Программы – не стихи, их надо проектировать, а не писать

Недавно (20.01.2017 г.) в газете «КоммерсантЪ» была опубликована статья «В «Росатоме» нашли проблемы с ядром» о легитимности используемого на атомных электростанциях программного обеспечения (ПО). Там, в частности, сказано, что имеющееся ПО не позволяет понимать, как программа будет себя вести при определенных воздействиях – одинаковых или разных, а также неизвестно, кто, как именно, когда и какие изменения в нее вносил. Более того, в статье говорится, что на этот софт нет никакой документации. Такая ситуация характерна не только для описанных объектов – иногда оторопь берет от осознания того, что в современном мире, в котором невозможно заранее спроектировать разве что человека, программы, управляющие сложнейшими процессами, пишутся, как стихи. Но стихами ядерными реакторами не управляют.

Стихи пишут, чтобы выражать чувства и эмоции, а программы – в том числе и для управления ответственными объектами. Программное обеспечение, которое создается под заказ, решает конкретную задачу пользователей, но как будет себя вести программа при определенных условиях, как правило, известно, в лучшем случае, разработчику. Как сделать процесс прозрачным и более понятным и эффективным, применив технологию автоматного программирования, знает заведующий кафедрой технологий программирования Университета ИТМО Анатолий Абрамович Шалыто.

Программисты во всем мире обычно не проектируют программу, отмечает профессор. Можно задаться вопросом, а зачем это делать, а если сразу можно написать код, основываясь на опыте и знаниях. Какой смысл в формализации?

«*Бардак с ПО творится почти во всем мире. Если аппаратура проектируется всегда, и только потом производится, то проектная документация на программы на практике выпускается крайне редко. Универсальный язык моделирования (UML), на который одно время у многих были большие надежды, используется далеко не всегда, причем даже в тех случаях, когда он применяется, диаграммы обычно строят одни люди – архитекторы, а другие – программисты в лучшем случае в них заглядывают. Есть и другие причины.*

Во-первых, изучать UML в российских вузах начинают примерно на 5-м курсе, когда программисты уже много лет пишут программы так, как их учили кого с пятого, а кого с восьмого класса школы. А тут приходит старый профессор и начинает им рассказывать про моделирование (проектирование) программ. Студенты рассматривают этот предмет, как один из многих, которые надо сдать для получения диплома, но не более того.

Во-вторых, когда они уже все знают и умеют и начинают работать, то тут уже или все куда-то торопятся, или у заказчика нет денег на проектную документацию, а у разработчика нет специальных людей, которые проектируют будущее ПО. На самом деле, это должен делать сам программист, но он либо не умеет, либо не хочет, либо на это нет времени и средств. Ну а если нужно что-то исправить в ПО, то программисты обычно просто берут код и на языке программирования исправляют эти ошибки так, как им удобно, а потом показывают заказчику работу измененной программы, а не измененную диаграмму. В этой ситуации мне очень жаль тех заказчиков, например, военпредов, которые по долгу службы должны согласовать

программное обеспечение так же, как они ***согласовывают документацию*** *на аппаратуру* ***при ее проектировании***», – обрисовал проблему Шалыто.

Частая ситуация – в команде разработчиков происходит «замена игрока», а в код надо внести изменения, но для понимания чужого кода нужно понять, как мыслил программист, какую архитектуру построил, какое поведение должно быть у программы. Для этого хорошо бы посмотреть проектную документацию на программу, из которой станет ясно, как она создавалась, и что получилось – тогда и согласовать с заказчиком техническое задание, по которому создается программное обеспечение, что особенно важно для ответственных объектов, станет легче. Но есть ли она? И везде ли?

Решение есть

Выход из ситуации Анатолий Шалыто предложил уже давно – еще в 1991 году. Его предложение называется «автоматное программирование» (http://is.ifmo.ru/books/_book.pdf), суть которого состоит в том, что при разработке ПО применяется тот же подход, который используется в инженерных науках при создании автоматизированных объектов управления. По словам ученого, изобретенный им «велосипед» вот уже почти 25 лет «стоит в гараже», из которого его иногда «выкатывают», чтобы «поездить», например, в ОАО «НПО «Аврора» (http://is.ifmo.ru/works/_volobuev.pdf и http://is.ifmo.ru/works/_automata_plis.pdf).

Автор предлагает применять этот подход всегда при создании программ со сложным поведением. Его суть состоит в том, что создаются автоматизированные объекты управления, состоящие из систем управления (автоматов) и объектов управления, которые могут быть и реальными, и виртуальными. При этом поведение этих систем смогут «прочитать и понять» и инженер, и программист-разработчик, и любой другой программист, и заказчик, и начальник. Это поможет при необходимости быстро разобраться в чужом коде.

