НАУКА И ЖИЗНЬ®

Копия текста со страницы http://www.nkj.ru/archive/articles/19595/

В гонке технологий победит тот, кто лучше пишет программы и считает инженерные процессы. Часть II: Считать по-русски.

Наука и жизнь №6, 2011 г.

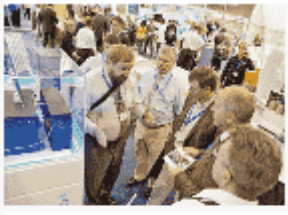
Елена Вешняковская.

Переславль. Вид на Плещеево озеро и Институт программных систем РАН.

Директор Института программных систем РАН Сергей Михайлович Абрамов с платой суперкомпьютера.

Негосударственный Университет города Переславля стремится дать студентам возможность работать с компьютерными технологиями завтрашнего дня.

Человек и суперкомпьютер.



Международная выставка в Гамбурге: суперкомпьютеры Сергея Абрамова вызывают живой интерес.

С одной стороны, переход к компьютерному проектированию и анализу неизбежен. С другой, стратегический выбор выглядит не очень привлекательным: либо российская промышленность продолжает технологически отставать, либо попадает в зависимость от программных инструментов, которые мы ещё не умеем делать сами. Есть ли третий путь? По мнению Сергея Михайловича Абрамова, директора Института программных систем им. А. К. Айламазяна РАН, оптимальная стратегия заключается в том, чтобы тонко пролавировать между тем, что в бизнес-науке называется «угрозами» и «возможностями», принимая для каждого конкретного случая свое, «сшитое на заказ» решение. Однако цена таких решений слишком высока, а требования к их компетентности слишком серьёзны, чтобы целиком доверить их управленческой государственной вертикали. Поэтому в Европе, например, в качестве инициатора таких решений и посредника между техноёмкими отраслями и государственными чиновниками используются специальные общественные некоммерческие структуры, объединяющие профессионалов отраслей и социальных институтов с общими сферами интересов. Эти объединения называются технологическими платформами. Правительство сочло европейский опыт интересным, и 1 апреля нынешнего года российская Комиссия по высоким технологиям и инновациям рассмотрела предложения профессиональных сообществ и подписала постановление о внесении в правительственный перечень 27 отечественных технологических платформ. В их числе Национальная суперкомпьютерная технологическая платформа (НСТП), которую в течение года кропотливо выстраивал Институт программных систем РАН. Концепция, которая легла в её основу, нацелена на то, чтобы выиграть время в технологической гонке и при этом не попасть в зависимость от чужих лицензионных технологий. Директор Института программных систем РАН, член-корреспондент РАН Сергей Михайлович Абрамов раскрывает детали.

Строить мост вдоль или поперёк?

— Сергей Михайлович, что такое технологическая платформа и сколько она захочет бюджетных денег?

— Нисколько. Технологические платформы — это своего рода клубы по интересам, саморегулируемые общественные организации, которые завязаны на определённую технологическую проблематику. Такой клуб может быть широким или специализированным. Например, Национальная программная платформа (НПП) объединяет всех, кто заинтересован в программном обеспечении. В сферу её интересов попадает программное обеспечение и для серверов, и для наладонников, и для айфонов; игры, антивирусы, инженерный расчёт, научный расчёт — это гигантская сеть. Она может пересекаться с другими сетями, это только увеличивает её функциональность; например, наш институт одновременно и участник

Программной платформы, и инициатор отдельной, более специализированной Суперкомпьютерной технологической платформы, где собираются люди, которые заинтересованы в суперкомпьютерных технологиях (в аппаратной и программной части, приложениях, сервисах и так далее). Платформа — это зародыш отраслевой сети, её задача — связать всех заинтересованных игроков между собой естественным, горизонтальным путём, а не министерским — не через управленческие вертикали. Это представительное профессиональное собрание, коммуникационная площадка, где каждый игрок может высказать своё мнение, быть услышанным, оспоренным или поддержанным. В рамках российского законодательства наиболее близкий духу этой структуры юридический формат — некоммерческое партнёрство.

— А где здесь интерес правительства?
— Попробую объяснить на примере суперкомпьютеров, и это будет верно, думаю, для всего остального тоже. У нас высокотехнологичная отрасль. Игроков в ней много. Интересы разные, конфликтующие. У каждого серьёзного игрока есть своя группа влияния, свои выходы на правительство. И все эти группы и ресурсы влияния попеременно бегают в правительство и убеждают. Одни, что мост надо строить вдоль. Другие — что поперёк. Третьи — что вообще в другом месте. Деньги тают, а ничего интересного на выходе нет, потому что нет синергии, концентрации усилий на стратегических направлениях. И тогда правительство предлагает: «Ребята, соберитесь. Соберитесь куда-нибудь все. Соберите хорошую платформу: такую, которая будет равномерно представлять все возможные отраслевые интересы и все конкурентные точки зрения. Тогда, если вы там о чём-то договоритесь и вынесете на наш уровень, мы готовы к этому прислушиваться. Нам надоели группы влияния. Мы хотим работать с отраслью в одно окно». Вот в этом и состоит интерес правительства. Общие правила «платформостроения» задаёт рабочая группа, которую возглавляет замминистра экономики. Там много дельных людей, они сформулировали совершенно разумные требования к платформам: хороший охват всех категорий игроков, хорошая проработка того, чем будем заниматься, как, какие проблемы, чего стоит их решить, какая нужна поддержка...

