Г.С. Суворов[1]

ЗАМЕТКИ О МЕТРОЛОГИИ ПРОГРАММНЫХ СИСТЕМ И МОДЕЛЕЙ

Точная оценка качества больших программных систем весьма затруднительна. Это свойство характерно для гуманитарных областей – искусства, некоторых видов спорта, военной стратегии и тактики, юридических наук и т.д. В программировании сложных систем и оценках его качества, как и в гуманитарных областях, значительную роль играет творческая составляющая человеческого сознания (человеческий фактор). Поэтому многие оценки качества программ носят субъективный характер. Применение к сложным программным системам примитивных, «точных» якобы математических методов вводит в заблуждение и потому вредно. Выход из положения следует искать в развитии методов квалиметрии и экспертных оценок.

Ключевые слова: *оценка качества больших программных систем, роль человеческого сознания.*

G.S. Suvorov

NOTES ON METROLOGY OF PROGRAM SYSTEMS AND MODELS

Exact measurements of big software systems quality are very difficult. Human element plays the key role in software of models as it plays in liberal arts. Therefore many quality ratings about it are subjective. Moreover, "exact" supposedly mathematical ratings of merit mislead and are injurious. To overcome these difficulties we must to seek out development methods of quality metrics measurements and expertise methods.

Keywords: *big software system quality, the key role of the human elements in software of models.*

При современном уровне развития информационных технологий все больше внимания уделяется созданию сложных программных моделей. Эти модели должны охватывать все стороны функционирования больших экономических, инженерных, медицинских, социальных, экологических, военных и прочих систем. Назначение этих систем сводится к одному – глубже изучить окружающий мир и тем самым обеспечить рациональное управление человеческой деятельностью в разных областях. Здесь обязателен учет разнообразия и сложности возникающих реальных проблем. При этом неважно, реализована ли в данный момент система, для которой создается модель, или эта система существует только в замысле. Различие взглядов на цели создания таких моделей, условия их функционирования и понимание этих условий непредсказуемо разнообразны, а обилие случайных внешних факторов создает ситуации, когда ничтожная случайность способна привести к огромным различиям в результатах работы – от безоговорочной победы до полного краха.

Чем активнее входит в нашу жизнь моделирование больших систем, тем больше значение программных систем как важной и обычно решающей части этих моделей. Почему же мы продолжаем говорить о программной системе, как *составной части* информационной или управляющей системы? Именно потому, что программная система не существует без компьютера и всегда составляет с ним (или с сетью компьютеров) единый комплекс. Но управляющие системы этим не ограничиваются: достаточно вспомнить системы управления самолетами, которым стали частенько давать термин «*авионика*». Вспомним многие инженерные, военные, медицинские и прочие системы, которые существуют, создаются или будут создаваться. В них, кроме вычислительных средств, заложенных в архитектуру самого компьютера или системы компьютеров, важную роль играют аппаратные (измерительные, сигнальные и обрабатывающие) средства, которые поставляют важнейшую информацию для обработки, осмысления ситуации и достижения поставленных целей. Для этого системы и создаются. Таким образом, мы всегда имеем дело с *аппаратно-программными системами*. Обе эти составные части – и аппаратная, и программная – имеют свои особенности, которые следует четко различать.

Аппаратная часть – это продукт промышленного материального производства. Он имеет намного более богатую историю, чем программная часть. Промышленность насчитывает столетия. Человечество всегда старалось обеспечить высокое качество произведенной промышленной продукции. Результатом этого было появление снача-

[1] Кандидат технических наук, старший научный сотрудник, доцент кафедры информационных технологий и естественно-научных дисциплин НОУ ВПО «Российский новый университет».

ла методов счета и измерений, а затем – возникновение таких динамичных отраслей науки, как *стандартизация* и *метрология*. Их развитие в материальном производстве идет параллельно с совершенствованием этого производства.

