Свойства метафор визуализации и выбор методов представления данных о функционировании программных комплексов

В.Л. Авербух, Д.Р. Исмагилов

Задачей данного исследования является анализ свойств метафор визуализации в связи с разработкой методов представления данных о функционировании программных комплексов. Рассматриваются популярные метафоры, например, метафора ландшафта и метафора города, и подходы к их применению для визуализации программного обеспечения, в том числе в системах отладки правильности и эффективности (настройки производительности) параллельных программ.

1. Введение

В наших предыдущих работах [1-3] рассматривается проблема выбора методик визуального представления данных при отладке правильности и производительности параллельных и распределенных программ. Отмечалось, что интерес к разработке визуальных отладчиков снизился, и это объяснимо, прежде всего, отсутствием адекватных средств визуализации. Действительно, в таких системах использовалась (и сейчас используется) простая двумерная графика с недостаточным уровнем выразительности. В тоже время из-за применения абстрактной образности и (зачастую даже неосознанного или неявного) использования неадекватных задаче метафор интерпретация результатов часто сводилась к расшифровке сложных визуальных текстов, понять и объяснить которые мог только специалист-разработчик. Однако с одной лишь неудачной графикой связаны причины застоя в разработке средств визуализации программного обеспечения параллельных вычислений? Возможно, что недостаточность средств визуализации есть последствие недостаточно четких моделей параллельных вычислений. В этой работе мы рассмотрим возможности таких известных метафор визуализации программного обеспечения, как комната, ландшафт, город, фабрика. Затем вспомним задачи, поставленные при разработке систем визуальной отладки параллельных вычислений еще в 90-ые годы. Цель исследований - поиск того, как отображать параллельные программы на визуальные объекты, порождаемые данными трехмерными динамическими метафорами, составляющими иерархическую последовательность “ комната” - “дом” - “город” [“фабрика”].

2. Метафоры визуализации

Под метафорой визуализации мы понимаем отображение, ставящее в соответствие понятиям и объектам моделируемой прикладной области систему сближений и аналогий и порождающее некоторый изобразительный ряд (набор видов отображения) и набор методов взаимодействия с визуальными объектами [4]. Отметим при этом, что метафора служит лишь толчком для развития идеи. Метафоры “живут” своей жизнью и могут отбрасывать не только логику реальности, но и ее образность. Во многих случаях имеет место превращение естественной образности в абстрактную, что не мешает пользователям, которые при правильно выбранной образности не обращают внимания на мешающую логику исходной области.

Известен целый ряд работ, посвященный использованию таких метафор как “комната” [1], система “комната”-“небоскреб” [5], “промышленный ландшафт” [6], “город” [7],[8],[9] для нужд визуализации программного обеспечения. Системы, построенные на базе этих метафор, используются для визуального программирования [1], отладки производительности параллельных программ [5] и, главное, для управления ходом разработки программных проектов [6], [7],[8],[9]. Простота восприятия и некоторые свойства, которые рассмотрим чуть

ниже, облегчают их применение в системах визуализации программного обеспечения. Отметим, что особый интерес к данным метафорам связан с их использованием в системах, разработанных на базе сред виртуальной реальности.

В [2] и [4] мы рассмотрели свойства метафоры комнаты. Среди них способность содержать какие-либо объекты внутри себя; ограничение контекста восприятия; замкнутость; включение в структуру (из комнат можно “строить здания”, рассматривая совокупность комнат, как элемент построения некоторой сложной конструкции); естественность метафоры.

Комната является естественной метафорой, с наличием соответствующего объекта в реальном мире. Это свойство делает интуитивно понятными все другие свойства. При использовании метафоры не возникает дополнительных аналогий и неестественных образов. Функциональность и характеристики реального объекта просто переносятся (хотя и в несколько расширенном понимании) в виртуальный мир.

Рассмотрим теперь свойства метафор города и ландшафта. Они во многом схожи, что естественно, так как схожи и сами метафоры.

1. *Неограниченный контекст*. При работе с метафорами города и ландшафта контекст пользователя не ограничивается искусственным образом. В результате, требуются дополнительные усилия со стороны пользователя для идентификации того или иного объекта среди множества других. Плюс это, или минус - зависит от конкретной реализации, а также целей, которые ставит система визуализации, использующая эту метафору. При визуализации большого количества данных, неограниченность контекста позволяет бегло окинуть взглядом всю “картинку” и быстро выделить ключевые места.

2. *Естественность метафор*. Известно, что естественность метафоры уменьшает усилия по интерпретации результирующего изображения. Применительно к метафорам города и ландшафта кроме естественности пространственной ориентации, имеет место также естественность навигации. В случае метафоры города способ навигации предлагается самой метафорой.

