Глава 4 | Количественный и качественный подходы

Введение

Одна из прочнейших традиций в науке – это почтительное отношение ко всему количественному, точному, строгому и безусловно истинному. Однако фактически в окружающем мире господствуют неточность и неопределенность, и безоговорочно утверждать что-либо о нем трудно. Фактом является и то, что точность и определенность обходятся дорого. В стремлении соблюсти подобное «почтение» мы склонны закрывать глаза на эти факты, упуская из виду величину издержек, связанных с достижением высокой точности и малой неопределенности. Другой аспект нашего стремления к почтительности – то, что в большей части научной литературы элегантность берет верх над насущной важностью.

Лотфи А. Заде

«Soft Computing and Fuzzy Logic»,
IEEE Software, ноябрь 1994, ©1994 IEEE

Раздел «Q» словаря невелик, и почти все слова там начинаются с «qu». Но на этих нескольких страницах выделяются две интеллектуальные вершины – слова quantitative (количественный) и qualitative (качественный).

«Количественный» относится к числам: «количество или число предметов; возможность измерения благодаря наличию размера, веса, количества или численности».

С другой стороны, «качественный» означает «имеющий качество или стремящийся к нему», где качество определено как «степень или уровень совершенства».

Размышляя над этими двумя определениями, я ощутил некоторую нерешительность. В обществе принято строго различать количественные и качественные методы. Одни вещи допускают численное описание, другие – нет. Исходя из этой традиции, я полагал, что данные два слова ясно и четко различаются между собой.

Но, постойте! Если понятия *качества* и *количества* столь различны, значит ли это, что о качестве нельзя говорить на языке чисел? Но если качество можно измерить количественно, то между этими двумя понятиями нет непреодолимой преграды.

Конечно, понятие качества весьма смутно. Оно завело в умственный тупик главного героя книги Пирсига «Дзен и искусство ухода за мотоциклом». Думаю, оно вызывает больше всего раздоров среди любителей давать определения в компьютерной области. (Чтобы проследить за ходом их мыслей, посмотрите обсуждение моей книги «Создание качественного программного обеспечения» («Building Quality Software», Prentice-Hall, 1992).) Если так трудно дать определение этому понятию и люди сходят с ума, просто пытаясь выяснить его смысл, удастся ли подойти к нему с количественной точки зрения?

Ответом будет «мы все время это делаем» или, по крайней мере, «все время пытаемся это делать». Мы оцениваем качество бегунов, измеряя скорость их бега. Мы оцениваем качество прыгунов в длину, измеряя расстояние, на которое они прыгают. И если нет другого выхода, мы оцениваем качество ныряльщиков и конькобежцев с помощью коллегий судей, которые показывают карточки с цифрами своих оценок за выступление. Если нет реальных количественных оценок для качества, можно их придумать.

В программировании мы тоже постоянно пытаемся измерять качество. Мы определяем количество ошибок, скажем, на строку кода или функцию, говоря при этом, что измеряем надежность. Мы подсчитываем операторы и операнды (и массу других вещей), говоря при этом, что измеряем сложность. Мы хронометрируем прогон программы, говоря при этом, что измеряем ее производительность. Мы подсчитываем инструкции и данные, говоря при этом, что измеряем эффективность использования памяти. Все, что мы здесь измеряем, относится к неуловимой вещи под названием качество.

Но не станем слишком далеко заходить в эту сторону. Несмотря на все упомянутые здесь количественные аспекты качественных оценок, между этими двумя понятиями все же существует фундаментальная разница. Одни вещи легко представить на языке чисел. Другие – нет. Это фундаментальное различие, и его значение в области программирования я собираюсь обсудить в этой главе.

Что общего между количественными методами, качественными методами и творчеством? Мое мнение таково: углубляясь в область гибких, эвристических, разумно достаточных методов решения задач, то есть тех аспектов творчества в программировании, которым посвящены предыдущие главы, мы переходим из той сферы, где легко делать измерения, в другую, где это не так просто. Мечтать о том, чтобы задачи этого мира можно было решать, в надлежащей мере применяя дисциплину, формальные методы и оптимизацию, – все равно что желать, чтобы все в этом мире легко допускало измерение.

