ОПЫТ ПРИМЕНЕНИЯ ОБЕРОНА ПРИ ИНТЕГРАЦИИ ИССЛЕДОВАНИЙ, ОБРАЗОВАНИЯ И ПРОИЗВОДСТВА

Ермаков Илья Евгеньевич

ООО «Метасистемы», ТИ ОГТУ, Орёл, ermakov@metasystems.ru

Доклад посвящен выводам, полученным в результате 4 лет использования в компании «Метасистемы» языков и технологий Оберон-семейства в исследовательской работе, обучении студентов и коммерческом программировании. Оберон продемонстрировал сочетание качеств, которые ставят его на особое место среди современных систем программирования, применительно к долгосрочным целям, особенно если эти цели требуют интеграции трёх названных направлений.

The report offers our summary about Oberon languages and technologies that is from 4 years of using their in research, education and commercial programming. Oberon reveals the union of properties, that rise it to exclusive place between modern programming systems, in applying to long-term purposes, especially if this purposes require integration of three areas noted above.

За последние несколько лет системы программирования семейства Oberon Н. Вирта получили достаточную известность в нашей стране. Центрами консолидации Оберон-сообщества являются проекты Информатика-21 [1] и OberonCore.ru. Специалистами из науки, образования и индустрии, использующими Обероны, была сформулирована единая система взглядов относительно преимуществ применения Оберонов на всех этапах обучения программированию [2, 3], а также для решения сложных научных и инженерных задач.

Данный доклад будет посвящён ряду выводов, полученных нами при работе в рамках Оберон-направления. Его инструменты и идеи на протяжении 4 лет являются для нашего коллектива долгосрочным фундаментом при разработке ПО, исследованиях в области системного

программирования и обучении студентов. Компания «Метасистемы» сформировалась на базе физико-математического факультета ОГУ и изначально строила свою деятельность в тесной связи с образовательной средой. С 2008 г. мы получили возможность перейти от спецкурсов к полной подготовке студентов «с нуля», начиная со ступени СПО, в Технологическом институте ОГТУ. (Эксперимент автора по работе на средней ступени интересен тем, что перекрывает сразу обе стадии из связки «старшие классы — младшие курсы», а Информатика-21 выстраивает концепцию единой системы обучения программированию именно для неё).

Два русла “mainstream”. При внимательном наблюдении за тенденциями массового ИТ последнего времени можно увидеть в нём два направления. Первое весьма традиционно для 90-х гг. (во всех странах) и связано с притоком в резко расширившуюся отрасль огромного числа деятелей, не получивших должного фундаментального образования — его можно несколько иронично назвать «программирование с разводным ключом». Второе течение, ставшее заметным недавно, во многом направлено на преодоление недостатков первого — это функциональное программирование (ФП), приносящее в отрасль некую математичность. Далее, если мы посмотрим на платформы Java и .NET, то увидим, что изначально их дебютной идеей было внедрение в хаотичный стиль программирования Си-языков ряда дисциплинирующих принципов и механизмов безопасности, существенно повысивших как качество, так и компонентность ПО. В этом смысле индустрия последовала опыту Оберон-систем, правда, отяготив его в конкурентной гонке как рудиментами, так и непроверенными и второстепенными новинками. В настоящий же момент мы наблюдаем активную интеграцию в эти платформы средств ФП. Таким образом, ведущие корпорации отрасли в своих продуктах пришли к некоему «великому объединению». Зададимся вопросом, чего же дальновидному профессионалу может не хватать в этих условиях «для полного счастья» - и какие позиции здесь имеет Оберон-направление.

Что за «принципом Калашникова»? Проектом Информатика-21 был сформулирован принцип с таким названием, звучащий как «Избыточная сложность - это уязвимость». Принцип резюмирует размышления многих видных деятелей Computing Science, таких как Дейкстра, Вирт, Хоар и мн. др. Однако у рядового деятеля ИТ, как показывают постоянные споры, этот тезис сочувствия не встречает. Действительно, на практике именно порождение избыточной сложности приносит много краткосрочных выгод — и индустрия интуитивно нашла в этом наиболее выгодный для себя путь. Увы, программисты, разбалованные лёгкостью «виртуальных игр», менее всех других инженеров понимают долгосрочную цену принимаемых решений. Но путь развития любой отрасли содержит резкие повороты (например, в ИТ — переход к многоядерному оборудованию), которые играют роль бритвы Оккама: переусложнённые технологии начинают испытывать серьёзные затруднения (правда, у лидеров отрасли накоплен опыт по перекладыванию этих затруднений на плечи рядовых разработчиков, а более всего — конечных пользователей).

Если идёт речь о долгосрочной, основательной деятельности (а такому подходу особенно способствует необходимость самостоятельно решать проблему обучения — в ситуации, когда «сам не научишь — никто не научит»), то очень быстро приходит понимание того, что стоит за «принципом Калашникова». Подход, связанный с предугадыванием «всех случаев жизни» и закладыванием этого в системы ПО, неизбежно подводит. Напротив, единственно возможным оказывается улавливание существенных моментов, **инвариантов**, которые будут неизменны продолжительное время — и построением на их основе не «универсальных решений», а **компактных расширяемых каркасов, готовых к длительной эволюции** при постоянно изменяющихся требованиях.

