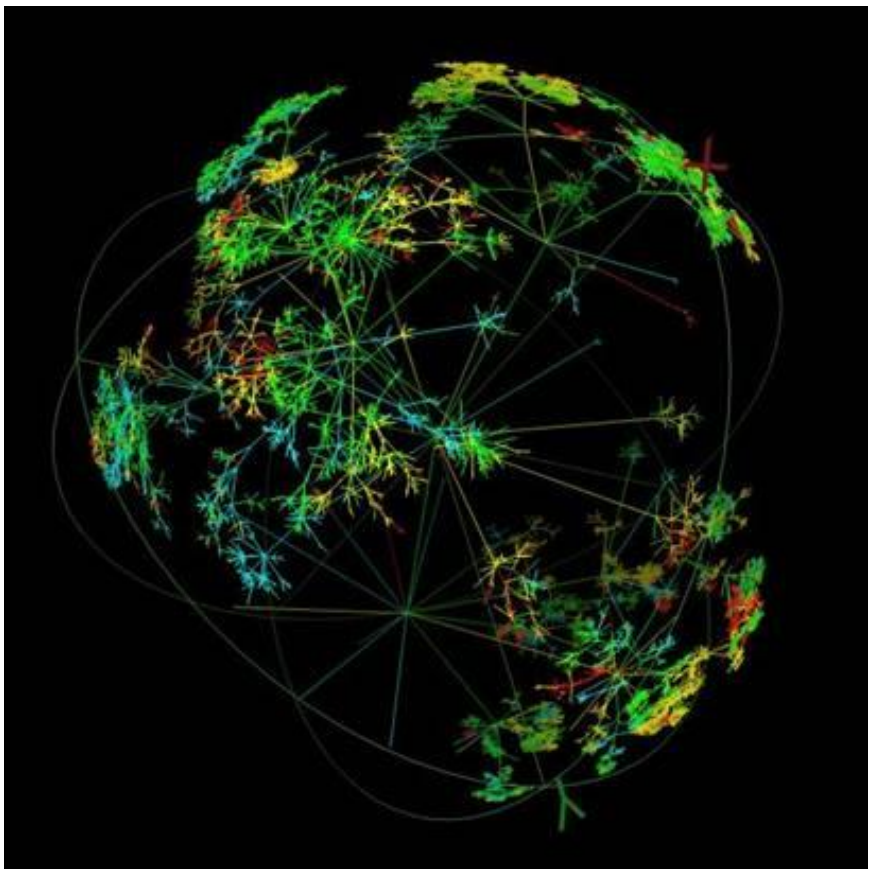

Рис. 3. Визуализация в Walrus

GEF (Graphical Editing Framework) 3D - представляет собой набор инструментов [17], разработанный организацией Eclipse Foundation для работы с диаграммами языка UML в среде разработки Eclipse. GEF 3D устанавливается как плагин в среду Eclipse и позволяет выполнять прямое и обратное проектирование.

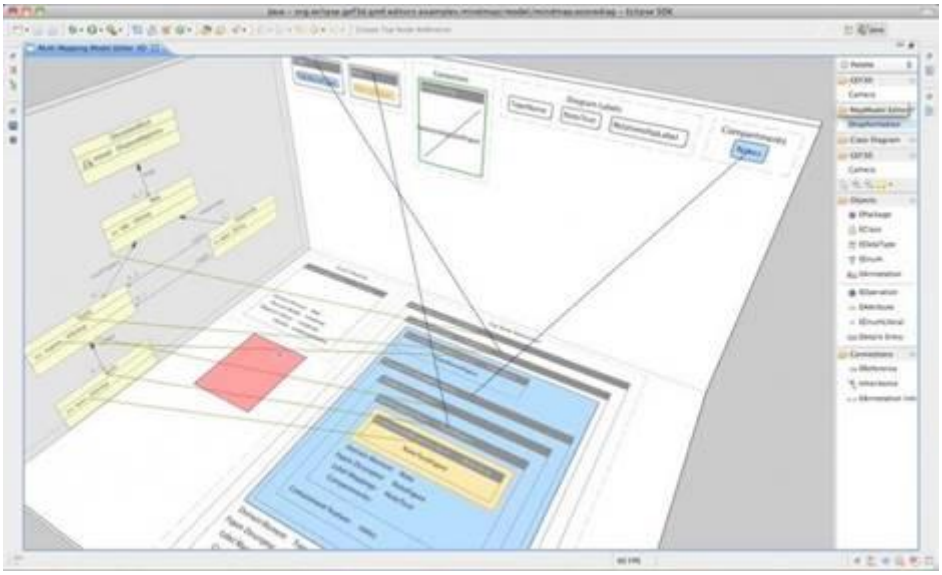

Рис.4. Визуализация программы в GEF3D.

Визуализация выполняется для UML модели, которая разворачивается в трехмерном пространстве в виде множества взаимосвязанных графов-проекций на плоскость [7]. На рисунке 4 представлен пример работы с плагином GEF 3D для среды Eclipse. А на рисунке 5 – диаграммы активности, классов и зависимости между ними.

