В. Сафонов, И. Сафонов
(Международный институт науки единства,
Арлингтон, Виржиния, США)

РИСКИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К концу прошлого тысячелетия эволюция информационных технологий (как, впрочем, и экономических, биологических и социальных) продемонстрировала одно из их существенных отличий от привычных и зрелых энергетических, транспортных и машиностроительных технологий – значительно более раннюю (и, к сожалению, преждевременную) подверженность глобальной унификации. Это привело не только к повышенному риску игнорирования специализированных интересов каждого конкретного заказчика, но и к уникальному по своему несовершенству и почти монопольно защищённому от развития рынку разработчика. Сказанное относится и к языкам программирования, и к операционным системам, и к системам управления базами данных, и к поисковым машинам, и к пользовательскому интерфейсу.

Патология рынка информационных технологий сделала процесс создания программных продуктов чрезвычайно рискованным занятием и привела к потере доверия многих заказчиков. Так, например, в 1995 году из 8000 программных проектов в США более 30 процентов не были завершены, а почти все остальные не вложились ни в отведенное время, ни в планируемый бюджет [34]. В последующие годы ситуация не улучшилась. До недавнего времени для подавляющего большинства разработчиков информационных технологий было очевидным традиционное разделение функций программного обеспечения между ***прикладными программами*** (включая и такие "универсально-специализированные" как, например, базы данных или поисковые машины) и ***операционными системами*** (независимо от того, являются ли они однозадачными, многозадачными или сетевыми). Первые обеспечивали принципиальное исполнение необходимых заказчику ***функций***, а вторые – поддержку важных для того же заказчика ***свойств*** этих функций (производительности, секретности, надёжности, безопасности, точности, адаптивности и т. п.). Однако, сложившиеся на рынке информационных технологий тенденции ***унификации операционных систем*** и ***персонализации прикладных программ*** привели к необходимости создания приложений с заданными каждым конкретным заказчиком свойствами. Возникла ***проблема оптимальной персонализации***.

К тому же, постепенно стало почти очевидным (для одного из авторов, в частности, ещё с конца шестидесятых годов), что требуемые ***свойства приложений*** (в первую очередь, их надёжность и производительность) должны проектироваться раздельно как дополнения базовых функций этих приложений. Постепенно создавалась ***методология раздельного***

проектирования *самих* функций (***внешнего поведения***) и их свойств (***внутреннего поведения***), наиболее полно исследованная авторами [25]. Центр тяжести проблематики ***проектирования доверия*** и ***управления рисками*** также переместился в приложения. Так, например, уже в семидесятые годы Игорем Кирилловым была модернизирована операционная система IBM-360, лишившаяся фиксированной максимальной кратности повторов при сбоях и передавшая функции (уже оптимального!) выбора кратности повторов конкретным приложениям.

Не вникая в технические детали и специфику использования ***принципа разделения внешнего и внутреннего поведений***, заметим, что именно эта методология регулярно применялась уже несколько десятилетий тому назад при разработке специализированных компьютерных систем и их программного обеспечения (для ракет, спутников, самолетов, подводных лодок, гидроакустических станций, кораблей на воздушной подушке и на подводных крыльях, автоматизированных систем управления производством и технологиями) авторами их коллегами [1-3, 10, 11, 13-17, 20-25, 39-43]. Число и прозрачность соответствующих публикаций весьма ограничены, но накопленный опыт не устарел и может быть эффективно применен в современных условиях для снижения уровня (в первую очередь!) эволюционных рисков при создании и модернизации сложных программных продуктов, а также при создании (инжиниринге) и совершенствовании (реинжиниринге) деловых и технологических процессов. Естественно, что и деловые, и технологические процессы становятся всё более автоматизированными, настолько тесно переплетаясь с информационными, что изолированное их проектирование становится невозможным. Методология раздельного проектирования требуемых свойств прикладных программ (а часто и аппаратурно-программных комплексов) широко применялась в металлургии, мелиорации, машиностроении, авиастроении, химии и других отраслях, а также рекламировалась и обсуждалась на десятках семинаров и конференций (в том числе, и международных), использовалась в университетских и других курсах лекций (в Украине, России, США и др. странах).