Вопреки нашим желаниям мир, в котором мы живем, – мир сложной культуры, интеллекта и творчества – оказывается не так прост. Сравните, например, эту книгу с другими статьями о программировании, которые вы, возможно, прочли. Как бы вы стали сравнивать ее с последними книгами Вейнберга, или Юрдона, или Демарко? Чего бы все мы хотели – в том числе я, как бы ни разочаровал меня результат, – так это дать какую-нибудь ясную количественную оценку каждой книге, чтобы можно было все их расставить по порядку. Скажем, Демарко получает 9,723 балла из 10 за свой светлый ум, нестандартные взгляды и юмор. Юрдон – 9,683 за исчерпывающее освещение, огромные связи и широчайший круг источников информации. Вейнберг – 9,706 за изобретательное сочетание психологических и технических идей, описание удивительных случаев и невероятную продуктивность. Гласс? Ему 1,738 балла за мучения.

Однако все это достаточно проблематично (даже если забыть о том, что я занял последнее место!). Немногие отнеслись бы к подобным числовым показателям с доверием. Во-первых, они субъективны. (Рискну предположить, что, к примеру, моя мама дала бы авторам следующие оценки: Гласс – 9,999, Демарко – 1,111, Юрдон – 1,111, Вейнберг – 1,111). Во-вторых, трудно оправдать значение составляющих эти оценки чисел, особенно по мере удаления вправо от запятой. Иными словами, мы ввели количественную оценку там, где она ни к чему. Качественный подход был бы здесь более уместен. Таки есть действительная разница между этими двумя подходами!

Чем же мы займемся в этой главе? Сначала мы рассмотрим один из вредных компьютерных трюизмов: «Нельзя управлять тем, что нельзя измерить». Как и большинство бабушкиных (и дедушкиных) сказок, он гораздо менее правдив, чем может показаться на первый взгляд.

Затем мы рассмотрим еще несколько примеров связи количественных методов с компьютерной обработкой, каждый раз убеждаясь, что качественные методы, хотя и менее привлекательные, иногда оказываются единственно доступными. Наконец, подобно столяру, с удовольствием забивающему в изделие последний гвоздь, мы закрепим свою точку зрения последним разделом. Интересно, как вы оцените эту главу по шкале от 1 до 10 баллов?

4.1. «Нельзя управлять тем, что нельзя измерить» – так ли это?

«Нельзя управлять тем, что нельзя измерить» – один из известных компьютерных и программистских трюизмов.

Но справедлив ли он? Правда ли невозможно управлять тем, что нельзя измерить?

Отвечаю: неправда. На самом деле, мы занимаемся этим постоянно. Приведу несколько достаточно общих примеров.

Начнем с того, что мы уже несколько десятилетий управляем программными проектами, обычно не пользуясь при этом никакими метриками. Конечно, можно говорить о том, что мы делаем это довольно плохо, но несомненно, что даже при «плохом» управлении мы успешно вступили в «компьютерную эпоху», которая войдет в историю как одно из самых важных предприятий человечества. Могло ли это произойти, если бы мы действительно не управляли тем, что не можем измерить?

Мы управляем исследованиями и разработками – деятельностью, фактически не поддающейся измерению, и результаты этой работы изменили облик мира.

Мы управляем руководителями. В иерархически организованной сфере корпоративного управления, чем более высокую должность вы занимаете, тем труднее количественно оценивать работу ваших подчиненных. И даже если вы будете оспаривать успешность наших действий, мы все же делаем это.

Фактически, самый общий из возможных здесь выводов состоит в том, что, управляя работниками информационной сферы, вы, весьма вероятно, окажетесь в области, где роль количественных методов невелика.

Поэтому, думаю, справедливо будет сказать, что указанный трюизм не верен. Мы на самом деле управляем тем, чего не можем измерить, и именно сейчас, когда вы читаете эти строки, этим занимаются тысячи руководителей по всему свету.

Конечно, хотелось бы управлять на количественной основе, имея для этого количественные показатели. Однако и в отсутствие соответствующих метрик мы справляемся с управлением.