Сейчас вершиной стандартизации в области качества в мире принято считать комплекс международных стандартов системы управления качеством серии ИСО. В Российской Федерации эти международные стандарты приняты в качестве национальных стандартов (ГОСТ Р ИСО 9000–2001). Они разработаны Международной организацией по стандартизации – ИСО (ISO), созданной в 1947 году. Наша страна (тогда Советский Союз) входила в число государств – основателей ИСО и сотрудничает с ней в настоящее время.

Особое внимание ИСО уделяет таким направлениям, как качество и безопасность продуктов питания, нефтяная и газовая промышленность, служба водоснабжения, комбинированные перевозки с использованием различных видов транспорта, радиоэлектроника, тяжелая промышленность, технологии в сфере здравоохранения, сельскохозяйственное оборудование и промышленная автоматика. Постоянно проводятся испытания продукции на соответствие требованиям технических регламентов и стандартов к ее безопасности, то есть защите от угрозы здоровью, окружающей среде, экономике. Все перечисленные направления опираются на прочную многолетнюю и постоянно развивающуюся метрологическую базу. Эта же оценка распространяется и на аппаратную часть систем управления.

История *программных систем* зародилась в середине XX века. За прошедшие с тех пор годы они претерпели стремительную эволюцию: сегодня стали повседневными те задачи, о которых совсем недавно даже не пытались и мечтать. В настоящее время выявились особенности создания программных моделей. Создание программных систем и программных моделей стало самостоятельной функцией промышленного производства.

В отличие от материального производства, имеющего громадный опыт выработки требований к создаваемой продукции и обеспечения выполнения этих требований, создание программных моделей и в седьмом десятилетии существования программирования имеет все признаки совершенно новой задачи.

Замысел новой системы и ее программной модели, как и всякая новая идея, рождается в сознании человека или, как принято говорить, носит виртуальный, то есть воображаемый, возможный характер. Замыслы материальных систем (конструкций), как правило, реализуются в реальных изделиях. Эта продукция может быть удачной и неудачной. Неизвестно, как много времени прошло, прежде чем мысль «*плоское тащи, круглое кати*» дала человечеству выдающееся изобретение – первое реальное колесо. Программные модели достаточно быстро реализуются в кодах (программах) и проверяются посредством запуска в компьютерах и, если необходимо, в сочетании с аппаратурой.

Стремительное развитие вычислительной техники и информационных технологий создает необъятные возможности перехода от воображаемых образов к их реализации в моделях. Здесь возникают не только новые возможности, но и новые препятствия продвижению «*к колесу*».

• Моделируемая сложная система *постоянно и стремительно изменяется*. Изменяется само понимание человеком системы, ее назначения, свойств и возможностей. Изменяются условия функционирования, внешние факторы, воздействующие на нее, потребности в результатах моделирования. Все это требует изменений модели на всех этапах ее жизненного цикла, что влечет за собой применение на всех этапах *итеративного метода* ее создания.

• В отличие от материальных систем, у программной модели, как правило, кроме замысла разработчика, нет иных материальных прообразов (или они только зарождаются и носят очень приблизительный характер). А заказчик (заказчиком может оказаться рынок) имеет свое понимание замысла модели. Модель в разных головах – разная модель. Из этого следует *расхождение между заказчиком и разработчиком* в понимании функций программы. Мало того, итерационная модель проекта должна учитывать начальные требования заказчика. Постепенно возникающие новые требования заказчика и последующие их уточнения приводят к структурным противоречиям, получается, что, как правило, невозможно заранее решить, какой должна быть архитектура модели. Поэтому необходима *постоянная обратная связь* между заказчиком и разработчиком.

• Итерационный характер программной модели и постоянная обратная связь «разработчик – заказчик» требуют от проекта максимальной гибкости. Отсюда вытекают требования к *гибкости и перенастраиваемости* программной части системы. Процесс создания модели превращается в преодоление сложности понимания решаемой задачи путем *представления в виде последовательности сравнительно небольших логических блоков*, интуитивно понятных и простых для программирования.