3. *Организация внутренней структуры*. Метафоры предполагают наличие внутренней структуры, причем в случае метафоры города эта структура диктуется самой метафорой, и задается она достаточно жестко – есть дома, кварталы, улицы, районы. При работе с метафорой ландшафта выбор структуры –свободный. В этом случае также можно говорить о вложенности ландшафтов.

4. *Наличие ключевых элементов*. Метафоры предполагают вывод достаточно большого объема информации, причем в большинстве случаев достаточно однородной в визуальном смысле. Для интерпретации этой информации, пользователю необходимо наличие ключевых элементов, “якорей”, которые являются отправными точками для интерпретации всех объектов. Например, в случае применения метафоры для выделения неких особенных (например - ошибочных) элементов среди множества других, эти элементы должны выделяться визуально, они и будут теми самыми “ключами”.

5. *Влияние метафоры города на результирующее отображение*. В отличие от метафоры ландшафта, при выборе метафоры города мы автоматически резко сужаем круг возможных видов отображения.

6. *Устойчивость к масштабированию*. Метафоры не только устойчивы к увеличению объема информации, но и можно говорить об увеличении оправданности использования метафор в этом случае. Метафоры ландшафта и города предполагают наличие некоторого достаточно большого объема предоставляемой информации для того, чтобы их применение было оправданным.

Фактически можно говорить о метафоре городского пространства, как об отдельном подвиде метафоры ландшафта, но со своим набором специфических свойств, таких как конкретная внутренняя структура, влияние на отображение. (Свойства метафоры города также рассмотрены в [10].)

Можно выделить в отдельный подвид и метафору “промышленного ландшафта”. По свойствам она ближе к метафоре города, но ей не присуще наличие ключевых элементов. Также можно говорить о частичном, не сильном ограничении контекста (что роднит ее с метафорой комнаты). Остальные свойства метафоры города присущи и метафоре

“промышленного ландшафта”. Кроме того, существует возможность “нагрузить” ее элементы смыслами, связанными с производством той или иной программной “продукции”.

В обоих случаях (то есть в случае метафоры города и промышленного ландшафта) использованию метафор помогает наличие транспортных артерий, которые могут рассматриваться как средство представления потоков управления, потоков данных, иных связей между программными конструкциями или частями программного комплекса.

3. Задачи отладки параллельных вычислений

Традиционные системы отладки обеспечивают функции, разбиваемые на три основных группы:

1) выбор состояния программы;

2) вывод состояния программы;

3) модификация состояния программы.

В дальнейшем мы будем иметь в виду исключительно визуальный аспект разработки и функционирования отладчиков, а проблемы инженерии программного обеспечения оставим “за кадром”. Среди проблем визуализации не до конца решенной остается важная проблема отображения динамики программы. Отметим, что динамическая природа программы настоятельно требует использования в отладке анимированных представлений потоков управления и данных, а также движущихся образов, соответствующих модельным объектам программы. Почти все визуальные отладчики умеют как-то отображать трассу программы (хотя и тут возникают некоторые проблемы). Однако при отображении взаимодействующих объектов и/или процессов могут возникнуть проблемы. В [3] описаны возможности динамического отображения в рамках метафоры города, где может отслеживаться анормальная работа программной системы, такая как взаимная блокировка, постоянное откладывание и т.п. Программные процессы здесь представляются динамикой уличного перекрестка, где имеют место связанные между собой управляющие движением объекты (светофоры) и взаимодействующие между собой, реагирующие на управление объекты (автомобили).

Отладка параллельных программ считается более сложной, чем традиционная последовательная отладка. Первая причина заключается в том, что имеется множество отдельных потоков управления (процессов) которые выполняются одновременно. Поэтому не всегда получается не только идентифицировать ошибочное выражение или команду, но даже и указать на процесс, содержащий это ошибочное выражение. Тот факт, что процесс асинхронно взаимодействует с другими процессами, еще больше усложняет задачу отладки. Действительно это не только вызывает новый класс ошибок (таких как гонки или тупиковые ситуации), но также способствует быстрому распространению ошибки на всю вычислительную систему (так называемый, “*эффект домино*”). Тем не менее, если процесс не выполняет то, что он должен делать, возможно, причина этого заключается в получении процессом неправильных данных, а не в том, что в нем есть ошибка. Другая проблема, возникающая при отладке параллельных программ, заключается в асинхронных взаимодействиях, которые могут произойти, а могут и не произойти, и тем самым чрезвычайно затрудняют повторное выполнение программы и препятствуют восстановлению ошибочной ситуации, создавая ***недетерминированность*** в поведении программы [11]. Данная параллельная программа при множестве успешных пропусков, имея на входе одни и те же данные, может получить и различную последовательность состояний, и различные результаты. Эти расхождения вызываются тем, что программа может содержать ***гонки сообщений (data races).***События, такие как получение сообщений, приходящих почти одновременно, могут вызвать различный порядок успешного выполнения программы, а программа может получить различие в результатах. Повторение симптомов гонки сообщений может оказаться затруднительным, так как добавка кодов для более полного изучения поведения программы может увеличить время ее работы, тем самым нарушить условия появления гонки [12].

