В. Сафонов, И. Сафонов

РИСКИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ОПТИМИЗАЦИЯ ПОТОКОВ РАБОТ

Международный Институт Науки Единства, Арлингтон, Виржиния, США
inusin@msn.com

Аннотация. Анализируется кризисная ситуация, сложившаяся в информационных технологиях в переходный период от многолетнего диктата разработчиков к нормальному рынку заказчиков. В качестве одного из возможных путей решения проблем эволюционных рисков предлагается инвариантная относительно языков и методологий программирования комбинация структурно-алгоритмического и экспертно-моделирующего подходов к созданию оптимально персонифицированных программных продуктов.

Ключевые слова: эволюционные риски, информационные технологии, рынок разработчика, рынок пользователя, проектирование доверия.

1. 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 отведённое 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, ба-

зы 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 в докторской диссертации старшего автора (1975) и изложенная в его монографии [25] в 1982 году. Центр тяжести проблематики **проектирования доверия** и **управления рисками** также переместился в приложения. Так, например, уже в семидесятые годы Игорем Кирилловым, одним из сотрудников старшего автора, по его заданию была модернизирована операционная система IBM-360, лишившаяся фиксированной максимальной кратности повторов при сбоях (насколько помнится, не более 2 для процессора и не более 18 для долговременной памяти) 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) старшим автором, его коллегами и учениками [1-3, 10, 11, 13-17, 20-25, 39-43]. По понятным причинам, 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 технологиче-

ские процессы становятся всё более и более автоматизированными, настолько тесно переплетаясь с информационными, что изолированное их проектирование становится невозможным (или, по крайней мере, бессмысленным). Сказанное никак не противоречит раздельному проектированию свойств этих автоматизированных процессов, хотя это разделение и становится более сложным. Кроме того, методология раздельного проектирования требуемых свойств прикладных программ (а часто и аппаратурно-программных комплексов) широко применялась в промышленности (в металлургии, мелиорации, машиностроении, авиастроении, в 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 проектирование 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.

Почти все вышеупомянутые тенденции были предвидены и великим Дийкстрой, и великим Глушковым и всегда были руководством к действию их учеников. Дийкстра, в частности, писал [30]: " Мы знаем, что программа должна быть **правильной**, и мы можем изучать её только с этой точки зрения; мы также знаем, что (она) должна быть **продуктивной**, и мы можем изучать её производительность на следующий день ... Но мы ничего не теряем – наоборот – учитывая эти различные **аспекты** одновременно. Это то, что я иногда называю "**разделением интересов** ..." (the **separation of concerns**)".

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** ... Хотя подобные глубокие преобразования программ являются, как правило, наиболее эффективным средством их **оптимизации**, однако их реализация в каждом конкретном случае может оказаться достаточно сложной".

Названная Глушковым проблема сложности оптимизационных преобразований программ и привела автора к необходимости создания и развития оптимизационных методов не только **преобразования**, но и **дополнения** программ (и вообще любых алгоритмов, в том числе и алгоритмов деловых процессов и технологий), 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 (побыстрее, ждёт очередной клиент!) существенно превалировали над интересами заказчиков (получше чем у конкурента!).

К концу прошлого столетия информационная индустрия (выкачав из заказчиков десятки миллиардов долларов на сомнительную с самого начала "проблему" двухтысячного года и из инвесторов не меньшего порядка сумму на "доткамы" и интернетовский бум) 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 продуктов, создаваемых для нетребовательных (в первую очередь, правительственных) клиентов. Но уровень квали-

фикации этих людей уже не сответствовал 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-ориентировнными. 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 продвинулись 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]".