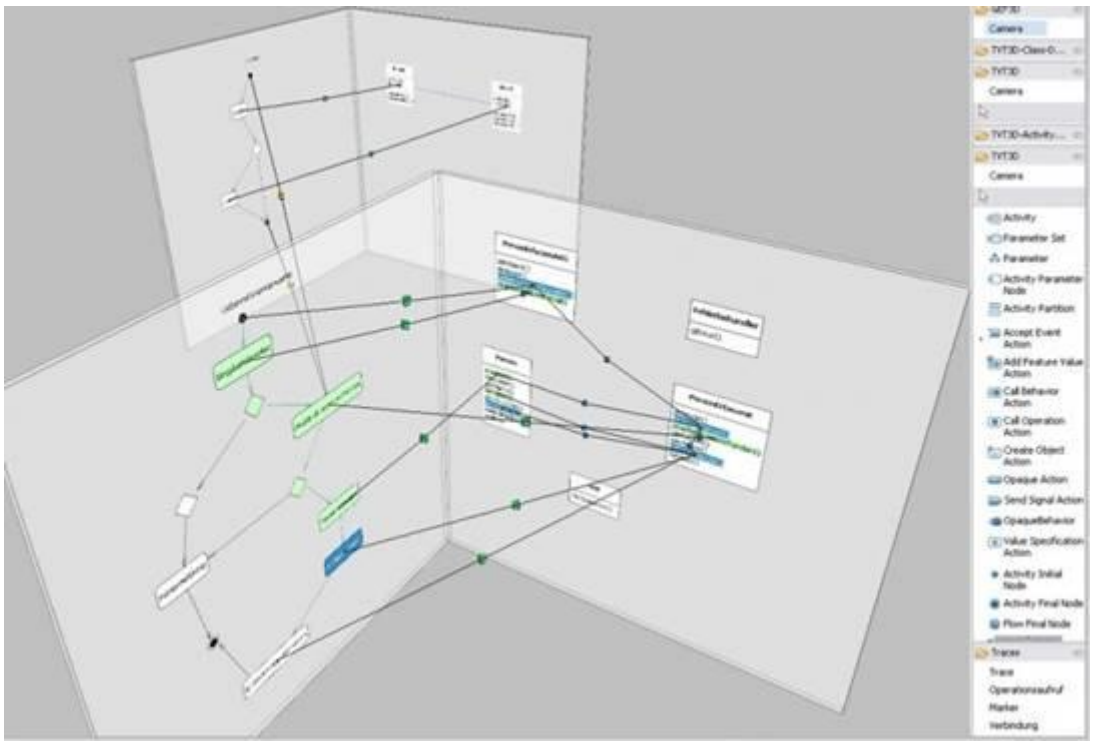

Рис. 5. UML в 3D представлении.

В трехмерном пространстве располагаются связи между отдельными элементами диаграмм, расположенные на одной из трех проекций. Отображаемая диаграмма визуально дает дополнительные возможности для анализа и проектирования программы, чем стандартное двумерное преставление большинства CASE инструментов. Важно отметить, что и обычное двумерное представление UML-диаграммы также поддерживается инструментами.

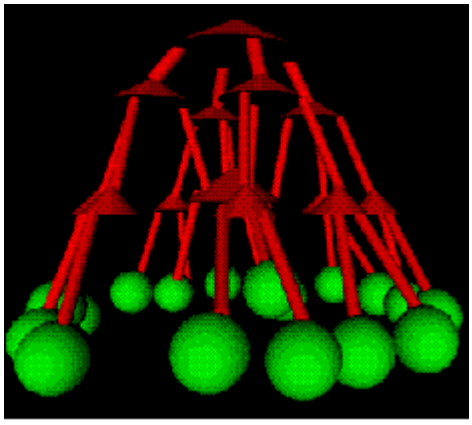

Рис. 6. 3D визуализация дерева [18]

В виде дерева может быть представлено множество классификаторов объектно-ориентированного языка,связанных отношениями между собой. Важной особенностью дерева (в отличие от графов) является отсутствие циклов, что упрощает расположение узлов в пространстве и делает наглядным представление наследования. Последнее, естественно, является важным для объектно-ориентированных систем. Типичный пример 3D визуализации дерева приведен на рисунке 6. Здесь представлена модель соединенных между собой узлов.

Примером системы для отображения соединенных узлов является PARC Information Visualizer Cone Tree [19]. Информация о самом узле (его структура) не раскрывается. Типичное использование – отображение дерева наследования (рис. 7).

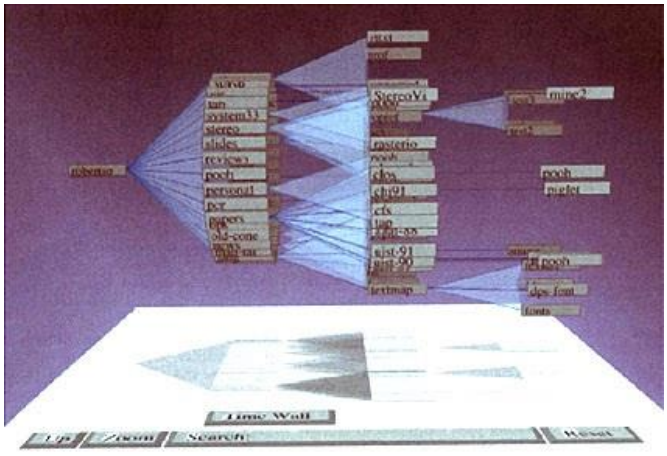

Рис. 7. Дерево наследования [20].

Вместе с тем, для анализа и визуализации метрик необходимо уметь “раскрывать” узлы дерева. Здесь может быть использована модель Circle Packing [21]. Там отображаются разноцветные круги, возможно вложенные. Элементы одного типа окрашены одинаковым цветом. Размер круга передает его характеристики (рис. 8).

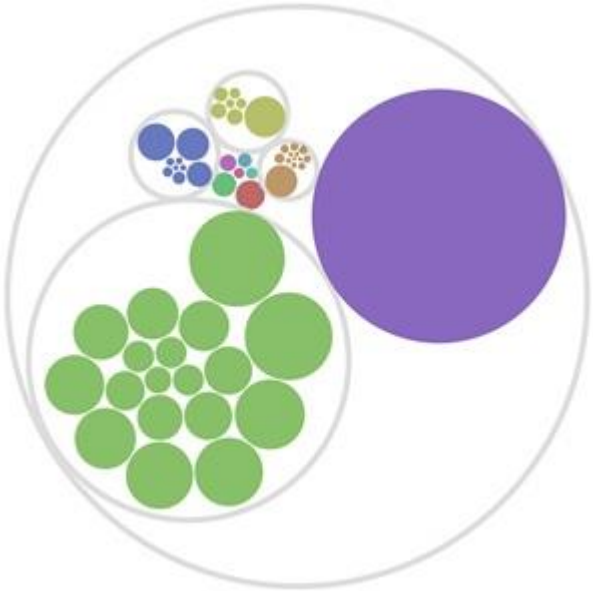

Рис. 8. Circle Packing [21]

Для трехмерного представления используется модель ландшафта [22]. Система формирует ландшафт, на котором располагаются граничные узлы, прорисовывается их содержимое (рис. 9). 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.