Тут нам, несомненно, повезло. Управление программными разработками десятки лет существует без надежных численных показателей, на которые можно опереться. Программные продукты, благодаря которым наступила компьютерная эпоха, пришли из той области, где не было ни надежных показателей, ни проверенных рекомендаций о том, как их применять.

Конечно, попытки были. Мой начальник в конце 1950-х пытался реализовать достаточно мощные по тем временам методы измерений для оценки разработки программного обеспечения. Мне было поручено определять и внедрять некоторые метрики. Насколько я припоминаю, они были достаточно детальны и конкретны и далеко не ограничивались привычным заданием контрольных точек, включая измерение таких вещей, как степень подробности комментариев, которыми снабжались все элементы данных в программе, а также единиц измерения этих элементов (думаю, можно назвать их «метриками метрик»).

Это была одна из первых интересных попыток управлять с помощью количественных показателей, но она оказалась мертворожденной. Большинство коллег этого руководителя не проявили интереса к управлению на таком уровне детализации, и никто больше не применил эти методы. Технические работники не приветствовали то, что казалось им вмешательством руководства в их работу, и всячески уклонялись от нововведений. Этот руководитель ушел на повышение, и вводимые им метрические подходы без него зачахли.

Лет десять спустя подобные опыты появились и в научных кругах. Мюррей Холстед (Murray Halstead), известный своей работой над созданием компиляторов, сделал примечательную попытку внедрить науку и измерения в программирование, предложив нечто под названием «научное программирование» (software science). Интересно отметить, что примерно тогда же появились термины «программная инженерия» (software engineering) и «программная физика» (software physics). Некоторые из этих понятий сохранились до нашего времени. «Научное программирование», как известно сейчас многим, занималось оценкой сложности программ путем подсчета количества операторов и операндов.

Работа Холстеда и его последователей вызвала огромные споры. Допустимость предлагаемых оценок критиковалась как учеными, так и практиками, причем ученые сомневались в правильности рассуждений Холстеда, а практики – в пользе от его изысканий. Страсти разгорались при одном лишь упоминании предмета. На одной из компьютерных конференций при обсуждении научного программирования один видных деятелей программной инженерии обвинил другого в приверженности «астрологии» (имея в виду лженауку). В настоящее время большинство авторитетов в этой области выражает сомнение в ценности этой идеи, но, тем не менее, количество посвященных ей публикаций в научной печати огромно, а также, что любопытно, появился ряд довольно успешных коммерческих продуктов, занимающихся расчетом этих метрик.

Те, кто оценивает практическое состояние дел, находят, что применение метрик в управлении программными проектами еще не вышло даже из

пеленок. На конференциях «Применение метрик в программировании» («Applications of Software Metrics») Билл Хетцел (Bill Hetzel), представляющий организацию-спонсора, регулярно докладывает о распространенности метрик на практике и считает ее ужасающей.

Любопытно, однако, что недостатка кандидатов в метрики нет. В действительности, в области метрик наблюдается раздвоение. Метрики, предлагаемые учеными, включая Холстеда и его многочисленных последователей (Зусе (Zuse) составил едва ли не энциклопедию по этой теме), развивались в одном направлении. Метрики же, предлагаемые практиками и якобы не используемые, довольно успешно развивались, но в другом направлении (для которого типичны работы Роберта Грейди (Robert Grady) из Hewlett-Packard).

(Разумеется, слово «успешно» в предыдущем абзаце употреблено в ироническом смысле. Как можно считать что-то успешным, если оно малоупотребительно? Однако справедливо будет заметить, что если сегодня какая-либо фирма-производитель программного обеспечения пожелает внедрить у себя программу применения метрик, она найдет богатый выбор как практических, так и теоретических предложений.)

Проблема тезиса «нельзя управлять тем, что нельзя измерить» связана с вызываемым им фанатизмом. Те, кто глубоко уверовал в полезность метрик, огорчаются отсутствием какого-либо прогресса в этой области. Чтобы привлечь внимание спонсоров и исполнителей, они придумывают броские фразы. Но такие фразы, прежде чем поверить им, следует хорошенько анализировать.

Можно ли управлять тем, что нельзя измерить? Конечно, да.

Если нам будет дан выбор, предпочтем ли мы методы управления, основанные на измерениях? И снова ответ утвердительный.

