Doc3D: Система трёхмерной визуализации архитектуры программных систем

Е. В. Чепурнова, А. В. Игнатенко
Московский государственный университет им. М.В. Ломоносова, Москва, Россия
lifekate@rol.ru, ignatenko@graphics.cs.msu.ru

Аннотация

В статье описывается система Doc3D, предназначенная для создания 3D-документации программного кода. Трехмерная документация способствует более быстрому и глубокому пониманию кода программистом и дополняет обычную документацию. Разработанная система применима для визуализации связей и структуры классов и пространств имен в C++ и Java.

Ключевые слова: визуализация структуры кода, понимание кода, трехмерная визуализация, интерфейс человек-компьютер.

1. ВВЕДЕНИЕ

Современные программные проекты, написанные с использованием объектно-ориентированных языков программирования, становятся всё сложнее и сложнее. Они зачастую насчитывают миллионы строк кода, тысячи классов. Для таких проектов серьезной проблемой является анализ структуры кода, особенно при ознакомлении с системой.

Существуют разнообразные подходы к визуализации структуры программных систем. Наиболее известный из них – моделирование с помощью языка UML (Unified Modeling Language) [1]. UML предназначен для проектирования и анализа различных объектно-ориентированных систем и является общепризнанным стандартом для описания моделей. Язык позволяет отображать структуру кода в виде двумерных графов (рис.1.), что облегчает восприятие (по сравнению с текстовым списочным представлением) и значительно упрощает работу программистов и разработчиков. Но с ростом числа классов, связей, пространств имён и т.д., воспринимать UML-модели становится сложно.

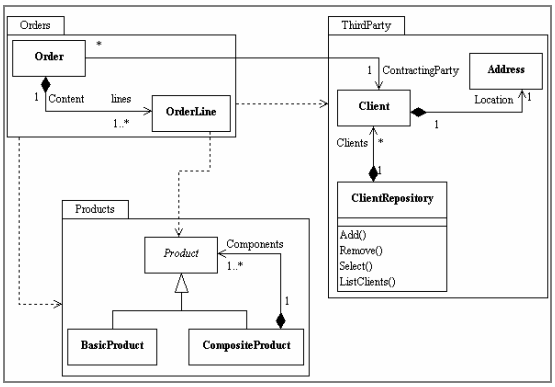

Рис. 1. UML – диаграмма, отображающая классы и различные связи между ними.

Другим подходом является трёхмерная визуализация основной структуры кода. Основная идея этого подхода заключается в использовании трех измерений для визуализации, что должно помогать разработчикам быстрее и проще понимать главные особенности проекта.

В работе [2] применяется подход на основе визуализации кода в виде трехмерного города. Главная идея – представление сложной информации таким образом, чтобы она легко и интуитивно воспринималась программистом, чтобы от программиста не требовалось дополнительных навыков. Подход заключается в том, что пользователь оказывается в знакомой ему трёхмерной среде, где есть такие же дома, дороги, деревья и светофоры, как и в реальном мире. Они означают какие-либо программные структуры. Дома, например, символизируют классы. В пространстве легко перемещаться, что даёт большую степень свободы, чем плоскость. Недостатком такого подхода является отсутствие четко выраженной иерархической структуры и большое количество отвлекающих деталей.

Другой подход [3] не использует аналоги реального мира для описания структуры кода. Вместо этого авторы опираются на систему виртуальной реальности и абстрактные трехмерные примитивы (кубы, сферы, плоскости т.п.). Этот подход предполагает наиболее подробную и точную визуализацию. Кроме классов и разнообразных связей между ними отображаются такие детали как функции и атрибуты классов. В зависимости от количества строк кода функции отображаются примитивами разного размера. Величина отображаемых классов зависит от количества функций и членов этих классов. Для выделения из всех функций конструкторов, деструкторов, перегруженных, виртуальных функций, широко используется цвет. Такая визуализация хорошо подходит для детального изучения относительно небольших систем, однако результат работы на больших системах оказывается чрезвычайно перегруженным деталями.