Теперь вернёмся к ситуации в “mainstream”. Оказывается, что оба русла, стремящиеся слиться в ведущих программных платформах, мало приблизились к решению содержательных задач программирования, а

продолжают заниматься играми со своими внутренними искусственными (с точки зрения специалистов-предметников) конструктами, порождая неубывающую сложность, - и это не окупается вводимой дисциплиной средств и даже внедрением математичности. В этих условиях по-прежнему не видно альтернативы Оберону как экстракту важнейших принципов и средств программирования, инварианту, от которого можно выстраивать дальнейшее долгосрочное развитие. Можно и нужно анализировать его недостатки и пути будущего развития технологий программирования, но совершенно очевидно, что Оберон попал в окрестность некоторой точки оптимума (из этого следует, в частности, что усовершенствовать его малыми изменениями, не покидая этот локальный оптимум, невозможно) — и отрицать этот язык так же глупо, как отрицать в математике дифференциальное и интегральное исчисления.

Что мы находим в Обероне? Что же скрывается за минимумом средств этого семейства языков, что позволяет говорить об их особенном положении среди инструментов программирования? Кратко ответим на этот вопрос, опираясь на собственный богатый опыт использования Оберона (язык Component Pascal в среде BlackBox). Во-первых, это динамичность и интеграция языка и системы разработки-выполнения, аналогичная интерпретируемым языкам, но сочетающаяся с «не-игрушечностью», высоким качеством каждой детали и прозрачной компиляцией в машинный код.

Во-вторых, Компонентный Паскаль является идеальной системой для построения компонентных, эволюционирующих систем. Сама система BlackBox — живая иллюстрация компонентных паттернов (которые описаны в известной книге E. Gamma & ... “Design Patterns” [4]), сопоставимая в этом плане, например, со Smalltalk-системами. Набор средств современного модульного языка поддерживает мощные архитектурные решения, в том числе, не имеющие аналогов в mainstream (речь о родовой шине сообщений). Подробнее об обероновских архитектурах можно прочесть в [5], [6]. Важно,

что в язык и включено всё необходимое, и не внесены какие-либо «подводные камни», характерные для Java и .NET. (В этих системах существуют на первый взгляд безобидные упущения, которые обнаруживаются только при попытке выразить в них то, что прямым образом выполнялось на Обероне. В частности, паттерн «родовая шина сообщений» в Java вообще не может быть реализован эффективно, из-за отсутствия стековых полиморфных структур данных. В C#/.NET при попытке эмулировать модульность были допущены некоторые серьёзные дефекты.)

В-третьих, Компонентный Паскаль предоставляет выразительный базис для структурирования данных, в современном понимании. В этом ещё одно (после компонентных паттернов) преимущество языка, органично расширенного от структурно-модульного до ООП (без введения отдельных понятий класса и объекта). Проектные решения в сложном алгоритмоёмком ПО — это отнюдь не только объединение данных и поведения (вошедшее в моду с ООП), но и аккуратное их расслоение, при сохранении на всех уровнях полиморфности и виртуализации. Здесь Обероны с их расширяемыми записями и сопряжёнными средствами качественно превосходят Java и C#. При этом в полной мере проявляется преимущество автоматического управления памятью при работе со сложными алгоритмами и структурами данных (часто можно говорить о принципиальном влиянии сборщика мусора на алгоритмику). Можно даже говорить, как это ни удивительно, о наличии в Обероне существенных аспектов, характерных для ФП. Такое наблюдение основано на нашем конкретном опыте разработки на Обероне транслятора функционального языка Рефал-0 [7] (ранее его высказывал координатор Информатики-21 Ф.В. Ткачёв, работающий в области вычислительных методов в физике высоких энергий).

И последнее наблюдение: Оберон прекрасно выполняет роль мостика через пропасть между задачами и оборудованием (про которую писал в своих работах Э. Дейкстра), давая прекрасные абстракции как для одной, так и для другой стороны. Если мы будем рассматривать программирование как

продолжение прикладной математики, её замыкание обратно на «физическую реализацию», то Оберон выглядит очень естественным интеллектуальным средством, гладким продолжением мысли. Кроме прозрачности, надёжности и простоты здесь важна ещё возможность отображать идеи напрямую в конструкции языка, без долгого этапа подбора и «игр» на этом уровне. Это свойство Оберона повлияло не в последнюю очередь при выборе, который мы делали между ним и Адой (которая в плане продуманности и строгости также является языком высокого качества).