• Ошибки возникают в деталях реализации

требований к продукции. Чем сложнее создаваемая программа, тем труднее человеку в процессе программирования представить и удержать в памяти все взаимосвязи между модулями. Количество этих внутренних связей постепенно возрастает. Ошибки аппаратуры, естественно, должны анализироваться в аппаратной части модели, чтобы оценить ее надежность. В программной части они оказываются ошибками входных сообщений. Поэтому если характеристики надежности аппаратуры указываются в исходных данных, то *в программной части должны учитываться статистические характеристики этих аппаратных ошибок* и приниматься соответствующие компенсирующие меры. Здесь необходимо обеспечить правильное истолкование правильных сигналов аппаратуры.

• Если речь идет об ошибках собственно программной системы, то их источником (независимо от того, возникли они непроизвольно или введены сознательно) может быть только *человек* или, как принято говорить, *человеческий фактор*. Эти ошибки, как правило, не поддаются прямому статистическому анализу ни в методах программирования, ни в методах организации программных систем. Особенно острой эта проблема становится, если моделируются большие системы. Даже понимание их работы достигается постепенно. Универсального способа написать большую рабочую программу без ошибок не существует. Качество программной модели зависит от того, насколько в самой *программной системе и в системе программирования* предусмотрены меры по минимизации ошибок и упрощения их поиска, а также от понимания архитектуры системы заказчиком и исполнителем.

Вопрос о выборе системы программирования всегда актуален в программном моделировании. Здесь стоит вернуться к истории программирования. Ведь языков и систем программирования создано несколько тысяч! И это всего за 70 лет существования вычислительных машин. В этом массиве языков и систем одни ушли в небытие, другие остались и действуют. В частности, остался и здравствует язык Бейсик, созданный еще в 1960-х годах, чтобы приобщить к компьютерам всех начинающих, в том числе – младших школьников. Ныне его версии процветают, особенно среди непрограммирующих профессионалов – квалифицированных специалистов в своей области – ученых, инженеров, врачей, то есть людей, занятых осмыслением научных и прикладных результатов.

При проектировании этого языка (по данным Википедии) использовались следующие восемь принципов, согласно которым новый язык должен:

1) быть простым в использовании *для начинающих*;

2) быть языком программирования *общего назначения*;

3) предоставлять *возможность расширения функциональных приложений*, доступную опытным программистам;

4) быть *интерактивным*;

5) предоставлять ясные сообщения об ошибках;

6) быстро работать на небольших программах;

7) не требовать понимания работы аппаратного обеспечения;

8) защищать пользователя от операционной системы.

Важно заметить, что большинство из этих принципов можно отнести и к языкам, и к программным системам, применяемым при создании больших программных моделей, и рассмотреть эти принципы в том же порядке.

1. Простота использования и простота восприятия (иными словами, понимания) обеспечивают не только легкость изучения программного комплекса, но повышают его устойчивость к программным ошибкам, облегчают подготовку и проведение испытаний программной модели.

2. Для программных моделей требование иметь общее назначение обычно, так как большинство этих моделей имеет такой общий характер.

3. Требование расширения функциональных возможностей обязательно для программных моделей сложных систем.

4. Требование интерактивности как взаимодействия человека-пользователя с компьютерной моделью обязательно, так как в этом состоит цель создания модели.

5. Ясные сообщения об ошибках стали необходимым и привычным свойством современного программирования, в частности объектно ориентированного программирования.

6. Требование быстро решать простые задачи вытекает из первого – простоты применения.

7. Седьмой принцип – это единственный, который, применительно к моделям больших систем, вызывает серьезные возражения. Даже если рассматривать только чисто программные модели, то в случае жизненно важных систем необходимо учитывать аппаратную составляющую, в том числе ресурсы и надежность компьютерной системы. Если аппаратура в системе – это не только компьютерная часть, но и постоянный источник информации о состоянии внешней для модели среды, то следует уже рассматривать не программную, а аппаратно-программную модель системы, о чем и говорилось в начале статьи. Но тог-

да требования к программной модели, для которой информация аппаратуры является входным сигналом, должны быть ясно изложены в задании на разработку.

8. Последний принцип можно истолковать так: язык не должен зависеть от операционной системы.