Мир гораздо сложнее, чем это следует из крылатых фраз, которыми мы его описываем. И вздорность обманчивых фраз вроде «нельзя управлять тем, что нельзя измерить», отнюдь не стимулирующих поиски, отчасти объясняет нынешнее состояние дел, то есть отсутствие каких-либо измерений.

4.2. Математика и компьютерные науки

Компьютерные науки выросли на факультетах математики. Там они делали первые шаги, там достигли известности и остаются в тесной связи с математикой. Во многих колледжах и университетах математика и компьютерные науки сосуществуют в рамках единой организационной структуры.

Но насколько эта связь с математикой важна для компьютерных наук? В особенности в том, что касается программирования?

Многие сочтут такой вопрос еретическим. «Как можно сомневаться? – скажут они. – Логика и точность математического представления – неотъемлемая часть компьютерных наук, так было всегда и будет впредь». И это трудно оспорить.

Однако есть и другая точка зрения. Она более сложна и необычна, и потому требует тщательной формулировки.

Когда-то компьютеры служили только для вычислений. Фактически, у них тогда и не было возможностей заниматься чем-то другим. Никаких других символов, кроме цифр, они не знали, и невозможно было представить, что эти удивительные новые устройства годятся для чего-то еще.

То была эпоха вычислений как раздела прикладной математики. Компьютеры выполняли с числами немыслимые трюки, которые прежде считались невозможными или требовали невероятного количества математиков и чудовищно долгого времени. Поначалу компьютеры могли повысить производительность труда только тогдашних математиков.

Но эта эпоха закончилась 50 лет назад. Как только набор символов, которыми оперировали компьютеры, вышел за пределы десяти цифр, они превратились, я бы сказал, в «информационные процессоры». Прошли те дни, когда шестибитный байт позволял писать только заглавными буквами, и сегодня, когда мы стремимся заложить в «вычислительную машину» все символы всех существующих в мире языков, основной материал для компьютера – уже не числа, а оцифрованная информация.

Возникли другие научные дисциплины, в которых вычисления и программирование рассматриваются с точки зрения информации, а не математики. Однако, так или иначе, компьютерные науки остаются самым влиятельным участником в этой области, а они продолжают рассматривать математику как сердцевину всего. Например, в правительственных докладах, посвященных будущему программирования, часто звучат призывы «добавить математики» во все еще незрелые теорию и практику программирования.

Между тем, за прошедшие десятилетия характер компьютерной практики изменился. Вначале вычисления, ориентированные на математику и информацию, представляли собой две весьма различные и слабо связанные дисциплины. Специалисты по вычислениям были воспитаны на механических калькуляторах и рассматривали компьютеры как чрезвычайно быстрые и надежные арифмометры. Специалисты по информации выросли на перфокартах и рассматривали компьютеры как чрезвычайно быстрый и надежный инструмент для обработки перфокарт. Математиче-

ские компьютеры разговаривали на языке двоичных слов, состоящих из фиксированного количества бит, и поначалу редко использовали какие-либо «символы», даже цифры. Напротив, информационные процессоры пользовались цифрами и символами, а «слово» рассматривалось динамически: программы устанавливали «метку слова», чтобы указать, сколько символов содержит текущий вариант отдельного информационного поля.

Постепенно различия в аппаратном обеспечении для нужд вычислений и обработки информации стерлись. В начале 1960-х было замечено, что двоичные компьютеры с возможностью обработки символов, а также десятичных и двоичных вычислений оказываются быстрее и дешевле, чем прежние машины для обработки информации, ориентированные на символы и цифры. Когда в середине–конце 1960-х на сцене появилась IBM 360, аппаратные различия между этими двумя областями фактически исчезли. Одни и те же компьютеры могли экономично выполнять обе функции, и необходимость разделения аппаратных платформ отпала. Да, в информационном процессоре по-прежнему десятичные вычисления выполнялись как дополнение к двоичным, но это было незначительное отличие по сравнению с остальными частями компьютера, большинство которых занималось более существенными вещами, чем просто арифметика.