— То есть в итоге платформа всё-таки захочет денег?
— Нет. Деньги через неё провести нельзя.

— Как тогда концентрировать ресурсы на стратегических направлениях?
— В Европе это делается так. Технологическая платформа готовит заключение: прогноз на перспективу; ближайшие цели; планы их реализации. Обсудив это внутри себя, участники выносят документ на правительственный уровень в виде экспертных предложений: надо делать такие и такие конкретные программы. Если правительство доверяет платформе, оно формирует программы и предоставляет им ту или иную поддержку, не обязательно финансовую, возможно — просто льготы. Затем для реализации программы формируется консорциум коммерческих предприятий, это будут уже руки, выполняющие практическую часть задачи. Но сама платформа — не руки и даже не голова, это просто язык...

— ...который озвучивает интересы коммерческих игроков; своего рода инструмент коммерциализации государственной стратегии.
— Не только коммерческих. Проблема модернизации заключается в том, что существуют четыре крупных института, чьи интересы должны где-то встретиться. Эти четыре института: бизнес, которому технологии нужны, чтобы получать прибыль; правительство, работающее в интересах всех своих налогоплательщиков; наука и, четвёртое, образование — потому что наша наука существует от образования отдельно. Площадкой для их встречи служит технологическая платформа.

— Своего рода отраслевой парламент?
— Да, только вход в него открыт для каждого игрока. Понимаете, мы сделали очень хорошую, без компромиссов, платформу. Туда вошли западные фирмы и СНГ — Белоруссия, Казахстан, Украина, туда вошли непримиримые конкуренты — Intel и AMD, Microsoft и «Альт Линукс»... Из отечественных организаций тоже удалось привлечь таких людей, которые зачастую в одном месте просто собираться отказываются. Это абсолютные антагонисты, но все они в составе платформы. Потому что надо договариваться. Не в жульническом смысле «договариваться», а говорить и быть услышанным.

— А что там делают западные компании?
— Это же коммуникационная площадка. Вы хотите, чтобы вас слышали? Входите. Или, пожалуйста, можете стоять за дверью, мы тут сами пообщаемся.

Терафлопсы — или диктатура заказчика?

— Зачем вообще такие сложности? Почему бы не создать «Министерство модернизации», и пусть оно централизованно тратит деньги налогоплательщиков в стратегических интересах страны?

— Если вы будете на бюджетные деньги создавать вещи, которые окажутся никому не нужны, это не прогресс. Если мы строим суперкомпьютер, то не для того, чтобы войти в Top500, а чтобы решить конкретную проблему заказчика. Логика суперкомпьютерного процесса заключается в том, что есть отрасли — они могут быть самыми разными: производство, экономика, образование, государственное управление, — которым для нормальной работы нужно считать, и считать хорошо. Но формулировать запросы они могут только на языке своих предметных областей. «Мне нужно новое лекарство», «нужен двигатель», «нужно видеть, сколько понадобится рабочих мест». Рядом с отраслевиками должны стоять матмодельеры, которые превращают их запросы в математические модели. За ними — программисты-алгоритмисты, которые реализуют эти численные схемы в виде алгоритмов и, в свою очередь, выдвигают требования специалистам следующего уровня: по системному и прикладному программному обеспечению. Те выдвигают требования на аппаратную часть, те — на элементную базу.

— «Диктатура заказчика»?

— Не совсем диктатура, есть и обратная связь. Специалист на любом уровне может обернуться к заказчикам и сказать: «Ребята, доказано, что эта проблема алгоритмически неразрешима, так что давайте пересмотрим свои «хотелки» (на программистском жаргоне — требования заказчика к продукту. — Прим. ред). Либо аппаратчик скажет: «Ребята, всей душой бы рад, но скорость света равна тому, чему она равна, и ничему другому. Поэтому извините, но давайте опять-таки пересмотрим...» Такая «слоеная» структура отрасли решает ключевую задачу: она обеспечивает каждому уровню производства заказчика, сбыт. Посмотрите на реестр участников Национальной суперкомпьютерной платформы — там не только те, кто делает суперкомпьютеры, но и те, кто их использует, кто владеет ими, кто предоставляет сервис.

— А что делает тем временем высшая школа?

— Пронизывает все семь уровней отрасли. Нужны аппаратчики, нужны специалисты-программисты, матмодельеры, эксплуатационщики, нужны те, кто предоставляет сервис и заполняет разрыв между заказчиками, не знающими, что такое расчёт, и переводчиками их запросов в матмодели. Образование должно готовить кадры конкретно для каждого из уровней, под его конкретные потребности. Например, специалистов по системам охлаждения сейчас в России просто нет...

— Поможет ли переход на болонскую систему решить проблемы специализации? Например, почему бы системы охлаждения не проектировать бакалавру?

