УДК 004:37

АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ СРЕДЫ ВИЗУАЛИЗАЦИИ АЛГОРИТМОВ

Алексейчук И.В.

Херсонский государственный университет

В статье рассмотрены история и перспективы развития программно-методического комплекса «Визуализатор алгоритмов поиска и сортировки»

Ключевые слова: *Визуализация алгоритмов, расширяемая архитектура, модульная система.*

Введение

История создания среды визуализации алгоритмов сортировки и поиска в Херсонском государственном университете насчитывает уже несколько десятилетий. Впервые концепцию наглядного представления алгоритмов предложил и обосновал ее методическую целесообразность доцент Львов Михаил Сергеевич. Сила данного подхода заключается в представлении абстрактных понятий о эффективности алгоритмов во вполне конкретном и наглядном виде.

Первый программный продукт, посвященный данной тематике, был создан в 80-х годах прошлого столетия силами университетских лабораторий. В рамках технологий тех лет, система была реализована в виде набора DOS-приложений, которые последовательно применялись к исходному алгоритму, в результате чего препроцессор дополнял данный алгоритм командами визуализации. Несмотря на решение всех возложенных на систему задач, она обладала рядом недостатков, в числе которых – дезинтеграция компонентов системы, отсутствие встроенной среды редактирования, сложность управления.

Следующим витком развития среды стало создание в 2005 году приложения «Видеоинтерпретатор 2.0». Данное приложение по сей день используется в университете при чтении лекций, посвященных основам программирования и эффективности алгоритмов. «Видеоинтерпретатор 2.0» представляет собой Windows-приложение, разработанное на языке программирования C++ при помощи технологии MFC. В системе есть встроенный редактор, позволяющий ускорить создание алгоритмов за счет вставки заготовок синтаксических конструкций. Из нововведений следует отметить добавление встроенной коллекции алгоритмов поиска и сортировки.

В середине первого десятилетия 2000-х годов на базе ХГУ стал активно развиваться другой продукт, впоследствии получивший название «Основы алгоритмизации и программирования». Продукт стал неотъемлемой частью одноименного курса и предоставил студентам и преподавателям адекватные на тот момент инструменты образования. Параллельно с этим началось обсуждение возможности интеграции системы «Видеоинтерпретатор 2.0» в вэб-систему. Единственным доступным, с точки зрения трудозатрат, способом интеграции существующего десктоп приложения с вэб-системой было его преобразование в ActiveX компонент. Это давало много новых возможностей, однако и накладывало ограничения на возможности продукта. Главным из которых было невозможность использования никакого другого браузера, кроме Microsoft Internet Explorer, а так же возможность просмотра только на платформе Windows. Следует заметить, что технология ActiveX является детищем корпорации Microsoft и широкого распространения среди других компаний не получила, по ряду причин в числе которых проблемы уязвимости. Так же результатом включения системы «Видеоинтерпретатор 2.0» в курс «Основы алгоритмизации и программирования» стало оснащение его инструментами исследования эффективности алгоритмов.

С тех пор прошло уже пять лет… Для бурного рынка ИТ это почти вечность. Теперь я предлагаю посмотреть на продукт с точки зрения тех технологий, которые нам предлагает современная индустрия, и тех, новых, требований, которые выдвигает современное образование.

В 2010 году на кафедре информатики факультета физики, математики и информатики Херсонского государственного университета было принято решение о создании новой версии системы видеоинтерпретации алгоритмов, удовлетворяющих современным требованиям.

Команда разработчиков состоит из шести человек, в числе которых архитектор, функции которого выполняет автор статьи, четыре разработчика и дизайнер. Новая версия продукта представляет собой магистерский проект представленной выше команды.

Основная часть

Требования, выдвинутые к продукту:

кросплатформенность – хороший продукт должен обеспечивать пользователю возможность работы вне зависимости от операционной системы, которую он предпочитает. Долгие годы корпорация Microsoft на поприще операционных систем имела лидирующие позиции, однако в последние годы наблюдается тенденция роста пользователей других операционных систем – Linux, Mac OS. Последняя операционная система особенно популярна в странах северной Америки.

возможность работы в автономном режиме – данное требование, возможно, покажется читателю архаизмом, в связи с общедоступностью Интернета и увеличением пропускных способностей каналов связи, однако не следует забывать, что прогресс семимильными шагами идет только туда, где развивается бизнес, обходя стороной глубинку.

расширяемость – если обратить внимание на историю развития нашей системы визуализации, можно заметить, что она развивалась скачкообразно – от версии к версии. Такую экстремальность можно объяснить несколькими факторами:

продукт создавался силами студентов разных годов выпуска, что, во-первых, создавало сложности передачи знаний между командами разработчиков, а, во-вторых, ограничивало более поздние команды старыми технологиями, которые применяли пионеры;

программный код, написанный студентами, так или иначе, далек от идеала, поэтому нескольким следующим поколениям разработчиков-выпускников приходилось его «вычищать» (хотя не всегда это его действительно улучшало);

И, в конце концов, продукт приходил в такое состояние, что проще было все переписать, чем продолжать поддерживать старую версию, да и набор технологий к этому моменту пора было обновлять (Рис.1). Наша команда в этом отношении не исключение.