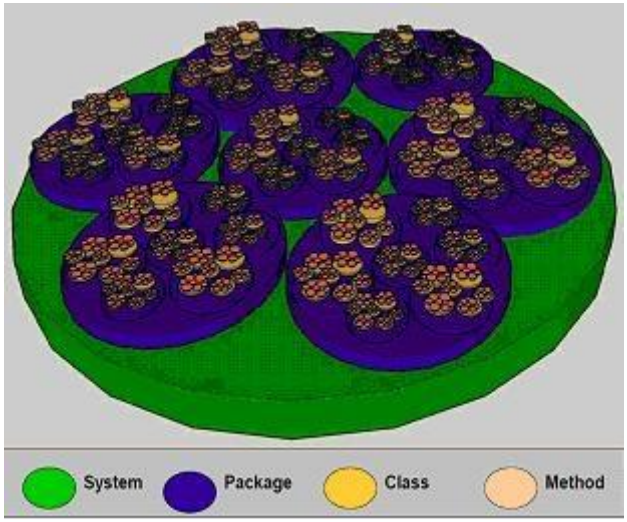

Рис.9. Визуализация программы в виде природного ландшафта [22].

Другой пример совместного отображения элементов программы и их связей как деревьев ландшафта в трехмерном пространстве – это Hierarchical Net 3D [23]. Этот инструмент позволяет, как отображать наследственные связи узлов дерева (классов программы), так и раскрывать их содержимое (рис. 10). Он позволяет изменять уровень детализации отображаемых узлов программы, а с помощью элементов навигации позволяет перемещаться в трехмерном пространстве во всей отображаемой структуре программы.

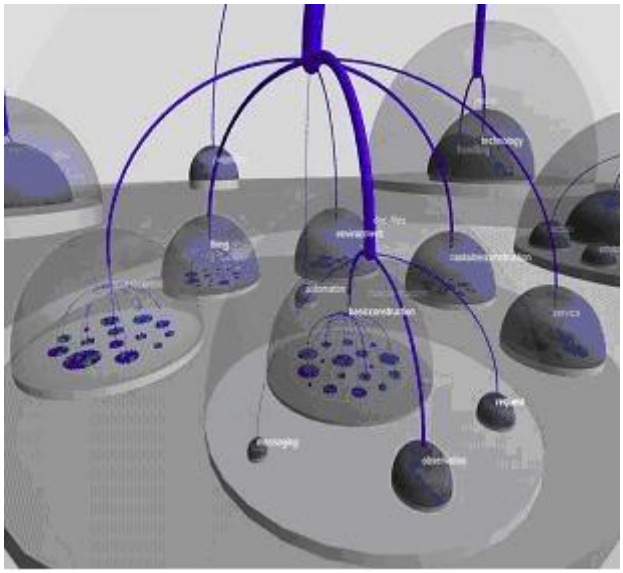

Рис.10. Ландшафт с отображением отношений.

Другой пример – это система Vizz3D [24]. Этот инструмент предоставляет дополнительные возможности для визуализации и анализа связей между элементами программы. Возможности навигации пользователя в трехмерном пространстве позволяют детально рассмотреть отношения связи между узлами. Пример визуализации отношений между элементами программы приведен на рисунке 11.

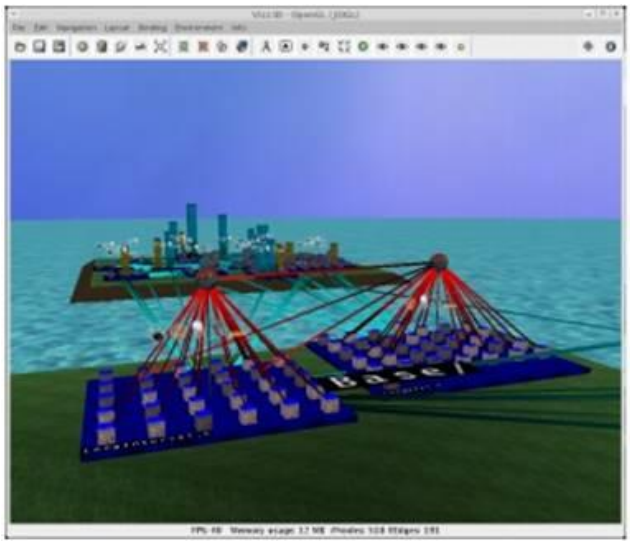

Рис.11. Визуализация отношений в трехмерном пространстве [24].

Общим моментом для визуализации графов является отсутствие понятной для пользователя ориентации графа в пространстве. Практически, во всех системах, ориентация графа (дерева) в трехмерном пространстве определена только алгоритмом его отображения (способом размещения данных на экране). Соответственно, ориентация не несет никакой информации.