— Дело не в делении на бакалавров и магистров. Российское образование отличалось от западного не только фундаментальностью, но и критическим импульсом, отсутствием авторитетов. Например, то обстоятельство, что в российской высшей школе всегда заставляли студентов учить доказательства, можно рассматривать как формализм, но для меня оно означает, что прививали навык не верить на слово, требовать доказательности. Своих студентов я учу совершенно не по-болонски. Например, в курсе про сети телекоммуникаций на западе им бы рассказывали, какой пакет имеет какую длину, какой байтик и битик что означают. А я рассказываю, с какими техническими проблемами столкнулись разработчики того или иного продукта, и заставляю студентов искать решение. И они в конце концов находят решение, которое нашли те. То есть учатся решать проблемы, а не просто запоминают факты. Отраслевая суперкомпьютерная платформа накладывает на образование определённые требования. Высшей школе предстоит не только транслировать студенту «неувядающие знания», но и учить его решать задачи с опережением, такие, которых ещё никто не решал; знакомить с технологиями, которых ещё нет.

— Но как?

— А как люди вообще учатся что-то делать? Берут и начинают делать. Конечно, нужна хорошая фундаментальная база, это незыблемо. Её получают в университетах, на лекциях и семинарских занятиях классическим способом — классическое образование в каждой отрасли своё. Для компьютерных технологий, например, — математика, хорошая дискретная математика, крепкая алгебра; кому-то ещё матфизику, вычислительную математику — но уже не обязательно всем. А вот следующий этап — учёба в бою, стажировка. Студентов надо погружать туда, где возникают эти технологии, привлекать к их созданию. В советской высшей школе это называлось базовыми кафедрами, и у себя в Переславле-Залесском мы осуществили эту модель: помимо Института программных систем у нас есть маленький негосударственный вуз — Университет города Переславля. Для его студентов не существует понятий «учебная задача» или «упрощённая постановка задачи»; они создают информационные

технологии завтрашнего дня и выходят из вуза уже актуальными специалистами; владеют тем, что создано сегодня, сейчас.

Покупать — или делать своё?

— Чтобы от суперкомпьютеров был толк, на них должны стоять программы, которые хорошо считают. Расчётная инженерная программа — это стратегический ресурс. Что делать с технологической зависимостью от западных лицензий?

— Во-первых, нельзя сказать, что отечественных расчётных пакетов нет совсем. Есть «Тесис», с очень конкурентоспособным продуктом для гидродинамики, который хорошо продаётся во всём мире. Есть ГДТ, это Тула, газодинамику быстротекущих процессов, всё, что связано с выстрелами, они считают просто замечательно, лучше многих. «Рок Флоу Динамикс» занимается расчётами для нефтегазовой отрасли. Есть конкуренция и на западе: выпускается много специализированных программ. Ведь инженерных задач много, и они очень разные. Есть задачи по деформации, упругой и неупругой. Есть гидродинамика, турбулентность, массотеплоперенос, горение — словом, множество процессов, для каждого существуют свои расчётные продукты.

— А если вам нужно посчитать двигатель внутреннего сгорания? В нём есть и механическая часть, и химическая, горение, течение, впрыски, выпуски, акустика... Как считать одно изделие десятью разными пакетами?

— Писать комплексирующие программные надстройки или использовать те, что уже есть. Они объединяют разные пакеты в то, что пользователем будет восприниматься как единый продукт. В рамках программы СКИФ-ГРИД, например, у нас была прекрасная отечественная работа, сделанная пятью участниками: СПбГУ, «Тесис», «Урал-Грид», ЮУрГУ — Южноуральский университет, головной, и «Сигма-технологии». Замечательные ребята: они интегрировали разные пакеты, включая западные (кстати говоря, и Ansys тоже) в гридовые оболочки.

— Но если вы хотя бы на один пакет в составе такого продукта получаете лицензию Ansys, то обязуетесь соблюдать его условия: в частности, ограничиваете номенклатуру пользователей своего продукта только теми организациями, которые у Ansysa в «белом списке». Нужны ли нам технологии такой ценой?

— Но ведь не Ansys'ом единым! Кроме него есть ещё море разрозненных специализированных продуктов: и западных, и отечественных. Главное — мы умеем делать над ними комплексирующую надстройку. Причём «Сигма-технологиям» удалось совместить комплексирование с многокритериальной оптимизацией, и я не вижу на Западе аналога их продукту.

— Можно пойти по пути связывания мелких пакетов в работающую систему, но ведь можно и найти одного сильного исполнителя, желательно из оборонной отрасли, и заказать ему «свой большой русский Ansys»?

— Разумеется. Откуда, по-вашему, сам Ansys? Один из очень известных ансисовских инженерных пакетов называется LS-dyna. Знаете, что означает LS? Ливермор — Сандия. То же самое, что для нас, скажем, Саров — Снежинск. Разработчики не собираются скрывать оборонного происхождения своего продукта, напротив, демонстрируют его. Когда-то американские суперкомпьютерные центры Ливермор и Сандия получили бюджетные деньги на цифровое моделирование ядерного оружия. Это задача очень сложная, она включает в себя и упругость, и деформацию, и тепломассоперенос, и обтекание — всё на свете. По ходу проекта накопилось огромное количество специализированных кодов. За много лет эти коды были «вылизаны», оптимизированы, доведены до определённого уровня совершенства. Затем возникли коммерческие компании, которые получили их в собственность и довели до коммерческого использования. Так это делается и работает, поэтому нам у себя никак нельзя игнорировать опыт наших крупнейших расчётных центров.