Многие считают, что автоматное программирование – это программирование с автоматами. Это не так, также как линейное программирование – это не программирование с линейками (http://is.ifmo.ru/download/2008-03-17_automata.pdf).

Автоматное программирование – это парадигма программирования, при использовании которой о программах предлагается думать, как об автоматизированных объектах управления.

«*Я считаю, что до того, как начинать программирование системы, необходимо разобраться с ее поведением, и предлагаю не начинать сразу писать программу, а сначала нарисовать «картинки» – графы переходов, которые опишут поведение будущей программы в терминах состояний, переходов между ними и действий, выполняемых в состояниях и/или переходах. Следовательно, сначала нужно формально описать поведение программы – решить и изобразить в наглядной графической форме, что она должна делать. Если говорить применительно к человеку, то описание можно проводить так: человек находится в одном из двух состояний: «жив» или «мертв». Если жив, то тогда он может перейти в одно из вложенных состояний: «здоров» или «болен», ну и т. д. Если такая система автоматов построена, то по ней вручную или автоматически можно создать программу, более того, автоматы мы умеем генерировать по их спецификации»,* – поясняет автор.

Есть еще один важный момент: техническое задание согласно ГОСТ окончательно оформляет разработчик. При наличии графов переходов заказчик может достаточно четко понять, что хочет сделать программист, и, при необходимости, внести в задание исправления. Согласованное и формализованное техническое задание затем будет исполнено в виде программного кода, которого может не увидеть не только заказчик, но и программист, если код по графам строится автоматически (http://is.ifmo.ru/present/2012/Yankin-Shalyto-PLIS.exe).

По мнению Шалыто, качество программ, в основном, обеспечивается не итоговым тестированием или верификацией, а тем, что на ранних стадиях создания программы заказчик и разработчик однозначно и верно поняли друг друга и согласовали техническое задание. При этом необходимо отметить, что автоматные программы могут весьма эффективно формально верифицироваться (http://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf и http://is.ifmo.ru/works/_2011_03_04_remizov.pdf).

Конечно, такой подход требует от заказчика понимания проектной документации на ПО и, в частности, графов переходов. Однако это все равно в разы понятнее и доступнее, чем изучение программного кода непрофессионалами или даже самими авторами программ через несколько лет после завершения разработки. Кроме того, существуют приложения для моделирования автоматных программ, где в интерактивном виде можно проследить все переходы от одного состояния к другому в автоматах, а потом сгенерировать код программы по согласованным с заказчиком, графам переходов (http://is.ifmo.ru/works/2014/yankin-control-block.pdf). Такое проектирование программ очень напоминает проектирование схем, которое понятно инженерам.

«*Это просто разные способы мышления: у инженера – одно, у программиста – другое.* ***Сегодня программирование – это не инженерная наука, а прикладная математика***. *Программистов готовят иначе, чем инженеров, они в первую очередь должны знать математику. И потом уже применять ее. А если бы у них была и инженерная подготовка в области управления дизелями, электроэнергетикой и ракетными комплексами, например, то это было бы уже инженерное программирование. Программисты – они ведь очень умные, они применяют автоматы, как и другие математические абстракции, так и тогда, когда считают нужным. Мое мнение – их надо всегда применять для описания процессов управления, где есть зависимость поведения от предыстории*», – сказал Анатолий Шалыто.

Пример **проектной** документации приведен здесь: http://is.ifmo.ru/unimod-projects/camera/, а вот пример такой документацией для сложного проекта по адресу: http://is.ifmo.ru/projects/dg/. С документацией для еще более сложного проекта можно ознакомиться здесь: http://is.ifmo.ru/automata/_s7300.pdf.

Особенно классно, если поведение программы удается описать в виде одного графа переходов, в котором число вершин и состояний совпадает. Это позволяет наглядно и точно показать, как будет работать система, и при автоматическом построении по ней программы она будет вести себя так, как была спроектирована. Это и удобно, и безопасно. Конечно, это не решит все проблемы с заказным ПО, но то, что такой метод программирования облегчит жизнь и заказчикам, а, в конечном счете, и разработчикам – это практически гарантировано.

В заключение Шалыто сказал: «Если не хотите применять то, что предлагаю я, и что представлено на моем сайте http://is.ifmo.ru/, используйте нечто аналогичное – то, что предлагают другие (http://www.swd.ru/files/pdf/brochures/Rhapsody.pdf). Меня и это устроит, но только сделайте так, чтобы программное управление ответственными объектами делалось по-человечески в хорошем смысле этого слова, а не так, как это делается в большинстве случаев сегодня».

Текст Е.В. Софроновой (e.sofronova@corp.ifmo.ru), Университет ИТМО, Управление по связям с общественностью.