Рис. 1. Развитие среды визуализации

Под данным требованием следует понимать возможность модификации среды – добавление нового функционала, изменение визуализации, в том числе добавление визуализаций новых классов алгоритмов.

В рамках своей магистерской работы, я решил произвести исследование возможности расширения среды и учесть такие факторы:

- невозможность написания красивого кода – как было уже замечено выше, код написанный студентами не идеален. Не следует расценивать это как упрек – магистерская работа, как правило, не имеет ни большой научной, ни практической ценности. Это всего лишь полигон для обучения выпускников в условиях, приближенных к реальным.

- нежелание разбираться в чужом коде – этот пункт вытекает из предыдущего
- необходимость расширять среду

Учитывая эти факторы, я сконцентрировал внимание на нестатической привязке компонентов системы. То есть вместо того чтобы напрямую использовать функциональность одного модуля в другом, мы опишем набор интерфейсов взаимодействия и запустим механизм связывания не на этапе компиляции, а на этапе выполнения.

Это предоставит нам возможность расширять функциональность не скачкообразно – с приходом новой версии и, соответственно, переписыванием продукта заново, а постепенно – добавлением новых модулей системы. Так же к несомненным плюсам можно отнести отсутствие необходимости разбираться в чужом коде – если определенный функционал не реализован или реализован с недостаточным качеством, необходимо соблюсти интерфейс модуля данного типа и подменить существующий.

Ключевым вопросом работы становиться не построение максимально полнофункциональной версии – необходимый функционал всегда можно добавить позже, а построение наиболее гибкого механизма связывания.

Итак, что же нам все это дает на практике. Во-первых, мы позволяем расширять среду в том направлении, которое востребовано в данный момент образованием. Во-вторых, мы избавляемся от необходимости постоянного переделывания среды с нуля. И, в-третьих, функционирование среды жестко не зафиксировано на ее компонентах, главное – ядро системы.

Компоненты ядра

В данном разделе описаны абстрактные компоненты системы, композиция которых во время выполнения будет обеспечивать выполнение задач.

Парсер и интерпретатор – исходный алгоритм представлен в формальном виде, как правило, на некотором языке программирования. Следовательно, парсер необходим для построения объекта структуры данных, описывающих формальную модель. Данную структуру, в силу нелинейности алгоритмов, удобнее всего представлять в виде дерева, где корневой узел – точка входа в программу. Интерпретатор же обеспечивает выполнение инструкций в узлах дерева и навигацию по нему.

Редактор – визуальное средство создания алгоритмов. Должен обеспечивать средства удобного отображения, выбора, изменения алгоритма. Так же в функции редактора входит формирование исходных данных, над которыми будет выполняться алгоритм. Во время выполнения редактор должен выделять выполняемую интерпретатором инструкцию.

Визуализатор – интерфейсный модуль, отвечающий за наглядное выполнение алгоритма. Должен предоставлять возможность отображать каждую из инструкций, выполняемых интерпретатором.

Отображение результатов – модуль представления статистики о выполнении алгоритма, должен предоставлять возможность ее представления в разных, удобных пользователю формах, экспорта и сохранения.

Технологические решения

В силу требований, выдвигаемых к системе, мы остановили свое внимание на ряде технологий, решающих наши задачи наилучшим образом:

Moonlight

Данная технология разрабатывается корпорацией Novel и является совместимой с технологией Microsoft Silverlight и предназначена для построения RIA-приложений (Rich Internet Application). Технология Moonlight привлекла наше внимание в силу того, что наиболее удобным образом решает задачи анимации и динамического отображения контента. Так же, поскольку в основе технологии Moonlight лежит кросплатформенная технология Mono, это соответствует требованиям об обеспечении работы на трех основных платформах: Windows, Linux, Mac OS.