В качестве еще одного примера можно привести другой язык программирования. Это известный и широко распространенный Паскаль. Он также был создан для первоначального обучения студентов. Однако вскоре его создатель Н. Вирт получил предложение дать версию языка для промышленного применения и разработать к нему компилятор. Это было выполнено. Язык программирования Паскаль получил промышленную разработку. Основным требованием к этому языку программирования также стала его простота применения. Следует обратить внимание на то, что успех применения и развития языков программирования и программных систем, независимо от их назначения, определяется их *воспринимаемостью*, то есть их *понятностью*.

Таким образом, оказывается, что среди программных систем к числу самых долгоживущих относятся языки программирования Бейсик и Паскаль (конечно, модифицированные за истекшие после их создания 50 лет). Их основным качеством является понятность, то есть простота восприятия любым мало-мальски подготовленным человеком. Это верно и для ученых, и для инженеров – для всех, кто занят созданием больших систем.

Одна характеристика качества программных систем – их воспринимаемость – отмечена, она признана во всем мире (за исключением некоторых отечественных «специалистов»). Но это только первый шаг к применению программных систем, для их создания нужно отметить и *простоту их применения*. Это второе качество, особенно важное при моделировании сложных систем.

Но всякая большая конструкция должна быть надежной. Как обстоит дело с программными системами? Ведь их надежность или ненадежность определяются, как уже не раз отмечалось, «человеческим фактором». Ошибки человека имеют много причин: это и его физическое и моральное состояние, профессиональное мастерство, множество других причин вплоть до простого непонимания архитектуры системы в целом, а также деталей этой архитектуры.

Здесь большими преимуществами обладает применение объектно ориентированных языков программирования. Заложенные в него основные принципы известны [3].

Программирование объектами, которые представляются на высоком уровне абстракции, то есть эффективного средства против сложности программирования, которое позволяет программисту сосредоточиться на существенных свойствах объекта, инкапсулируя менее важные свойства.

Программирование-моделирование. Программы моделируют объекты и явления реального мира.

Вычисления путем обмена сообщениями. Методы определяют, как объект будет отвечать на сообщение.

Программирование с применением полиморфизма. Сообщения обрабатываются разными получателями разными методами.

Программирование с применением наследования. Новые классы определяются как уточнения существующих классов.

Программирование с применением библиотеки классов.

Объектный (объектно ориентированный) подход по праву признан всеми известными экспертами программной инженерии как наиболее эффективный в современных условиях. Объектное программирование, конечно, требует усилий при использовании в практической деятельности. С помощью объектных технологий можно создавать проекты, в десятки раз более сложные и объемные, чем при использовании традиционного, даже модульного подхода. Объектный подход позволяет фиксировать ошибки и определять их положение в программе. Таким образом, в значительной степени преодолевается влияние человеческого фактора – основного источника ошибок в программах. То есть применение соответствующих методов программирования, в частности объектного подхода, позволяет добиться повышения надежности программных моделей,

Кроме того, применение основанного на этом подходе унифицированного языка объектного моделирования *UML* (*Unified Modeling Language*) значительно совершенствует анализ и проектирование программных моделей сложных больших систем.

Итак, мы выделили три основных качества программных систем. Это легкость их *восприятия* (понятность), *простота создания* программ (в том числе – программ и методов испытаний), *надежность*. Но есть и обязательное четвертое качество – *стоимость* программной модели. По сравнению с предыдущими тремя стоимость сравнительно просто поддается количественной оценке. При этом важно учитывать стоимость подготовки персонала, его работы над созданием модели системы и ее испытаний на всех этапах ее жизненного цикла.

Наши заметки будут неполными, если в них не затронуть тему испытаний программных моделей больших систем. Именно эти испытания являются лучшим доказательством надежности системы и ее соответствия требованиям технического задания. Но проведение испытаний требует глубокого понимания архитектуры конкретных больших систем и потоков информации, которыми обмениваются модули системы и ее аппаратура. Испытания системы – важнейшая часть ее разработки и сопровождения. Здесь многообразие возможных решений определяется многообразием сложности возможных больших систем. Научное обеспечение этой работы нуждается в тщательной разработке.