2. ОПТИМИЗАЦИЯ ПОТОКОВ РАБОТ

Поведенческий подход позволил нам системно объединить, а часто и синергетически соорганизовать, многие традиционные методологии обеспечения безопасности и оптимизации функционирования структурно-алгоритмических систем со случайными нарушениями [38-43]. Этот сим-

биоз упреждающего **Инжиниринга Доверия** и реагирующего **Риск Менеджмента** создаёт условия для наиболее эффективного обеспечения не только безопасности, но и других важных свойств (надёжности, точности, секретности, производительности и т. п.) в общих рамках жизненного цикла проектирования и совершенствования систем. При этом реабилитируется прикладная математика, частично дискредитированная негибким применением (однокритериального!) математического программирования и в ещё большей степени расслабляющим влиянием диктата (к сожалению, не всегда инженерно образованных) разработчиков и программистов. Известно, что решения, основанные на математике, более долговечны чем решения, основанные на технологии. Достаточно вспомнить булеву алгебру, теорию графов, формулу Байеса и дискретные аналоги дифференциального исчисления. Для образованного и опытного инженера простота и точность прикладной математики очевидны. "Если люди не верят в простоту математики, то только потому, что они не представляют себе как сложна жизнь", – говорил Джон фон Нейманн. Более того, такие решения делают нас (разработчиков!) и, следовательно, наши решения обоснованно независимыми от влияния поставщиков и строго ориентированными на интересы заказчика.

Моделирование и оптимизация помогают с достаточной точностью понять проблемы и наилучшим образом решить их. В этом случае неадекватное использование терминов "моделирование" и "оптимизация" является не только нежелательным, но и недопустимым. К сожалению, любое иллюстративное и неполное описание (эскиз, набросок) проблемы, как правило, называется "моделью" проблемы, а любой шаг в кажущемся правильным направлении улучшения обычно называется "оптимизация". Сомневающиеся в состоятельности моего утверждения могут легко его проверить путём поиска в Интернете соответствующих терминов и анализа их интерпретаций. Сказанное выше в первую очередь относится к тысячам рекламных сообщений о методах и инструментах управления потоками работ (workflows) и оптимизации деловых процессов (business processes). Поскольку один из авторов является пионером оптимизации потоков работ, мы рады возможностям соответствующего многомиллиардного рынка [http://www.internettime.com], но как профессионалы мы огорчены дискредитацией терминов и концепций моделирования и оптимизации в этом контексте. Заказчики и пользователи информационных технологий и автоматизированных процессов! Берегитесь амбициозных дилетантов в инжиниринге и менеджменте! Они злоупотребляют вашим доверием.

Производительность и качество, надёжность и долговечность, безопасность и секретность ответственных процессов и технологий нуждаются в доверии к их создателям и пользователям, равно как и к советникам и

консультантам руководителей. Убедитесь сначала, что вам предлагают действительно **лучшие (строго обоснованные) решения**. Мы сможем гарантировать нашим клиентам корректное моделирование и оптимальное решение их проблем, если таковые в принципе разрешимы и когда эти решения приемлемы для клиентов даже с учётом причин, выходящих за рамки технического задания. Проблемы же однокритериальной (всё реже!) и многокритериальной (всё чаще!) оптимизации, основанной на персонально-ориентированном моделировании ситуаций и процессов для автоматизации бизнеса и обработки данных, решаются в четырёхмерном пространстве **целей**, **ресурсов**, **структуры** и **поведения**, первично определяемых и часто корректируемых клиентами. Поскольку однокритериальные проблемы (с нетривиальным множеством ограничений) до настоящего времени являются массовыми, а также в силу строгости и простоты их формулировок самими заказчиками, начнём именно с них. Многокритериальные проблемы оптимизации потоков задач будут нами рассмотрены в последующих публикациях.

Двумя основными жизненно важными функциями человека в его взаимодействии с природой, технологиями и себе подобными являются **принятие решений**, акты и процессы которого будем обозначать маленькими буквами латинского алфавита {a1, a2,…, b1, b2,…, c1,…}, и **выполнение действий,** акты и процессы которого будем обозначать прописными буквами латинского алфавита {A1, A2,…, B1, B2,…, C1,…}. Элементарными (атомарными) актами принятия решений и выполнения действий являются **решения** (a) и **действия** (A), соответственно, рассматриваемые как неделимые на рационально установленном для каждой задачи уровне абстракции её моделирования, анализа, оптимизации и синтеза. Синтаксис, семантика и прагматика корректной композиции решений и (или) действий определяют с необходимой степенью детализации дискретные и непрерывно-дискретные процессы человеческой активности, как правило, более или менее автоматизированной. Если эти процессы описывают (моделируют) материальные, энергетические, финансовые, информационные и (или) иные производства и (или) услуги, то соответствующие решения и действия принято называть **работами**, а сами процессы моделирования принятия решений и (или) выполнения действий – **потоками работ** (**workflows**).