— Мы и не игнорируем. Проект по развитию суперкомпьютеров и грид-технологий на базе Саровского ВНИИЭФа, уже профинансирован и Росатом отчитался о завершении первого этапа. С одной стороны, проект нацелен, в числе прочего, на передачу в «мирные» или «почти мирные» отрасли расчётного потенциала, накопленного отечественной оборонкой, с другой — немного смущает, что программные пакеты поставляются только на саровских же суперкомпьютерах. Почему не отдельно?

— Возможно, государство пытается продвинуть на отечественный рынок стопроцентно российские продукты. Скепсис по этому поводу, конечно, высказывался, зато можно быть уверенным, что в Сарове умеют считать многие задачи лучше других. Просто исходя из здравого смысла: ядерное оружие есть, с ним всё в порядке, значит, считать его создатели умеют. Потому что, не умея считать, его не построишь. Значит, и готовый код у них уже имеется. Везде ли он применим? Наверняка не везде. Насколько он применим конкретно в судостроении, или у Сухого, или на КамАЗе? Как раз это и предстоит узнать. Организации, получившие от Сарова персональные суперкомпьютеры, инженерные пакеты и доступ к более мощной машине, заняты сейчас верификацией программ: проводят расчёты с моделью, потом эксперимент с изделием и сравнивают результаты: какое

отклонение? Насколько можно этому пакету доверять? Кроме того, они сравнивают саровские пакеты с западными, и, судя по тому, что видел я, саровские демонстрируют очень достойное качество. Некоторые вещи они считают лучше западных — не вообще всех, а тех, на которые нам дают лицензии.

— Нет ли в обороноцентрическом подходе риска, что модернизация в России снова ограничится военными и имиджевыми достижениями и так и не доберётся до лекарств, велосипедов и соковыжималок? Отдельно взятую гонку вооружений мы уже проходили и знаем, что она заканчивается полным экономическим истощением системы.

— Сегодня всем очевидно, что технологическая независимость не сводится к вооружениям; она обеспечивает состязательность по самому широкому экономическому фронту. На западе суперкомпьютерные технологии рассматриваются однозначно: как единственный инструмент получения конкурентного превосходства во всём. Председатель американского Совета по конкурентоспособности (представительная общественная организация, которая объединяет технологических и бизнес-экспертов США и даёт стратегические советы правительству. — Прим. ред.) Дебора Уинс-Смит сказала буквально так: «Ресурсы, деньги и таланты есть у многих стран. Поэтому США сталкиваются с неожиданными экономическими вызовами. Страна, которая побеждает в вычислениях, побеждает в конкуренции (The country that outcomputes, outcompetes)». Обратите внимание, это сказал не программист и не электронщик, а экономист. А теперь вопрос: нам продадут инструмент победы в конкурентной борьбе? Альтернативы выстраиванию собственной вычислительной отрасли нет. То есть теоретически есть: можно, например, решить, что особый путь России — это экстремальный туризм, и строить свою экономику на нём. Но если у нас есть какие-то амбиции на уровне G7, нам придётся жить немножко по-другому.

— Амбиции должны соответствовать потенциалу. Реалистично ли — в одночасье создать такой же расчётный потенциал, какой «семёрка» нарабатывала десятилетиями?

— Нам совершенно не обязательно целиком проходить тот же самый путь.

— А где можно «срезать»?

— Расчётные инженерные пакеты устроены многослойно. В них есть компоненты, которые могут заменяться от задачи к задаче; в конце концов всё сводится к системам линейных уравнений и поэтому работает. Например, один из компонентов пакета — солвер, «решалка», которая решает эти системы уравнений. Насколько хорошо пакет считает, то есть на сколько процентов посчитанное в модели расходится с практикой, — зависит от качества солвера. И конечно, нам никто не продаст солвер, ориентированный на двигателестроение, вообще, ни по какой цене. Но это только половина правды. Вся правда в том, что эти солверы, вообще никто никому не продаст. Их придерживает не государство, а конкретные фирмы, которые разработали их для собственных целей. У General Electric — свой солвер, у «Боинга» свой, у всех серьёзных игроков в промышленности свои. Организация создаёт под себя инструмент своего конкурентного превосходства — и втыкает его в стандартную покупную платформу. В нашем правительстве сейчас борются два подхода: купить технологии или делать свои? Правильно и то и другое: что можно выгодно купить, надо покупать, а что нельзя купить, то необходимо делать самим и комбинировать с покупным.

«По любви» или за деньги?

— Весь двадцатый век Россия только и делала, что кого-то догоняла и перегоняла. Есть ли в вычислительных технологиях ресурс, который бы позволил двигаться не «вдогонку», а, так сказать, «перпендикулярно»? Где в индустрии инженерных программ вы видите потенциал для появления каких-то новых козырей, которые позволят пересмотреть правила игры?