Вслед за пионерскими попытками раздельного проектирования наиболее приоритетных свойств надёжности и производительности программных приложений, послужившими экспериментальной базой соответствующей ***методологии оптимизационного проектирования*** (Проектирование Надёжности или Надёжностное Проектирование, Проектирование Производительности), стало принципиально возможным и технически рациональным оптимальное удовлетворение потребностей заказчиков в придании их приложениям свойств защищённости от несанкционированного доступа, приоритизации базовых функций, упрощаемости пользовательского интерфейса и т. п. Также стало возможным параллельное проектиро-

вание разделённых (когда это целесообразно) аспектов поведения прикладных программ и альтернативное (многовариантное) проектирование их базовых функций, а также оптимизационное проектирование новых и совершенствование старых операционных систем.

Основные идеи, модели и методы формального преобразования программ с целью их совершенствования и оптимизации были предложены выдающимся советским учёным и инженером Виктором Михайловичем Глушковым, впервые опубликованы в 1965 году в журнале "Кибернетика" [5, 6] и развиты им самим и его учениками [7-9, 12, 20-25]. Глушков [7] шёл ещё дальше Дийкстры: "***После того как программы тем или иным образом составлены, дальнейшее их усовершенствование может выполняться формальными методами.*** С этой целью употребляются различные приёмы для ***формального преобразования программ...***".

Названная Глушковым проблема сложности оптимизационных преобразований программ и привела автора к необходимости создания и развития оптимизационных методов не только ***преобразования***, но и ***дополнения*** программ, что является более естественным и привычным для инженера и экономиста, а также значительно упрощает оптимизацию программ (и алгоритмов) по практически любым критериям. В частности, стало проще учитывать и преодолевать ресурсные конфликты между отдельными аспектами (например, надёжностью и быстродействием, надёжностью и секретностью, безопасностью и производительностью) в процессе их учёта при оптимизации – именно здесь разделение интересов оказалось наиболее естественным и плодотворным.

И наконец, нельзя забывать о реалиях корпоративных интересов информационной индустрии, не успевшей избавиться от проблемы объектно-ориентированного перепрограммирования (иногда, якобы) морально устаревших программ, унаследованных заказчиками от разработок семидесятых и восьмидесятых годов (КОБОЛ → Реляционные СУБД, С → С++, С → Java и т.п.), и получившей к концу девяностых годов мешанину уже объектно-ориентированных программ с их колоссальными библиотеками классов и интерактивными инструментами разработки [17, 18, 21, 26]. При этом ***интересы самих разработчиков*** (быстрее, ждёт очередной клиент!) существенно ***превалировали над интересами заказчиков*** (лучше, чем у конкурента!).

К концу прошлого столетия информационная индустрия впала в состояние "грогги" к моменту и перед лицом нормализации информационно-технологического рынка и экономического уравнивания его с традиционными рынками других технологий, где уже давно господствовал ***рынок покупателя***. Рынок труда переполнился безработными программистами, ранее занимавшимися рутинными и примитивными операциями кодирования, тестирования и документирования программных

ния, тестирования и документирования программных продуктов, создаваемых для нетребовательных (в первую очередь, правительственных) клиентов. Но уровень квалификации этих людей уже не соответствовал новым повышенным требованиям повзрослевшего рынка информационных продуктов и услуг.