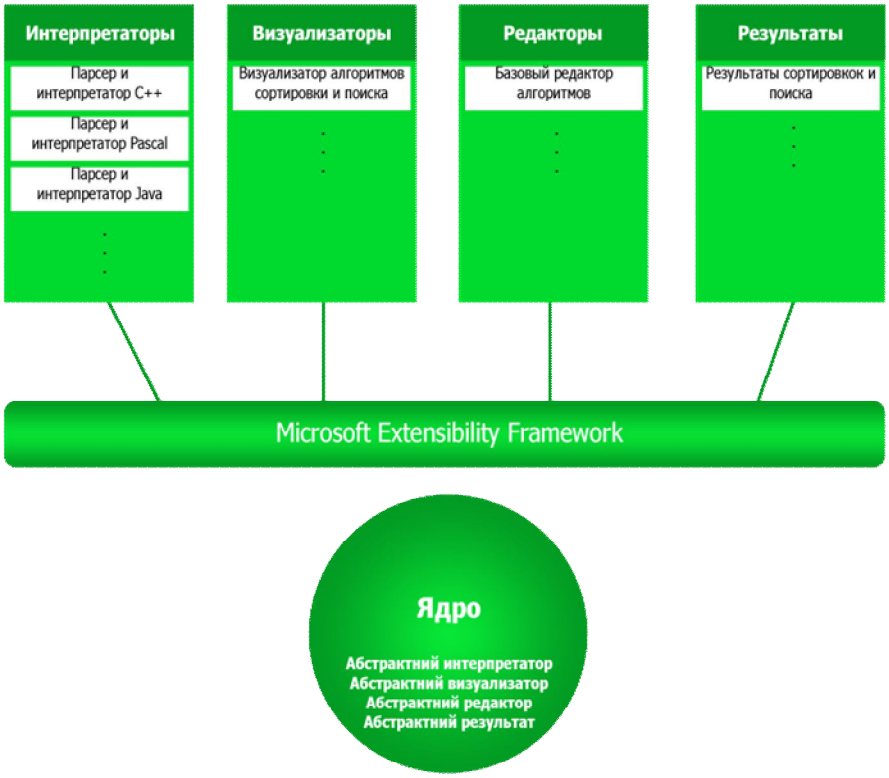

Рис. 2. Модули системы

ANTLR 3.5 (Another Tool for Language Recognition)

Система ANTLR представляет собой наиболее удобное средство для автоматической генерации парсеров и интерпретаторов, описанных в БНФ (Бекуса-Наура форма). Данный подход был позаимствован у предыдущей версии видеоинтерпретатора, однако в нем был использован ANTLR версии 2.7.

MEF (Microsoft Extensibility Framework)

Технология MEF обеспечивает механизм композиции модулей системы на этапе выполнения. Библиотека MEF в «Видеоинтерпретатор 3.0» входит как составляющая ядра системы и обеспечивает наращивание функциональности без необходимости перекомпиляции всех ее модулей.

Выводы

На рисунке 1 представлены этапы развития продукта, ключевыми из которых были 1986 – год создания первой версии, 2005 – создание второй версии, 2010 – принятие решение о создании новой версии. Градиент на фоне от зеленого к красному отображает возмущения: устаревание технологий, сложность поддержки версии и добавления нового функционала. Так, после каждого нового релиза количество возмущения только увеличивалось. Концепция построения расширяемой системы над ядром, предоставляющем абстрактные интерфейсы позволит кардинально изменить ситуацию (рис. 3).

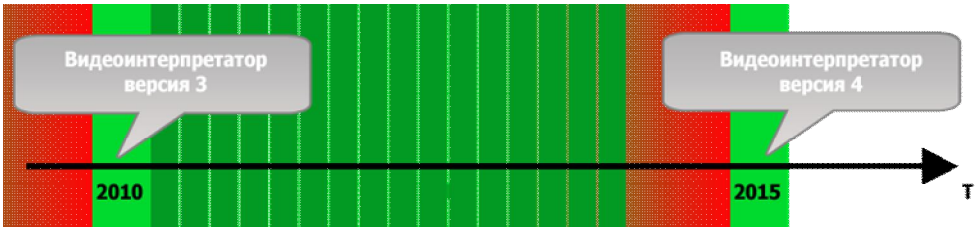

Рис. 3. Прогнозирование развития продукта

Темно-зеленые вертикальные линии представляют написание новых модулей системы. Очевидно, что в конечном итоге систему ожидает участь ее предшественниц. Но при этом мы позволяем сохранить множество созданных на предыдущем этапе компонентов системы и, при условии реализации обратной совместимости, мгновенно наполняем ее функционалом.

Прогнозирование технологической актуальности

Несмотря на быструю смену технологий, в мире ИТ принято доверяться трендам. То есть если крупная корпорация делает ставку на определенную технологию – проводит ее рекламу, активно развивает, пытается завоевать новые рынки, естественно желая получить дивиденды, то и рядовому разработчику имеет смысл доверять данной технологии.

В нашем случае технология Moonlight была выбрана не только в силу удобности и применимости в нашей задачи, но и в силу того, что на данную технологию возлагает определенные надежды крупнейший игрок в мире ИТ – Microsoft. Silverlight, как старший брат Moonlight, активно продвигается Microsoft на рынке мобильных платформ, как средство разработки для смартфонов под управлением ОС Windows Phone 7. И это укрепляет нашу веру в эти технологии.

К сожалению, достойной альтернативы средству автоматической генерации парсеров мы не нашли – большинство инструментов в этой области очень специфичны. Так что ANTLR обладает монополией в этой сфере. Однако наличие обширного комьюнити и постоянное обновление версий так же свидетельствуют о правильности нашего выбора.

Технология MEF еще достаточно молода и развивается порядка полутора лет, но уже завоевала широкую популярность среди .Net-разработчиков. И что немаловажно – была включена в состав Framework 4.0.

В связи со всеми данными событиями мы полагаем, что актуальность технологий сохраниться как минимум на 3 – 4 года и при удачном стечении обстоятельств и на 5 лет.

Источники:

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Інформаційні технології в освіті. Випуск 3. Херсон – 2009.
2. http://www.antlr.org
3. http://www.stackoverflow.com