— На рынке инженерных расчётов есть один игрок, которого нельзя сбрасывать со счетов: это пакеты с открытым кодом, свободное инженерное обеспечение. Оно существует и развивается. Несомненно, в недалёком прошлом оно проигрывало коммерческому по качеству, но чем дальше, тем более становится с ним сопоставимо. С учётом того, что пакеты состоят из взаимозаменяемых компонентов и там неплохая стандартизация, скоро, наверное, можно будет собирать на их основе вполне функциональные программные гибридики: опираясь на это свободное программное обеспечение и дополняя его сильными отечественными разработками, делать собственные пакеты. В суперкомпьютерную технологическую платформу пришло много людей, которые считают использование открытого ПО перспективным, и я разделяю это мнение.

— И Великобритания, и многие страны в Европе официально объявили открытые ресурсы основой своих государственных IT-стратегий. Таким образом правительства демонстрируют экономию бюджетных средств. Но экономия не равна технологической независимости. Что на самом деле могут нам позволить открытые коды и в чём они будут нас лимитировать? В частности, имеем ли мы право продавать продукты, сделанные на основе открытого кода?

— То, что продукт на основе открытого кода нельзя коммерциализовать, — одно из самых распространённых заблуждений. Существует целый веб-ресурс, посвящённый исключительно росписи лицензий открытых продуктов: какая лицензия что позволяет. Лицензий сотни, возможностей — ещё больше. Если в открытых кодах и есть что-то ограничивающее, то это так называемое свойство «вирусности», или «зара-зительности», лицензий Линукс (самый известный и мощный из существующих в мире открытых кодов. — Прим. ред.). Всё, что включает их продукт, должно тиражировать эту линуксовую лицензию. Но она не запрещает коммерческое использование Линукса; и вообще коммерческая и открытая продукция не настолько непримиримы между собой, чтобы одно исключало другое. У меня, например, было два контракта с Microsoft, мы писали для них системную программку — и по условиям контракта должны были выпустить её под лицензией FreeBSD: в качестве маленькой бесплатной вишенки, которая полагается тому, кто купит у Microsoft большой торт. С другой стороны, на Линуксе можно строить бизнес: как по продаже программных продуктов, так и в области сервиса, обслуживания и так далее.

— А почему вообще существует открытый продукт? Это же всё равно, что вырастить в своём саду яблоки и начать их раздавать бесплатно.

— Пример с яблоками не работает. После того как программный продукт создан, его тиражирование практически ничего вам не стоит. Если бы, вырастив одно яблоко, вы смогли дальше без затрат размножать его до бесконечности, не исключено, что вы бы тоже стали их раздавать. Несомненно, всё началось с того, что Торвальдс, основоположник Линукса, написал операционную систему для себя, не думая ни о глобальной известности, ни об экспансии, ни о доходности. Сделал, она ему понравилась, он получил удовольствие. Тогда он пригласил желающих пользоваться ею бесплатно, развивать её и поддерживать Линукс-движение. В этом движении до сих пор очень сильна идеологическая составляющая.

— В чём она заключается?

— Хорошее программирование приносит эстетическое наслаждение... Кстати, это правда, это действительно так: программист, которому удалось хорошо написать программу, получает огромное удовольствие. Чрезмерно наживаться на этом и открыто жадничать — как бы «нехорошо».

— А рациональные достоинства у открытого кода есть?

— Все участники движения получают гигантские преимущества перед коммерческим программированием. Обновления выходят чаще, чем коммерческие, качество проработки кода гораздо выше.

— Потому что на энтузиазме люди работают лучше, чем за деньги?

— Потому, что этих людей очень много. Назовите мне хотя бы одну коммерческую компанию, в которой число сотрудников исчислялось бы миллионами! А линуксовых программистов миллионы, они умные, они работают не за страх, а за совесть, для себя. Поэтому всё происходит стремительно: не успела появиться новая железка, мгновенно кто-то пишет под неё драйвер. Наш Институт программных систем РАН давно включился в это движение, мы стараемся опираться на него и работать в его рамках.

— Программисты, особенно энтузиасты, — это отдельный «этнос», обитающий поверх административных границ. А мы всё-таки думаем государственными интересами. Насколько актуально для открытого кода понятие госграницы — той, которая должна быть «на замке»?

— Вряд ли оно актуально для каждого отдельного участника движения, но у нас имеется компания «Альт Линукс», участник и нашей платформы, и программной, у которой очень сильный козырь — собственная, российская платформа сборки. Дело в том, что все многочисленные клоны Линукса базируются на репозитариях, куда программисты присылают свои изменения и дополнения. Ведь миллионы людей не просто пользуются Линуксом, но и постоянно что-то в нём правят и дописывают, поскольку открытый код это позволяет. Репозитарий — это точка сборки всех изменений. Ею могут быть организация, сервер, сотрудники — инфраструктура, куда присылают исправления, а она тестирует их, отделяет, так сказать, зёрна от плевел и время от времени выпускает новый дистрибутив — программное обновление. «Альт Линукс» располагает инфраструктурой со своим программным обеспечением, которая физически находится в России. Это важно с точки зрения информационной безопасности: случись что, при любом раскладе этот ресурс остаётся нашим. Более того, когда мы сами принимаем изменения, то точно знаем, в чём они заключаются. Можно обсуждать, плохо это или хорошо с идейной точки зрения, но мне важно, что у «Альтов» есть свой инструмент сборки и свой инструмент тестирования.