Пожалуй, в настоящих заметках достаточно много сказано о том, что аппаратная и программная части аппаратно-программных комплексов имеют свои специфические особенности. Эти особенности необходимо учитывать при создании моделей больших систем. Но возникает естественный вопрос: *как измерять качество и надежность программных систем?* В аппаратной части (на языке тех, кто работает с моделями, – в «железе», *hardware*, по-английски) этот вопрос решается в течение многих лет. Оценки качества программной составляющей («софта» – *software*) вызывают недоумение даже у специалистов. Нередко можно услышать возмущенный вопрос «Какая может быть метрология у программных систем?» Ответ на него обычно ищут у аппаратной части, порой с получением «научных» результатов, не имеющих отношения к делу. Выше был дан ответ на вопрос о том, какие характеристики качества имеют программные модели. Эти характеристики – не наше изобретение. В мировой компьютерной литературе эти характеристики общеизвестны. Например, взять известную работу [1], которая к 2009 году выдержала 7 изданий [2]. Это отмечается и в других работах, где нередко обращается внимание на «читабельность» – *readable*, по-английски, то есть воспринимаемость. Но нужно ответить и на вопрос, *как измерить* эти характеристики качества (кроме стоимости программной системы, которую можно постараться как-то измерить)?

На этот вопрос отвечает наука *метрология*. Согласно ей, измерения являются инструментом познания объектов и явлений окружающего мира. Поэтому метрология относится к науке, занимающейся теорией познания, – гносеологии. Объектами измерений являются и физические, и нефизические величины. Последние особенно распространены в экономике, медицине, искусстве, спорте, военном деле, то есть там, где велика роль человеческого фактора. Теперь мы видим, что он большое значение имеет и в оценке качества программ.

Если не измерять качество программной модели ее объемом и весом сопровождающей технической документации, то естественным оказывается вывод: *лучшим оказывается то программное обеспечение, которое легко воспринимается в первый раз и легко восстанавливается в памяти даже через годы*! Вопрос о простоте применения при создании модели и ее испытаниях в жизненном цикле решается таким же образом, но с применением более совершенных методов программирования моделей больших систем.

На вопрос о надежности программного обеспечения отвечает само развитие программных языков и методов программирования. Только при их развитии можно достичь повышения надежности больших программных систем, не обманывая себя поверхностными статистическими данными.

Особенно остро стоит вопрос о надежности именно больших систем, в частности о соответствии архитектуры моделей больших систем поставленным перед ними задачам. Действительно, сами объекты моделирования настолько сложны (возьмем в качестве примера систему жизнеобеспечения современного города), что простыми проверками (тестированием) никаких гарантий адекватности модели самой системе получить невозможно, нужна научная разработка программ испытаний (программ в смысле планов) и соответствующих методов испытаний программных моделей больших систем. Он может разрешаться при научном решении проблемы испытаний моделей таких систем.

Как же быть с метрологией программного обеспечения? Может быть, действительно ее не существует? Раздел метрологии, посвященный измерению качества, носит название *квалиметрии*. Он выделяет единичные и комплексные показатели качества. Единичные относятся к одному из свойств продукции, а комплексные характеризуют сразу несколько свойств. Мы уже выбрали четыре характеристики качества для моделей больших программных систем. Из них только стоимости можно дать измеряемую оценку. Эта оценка важна при расчете расходов на обучение персонала, работы персонала над созданием системы, а также на подготовку и проведение ее испытаний. Но испытаниями и сдачей заказчику большой системы ее жизненный цикл не кончается. Начинается этап ее сопровождения, который включает корректировки модели в связи с новыми источниками информации, получением дополнительных данных, возникновением ранее не проявившихся

исключительных ситуаций (новых ошибок в программном комплексе и влиянием на него ранее непредвиденных источников информации). В конечном счете формируется новое понимание явления и новое представление о явлении. Оценка этих результатов, то есть целесообразности и осуществимости предлагаемых стратегических решений, носит априорный и в значительной степени субъективный характер и потому требует нового моделирования.