Анализ возможных ситуаций, оптимизация и синтез потоков работ являются стадиями **проектирования потоков работ**, равно как контроль, оптимизация и коррекция выполнения отдельных работ или их взаимосвязанных совокупностей – стадиями **управления потоками работ**. Заметим, что **проектирование доверия** и **управление риском** являются частными, но важными специфическими случаями интегрированного процесса проек-

тирования потоков работ и управления ими в условиях риска и неопределённости. А поскольку риск и неопределённость являются неизбежными атрибутами сложных потоков работ, то и важность **оптимального проектирования доверия и управления риском** не вызывает сомнения. Чем сложнее поток работ, тем менее наглядным становится его описание и более сложной – его модель. Чем выше уровень автоматизации потока работ, тем меньше смысла имеет визуализация этого потока для его автоматизированного (а иногда и автоматического) проектирования и управления. Принцип Эшби ("Теория сложных систем есть теория упрощения") постепенно вытесняется принципом Глушкова ("Система управления должна быть по крайней мере той же сложности, что и управляемый объект"). Для инструментов автоматизированного проектирования и управления определяющим их эффективность критерием становится уровень формализации объектов и процессов проектирования и управления, а их наглядность уходит на второй план. Смысл и необходимость описательности вытесняются требованиями повышенных строгости и точности моделирования.

И, наконец, для оптимизации необходимы соответствующие метрики с известными или вычислимыми зависимостями между ними. Поэтому формальные (алгоритмические, алгебраические) модели потоков работ предпочтительнее графических, что конечно не исключает и последние. Подобного же рода соображения и накопленный опыт говорят в пользу теоретико-игровых моделей и методов по сравнению с аналитическими, и многокритериальных и "размытых" – по сравнению с классическими моделями и методами математического программирования, при анализе и оптимизации процессов принятия решений.

Предположим, что имеется несколько (j) альтернатив реализации каждого функционального (**внешнего**) или аспектного (**внутреннего**) действия A(i) потока работ (алгоритма, процедуры) A представляющих также альтернативные реализации всего требующего оптимизации потока работ.

$$A(i) \rightarrow \{ A(i, j) \},\ j = 1, 2, \ldots , n(i),\ j = 1, 2, \ldots, N.$$

Каждая возможная реализация действия A(i, j) характеризуется вектором параметров (свойств, критериев, метрик) производительности, качества, надёжности, безопасности и т. п.

$$\{ f(i, j), r(i1, j), r(i2, j), \ldots , r(iS, j) \},$$

где f – функциональные (**целевые**) и r – аспектные (**обеспечивающие свойства**) требования или ограничения;
и весь поток работ A оценивается вектором параметров

$$\{ F, R1, R2, \dots , RS \},$$

где $F = F (f1, f2, \dots , fN)$, $Rk = Rk (r1k, r2k, \dots , rNk)$.

Необходимо выбрать наилучшую **каноническую** (последовательную, параллельную, альтернативную, и (или) циклическую) **композицию** (архитектуру) реализационных альтернатив действий A(i), i = 1, 2, ... , N, потока работ A, которая в результате приводит (это не самый общий, но достаточно часто используемый на практике, случай) к глобальному экстремуму целевой функции (функционала) F при заданных ограничениях вектора параметров этого потока работ:

$$F \rightarrow extr, \quad Rk <= Rk0,$$

где все Rk0, k = 1, 2,..., S, заданы.

Наиболее часто используемыми параметрами являются Время Выполнения Потока Работ (Performance), Готовность и (или) Вероятность Правильного Выполнения (Reliability), Уровни Безопасности (Safety) и Секретности (Security), Сложность (Complexity), Цена (Price), Эффективность Капиталовложений (Return on Investment – ROI), etc. Очевидно, что каждый из этих параметров (а также количество типов функциональных и аспектных операторов – при необходимости их унификации) может быть выбран в качестве целевой функции F. В этом случае остальные параметры должны удовлетворять установленным ограничениям. Известно, что любой поток работ может быть представлен композицией названных выше четырёх канонических форм – последовательной (линейной), параллельной (конъюнктивной), альтернативной (дизъюнктивной) и (или) циклической (итеративной).