Одной из самых перспективных за последние годы представляется идея ***сравнительной отладки*** (*relative debugging*), заложенной в основу реализации отладчика Guard [13]. Guard

предполагалось использовать при отладке программ, реализованных за счет эволюционных методик разработки программного обеспечения. Эта методика делает возможным применение ранних версий программы (о которых заведомо известно, что они функционируют правильно) для генерации значений, служащих для сравнения с новой разрабатываемой программой. Одной из форм сообщения пользователю о наличии различий в версиях программы (при вычислении массивов) служило использование битовых карт, когда информация выдавалась в виде прямоугольной битовой карты на экране дисплея, на которой белыми пикселями обозначаются значения массива, одинаковые в обоих вариантах программы, а черными - элементы с различными значениями. Такая, достаточно простая визуализация массивов особенно полезна при определении ошибок, связанных с заданием циклов, или с адресных выражений, так как эти типы ошибок обычно генерируют четко различаемые образы. Вид отображения типа битовая карта может использоваться и для визуализации различий в многомерных массивах. Однако более мощные методики визуализации различий поддерживаются за счет применения трехмерной графики. Именно они лучше всего подходят для визуализации различий в многомерных массивах. В этом случае возможно применение анимации для отображения развития в различиях, возникающих по ходу выполнения программы.

Существует еще две проблемы с визуализацией в отладчиках для многопроцессных программ. Первая проблема заключается в создании механизма указания на то, какие из процессов программы должны получить и (выполнить) соответствующие операции управления. Вторая связана с необходимостью обеспечить пользователя методами поиска и извлечения информации о состоянии нужного процесса из сложного состояния всей многопроцессной программы. Решение обеих проблем возможно за счет описания интересующих пользователя процессов по ходу отладки и требует **навигации по процессам** [14].

Визуальная составляющая систем измерения и настройки производительности параллельных программ прежде всего связана с выводом “узких мест” производительности. Кроме того, необходимо профилировать программу, то есть выделять характеристики тех изолированных участков программы, чья производительность определяет производительность всей программы. Среди причин “узких мест” производительности можно назвать *дисбаланс загрузки*, *конкуренция при передаче сообщений*, *большие накладные расходы при взаимодействии*, *слабое использование линий связи*. [15].

Другой способ описания производительности связан с выводом метрик производительности работы аппаратных и программных частей параллельных систем. При этом различные приложения генерируют огромное количество данных, требуемых для оценки производительности. Отмечается: что методики научной визуализации сосредоточиваются на интуитивном выводе регулярно располагаемых, n-мерных наборов данных. А вот данные о производительности параллельных систем нерегулярны ни по пространственным, ни по временным соотношениям и поэтому приходится использовать методики представления, заимствованные из статистической графики [5].

При создании модели производительности [16] могут быть определены абстракции тех аспектов параллельной программы, которые воздействуют на производительность и отвечают на три базовых вопроса:

1) Почему данная прикладная программа имеет низкую производительность?

2) Где расположено “узкое место” производительности?

3) Когда проявилось воздействие данного “узкого места” на производительность программы?

В литературе чаще всего рассматривалась отладка эффективности на уровне операций посылки сообщений, но возможна оптимизация самого кода программы при ее распараллеливании [17]. В этой связи ставится задача предсказания и оценки производительности параллельной программы и отмечается, что для решения проблемы получения оценок производительности необходимо ответить на ряд важных вопросов:

- Как должны обрабатываться неизвестные программы?
- Какая информация о производительности должна предоставляться программисту?

- Насколько точно необходимо оценивать производительность?
- Где находятся части программы, требующие увеличения производительности?
- Сколько времени займет вычисление необходимых данных о производительности?
- Как данные о производительности должны фильтроваться и визуализироваться?