Ещё одним важным направлением является реализация отображения классов в 2.5D-пространстве [4]. 2.5D означает расположение каких-либо объектов на одной плоскости (видимой или невидимой), но не в 2D, а в 3D. Расположение на одной плоскости делает сцену наглядной, а 3D позволяет свободно перемещаться в пространстве. Так же большую роль играет объём и размер объектов. В 2D понятие объёма не существует, а размер – величина достаточно ограниченная, т. к. сильно влияет на восприятие. В 2.5D третье измерение даёт право делать объекты любого размера, при этом общий вид сцены сильно не страдает.

Существуют и другие методы визуализации: аналоги видеоигр, уплотненные параллелепипеды [5] и другие.

Общими недостатками всех трехмерных методов визуализации кода является плохо читаемый текст и большое

количество степеней свободы наблюдателя, что может приводить к потере ориентации пользователем.

Далее в статье описывается предлагаемая система визуализации Doc3D. Основным отличием разрабатываемой системы является применение трехмерного отображения структуры кода не вместо традиционной текстовой документации, а в дополнение к ней, что позволяет объединить достоинства обоих подходов.

В п.2 описаны основные цели разрабатываемой системы. Входные данные для системы описаны в п.3. Пункт 4 посвящен описанию способов визуализации пространств имен, классов. В п.5 описывается взаимодействие пользователя с системой и результаты тестов. Заключение и направления будущих исследований приведены в п.6.

2. ОСНОВНЫЕ ПРИНЦИПЫ

При разработке системы Doc3D были поставлены следующие задачи:

- Система разрабатывается для документации существующего кода с целью анализа его базовой структуры, взаимосвязей классов и пакетов.
- Система должна дополнять текстовую документацию, которая создается вручную или генерируется по комментариям в коде такими программами, как Doxygen [6].
- Система должна обеспечивать удобную навигацию, избегать возможности потери ориентации пользователем.

3. ВХОДНЫЕ ДАННЫЕ

Входными данными для Doc3D являются файлы xml-формата, которые содержат всю необходимую информацию для визуализации. Эти файлы генерируются популярной программой Doxygen [6], которая позволяет создавать xml- и html-документацию для C/C++ и Java кода. Основной задачей стояла визуализация классов C++, но, т.к. Doxygen создаёт стандартизованную документацию, то возможна так же визуализация и классов Java.

4. ВИЗУАЛИЗАЦИЯ

Были разработаны два основных режима: режим просмотра классов и режим просмотра пространств имён и пакетов. Каждый из режимов имеет несколько различных представлений.

4.1 Визуализация пространств имён и пакетов

В Doxygen существует понятие пакетов (групп), в которые программист может объединять классы и функции. Обычно с помощью пакетов выделяются логически связанные группы классов.

Пакеты связаны между собой отношениями зависимости и использования, причем эта информация обычно является наиболее значимой для анализа системы, т.к. позволяет разделить систему на слои (представления, логики и т.п.) для дальнейшего раздельного анализа.

Разработанная система анализирует зависимости пакетов и располагает пакеты в трехмерном пространстве с учетом этой информации.

Каждый пакет изображается параллелепипедом. Размер параллелепипеда зависит от количества входящих в пакет классов. При этом имеются ограничения на его минимальный и максимальный размер, т.к. может возникнуть ситуация, когда один пакет содержит в десятки раз больше классов, чем другой. Ограничения на размер позволяют поддерживать наглядность информации в любом случае.

Было создано представление, в котором пакеты располагаются на одной плоскости (см. рис 2), а связи между ними отображаются отрезками разного цвета. На рис. 2 красными отрезками показана вложенность. Кроме того, отображаются связи по наследованию классов (рис 3). Если класс A входит в пакет Gr1, а класс B – один из наследников класса A (не обязательно на первом уровне наследования) и B принадлежит пакету Gr2, то пакет Gr2 зависит от Gr1 по наследованию классов. Такие связи отображаются чёрными стрелками.