Как правило, виды функциональных зависимостей $F = F (f1, f2, \dots , fN)$ и $Rk = Rk (r1k, r2k, \dots , rNk)$ для канонических форм уже известны или тривиально определимы, а для произвольного потока работ определяются его алгоритмической структурой. В самом общем случае, для решения оптимизационных задач может быть применён метод неявного перебора, использующий схему ветвей и границ. Решением задачи оптимизации будет N-мерный вектор x, координаты которого принимают значения из некоторого конечного множества (которые удобно интерпретировать как целочисленные). Например, x(i) = j означает, что выбрана j-ая версия i-го действия (акта), и каждая x(i) может принимать любое целочисленное значение из отрезка [1, ni], где ni – число версий i-го действия.

Если в процессе решения оптимизационной задачи версии реализации некоторых действий уже выбраны, то фиксируются соответствующие им координаты, и подмножество фиксированных координат называется **частичным решением**. Не фиксированные до этого момента координаты будем называть **свободными координатами**. Всякий конкретный набор целочисленных значений свободных переменных называется **дополнением** соответствующего частичного решения. Частичное решение представляет собой упорядоченное множество пар (1, z1), (2, z2), …, (L, zL), где первой компонентой каждой пары является номер соответствующего действия (при необходимости решения могут включать в себя действия, например, тестовые или диагностические), а второй – номер альтернативы (варианта) этого действия. Каждому же частичному решению ставим в соответствие число k нецелочисленных ограничений оптимизационной задачи. Таким образом, объектом оптимизационного анализа является упорядоченное множество

$$\{F, R1, R2, \ldots, Rk, (1, z1), (2, z2), \ldots, (L, zL)\}.$$

Оценки свойств определяются следующим образом. Для свободных переменных полагаем x(i) = 0. При x(i) ≠ 0 величина свойства rni фиксирована, а при x(i) = 0 будем использовать наилучшее значение соответствующего свойства (т. е. значение, минимизирующее вклад этого свойства в ограничение ≤). Например, если rn – время, то rni = Ti = Tix(i) x(i) ≠ 0, иначе Ti = Tij, 1 ≤ j ≤ ni, где Tij время работы j-го варианта i-го действия (подробнее в [17]).

В большинстве практически важных случаев оптимизация потоков работ может быть упрощена путём учёта специфики конкретных деловых, производственных, технологических, информационных и иных процессов. Например, градиентные [13] или аналитические [1, 2] методы находят широкое применение при проектировании в условиях дефицита производительности (Design for Performance), безопасности (Design for Safety), секретности (Design for Security) и (или) надёжности (Design for Reliability). В более сложных ситуациях мы использовали выпуклое программирование [10], метод ветвей и границ [15], динамическое программирование [11] и другие методы. Применимость моделей и методов оптимизации проверялась путём специальных исследований их адекватности конкретным ситуациям [10] и устойчивости к неточности исходных данных [2]. Мы предлагаем унифицированную (единую) методологию и формализованные методы для оптимального удовлетворения требований к безопасности, секретности и надёжности алгоритмов деловых процессов и программ атоматизации этих процессов. В основу этой методологии положены концепция

и архитектура структурно-алгоритмической оптимизации [25] . При этом учитываются принципиальные отличия между проектными и операционными (эксплуатационными) аспектами безопасности, секретности и надёжности, а также учитывается корреляция между этими аспектами. Это приводит к необходимости разделения интересов (полномочий), связанных с влиянием 1) неадекватности моделирования, 2) проектных и программных ошибок, 3) несовершенством защиты и злонамеренных вторжений, 4) неисправностей и сбоев оборудования (аппаратуры), и 5) недооценкой опасности действий собственного персонала и возможных материальных потерь.