3. 3D визуализация для программного обеспечения: модель города

Идея этого подхода как раз и состоит в том, чтобы задействовать ориентацию объектов. Соответственно, необходимо разрешить пользователю каким-то образом “перемещаться” между объектами в пространстве. Способ визуализации в виде города позволяет пользователю более привычным образом ориентироваться в трехмерном пространстве. Код программы представлен с использованием аналогии с реальным городом. Например, 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. Другой пример из работы [25], относящийся к визуализации программы написанной на языках C и C++. Рисунок 12 показывает представление файлов в зависимости от их размера:

- Дом – от 0 до 50 строк кода
- Апартаменты – от 51 до 200 строк
- Офисное здание – более 200 строк кода
- Ратуша – файлы заголовков (.h)
- Фигурка – функция или метод класса

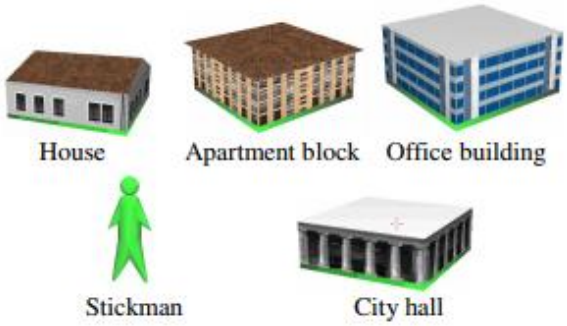

Рис. 12. Представление программных файлов [25]

И в соответствии с такой “кодировкой” уже отображается структура программы (рис. 13). Важно отметить, что при отображении программы могут быть визуализированы также различные метрики (правая часть рисунка 13). Например, в объектно-ориентированной системе это может быть размер класса, число методов, число потомков. Как правило, отображаемая метрика влияет на ширину и/или высоту отображаемого сооружения, что позволяет визуально оценить высокие или широкие здания. Так визуально будет проявляться возможный результат неправильного проектирования пакетов или классов (так называемая, избыточная ответственность класса).

Рис. 13. Программная архитектура и метрики

Одним из наиболее распространенных инструментов для отображения объектно-ориентированных метрик является система CodeCity [26]. На рисунке 14 представлен пример визуализации в этой системе.

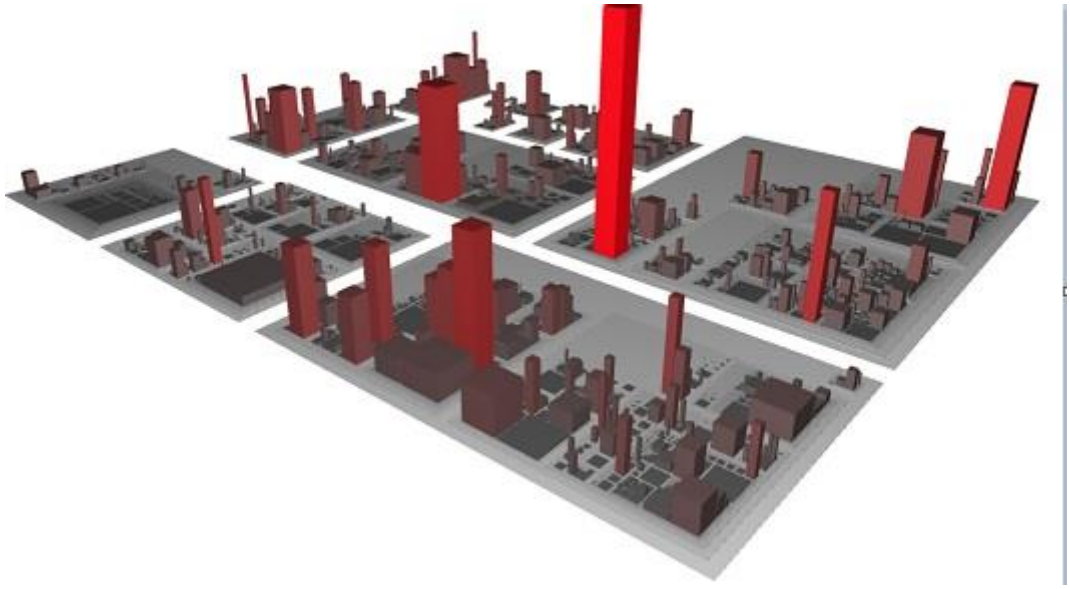

Рис.14. Программа-город.

Классы в программе отображаются красным цветом, размер класса определяется метриками класса (например, число методов и число атрибутов), 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 в такого рода системах используются различные уровни рельефа города, что наглядно позволяет визуализировать вложенность пакетов в визуализируемой программе (рис. 15).

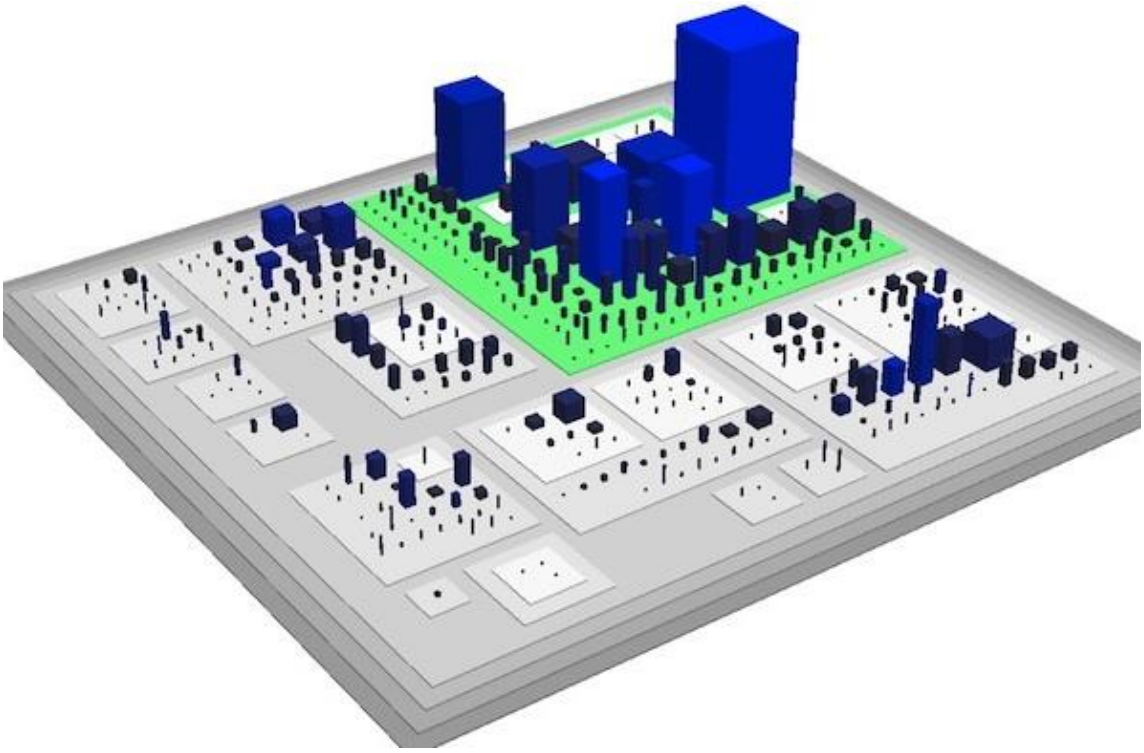

Рис.15. Визуализация вложенности в виде рельефа.