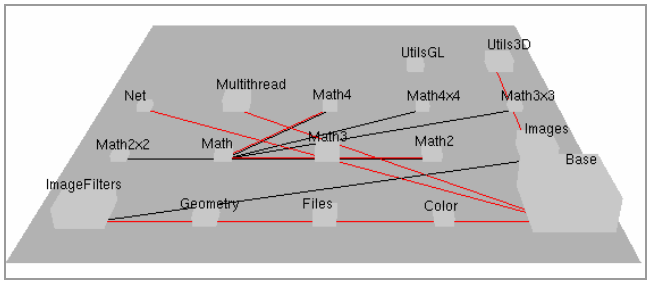

Рис. 2. Визуализация пакетов и связей между ними.

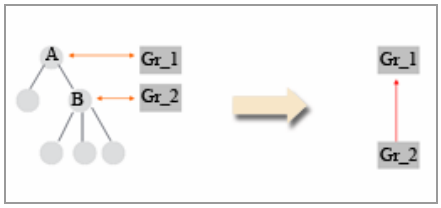

Рис. 3. Связи между пакетами по наследованию.

Другим вариантом представления пакетов является представление для анализа зависимостей пакетов. В этом режиме можно проанализировать все связи и расположить пакеты на разных плоскостях. На рис. 4 показана реализация этой идеи. На нижнем уровне располагаются пакеты, не зависящие от каких-либо других, на следующем - зависящие только от нижних и т. д.. Таким образом, на n-ом уровне располагаются пакеты, зависящие обязательно от пакетов (n-1)-ого уровня и, возможно, от пакетов более нижних уровней. В этом представлении программист может легко понять основные зависимости и определить базовые пакеты, а также пакеты верхнего уровня. Оно также хорошо подходит для оценки корректности архитектуры программы – большое количество взаимных зависимостей и связей через уровень наглядно свидетельствуют о необходимости оптимизации структуры программы.

Аналогично пакетам реализована и визуализация пространств имен.

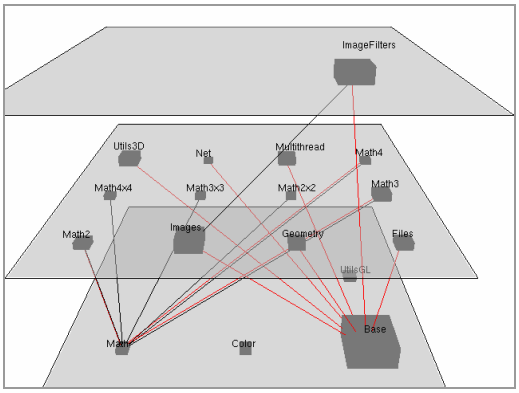

Рис. 4. Визуализация пакетов на разных плоскостях.

4.2 Визуализация классов

Было разработано несколько представлений для отображения наследования классов (как внутри пакетов, так и при отображении всех классов проекта).

Представление на цилиндре (рис 5.) используется при отображении классов внутри пакетов или для визуализации небольших систем (количество классов не больше 50). Базовые классы отображаются сферами и располагаются по окружности цилиндра, ветви производных классов отходят вправо вдоль поверхности. Такое представление более удобно, чем традиционная древовидная двумерная структура наследования, т.к. позволяет одним взглядом оценить количество классов и глубину наследования. В таком представлении пользователь может вращать цилиндр.

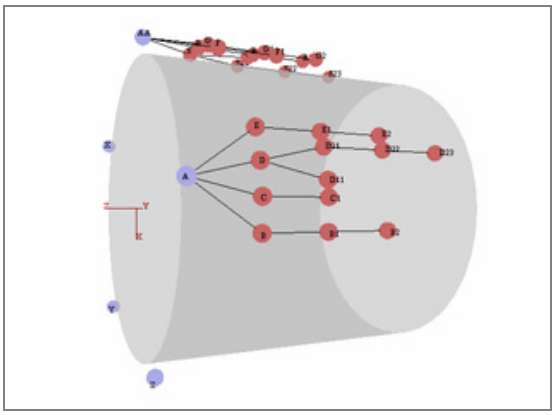

Рис. 5. Представление классов и наследования.