Современные компьютеры, и это придется признать даже самым упорным приверженцам математики, применяются гораздо шире, чем только для арифметики и математики. Существуют текстовые процессоры, тренажеры, игры и различные инструменты программирования, такие как компиляторы, в которых практически нет математики. Конечно, есть и электронные табличные процессоры, выполняющие значительный объем расчетов, а также научные и инженерные программы, где математика существенна, но все математические приложения, безусловно, занимают меньшую часть всего спектра практического применения компьютеров.

Закончив экскурс в историю, вернемся к нашему исходному вопросу. Насколько важна для компьютерных наук связь с математикой? Как следует из нашего краткого исторического обзора, значение математики как одной из функций сильно понизилось.

Но, может быть, она сохраняет свое скрытое значение? То есть, существуют математические идеи и мастерство, которыми важно владеть компьютерному специалисту, даже если ему не приходится применять математические методы? Многие компьютерные специалисты придерживаются такой точки зрения.

Любопытно, что при ответе на этот вопрос возникает дихотомия между аппаратным и программным обеспечением. Специалисту по аппаратной части ответ явно кажется положительным. Но специалисты по компью-

терным наукам простирают эту важность и на сферу программного обеспечения.

Я называю это «синдромом латыни». Давным-давно, когда в школах постепенно переставали преподавать латынь, кто-то во всеуслышанье заявлял, что умственные навыки, приобретаемые при изучении латыни, имеют большое значение для освоения других учебных дисциплин. Благодаря именно такому ходу мысли латынь просуществовала в качестве учебного предмета гораздо дольше, чем требовало ее практическое значение. Думаю, математика может оказаться для компьютерных наук тем же, чем в свое время стала латынь для прочих образовательных предметов.

Следует, конечно, правильно понимать эту аналогию. Математика не является умирающей наукой и никогда ею не станет. Ее вклад в другие науки и нашу жизнь в целом велик и значителен, а логика и точность, необходимые для занятий математикой, составляют важную часть работы программного обеспечения. В самом деле, программные системы – это едва ли не олицетворение логики и точности. Программный продукт, в котором отсутствуют логика и точность, бесполезен и даже опасен.

Однако здесь возникает вопрос. Действительно ли математика лучше всего подходит для обучения программистов логике и точности? Прежде всего, насколько мне известно, никто никогда не изучал этот вопрос. Математиков, перешедших в программисты, он не интересует, а всем остальным, испытывающим благоговение перед красотой и значительностью математики, претит поднимать или изучать такой спорный вопрос. Как я уже говорил, здесь пахнет ересью.

Должен признаться: я бывший математик. Я получил диплом в одной из математических цитаделей несколько десятилетий назад. И хотя мой диплом получен давно, а знания очень устарели, я все же полагаю, что обладаю некоторой квалификацией, дающей мне право критиковать мою родную науку.

Едва начав движение от науки к практическому программированию, я заметил, сколь мало то, чему меня учили, соотносится с моей практической работой. Я проходил подготовку как «чистый» математик, среди ученых, которые, как и многие их коллеги-теоретики, свысока смотрели на любое практическое применение своей науки. А работа на производстве, куда я попал, состояла в применении математики. В той среде, где я очутился, почти все прослушанные мной курсы оказались ненужными, за исключением пары компьютерных курсов.

Все это, как я уже говорил, давняя история. Разумеется, нетрудно заметить, что математика, как ее преподавали несколько десятилетий назад, может иметь слабое отношение к современному программисту. Отсюда

следующий очевидный вопрос: может быть, для практического программирования полезнее современная математика?

Я просмотрел много книг, рекомендованных моими коллегами математиками и программистами, чтобы найти связь между их содержанием и тем, чем я занимаюсь уже больше полувека, и я нахожу эту связь весьма слабой, если она вообще имеется. Если эти математические книги, входящие в обязательный круг чтения современных компьютерных специалистов, действительно важны для их работы, то это относится к той области программирования, с которой я не имел дела. А надо сказать, что я (в отличие от своих ученых коллег) потрудился едва ли не во всех его областях!