Такого рода визуализация позволяет наглядно показать дефекты проектирования. Например, Java-класс, обладающий определенным сочетанием значений метрик, показывается на карте города определенным цветом. На рисунке 16 показан пример такой раскраски дефектов проектирования в Java SDK.

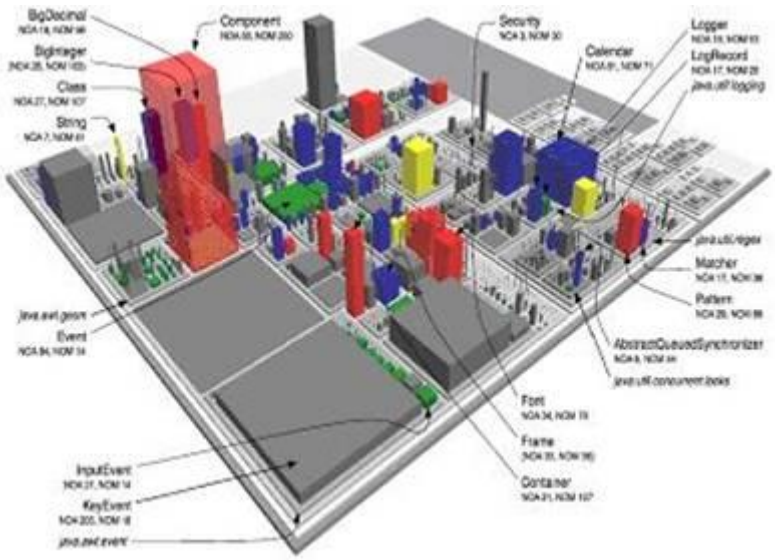

Рис.16. Дефекты проектирования

Возможность визуализации эволюции программной системы в трехмерном программном городе дает трактовка пакетов программной системы не как кварталов города, показываемых рельефом города, а как улиц города. В системе [28] классы и пакеты-улицы показываются в двумерном пространстве, как на рисунке 17.

Рис.17. Здания - классы системы, улицы - пакеты системы.

В этом случае ширина и высота зданий описывают метрики классов. Третье измерение при таком подходе используется для визуализации эволюции системы, как это показано на рисунке 18.

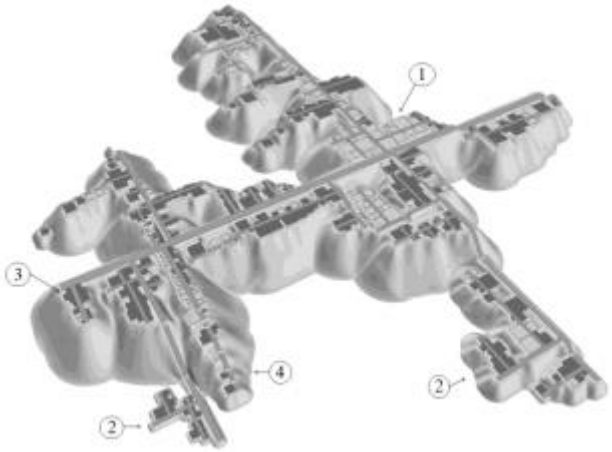

Рис. 18. Использование рельефа для визуализации эволюции системы.

В этом случае части системы, имеющие больший возраст, расположены на возвышенностях. Третье измерение классов может использоваться для визуализации метрик класса.

4. 3D визуализация для программного обеспечения: виртуальная реальность

Это сравнительно новое направление. Связано с развитием популярности систем виртуальной реальности и соответствующего оборудования. Соответственно, это VR-оборудование используется, как для просмотра, так и для навигации в “программном городе”. На рисунке 19 показан “программный город” в гарнитуре Oculus Rift [2].