Как измерить остальные три характеристики? Во многих областях измерений, объективные оценки характеристик качества продукции добываются с большими трудностями и даже по прошествии многих лет. Эти оценки имеют *субъективный* характер, и замена их объективными пока оказывается невозможной. Это наблюдается в гуманитарных науках – искусстве, литературе, музыке, исполнительском мастерстве. Это мы наблюдаем в спорте, в военном деле, да и в политике. Очевидно, это наблюдается и в создании программных систем. Слишком много факторов влияет на субъективные оценки многих людей, слишком много личного опыта, симпатий и привычек привносит в оценки каждый «судья». В перечисленных областях деятельности, где номенклатура основных величин не определена, теория прямых измерений не находит пока эффективного применения. В ней принято различать пять типов шкал: *наименований, порядка, разностей* (*интервалов*), *отношений* и *абсолютные*.

Самая простая шкала – это ***шкала наименований***. Она характеризуются только отношением эквивалентности (равенства). Примером такой шкалы является распространенная классификация (оценка) цвета по наименованиям (атласы цветов насчитывают до 1000 наименований).

Для нашего случая измерения качеств программных моделей пригодны две шкалы.

Шкала порядка – требует расположения оценок в порядке возрастания или убывания измеряемой величины. Такая расстановка измерений по шкале порядка называется *ранжированием*. Для облегчения измерений по шкале порядка некоторые точки на ней можно зафиксировать в качестве опорных (реперов). Недостатком реперных шкал является неопределенность интервалов между реперами. Эти шкалы применяются при оценке знаний студентов по баллам, результатов тестирования, а также мощности землетрясения – по 12-балльной шкале, силы ветра – по шкале Бофорта, чувствительности фотопленок и т.д.

Шкалы разностей (интервалов) отличаются от шкал порядка тем, что по ним можно уже судить не только о том, что одна величина оценки качества больше другой, но и оценить, насколько она больше.

Попытки применения ***шкал отношений*** и ***абсолютных шкал*** пока неизвестны, так как невозможны точные измерения. Стоит ли в связи с этим приходить в отчаяние? Действительно, создание программных систем располагается на границе точных и гуманитарных наук, но сама эта наука не является чисто гуманитарной, хотя пользуется их методами. По нашему мнению, положение не безнадежно. Да, инструментальных методов измерения характеристик качества программных систем сегодня не существует, но существует и стремительно развивается технология программирования.

Сегодня квалиметрия насчитывает пять характеристик качества, но совсем недавно применение квалиметрии для оценки качества программных моделей даже не рассматривалось. По любой из их характеристик качества мы можем сегодня применить для оценки шкалу измерений или шкалу разностей, а также провести измерения ***экспертным методом***, который широко применяется в квалиметрии, искусстве, спорте, медицине и других областях и опирается на субъективные оценки ряда квалифицированных специалистов.

Конечно, трудно в ближайшее время ожидать исчерпывающего решения проблемы точных измерений качества программных систем и моделей. Очень трудно методы метрологии применять к программированию, детищу математики! Но на примере программных моделей сложных систем мы видим, что этот раздел математики уже вторгся в область человеческого сознания. Не стоит закрывать на это глаза и применять к сложным объектам примитивные, якобы математические методы. Перефразируя далеких предков, можно сказать: «человек познает самого себя!» Не только программирование сложных систем, но и методы их испытаний развиваются стремительно. Сегодня программирование далеко не то, что было двадцать и даже десять лет назад. Не могут не развиваться методы измерений качества сложных программных систем. Но не следует ожидать их мгновенной и полной формализации. Понимание человеком глубины этой проблемы развивается, как и сама метрология.

Литература

1. Себеста, Р.У. Основные концепции языков программирования. – М. : Вильямс, 2001.

2. Sebesta, R.W. Concepts of Programming Languages (7ed). – Addison Wesley, 2005.

3. LaLonde, W.R., Pugh, J.R. Inside Smalltalk. – London, Prentice-Hall, 1996.