Проблемы аспектной оптимизации безопасности, защищённости и надёжности неизбежно фокусируются на сбалансированности с требованиями и ограничениями к параметрам производительности, точности, (потенциальных) возможностей, энергопотребления, стоимости (цены) и других. Формализованы и исследованы проблемы устойчивости используемых моделей, методов и программ к информационным флюктуациям, ситуациям принятия решений и процедурам выполнения действий. Предлагаются актуальные постановки и эффективные методы тестирования (не только автоматизированных) деловых процессов и процессов автоматизированной обработки данных с корректирующей избыточностью и без неё. Оригинальное программное обеспечение оптимизации потоков работ находится в процессе разработки. Наша методология находит широкое коммерческое применение при разработке автоматизированных систем технологического и организационного управления, САПР, инструментов прогнозирования процессов и поддержки решений, специализированных управляющих компьютеров, средств обработки документов, мультипроцессорных систем, поисковых машин и т. п.

Литература

1. Бондарь Ю. и Сафонов И. В. Метод оптимального использования алгоритмической избыточности. – Управляющие системы и машины, 1975, № 3.

2. Бондарь Ю. и Сафонов И. В. Устойчивость модели оптимального обнаружения и исправления ошибок. – Разработка и применение АСУ и средств автоматизации. – Киев: Институт автоматики, 1977.

3. Борисюк А. А., Гук К. Н., Коссовский В. Г. и Сафонов И. В. Обеспечение надёжности СЦВМ управляющего комплекса. – Диагностика, контроль, надёжность систем управления. – Киев: Институт автоматики, 1976.

4. Буч Г. Объектно-ориентированное программирование с примерами применения. – Москва: Конкорд, 1992. – 519 с.

5. Глушков В. М. Теория автоматов и микропрограммные алгебры. – Кибернетика, 1965, № 1.

6. Глушков В. М. Теория автоматов и формальные преобразования микропрограмм. – Кибернетика, 1965, № 6.

7. Глушков В., Барабанов А., Калиниченко Л., Михновский С. и Рабинович З. Вычислительные машины с развитыми системами интерпретации. – Киев: Наукова Думка, 1970.

8. Глушков В. М., Капитонова Ю. В. и Летичевский А. А. Автоматизация проектирования вычислительных машин. – Киев: Наукова Думка, 1975.

9. Глушков В. М., Цейтлин Г. Е. и Ющенко Е. Л. Алгебра. Языки. Программирование. – Киев: Наукова Думка, 1974.

10. Демьянчук А. П. И Сафонов И. В. Применимость метода выпуклого программирования для надёжностной оптимизации систем реального времени. – Диагностика, тестирование и надёжность управляющих систем. – Киев: Институт автоматики, 1976.

11. Демьянчук А. П., Карась В. М. и Сафонов И. В. Надёжностная оптимизация алгоритмов АСУ методом динамического программирования. – Проблемы надёжности систем и средств автоматики. – Киев: Техника, 1976.

12. Капитонова Ю. В. и Летичевский А. А. Математическая теория проектирования вычислительных систем. – Москва: Наука, 1988.

13. Карась В. М. и Сафонов И. В. Обеспечение надёжности функционирования и развития АСУ (Оптимизация алгоритмов). – Киев: РДНТП, 1974.

14. Коссовский В. Г. и Сафонов И. В. Формализованное надёжностное проектирование специализированных цифровых машин. – Проблема надёжности систем управления. – Киев: Наукова Думка, 1973.

15. Куриленко В. Т., Левченко С. Н. и Сафонов И. В. Надёжностная оптимизация алгоритмов методом ветвей и границ. – Алгоритмы, программное и техническое обеспечение автоматизированного управления производством. – Киев: Институт автоматики, 1978.

16. Макаренко Г. И. и Сафонов И. В. Система производства программ для управляющих ЦВМ. – Логическое управление. Вып. 3. – Москва: Энергоиздат, 1981.

17. Мамиконова О. А. и Сафонов И. В. Оптимизация структурированных алгоритмов и программ. – Средства реализации систем программирования. – Киев: ИК АН УССР, 1983.

18. Мурса Л. Г. и Сафонов И. В. "Селена": кто она такая? – Искусство кино, 1988, № 11.