— Развитие кода — долгий, сложный процесс, и обычно чужие коды никто дописывать и править не любит — неизвестно, какие вылезут проблемы. Как Линукс-движение ухитряется работать над Линуксом в миллионы рук?
— Это специфика открытого кода. Его общий объём — десятки миллионов строк; действительно, ни один нормальный человек в таком тексте не разберётся. Но если установить структуру кода, поделить его на модули и чётко специфицировать интерфейсы между ними — как эти модули взаимодействуют, — то объём каждого модуля становится разумным: одна-две тысячи строк, иногда даже сотни. С ними уже вполне можно работать. Правки небольшие, исходный текст небольшой. Сборщик тестирует правку и либо принимает её, либо отвергает, либо говорит: «Да, такая правка нужна, но она делается не так, а как, я сейчас покажу». И переписывает по-другому.

— То есть мало того, что Линукс огромен и начисто вылизан от ошибок, он ещё и особенно хорошо управляем?
— Управляем, стабилен и располагает развесистой кухней, которая называется многопользовательской системой управления версиями. Любое количество соавторов может работать над одним и тем же документом вместе или утащив его каждый к себе. Аналогичная система используется, чтобы создавать пользовательскую документацию.

— Но какие-то части Линукса написаны не у нас. А вдруг «мировая закулиса» спрятала там что-то нехорошее?
— Во-первых, этот код открыт, прочитать его потенциально можно. Но, во-вторых, не нужно. Потому что его уже читали миллионы людей, причём не просто людей, а фанатиков. Конечно, не все энтузиасты линукса фанатики, но даже если их 10%, то, появись в коде хотя бы одна сомнительная строчка, они подняли бы вой на всю вселенную. Эти люди пользуются термином «копилефт» вместо копирайта, они помешаны на абсолютной свободе.

— Итак, в лице Линукс-движения мы имеем огромный штат, высокую информационную безопасность, возможность, если что-то не нравится, самим поправить...
— ...и практически единственный путь России в области программного обеспечения. Сейчас объясню почему. Если взять софт как явление, то 99% его — это рутинная работа, на уровне техникума. То, что в нашей среде называется «делать табуретки». И вот в этой части мы никогда не сможем конкурировать с остальным миром.

— В производстве «табуреток»? Почему?
— В силу ограниченных ресурсов. В Индии, в Китае человеческий ресурс исчисляется миллиардами, и там налажен конвейерный колледжный — слава богу, что они пока остановились на нём, — выпуск изготовителей «табуреток». Эти специалисты работают как коммерчески, так и над открытыми кодами. В результате все рутинные компоненты в открытых кодах сделаны очень и очень хорошо, их можно брать как есть. А модульность кода позволяет нам вставить туда что-то своё; отдельный компонент, который мы придумаем и сделаем не на уровне табуретки, а на уровне наукоёмкого шедевра: вложив мозги, образование и критическое мышление. И так сумеем поднять весь пласт. Это же касается инженерных расчётов. Если мы обопрёмся на открытый софт, а наукоёмкие компоненты сделаем сами, использовав опыт национальных лабораторий, которые годами развивали довольно качественный код, всё может получиться очень неплохо.

Гонка технологий или борьба бюрократий?

— Обычно царство здравого смысла и интеллекта существует до появления бюджетных денег. А как только пойдёт первая копейка...
— У института есть опыт и хорошая репутация. Конечно, не миллиарды, но 700 миллионов нам удалось потратить, ведя себя прилично: делать дело, а не перетягивать одеяло.

— Кстати, об одеяле. Чем крупнее организация-разработчик, тем больше риск, что финансирование растворится в её жизнеобеспечении, а до стратегического острия добежит только его несущественная часть. Как крупной РАНовской структуре — Институту программных систем — удаётся концентрировать ресурсы на задаче, не перегружая её накладными расходами?
— С помощью проектной, грантовой системы и соотношения 20/80. В ИПС РАН всё устроено предельно просто: из любого гранта 20% идёт на содержание института, 80% — исполнителям. При этом исполнители обеспечены всем, начиная от света и тепла и заканчивая транспортом. Из Переславля в Москву, в Тверь, в Ярославль — они берут машину и едут, не думая, что такое оплата работы водителя, бензин, техобслуживание и прочее. А 80%

финансирования они распределяют сами, как считают нужным, и я не имею права вмешаться ни в одну статью. 20% накладных расходов — это много?

— Многие коммерческие организации могут этому только позавидовать.
— Это — Российская академия наук. Она очень разная, но с нашей стороны она такая. Так было в институте до моего директорства и так будет после, потому что всё остальное не работает, это тоже точно известно. Если сказать: ребята, отдавайте на структуру 25% — всё. С проектами можно будет проститься.