Кого и чему учить? В ООО «Метасистемы» имеется постоянный опыт вовлечения способных студентов в серьёзные проекты. Участие в разработке сложной программной системы «с нуля» (например, компилятора) оказывает огромное влияние на развитие начинающего специалиста. Разработка «с нуля» существенна — она позволяет показать проектирование и принятие технических решений в чистом виде, как оно должно быть «по уму», в то время обычная сегодня работа по сборке сторонних компонент примешивает множество случайных деталей, вызванных ошибками проектирования, обратной совместимостью, требованиями конкретных ситуаций в прошлом (проблема в том, что сами эти ситуации и их влияние для студента уже невидимы, остаются только «призраки» в виде непонятных деталей, которые постепенно запоминаются как норма вещей). Для разработки ПО «с нуля» с привлечением начинающих специалистов Оберон незаменим, т. к. позволяет сосредоточиться исключительно на мышлении, проектировании и воплощении идей в алгоритмах и программных структурах.

Однако имеется следующий парадокс. Наши студенты, прослушавшие основательные курсы по конструированию программ и уже получившие хороший опыт в 1-2 почти самостоятельных проектах, на голову выше по квалификации, нежели их собратья, занимавшиеся типичным «саморазвитием» массового программиста. Однако они проигрывают последним по ориентации в тёмных закоулках конкретных технологий. Это сказывается в проектах, которые должны опираться на своём нижнем уровне

на такие технологии. Здесь наши качественно обученные и способные студенты начинают «буксовать», тратя много времени на прояснение того, что познаётся большой практикой работы с этими технологиями (впрочем, такая практика не сильно полезна для умственного развития). Привлекать типовых специалистов нецелесообразно, т. к. основная часть работы состоит в закрытии платформы качественными абстракциями, которую они выполнить не могут. Свои сотрудники, способные качественно построить систему и одновременно имеющие за плечами большой опыт работы с массовыми платформами, оказываются немногочисленными и настолько ценными, что встаёт вопрос о целесообразности их задействования под конкретный проект. Конечно, способные новички быстро осваивают и любую конкретику, но на это впустую уходит драгоценное время.

Корень такого противоречия — не в образовании, а в сложившейся в отрасли ситуации. Эта ситуация проанализирована в аналитических статьях таких учёных, как Э. Дейкстра[1] и Н. Вирт[2]. Она же находится в центре внимания проекта Информатика-21[1], который говорит о «раковой опухоли избыточной сложности в ИТ» и ищет пути к её преодолению.

К чему приводит сегодня типичный процесс обучения по ИТ-специальности? В то время, когда требуется показать концепции, поставить мышление и умение ясно строить программные структуры, внимание студентов оказывается сосредоточено борьбе с дефектами конкретного инструмента. Наиболее слабые теряют интерес к развитию, сильные успешно преодолевают препятствия, но у них складывается превратное представление, что именно эта «возня с трубами разного диаметра» и составляет существо программирования. Если она увлекает и приносит удовлетворение (и самомнение), то такие студенты часто оказываются навсегда потеряны для серьёзного программирования. Они преодолели «полосу препятствий» и уже «вошли в касту». Нам приходилось наблюдать на специальности

1 Э. Дейкстра «Почему американская информатика кажется неизлечимой», «Смиренный программист», «Два взгляда на программирование» и др.

2 Н. Вирт «Долой жирные программы», «Преподавание информатики: потерянная дорога».

«Прикладная математика и информатика» уход в чистую математику тех, кто не воспринял такого программирования. Возможно, из них получились бы самые лучшие программисты. Информатика-21 имеет одной из целей делать качественных программистов из тех, кто в текущих условиях не хочет и не может в это погружаться (проблема обучения специалистов-предметников).

Так или иначе, переломить положение дел можно только в образовании. Ориентация на запросы индустрии с каждым годом усугубляет ситуацию. Уместно привести слова А.Н. Колмогорова, которые полностью приложимы к обучению программистов: «Нашей стране необходимо иметь много математиков-исследователей, способных делать открытия в самой математике и применять её нестандартным образом, требующим большой изобретательности. ... Откладывая вовлечение молодых людей в напряжённую научную работу, мы безвозвратно теряем многих из тех, кто мог бы сделаться творчески активным ученым.»

Используя Оберон, мы получили возможность вовлекать студентов, начиная с младших курсов, в серьёзное программирование, которое при правильном подходе к делу требует преимущественно острого ума и развитого абстрактного мышления.

Ссылки

1.Информатика-21 — http://www.inr.ac.ru/~info21

2.http://www.inr.ac.ru/~info21/texts/2006-09-SFO/v2public.pdf

3.И.Е. Ермаков. Проблема обучения программированию и пути её решения. - http://metasystems.ru/edu.php?page=pub

4.Э. Гамма и др. Приёмы ОО-проектирования. - СПб.: 2001.

5.И.Е. Ермаков. Оберон-технологии: что это такое? *

6.И.Е. Ермаков. Некоторые идеи архитектуры Оберон-систем. *

7.И.Е. Ермаков. Встраиваемый язык обработки текстов Рефал-0 и разработка его транслятора на Компонентном Паскале. *

* см. http://metasystems.ru/science.php?page=pub