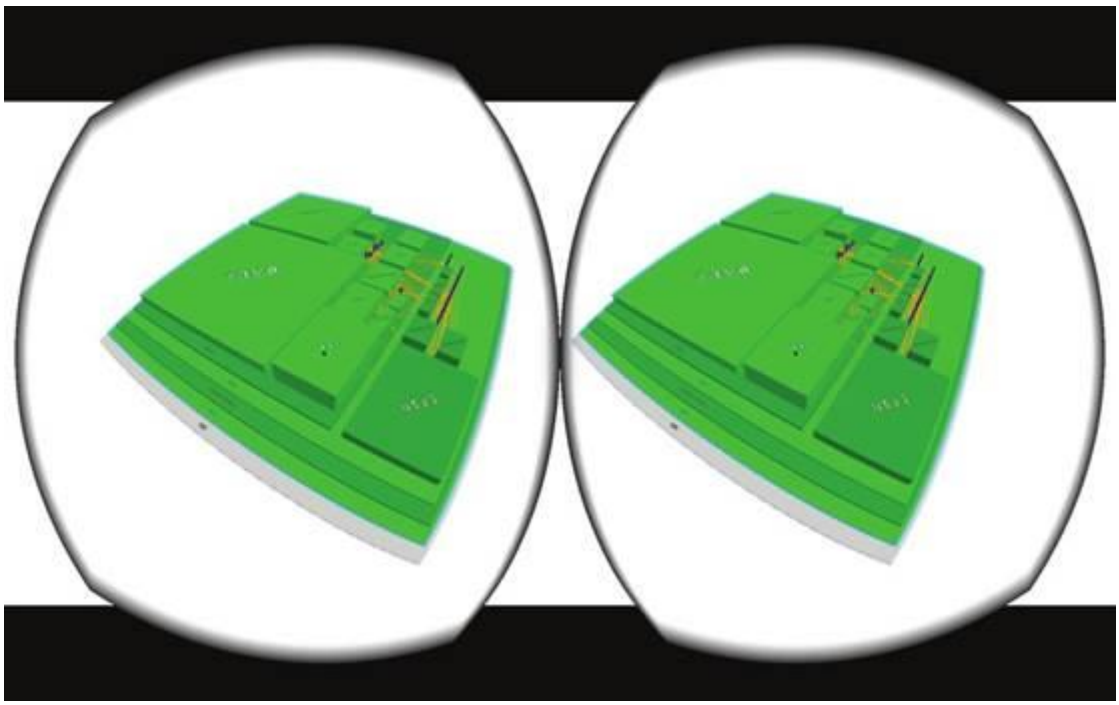

Рис. 19. Программный город и VR-очки [2]

При этом для навигации могут использоваться сенсоры Kinect, которые распознают движения руками пользователя.

В работе [27] пропагандируется подход на основе дополненной реальности. Здесь модель “программного города” встраивается в реальную сцену (рис. 20). Последовательность шагов этого процесса и проиллюстрирована на рисунке: a) камера снимает сцену с точкой привязки (маркером), b) изображение переводится в черно-белый формат, c) выстраивается масштабная 3D модель

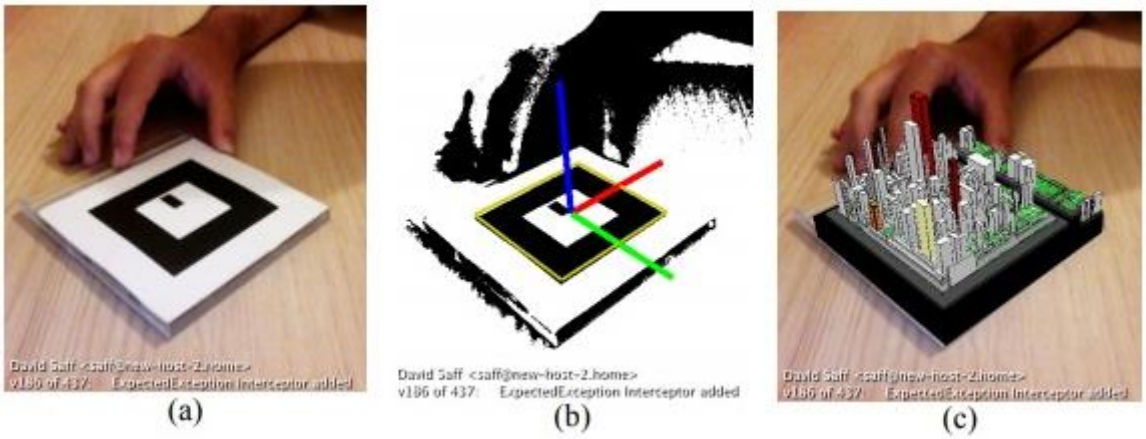

Рис. 20. Дополненная реальность для “программного города” [27].

5. 3D визуализация Java приложений

В этом разделе мы хотели бы остановиться на инструменте 3D визуализации для Java компонент, разработанном в лаборатории ОИТ факультета ВМК МГУ имени М.В. Ломоносова [7] в рамках работ по направлению программной инженерии. Для визуализации объектно-ориентированных метрик в трехмерном пространстве было выбрано представление программы в виде города. В качестве метрик были выбраны:

- KLOC – размер класса в килобайтах
- NOA – число атрибутов класса
- NOM – число методов класса

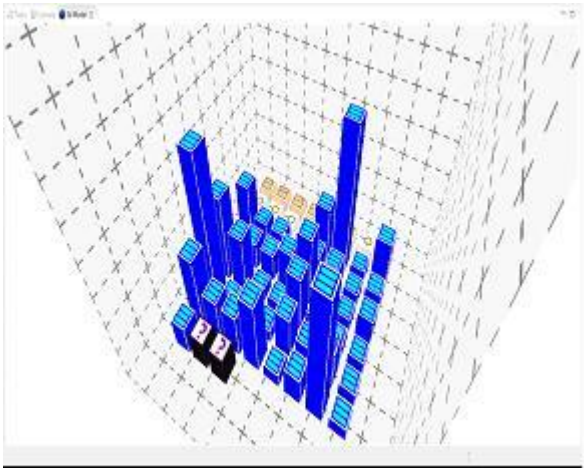

Рис. 21. “Здания” пакета JFace

Максимальное значение метрики в рассматриваемом пакете определяет максимальную высоту здания. Остальные отображаемые элементы визуализируются относительно отображаемого элемента. На рисунке 21 изображена структура пакета JFace из среды Eclipse

При этом “крыши” зданий окрашены в разные цвета, в зависимости от того, является ли данный объект классом или интерфейсом, публичный он или приватный. С помощью инструментов пользовательского интерфейса можно расцвечивать классы и интерфейсы в корне/листьях/середине иерархии наследования, или обладающие каким либо свойством, например, абстрактностью класса.

Инструменты навигации, в первую очередь, предназначены для поворотов и смещения начала координат (рис. 22). Это позволяет отобразить проекцию города из любой точки пространства под любым углом.