На рис. 6 показан другой вариант визуализации, в котором базовые классы расположены на плоскости. Особенностью этого режима является возможность перемещения пользователя в двух плоскостях: вдоль базовой плоскости и по вертикали.

Однако, для систем большего размера этот подход уже не подходит. На рис.7 показан пример визуализации программной системы, состоящей из 600 классов. Разобраться в структуре уже крайне сложно.

Для решения этой проблемы был создан специальный режим, в котором визуализируются только те классы, у которых много связей, с возможностью дополнительного перехода к скрытым классам. Точное количество отображаемых классов зависит от размера системы и максимального числа производных классов. Кроме вышеуказанных классов при таком подходе отображаются ещё и их базовые классы, для сохранения наследования.

Для наглядности классы рисуются сферами разного цвета и размера в зависимости от количества наследников. Пример такой визуализации показан на рис. 8 и рис. 9. На рис. 8 показана та же система, что и на рис. 7.

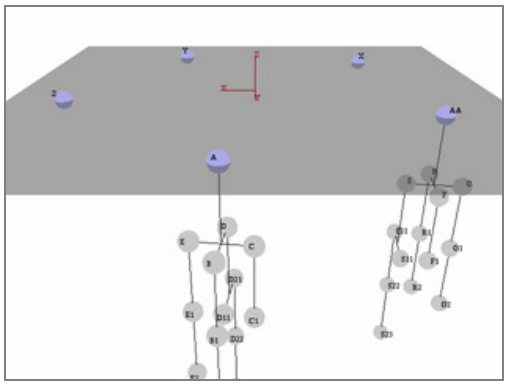

Рис. 6. Другой вариант представления классов и наследования.

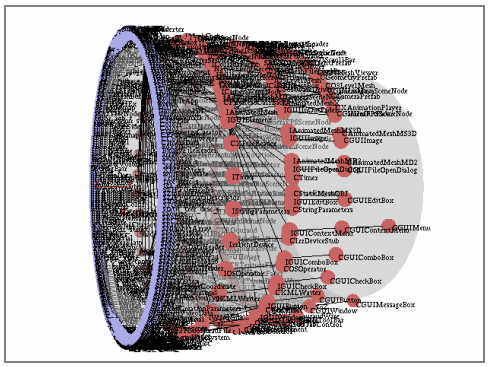

Рис. 7. Пример визуализации с помощью цилиндрического представления на системе из 600 классов.

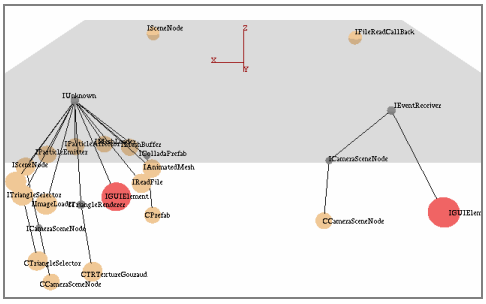

Рис. 8. Режим просмотра классов, имеющих относительно большое число производных (та же система, что и на рис. 7).

5. ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С СИСТЕМОЙ И РЕЗУЛЬТАТЫ ТЕСТОВ

Взаимодействие пользователя с разработанной системой происходит по следующему сценарию.

- Пользователь строит xml-документацию с помощью Doxygen и запускает систему Doc3D.

- После запуска система переходит в режим визуализации пакетов или пространств имён (в зависимости от структуры кода).
- Далее пользователь может переключаться между разными представлениями для пакетов, переходить в режимы визуализации классов и просматривать html-документацию этого класса (пакета), созданную Doxygen.