19. Редько В. Н. Дескриптологические основания программирования. – Кибернетика и системный анализ, 2002, № 1.

20. Сафонов И. В. К вопросу введения корректирующей избыточности путём преобразования алгоритмической структуры цифрового автомата. – Теория автоматов, Вып. 1. – Киев: ИК АН УССР, 1969.

21. Сафонов И. В. Об одном способе задания исходной информации при формализованном надёжностном синтезе. – Четвёртая республиканская научная конференция молодых исследователей по системотехнике. II том. – Киев: ИК АН УССР, 1969.

22. Сафонов И. В. О формализованном надёжностном синтезе дискретных устройств. – Кибернетика, № 5, 1971.

23. Сафонов И. В. Об алгоритмическом этапе формализованного надёжностного проектирования ЦВМ. – Управляющие системы и машины, № 1, 1972.

24. Сафонов И. В. Оптимизация при автоматизированном проектировании систем управления. – Автоматизированное проектирование АСУ. – Москва: Финансы и статистика, 1981.

25. Сафонов И.В. Надёжностное проектирование алгоритмов управления. – Владивосток: Институт автоматизации и процессов управления, 1982.

26. Соложенцев Е. Д., Карасев В. В. и Соложенцев В. Е. Логико-вероятностные модели риска в банках, бизнесе и качестве. – Санкт-Петербург: Наука, 1999.

27. Backus J. Can programming be liberated from the Neumann style? A functional style and its algebra of programs. – Comm. ACM, 1978, 21(8). – P. 613-641.

28. Bollella, et al. The Real-Time Specification for Java. – Boston, MA: Addison-Wesley, 2000.

29. Booch, Grady, James Rumbaugh, and Ivar Jacobson. The Unified Modeling Language User Guide. – Reading, MA: Addison-Wesley, 1999.

30. Dijkstra, E. W. On the Role of Scientific Thought. – E. W. Dijkstra Archive (EWD447), August 1974.

31. Fowler, Martin, et al. Refactoring: Improving the Design of Existing Code. – Boston, MA: Addison-Wesley, 2000.

32. Gamma, Erich, et al. Design Patterns: Elements of Reusable Object-Oriented Software. – Reading, MA: Addison-Wesley, 1995.

33. Henderson-Sellers, Brian. Object-Oriented Metrics: Measures of Complexity. – Upper Saddle River, NJ: Prentice-Hall, 1996.

34. Hoch, Detlev J., et al. Secrets of Software Success. – Boston, MA: Harvard Business School Press, 1999.

35. Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In ECOOP'97 – Object-Oriented Programming, 11th European Conference, LNCS 1241, pp. 220—242, 1997.

36. Kiselev, I. Aspect-Oriented Programming with AspectJ. Indianapolis, IN: SAMS, 2002.

37. Kruchten, Philippe. The Rational Unified Process. – Boston, MA: Addison-Wesley, 2003.

38. Safonov, Igor. Trust Engineering and Risk Management of Complex Systems. – Modeling and Analysis of Safety, Risk and Quality in Complex Systems. Proceedings of International Scientific School. – Saint-Petersburg: RAS, 2001.

39. Safonov, Igor. Aspect-Oriented Software Reliability Engineering. – Modeling and Analysis of Safety and Risk in Complex Systems. Proceedings of Third International Scientific School. – Saint-Petersburg: RAS, 2003.

40. Safonov, Igor, and Vadim Safonov. Forecasting and Planning of Corporate Business Activity and Data Processing for Optimal Trust Engineering and Risk Management. – Modeling and Analysis of Safety and Risk in Complex Systems. Proceedings of Forth International Scientific School. – Saint-Petersburg: RAS, 2004.

41. Safonov, Igor. Evolution Risks of Informational Technologies. – Ibid.

42. Safonov, Igor, and Vadim Safonov. Security Engineering and Patch Management for Safety of Information Technologies. – Thirteen Scientific-Technical Conference “Safety Systems” of the International Informatization Forum. – Moscow: IIA, 2004.

43. Safonov, Igor. Workflow Optimization for Safety Performance”. – Ibid.

44. Wake, William C. Refactoring Workbook. – Boston, MA: Addison-Wesley, 2004.