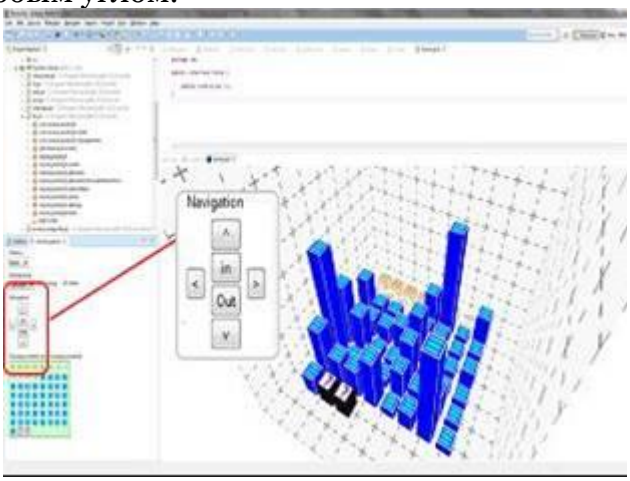

Рис.22. Панель навигации пользователя в программе-городе.

6. Заключение

В данной статье мы остановились на трехмерном визуальном представлении метрик и архитектуры программного обеспечения. В работе рассмотрены вопросы, связанные с трехмерным представлением графов и деревьев для визуализации структуры и метрик программных пакетов. Также описаны походы к визуализации на основе модели города. Перечислены основные инструментальные средства и их особенности. Из новейших разработок в данной области, статья описывает применение моделей дополненной и виртуальной реальности для визуализации программных компонент. В заключение представлен оригинальный пакет трехмерной визуализации для Java классов. Нам представляется, что описанные в данном обзоре подходы должны входить в арсенал исследователей и практиков в области программной инженерии.

Список литературы

1. Романов В. Ю., Намиот Д. Е. ВИЗУАЛЬНЫЙ АНАЛИЗ ПАКЕТОВ ПРОГРАММ //Научная визуализация. – 2017. – Т. 9. – №. 1. – С. 26-40.
2. Fittkau F., Krause A., Hasselbring W. Exploring software cities in virtual reality //Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on. – IEEE, 2015. – С. 130-134.
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. 11-17.
5. Merino L. et al. On the impact of the medium in the effectiveness of 3D

software visualization //In Proc. of VISSOFT. – 2017.
6. 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.
7. Романов В. Ю., Шульга И. В. Инструментарий для визуализации программного обеспечения в трехмерном пространстве //International Journal of Open Information Technologies. – 2015. – Т. 3. – №. 8. – С. 8-16.
8. Касьянов В.Н., Касьянова Е.В. Визуализация информации на основе графовых моделей // Научная визуализация. – 2014. – Том. 6, N 1. – С. 31 – 50
9. Касьянов В.Н., Касьянова Е.В. Визуализация графов и графовых моделей. – Новосибирск: Сибирское Научное Издательство, 2010
10. Panas T., Berrigan R., Grundy J. A 3d metaphor for software production visualization //Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on. – IEEE, 2003. – С. 314-319.
11. Teyseyre A. R., Campo M. R. An overview of 3D software visualization //IEEE transactions on visualization and computer graphics. – 2009. – Т. 15. – №. 1. – С. 87-105.
12. Tominski C., Abello J., Schumann H. CGV—An interactive graph visualization system //Computers & Graphics. – 2009. – Т. 33. – №. 6. – С. 660-678.
13. Baur M. et al. Drawing the AS Graph in 2.5 Dimensions //Graph Drawing. – 2004. – Т. 3383. – С. 43-48.
14. von Pilgrim J., Duske K. Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors //Proceedings of the 4th ACM symposium on Software visualization. – ACM, 2008. – С. 95-104.
15. CGV Coordinated Graph Visualization https://vcg.informatik.uni-rostock.de/~ct/software/CGV/ Retrieved: Oct, 2017
16. Walrus - Graph Visualization Tool http://www.caida.org/tools/visualization/walrus/ Retrieved: Oct, 2017
17. GEF3D http://eclipse.org/gef3d/ Retrieved: Oct, 2017
18. Cone Trees http://www.infovis-wiki.net/index.php?title=Cone_Trees Retrieved: Oct, 2017
19. Card S. K., Robertson G. G., Mackinlay J. D. The information visualizer, an information workspace //Proceedings of the SIGCHI Conference on Human factors in computing systems. – ACM, 1991. – С. 181-186.
20. THE INFORMATION VISUALIZER https://kisd.de/~rbaehren/information_visualizer.htm Retrieved: Oct, 2017
21. Circle Packing https://medium.com/@EvanSinar/7-data-visualization-types-you-should-be-using-more-and-how-to-start-4015b5d4adf2 Retrieved: Oct, 2017
22. Wang W. et al. Visualization of large hierarchical data by circle packing //Proceedings of the SIGCHI conference on Human Factors in computing systems. – ACM, 2006. – С. 517-520.
23. Balzer M., Deussen O. Hierarchy based 3D visualization of large software structures //Proceedings of the conference on Visualization'04. – IEEE Computer Society, 2004. – С. 598.4.
24. Panas T. et al. Communicating software architecture using a unified single-view visualization //Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on. – IEEE, 2007. – С. 217-228.
25. Alam S., Dugerdil P. EvoSpaces: 3D Visualization of Software Architecture //SEKE. – 2007. – Т. 7. – С. 500-505.

26. Wettel R., Lanza M. Codecity: 3d visualization of large-scale software //Companion of the 30th international conference on Software engineering. – ACM, 2008. – C. 921-922.
27. Souza R. et al. SkyscrapAR: An augmented reality visualization for software evolution //Proc. of 2nd Brazilian Workshop on Software Visualization (WBVS 2012). – 2012.
28. F. Steinbrückner and C. Lewerentz. Understanding software evolution with software cities. SAGE, 12(2):200–216, 2013.