Недюжинный интеллект информационной индустрии, оставшись наедине с проблемами наследства уже объектно-ориентированных технологий и не успев остыть после гонки за количеством, наконец, ощутил на себе холодное дыхание проблемы качества. Наиболее чуткие новаторы уже к середине девяностых годов стали пытаться переломить ситуацию. Возникло множество методологий совершенствования и развития процесса объектно-ориентированного программирования. Наступила эра аспектного программирования [35, 36], частично следующего советам Дийкстры в неадекватных условиях объектной ориентации, и программного рефакторинга [31, 44], даже частично не реализовавшего продуктивных идей и рациональной методологии Глушкова. Время покажет, насколько всё это полезно, а тем более эффективно. Но уже очевидно, что современный заказчик информационных продуктов и услуг достаточно грамотен для того, чтобы строго сформулировать свои требования разработчику, и достаточно информирован об альтернативных возможностях информационного рынка для того, чтобы сделать наилучший (т. е., оптимальный) выбор технологии и (или) исполнителя. Если задача выбора исполнителя является относительно простой и модельно близкой традиционной задаче анализа кредитных и инвестиционных рисков [26], то задача выбора технологии на порядок сложнее, во-первых, из-за процедурного характера сопоставляемых технологий, и, во-вторых, из-за конъюнктурной дискредитации всех технологий, не являющихся объектно-ориентированными. Лучшей же на сегодняшний день остаётся методология выбора, основанная на экспертных оценках и многокритериальном ранжировании в сочетании с аналитическими и статистическими методами и являющаяся одной из компонент экспертно-моделирующей системы СЕЛЕНА, диапазон применения которой практически не ограничен [18].

Объектно-ориентированное программирование до сих пор не смогло (и вряд ли сможет) решить проблему рационального моделирования задач анализа, оптимизации и синтеза программных продуктов, тем более для работы в условиях постоянного риска проектной и эксплуатационной ненадёжности, информационного вандализма и терроризма, промышленного шпионажа и, наконец, с учётом требований безопасности применения самих программных продуктов в ответственных процессах и критических ситуациях. Зациклившись на поиске эффективных метрик (см., например, [33]), апологеты объектно-ориентированного программирования не про-

двинулись далее примитивных эвристик оценивания сложности и процедур тестирования готовых или почти готовых продуктов, когда их совершенствование становится более трудоёмким чем полный реинжиниринг.

К сожалению, искусство программирования иногда является тормозом рациональной индустриализации информационных технологий, как впрочем уже неоднократно случалось в истории научно-технического прогресса, когда талантливые ремесленники препятствовали внедрению перспективных технологий. Корпоративные интересы редко стимулировали технологический прогресс, но почти всегда были стимулом примата умения над знанием, а не их синергетического симбиоза. У нас же созрела уверенность в том, что давно наступило время замены монокулярного зрения информационных стратегов, часто предпочитающих близорукость визуального подхода дальнозоркости формальных методов, бинокулярным зрением симбиоза визуализации и формализации, ориентированых в первую очередь на интересы заказчика и только потом на удобства разработчика (с учетом того, что определённые удобства разработчика также могут и должны способствовать удовлетворению интересов заказчика). Понимание сложившейся ситуации большинством создателей и пользователей информационных технологий существенным образом уменьшит риск несовершенства (более того, незавершенности) создаваемых программных продуктов, сделает, наконец, ощутимо эффективным их применение (о неэффективности информационных технологий так много написано, что не хочется повторяться) и увеличит доверие общества к информатизации.

Особого внимания заслуживает ***проблема сближения процессов обработки данных и управления организациями***. И в этом контексте эволюция информационных технологий должна быть проанализирована с точки зрения её стратегической целесообразности. Теоретико-методологическую основу такому анализу даёт дескриптивный взгляд на программы и программирование Владимира Редько. Вот что он пишет [13]: "Построение оснований любой дисциплины сопряжено с выбором концептуально единой точки зрения на предмет её изучения. Следуя объектно-ориентированному стилю программирования [4], на программы нужно смотреть как на объекты. Но эта точка зрения на предмет программирования как на целостную дисциплину не может быть концептуально единой. Более содержательна функциональная точка зрения, лежащая в основе всех разновидностей функционального программирования [27]".

Примечание: Продолжение тематики данного доклада и литература к нему в докладе "Риски информационных технологий и оптимизация потоков работ".