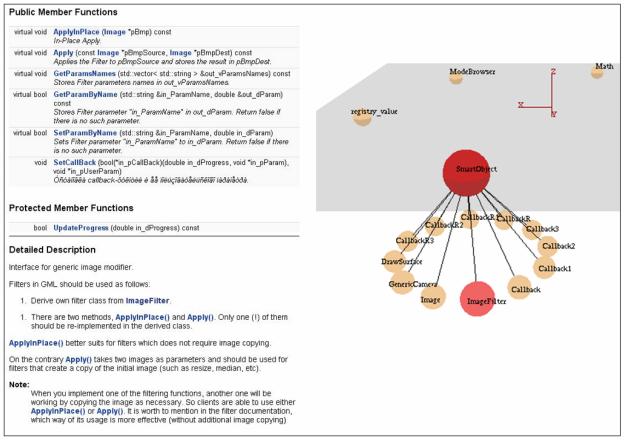

Рис. 9. Совместная двух – и трехмерная документация

Для анализа эффективности и удобства работы с системой были проведены следующие тесты. Программистам предлагалось проанализировать две незнакомые системы, одну небольшую (около 50 классов), другую более сложную (около 600 классов), на основе документации Doxygen и с помощью Doc3D, и сравнить удобство работы с документацией.

Результаты тестов показали, что работа с трехмерным представлением достаточно необычна и требует первичного привыкания. Однако после некоторого времени работы с системой программисты отмечали, что трехмерный вариант визуализации оказывается более наглядным, особенно во время работы с небольшой системой. В то же время можно заключить, что наибольшие неудобства возникали при анализе структуры сложной системы – как в трехмерном, так и в текстовом представлении документациии.

6. ЗАКЛЮЧЕНИЕ

Представленная система на данном этапе не является завершенным проектом. Однако на данном этапе можно сделать вывод о том, что в задачах анализа архитектуры программной системы трехмерная документация является жизнеспособной альтернативой традиционной двухмерной документации на основе текста и UML-моделей.

Работа над системой продолжается в следующих направлениях: развитие системы для более удобного представления систем с большим количеством классов, визуализация агрегации и других связей между классами и связей между пространствами имён, визуализация UML-моделей. Планируется реализация системы в виде плагина к web-браузеру для возможности создания интернет-документации.

7. БИБЛИОГРАФИЯ

[1] *Unified Modeling Language, Version 2.0 specification. ISO/IEC 19501.*

[2] Michael Balzer, Andreas Noack, Oliver Deussen, Claus Lewerentz. *Software Landscapes: Visualizing the Structure of Large Software Systems.* In proceedings of VisSym 2004, 2004.

[3] Jonathan I. Maletic, Jason Leigh, Andrian Marcus, Greg Dunlap. *Visualizing Object-Oriented Software in Virtual Reality.* In Proceedings of International Workshop on Program Comprehension (IWPC01), 2001.

[4] Thomas Panas, Rebecca Berrigan, John Grundy. *A 3D Metaphor for Software Production Visualization.* In proceedings of the Seventh International Conference on Information Visualization, 2003.

[5] Andrian Marcus, Louis Feng, Jonathan I. Maletic. *3D Representations for Software Visualization.* In proceedings of the ACM symposium on Software visualization, 2003.

[6] *Doxygen.* http://doxygen.org (26.04.06).

Об авторах

Екатерина Викторовна Чепурнова – студентка факультета вычислительной математики и кибернетики Московского Государственного Университета им. М.В. Ломоносова

E-mail: lifekate@rol.ru

Алексей Викторович Игнатенко – младший научный сотрудник факультета вычислительной математики и кибернетики Московского Государственного Университета им. М.В. Ломоносова

E-mail: ignatenko@graphics.cs.msu.ru

Doc3D: A system for three-dimensional visualization of software system architecture

Abstract

This article describes Doc3D: a system for generating 3D documentation of software systems source code. Three-dimentional documentation facilitates code understanding and analysis, and supplement ordinary text documentation. Developed system is applicable for visualization of C++/Java classes and packages relations.

Keywords: *code visualization, code understanding, 3d visualization, human-computer interface.*

About the authors

Ekaterina Chepurnova is a student at Moscow State University, Department of Computer Science. Her contact email is lifekate@rol.ru

Alexey Ignatenko is a researcher at Moscow State University, Department of Computer Science. His contact e-mail is ignatenko@graphics.cs.msu.ru