Scientific Visualization, 2018, volume 10, number 5, pages 123 - 139, DOI: 10.26583/sv.10.5.08

3D visualization of architecture and metrics of the software

D.E. Namiot[1], V.Yu. Romanov[2]

Lomonosov Moscow State University, Russia

[1] ORCID: 0000-0002-4463-1678, dnamiot@gmail.com
[2] ORCID: 0000-0001-5140-9576, vladimir.romanov@gmail.com

Abstract

This article provides an overview of the methods of 3D visualization of software architecture and metrics. Metrics for programs (packages, classes, repositories) form one of the most actively used directions in software engineering. This line of research refers to the analysis of software, and visual analysis here is one of the most frequently used tools. This kind of visualization is usually part of the software quality analysis process. It can be used for training, for refactoring programs, and for integrating (combining) individual components (packages) into complex software systems. Obviously, visualization facilitates and speeds up the process of understanding the structure of software components. This is becoming more and more relevant, because now many open-source software components (what most often integrate into other systems), for example, are large and rather complex software packages. Accordingly, their integration into a new project is a very difficult task. We note that the integration problem becomes even more complicated if there is no access to the source texts of components in questions. In this case, the visual representation of metrics is, in fact, the main element of analysis. Analysis of third-party components is not the only area of application. Exactly the same problems arise in corporate development when separate groups work on a large project, which, moreover, can often vary in the composition of performers. In this paper, we consider, for example, the methods of visualization and analysis of the structure of a program in 3D space, which are based on the metaphor of the representation of a software component as a city, which consists of individual buildings that are grouped into districts, etc. We also consider the use of virtual reality for the presentation of software metrics.

Keywords: software engineering, software metrics, 3D visualization, software analysis.

References

1. Romanov V. Ju., Namiot D. E. VIZUAL"NYJ ANALIZ PAKETOV PROGRAMM //Nauchnaja vizualizacija. – 2017. – T. 9. – #. 1. – S. 26-40.
2. Fittkau F., Krause A., Hasselbring W. Exploring software cities in virtual reality //Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on. – IEEE, 2015. – S. 130-134.
3. Romanov V.Ju. Vizualizacija programmnyh metrik pri opisanii arhitektury programmnogo obespechenija //International Journal of Open Information Technologies. – 2014. – T. 2. – #. 2. – S. 21-28.
4. Romanov V.Ju. Analiz ob"ektno-orientirovannyh metrik dlja proektirovanija arhitektury programmnogo obespechenija//International Journal of Open Information Technologies. – 2014. – T. 2. – #. 3. – S. 11-17.
5. Merino L. et al. On the impact of the medium in the effectiveness of 3D software visualization //In Proc. of VISSOFT. – 2017.
6. Romanov V. Ju. Vizualizacija i analiz bol'shih programmnyh sistem s pomoshh'ju ih trehmernogo predstavlenija //International Journal of Open Information Technologies. – 2014. – T. 2. – #. 5. – S. 1-9.

7. Romanov V. Ju., Shul'ga I. V. Instrumentarij dlja vizualizacii programmnogo obespechenija v trehmernom prostranstve //International Journal of Open Information Technologies. – 2015. – T. 3. – #. 8. – S. 8-16.
8. Kas'janov V.N., Kas'janova E.V. Vizualizacija informacii na osnove grafovyh modelej // Nauchnaja vizualizacija. – 2014. – Tom. 6, N 1. – S. 31 – 50
9. Kas'janov V.N., Kas'janova E.V. Vizualizacija grafov i grafovyh modelej. – Novosibirsk: Sibirskoe Nauchnoe Izdatel'stvo, 2010
10. Panas T., Berrigan R., Grundy J. A 3d metaphor for software production visualization //Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on. – IEEE, 2003. – S. 314-319.
11. Teyseyre A. R., Campo M. R. An overview of 3D software visualization //IEEE transactions on visualization and computer graphics. – 2009. – T. 15. – #. 1. – S. 87-105.
12. Tominski C., Abello J., Schumann H. CGV—An interactive graph visualization system //Computers & Graphics. – 2009. – T. 33. – #. 6. – S. 660-678.
13. Baur M. et al. Drawing the AS Graph in 2.5 Dimensions //Graph Drawing. – 2004. – T. 3383. – S. 43-48.
14. von Pilgrim J., Duske K. Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors //Proceedings of the 4th ACM symposium on Software visualization. – ACM, 2008. – S. 95-104.
15. CGV Coordinated Graph Visualization https://vcg.informatik.uni-rostock.de/~ct/software/CGV/ Retrieved: Oct, 2017
16. Walrus - Graph Visualization Tool http://www.caida.org/tools/visualization/walrus/ Retrieved: Oct, 2017
17. GEF3D http://eclipse.org/gef3d/ Retrieved: Oct, 2017
18. Cone Trees http://www.infovis-wiki.net/index.php?title=Cone_Trees Retrieved: Oct, 2017
19. Card S. K., Robertson G. G., Mackinlay J. D. The information visualizer, an information workspace //Proceedings of the SIGCHI Conference on Human factors in computing systems. – ACM, 1991. – S. 181-186.
20. THE INFORMATION VISUALIER https://kisd.de/~rbaehren/information_visualizer.htm Retrieved: Oct, 2017
21. Circle Packing https://medium.com/@EvanSinar/7-data-visualization-types-you-should-be-using-more-and-how-to-start-4015b5d4adf2 Retrieved: Oct, 2017
22. Wang W. et al. Visualization of large hierarchical data by circle packing //Proceedings of the SIGCHI conference on Human Factors in computing systems. – ACM, 2006. – S. 517-520.
23. Balzer M., Deussen O. Hierarchy based 3D visualization of large software structures //Proceedings of the conference on Visualization'04. – IEEE Computer Society, 2004. – S. 598.4.
24. Panas T. et al. Communicating software architecture using a unified single-view visualization //Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on. – IEEE, 2007. – S. 217-228.
25. Alam S., Dugerdil P. EvoSpaces: 3D Visualization of Software Architecture //SEKE. – 2007. – T. 7. – S. 500-505.
26. Wettel R., Lanza M. Codecity: 3d visualization of large-scale software //Companion of the 30th international conference on Software engineering. – ACM, 2008. – S. 921-922.
27. Souza R. et al. SkyscrapAR: An augmented reality visualization for software evolution //Proc. of 2nd Brazilian Workshop on Software Visualization (WBVS 2012). – 2012.
28. F. Steinbrückner and C. Lewerentz. Understanding software evolution with software cities. SAGE, 12(2):200–216, 2013.