— Насколько сложна нормативная база национальных технологических платформ?
— Предельно проста: постановление правительства номер четыре «Порядок формирования реестра технологических платформ». Это означает: технологические платформы создавайте сами, как хотите, мы регулируем только процесс регистрации. Телефонное право, конечно, никто не отменял, но правительство хотя бы попыталось встать в стороне. Чиновников у нас любят критиковать, и часто есть за что, но на самом деле иногда — не так уж редко — они действительно хотят сделать как лучше.

— Сенсационное заявление.
— Я понимаю. Но я купился именно на это. Я потратил год жизни на создание платформы, потому что верю, что они действительно хотят как лучше, и обратного пока не доказано, хотя, конечно, видно, что подковёрные игры идут.

— Со стороны процесс «платформостроения» выглядит невыносимо бюрократическим, но безупречная бюрократия — это попытка «победы легитимности над подковёрностью». В каком состоянии суперкомпьютерная платформа сейчас?
— Для нас был крайне важен процесс легитимизации: поэтому он и занял год. Сначала группа семерых, фактически «самозванцев», стартовала с идеей платформы. Затем мы собрали 120 организаций и провели конференцию, в которой участвовали больше 30 из них, — согласитесь, это уже легитимнее, чем семеро. На данный момент платформа объединяет более 180 участников, почти половина из них — коммерческие организации самого разного масштаба и направленности. Сейчас, перед общим собранием, приём временно приостановили, чтобы подстраховаться от странных вбросов в последний момент, но когда оно пройдёт, доступ снова будет открыт.

— Странные вбросы — это?..
— Культура внепроцедурного давления в принятии государственных решений пока ещё очень сильна.

— Но, наверное, чем легитимнее и представительнее «отраслевой парламент», тем труднее будет отодвинуть его от принятия решений?
— На это вся надежда. Если платформе удастся ориентировать суперкомпьютерную инфраструктуру не на гонку за терафлопсами, а на решение отраслевых задач, у нас есть шанс превратиться в производящую, конкурентоспособную страну.

Редакция благодарит за предоставленные иллюстрации Институт программных систем им. А. К. Айламазяна РАН, ООО «Тесис» и коллектив разработчиков FloEFD.

***

Чем он только думает?

Сергей Боченков, кандидат химических наук, руководитель IT-отдела проектной компании «СК КО-Восток».

Всё, что вы хотели знать об интеллектуальной начинке своего железного друга, но стеснялись спросить.

Число можно записать в двоичном коде: с помощью нулей и единиц. Ноль можно представить как отсутствие, а единицу — как наличие напряжения.

Это сделало возможными электронные вычисления: запись числа и операции над числами посредством физических процессов в полупроводнике. «Интеллект» компьютера начинается на нижнем, аппаратном уровне («договариваться» с которым по-человечески крайне неудобно) и вот уже много лет эволюционирует в сторону человеческой логики с помощью всё новых слоёв программных надстроек.

Первый уровень «компьютерного интеллекта», который выходит за пределы аппаратной части — уровень драйверов. **Драйвер** — это программная прослойка между программой и аппаратным обеспечением, которая позволяет программисту управлять устройствами любой сложности с помощью «человеческих» команд: не на уровне электрических сигналов, подаваемых на элементы платы, а на уровне логики. Представьте на месте компьютера лифт. Если бы, чтобы ехать, пассажиру нужно было переключать рубильники, подавать на одну обмотку мотора одно напряжение, на другую — другое, какое-то устройство запускать, другое заранее тормозить, чтобы лифт успел остановиться на нужном этаже, он бы как минимум переутомился. Поэтому драйвер лифта (компьютерной платы) дополняет устройство кнопками с номерами этажей (набором команд).

Следующий уровень — **операционная система**. Это программа, которая может интегрировать в себя множество драйверов, в неё зашиты возможности «на все случаи жизни». Почему, например, Windows такая большая? Потому что там очень много того, чего один отдельно взятый человек никогда не использует. Благодаря операционной системе конечные пользователи — обычные люди, не специалисты — могут уверенно работать с логикой папок, файлов и отдельных программ, не вникая ни в конфигурацию компьютера, ни уж тем более в то, как микросхемы передают друг другу электрические сигналы.

Пользовательские программы, которые находятся над операционной системой, — ещё дальше от машинной логики и ещё ближе к человеческой: к логике специалиста. Например, бухгалтеры ничего не знают ни о компьютере, ни об операционной системе, но если запустить им бухгалтерскую программу, будут в ней прекрасно работать, пользуясь своей бухгалтерской логикой.

Платформа — это программа, которая позволяет устанавливать на неё что-то ещё. Строго говоря, операционная система — это тоже платформа, потому что на неё устанавливаются все остальные программы, но чаще имеется в виду платформа специализированная, на которую можно в разных комбинациях устанавливать готовые модули для решения узких задач. В платформе есть ядро, куда эти модули втыкаются, — аналогично материнской плате, куда втыкаются разные устройства. За счёт готовых модулей функциональность такого комплекса можно расширять.

Код и программа — это по сути одно и то же: то, что со стороны пользователя — программа, со стороны программиста представляет собою код. Если мы используем её, как орудие, не залезая внутрь, то чаще говорим о программе, а если переписываем или дописываем, — то о коде.