4. Возможности метафор визуализации и отладка параллельных программ

Теперь, когда мы рассмотрели общие задачи отладки параллельных программ, снова вернемся к метафорам визуализации, их свойствам, выбору подходящих “хороших” метафор. Именно набор свойств является основой для сравнения метафор. Метафора связана с отображением моделируемой предметной области на “область визуальных объектов”. Нужно оценивать не метафору саму по себе, но связку “метафора – моделируемая область”. Соответственно тут видятся две задачи: оценка данной связки на адекватность, и либо подбор метафоры под конкретную область (прямой ход), либо подбор области под данную метафору (обратный ход). Однако выделить сущности, которые и поддаются визуализации и полезны при одно- двух- шаговой интерпретации достаточно сложно.

Попытаемся оценить использование этих метафор для задач отладки производительности параллельных программ (оцениваем в связке с моделируемой областью). Использование метафор ландшафта и города напрашивается для визуализации потока данных программы, распределения данных по узлам вычислений, отображения хода их обработки, фильтрации результатов и т.д. Обе (принципиально трехмерные) метафоры пригодны для решения таких задач, как навигация по процессам, передача динамики процессов и пр. В обоих случаях в рамках метафор возможно проектирование систем с использованием виртуальной реальности, что весьма перспективно для задач отладки параллельных вычислений (см.[5]). Трехмерность и реалистичность метафоры города могут использоваться при реализации новых возможностей при сравнительной отладке.

Теперь возникает ряд вопросов, в частности:

- можно ли, пользуясь лишь набором свойств, выбрать метафору из этих двух?
- каковы, исходя из задачи, положительные и отрицательные стороны их применения?

Сначала пробежимся по свойствам, совпадающим у метафор. Неограниченный контекст предоставляется обеими метафорами, и это скорее положительная сторона при применении к нашей задаче: пользователю скорее будет важно знать - как считаются данные на всех узлах в целом (нет ли дисбаланса загрузки, не простаивают ли узлы), чем рассматривать какой то отдельный узел из сотен или тысяч ему подобных. Естественность метафор всегда является положительным свойством (упрощение интерпретации, конечно при сохранении размерностей сущностей – [18]), при прочих равных дающим приоритет перед искусственной метафорой. Также и наличие ключевых элементов не выделяет ни одну из метафор. Причем тут это свойство прибавляет работы по реализации системы (выделение “якорей”, их отображение). Кроме того – обе метафоры “способны к масштабированию” - увеличение объемов данных (при неизбежном наращивании количества узлов) лишь увеличивает обоснованность их применения. Пока кажется, что обе метафоры оправданы для применения в задачах визуализации потока данных параллельных программ. Для нужд отладки полезно показывать связь между данными и программными конструкциями. В развитии метафоры ландшафта предлагается метафора оболочки, когда визуализация данных и программных объектов проводится на расположенных одна над другой поверхностях, подобных поверхности земли и небесной сфере (или летающим островам). Объекты, расположенные на этих поверхностях, могут связываться между собой, демонстрируя внутрипрограммные связи и потоки данных. Данная метафора может использоваться при представлении параллельных программ, построенных на базе моделей общей памяти и параллелизма по данным [3].

Рассмотрим теперь различия метафор. Метафора города уже обладает внутренней структурой. Причем она хорошо ложится на некоторые топологии параллельных систем. Следовательно, нет дополнительной нагрузки при интерпретации (а также при реализации

системы). Влияние метафоры городского пространства на результирующее изображение нельзя отнести к положительным или отрицательным качествам, так как тут велика роль реализации.

В итоге получаем, что обе метафоры в принципе подходят для наших задач. Метафора городского пространства накладывает некоторые ограничения, однако они в тоже время несколько упрощают интерпретацию результатов и реализацию системы. Кроме того, в рамках метафоры города имеет место иерархический семантический зуминг – “комната, здание, город (фабрика)”. То есть, возможно масштабирование с переходом из помещения на улицу, затем в квартал и весь город [завод] в целом. Природный ландшафт из-за отсутствия иерархии объектов должен ограничиться простым изменением масштаба изображения.

Отладка чрезвычайно сложных и поэтому ненадежных параллельных программ настоятельно требует использование визуализации. Почему же визуализация пробуксовывает в последние годы? Наше мнение таково, что причины лежат, в том числе, в отсутствии ясного понимания того, что может визуализация. Ее главная задача - показывать состояния объектов и процессов, их особенности и переходы из состояния в состояние, структуры и связи между ними. Однако не все можно комплексно показать на сравнительно малом пространстве экрана. Даже использование больших экранов не всегда дает приемлемый результат. Масштабируемая трехмерная динамическая метафора может помочь в этом деле.

