УДК 004.855.3

В.Е.Казанов (4 курс, каф. ЭСиС), Е.Н.Попков, д.т.н., проф.

АРХИТЕКТУРА НАУЧНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Работа содержит ряд предложений по модернизации архитектуры уже существующего программного проекта – вычислительного комплекса «РИТМ».

Можно выделить несколько основных требований, предъявляемых к современному научному программному проекту.

Первое требование - простота поддержки и модернизации приложения. К разработке университетского приложения следует привлекать прежде всего студентов, специалистов только развивающихся, поэтому важно сохранять наглядность и ярко выраженную модульность архитектуры.

Практически ни одно научное приложение не может обойтись без элементов численных методов. Последние всегда требуют больших вычислительных затрат, что накладывает второе ограничение на вычислительное ядро - по скорости и, следовательно, по выбору средств разработки. Одновременно ядро, как наиболее трудоемкая составляющая, должно использовать традиционные технологии, заведомо не устаревающие.

Программа должна быть представлена по крайней мере на двух платформах: популярной настольной Windows и более технологичной бесплатной Linux. Первое необходимо студентам для домашнего использования, второе открывает широкие возможности распределенных вычислений, использования бесплатного легального программного обеспечения.

Большая часть существующих приложений не удовлетворяет ни одному из перечисленных требований: одни созданы для архаичных платформ, исходный код вторых утерян и не может быть модернизирован, третьи изначально создавались как временное негибкое решение и, соответственно, их разработка не может быть качественно продолжена.

Существует несколько основных вариантов выбора технологии разработки и/или архитектурного решения.

Последнее время большую популярность приобрели языки высокого уровня, использующие виртуальную машину для обработки приложения (фреймворки JDK и .Net). Приложения, основывающиеся на подобных решениях, обладают двумя основными преимуществами: быстротой и простотой разработки, свойственной языкам высокого уровня; независимостью от платформы, гарантированной концепцией виртуальной машины.

Главный недостаток - существенная потеря производительности (в двадцать и более раз) по сравнению с классическими компилируемыми языками. Их целесообразно использовать в разработке интерфейсов, сетевых приложений и любых других программ, в которых производительность не является критичной.

Обычным решением является использование инструментария сред Visual Studio/Borland Developer Studio для создания моноплатформенного приложения, или QT/GTK для многоплатформенных программ. Такая методика означает создание технологически монолитного приложения, ограниченного возможностями и лицензией, предоставленными компанией-производителем.

Дополнительным недостатком любой фирменной технологии для проекта, рассчитанного по крайней мере на несколько лет, является корпоративная гонка технологий, заставляющая производителей инструментариев ограничивать поддержку прежних разработок в пользу новых.

В свете приведенных выше соображений имеет смысл разбить приложение на две составляющие, фактически разные программы: вычислительное ядро с минимальным интерфейсом и вспомогательные интерфейсные программы. Ядро должно быть

разработано на одном из классических компилируемых языков (C/C++ или Fortran) с использованием средств библиотек, описанных в ключевых стандартах. Эти языки обладают наилучшей производительностью, представлены на большом количестве платформ, и для них разработано множество компиляторов.

Наличие интерфейсной части только в простейшей форме командной строки существенно облегчает переносимость приложения и включение ядра в технологически неоднородные программные комплексы. Удобно при этом использовать бесплатный кроссплатформенный набор компиляторов - gcc, представленный на огромном количестве платформ.

На пользовательский интерфейс не накладывается никаких ограничений. Возможно, к примеру, подключение к ядру веб-интерфейса или создание внешне монолитной программы с использованием любых специфичных для данной платформы средств.

Интересной возможностью представляется разработка распределенной по сети вычислительной системы на основе подобных вычислительных ядер, или запуск в пределах одного многопроцессорного компьютера нескольких ядер.