Код и алгоритм соотносятся как текст и его смысл. Допустим, нужно посчитать скорости движения жидкости в трубе. Формулы нам даст математик. Алгоритм — это описанные на человеческом языке шаги, которые человеку нужно сделать, чтобы из исходных данных, например диаметра трубы и её длины, получить новые необходимые данные. Например: «Взять формулу номер один. Подставить в неё значения. Рассчитать... Взять формулу номер два...» Имея алгоритм, можно получить искомый результат, даже не понимая, что и зачем мы делаем. Код — это текст, который «объясняет» компьютеру, как ему выполнять алгоритм, воплощение алгоритма на одном из языков программирования.

***

Что надо знать заказчику программного продукта
Сергей Боченков, кандидат химических наук, руководитель IT-отдела проектной компании «СК КО-Восток».

Почему программисты не любят иметь дело с чужим кодом?

Идеальный программный проект предполагает постановку задачи полностью. Определяется задача проекта, планируются этапы, поочерёдно выполняются, документируются и так далее. В результате мы получаем хорошо задокументированные работающие программные модули, в которых может легко разобраться любой программист. В программистской реальности таких проектов бывает немного. В остальных случаях сроки поджимают, процессы слегка путаются, и в результате получается «чёрный ящик» — нечто очень сложное, в красивой упаковке с кнопочками, нажимая которые, ты в 99,9% случаев получишь то, что ожидаешь, но как эта штука работает, даже те, кто его писал, через некоторое время забывают. Самый страшный вариант — это «эволюционное развитие проекта»: задание получено, программа пишется, и тут приходит заказчик и говорит: а ещё мне нужно вот что... Если бы это «вот что» было известно сразу, вся программа писалась бы по-другому. Но, естественно, заказчик не готов платить за переделку всего, поэтому в программе делают «заплатку»: втыкают что-то сверху, каким-то образом, не всегда очень логично, подсоединяют, и оно работает (кроме того,

продолжает работать и всё остальное, что немаловажно). Если сделать такое несколько раз, программа превращается в лоскутное одеяло. Теперь даже те, кто её писал, боятся её трогать, а программист, который видит этот код впервые, почувствует абсолютный ужас и желание переписать всё заново. Это нормальное, естественное желание; дописывать подобный лоскутный код — как идти по болоту, каждый следующий шаг страшнее предыдущего.

Когда организация хочет, чтобы новый исполнитель доделал то, что у неё уже есть, программисты часто кривятся: «Кто вам это писал? Чудо, что оно вообще работает. Надо всё переписывать целиком». И что здесь будет правильным и неправильным — большой вопрос. С одной стороны, из заказчика вроде бы выбивают деньги на весь проект, вместо того чтобы доделать только часть, но с другой — проекты редко делают «в пустоту», обычно их предстоит потом поддерживать и сопровождать. А сопровождать проект, который работает неизвестно как, — это настоящий кошмар и для программиста, и для клиента. Поэтому в реальности мы пытаемся найти баланс между разными способами борьбы за управляемость кода. Существует, например, такой метод, как рефакторинг: изменение внутренней структуры кода без изменения его функциональности. Неизвестный нам код как бы упаковывается «в несколько новых коробок», на каждой из которых написано «мы не знаем, как это работает, но оно должно реагировать так-то на такие-то команды». Вокруг «коробок» пишут новую оболочку, которая использует их как чёрные ящики, не вникая в их содержимое. Это позволяет, не переписывая весь код с нуля, понимать, «в какой коробке» искать проблему, если что-то срабатывает не так. Отдельных денег за рефакторинг никто никогда не платит, исполнители включают его в проекты по необходимости, чтобы увеличить управляемость кода.

Полезно ли коду быть старым?

Смотря какой код. Расчётному, скорее, полезно: многолетняя эксплуатация означает, что в нем всё вылизано и про него всё известно. Правда, подходы к программированию со временем меняются, например, когда-то в нём не было понятия объектов, классов и так далее. Раннее программирование было процедурным: есть функция, ты в неё что-то вложил, она тебе что-то отдала назад. А современное программирование оперирует объектами, с которыми соотносятся определенные наборы свойств и методов. Если старый код представляет собой отдельную процедуру, выдающую, пусть даже по совершенно чудовищному алгоритму, результат, то её лучше не трогать. Обычно её просто вшивают в новую оболочку, уже представляющую собою объект. Но если старый код — это просто неправильно построенные объекты или какие-то чудовищные, непонятно как сгруппированные библиотеки методов (библиотека — это огромная «книга с алгоритмами»), тогда их приходится сначала разбивать по тематикам, а потом уже встраивать в объекты: каждому свою часть. Тогда объекты, которые связаны с гидравликой, будут включать одну часть процедур; связанные с гидродинамикой, — другую и так далее.

Объектное разбиение увеличивает потребность машины в ресурсах, но радикально снижает сложность общения с ней. Вообще, чем дальше мы уходим от машинной логики «один — ноль», тем машине сложнее: ей приходится обрабатывать больше операций.