Литература

1. Байдалин А.Ю., Исмагилов Д.Р. Средства представления структур в системах визуализации программного обеспечения // ГрафиКон'2006, 1-5 июля 2006. Россия. Новосибирск, Академгородок. Труды Конференции. Новосибирск. ИВМиМ. 2006. Стр. 271-274.
2. Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р., Казанцев А.Ю. Трёхмерные методики визуализации программного обеспечения параллельных и распределённых вычислений // ПаВТ'2008: Труды международной научной конференции (Санкт-Петербург, 28 января - 1 февраля 2008 г.), Электронное издание, Челябинск, Издательство ЮУрГУ, 2008. Стр. 283-288.
3. Авербух В.Л., Байдалин А.Ю., Исмагилов Д.Р., Флягина Т.А. Трехмерные и динамические метафоры в визуализации программного обеспечения параллельных и распределенных вычислений // Тезисы 10-го Международного семинара “Супервычисления и Математическое моделирование”, РФЯЦ-ВНИИЭФ, Саров, 2008, с. 16-17.
4. Averbukh V.L., Bakhterev M.O., Baydalin A.Yu., Gorbashevskiy D. Yu., Ismagilov D.R., Kazantsev A.Yu., Nebogatikova P.V., Popova A.V., Vasev P.A. Searching and Analysis of Interface and Visualization Metaphors // Human-Computer Interaction, New Developments / Chapter 3, Vienna, In-teh., pp. 49-84.
5. Reed D., Scullin W., Tavera L., Shields K., Elford Ch. Virtual Reality and Parallel Systems Performance Analysis // IEEE Computer, V.28, N 11, (November 1995) pp. 57-67.
6. Christensen H. B. Utilising a Geographic Space Metaphor in a Software Development Environment // Proceedings of the IFIP Seventh Working Conference on Engineering for Human-Computer Interaction, 1998, pp. 39-56.
7. Panas Th., Berrigan R., Grundy J. A 3D Metaphor for Software Production Visualization // Seventh International Conference on Information Visualization, 2003. Proceedings. 16-18 July 2003, pp. 314-319.
8. Balzer M., Noack A., Deussen O., Lewerentz C. Software Landscapes: Visualizing the Structure of Large Software Systems // Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization, 2004, pp. 261-266.
9. Wettel R., Lanza M. Visualizing Software Systems as Cities // 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007). 24-25 June 2007, pp. 92-99.
10. Russo Dos Santos C., Gros P., Abel P., Loisel D., Trichaud N., Paris J. P. Experiments in Information Visualization Using 3D Metaphoric Worlds // Proceedings of the 9th IEEE

International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, pp. 51-58.
11. Caerts C., Lauwereins R., Peperstraete J.A. PDG: A Process-Level Debugger for Concurrent Programs in the GRAPE Rapid Prototyping Environment // 4th IEEE Int. Workshop on Rapid System Prototyping, Research Triangle Park, North Carolina, June 28-30, 1993.
12. May J., Berman F. Retargetability and Extensibility in a Parallel Debugger. J. Parallel and Distributed Computing 35(2), 15 June 1996. Pp. 142-155.
13. Abramson D., Foster I., Michalakes J., Sosic R. Relative Debugger: A New Methodology for debugging Scientific Applications // Communication of the ACM. V. 39, No. 11 (November 1996), pp. 69-77.
14. Cheng D., Hood R. A Portable Debugger for Parallel and Distributed Programs // Proc. Supercomputing'94, Washington, D.C., November 1994. IEEE Comp. Soc. Press. Pp. 723-732.
15. Yan J., Sarukkai S., Mehra P. Performance Measurement, Visualisation and Modeling of Parallel And Distributed Programs using AIMS Toolkit // Software - Practice and Experience. Vol. 25, No.4 (April 1995) pp.429-461.
16. Miller B., Calaghan M., Cargille J., Hollingsworth J., Irvin R.B., Karavanic K., Kunchithapadam K., Newhall T. The Paradyn Parallel Performance Measurement Tool // IEEE Computer, V.28, N 11, (November 1995), pp.37-46.
17. Fahringer Th. Estimating and Optimizing Performance for Parallel Programs // IEEE Computer, V.28, N 11, (November 1995), pp. 46-56.
18. Baydalin A Representation view's adequacy criterion // CSIT ' 2008 Proceedings of the 10th International Workshop on Computer Science and Informational Technologies Antalya, Turkey, September 15-17, 2008 Ufa State Aviation Technical University, 2008, vol.1 pp.152-158.