Наверное, поэтому я и придумал «синдром латыни». Я пришел к убеждению, что мы продолжаем делать вид, будто математика связана с программированием, несмотря на то что времена, когда это действительно было так, давно миновали. Важно отметить, что я не отрицаю пользу математики для программиста. Я просто хочу сказать, что для большинства программистов эта польза очень невелика, если вообще есть. Не исключено, что изучение латыни было бы не менее полезным в смысле воспитания логики и точности.

В последние годы стало модным связывать новые идеи в программировании с математическим мышлением. Например, идея структурного программирования целиком основана на математической теории, согласно которой все языковые формы могут быть представлены посредством нескольких значимых. Другой пример – идея формальных методов, основанная на том предположении, что математические представления и доказательства являются – или должны быть – существенной частью практики программирования. Здесь имеет место некая «мощь в соучастии», что-то вроде «вины в соучастии» наоборот. Мы так часто слышали, что математически обоснованные идеи в программировании неким образом превосходят те, у которых такого основания нет, что перестали критически относиться к подобным заявлениям. Я склонен рассматривать это как «синдром непорочного зачатия». Некоторые церковники когда-то приняли ту точку зрения, что рождение ребенка девственницей в определенном отношении удовлетворительнее его альтернативного, возможно, менее безупречного варианта. Разумеется, глубокое удовлетворение вызывает вера в то, что применение компьютеров, основанное на математике, в каком-то смысле чище и лучше, чем его менее безупречная альтернатива.

Но насколько важна такая безупречность? В сложном и запутанном мире реального программирования основное правило, скорее, состоит в том, что «красиво то, что полезно». Ценность идеи в ее собственных достоинствах, а не в том, имеет она математическое основание или нет. Структурное программирование полезно, потому что позволяет создавать хорошие

программные продукты, а не потому что оно основано на математике. Если формальные методы когда-нибудь станут полезными – а на сегодняшний день они себя не проявили с хорошей стороны, – то опять-таки, потому что они дадут нам хорошие программные продукты, а не потому что основаны на чистой и красивой математике. Математика не должна превращаться в подпорку идей, ценность которых не находит подтверждения.

Стоит ли уделять этому вопросу так много места? То есть в самом ли деле все эти проблемы с математикой вредят компьютерной области или к ней надо относиться, как к латыни – раз польза есть, к чему беспокоиться о деталях? Полагаю, проблема здесь имеется.

Наши собратья-математики, развивая методологию исследований, пришли к выводу, что основными инструментами исследований являются анализ и доказательство. Если теория доказана, она правильна. Математически говоря, более сильных утверждений не существует.

Компьютерные науки применяли тот же подход к исследованиям. Я бы сказал, в основном они заключаются в том, что новая идея пристально анализируется, затем делаются некоторые попытки доказать ее корректность, после чего ее начинают пропагандировать. В этой новой области знаний доказать что-то сложнее, чем в математике. На компьютерные научные идеи влияет окружающая реальность, и доказательства неотделимы от ее несовершенства. В итоге большинство доказательств в компьютерных исследованиях оказывается просто набором доводов в пользу открытий, сделанных на стадии анализа. Получается, что в компьютерных науках слабые математические исследования выдаются за добротные.

Заявляю, что математическое исследование – просто неподходящая модель для компьютерных наук. Думаю, более подходящей можно считать ту модель, которая здесь применяется. Научный метод, как известно, состоит в формулировке гипотезы, проверке этой гипотезы и сообщении о результатах проверки. Отстаивание гипотезы должно опираться на достаточно веские данные.

В компьютерных науках редко используется научный метод. Обычно в этих исследованиях есть какие-то зачатки формулировки гипотезы, очень незначительный объем проверки и чрезмерно активная пропаганда. В результате в компьютерных науках много теорий, основанных на вере в сомнительные вещи, стремящихся игнорировать реальность и уповающих на гениальность. Не имею ничего против гениальности, но весьма опасно игнорировать реальность в сфере производства продуктов, взаимодействующих с этой реальностью и, фактически, быстро изменяющих ее.

Есть определенная ирония в том, к чему мы, кажется, пришли в поисках ответа на первоначальный вопрос. Насколько важна для компьютерных наук связь с математикой? Я бы сказал, что математические корни при-

вели компьютерные науки к такому состоянию, когда исследования здесь нельзя считать высококачественными ни с точки зрения математических принципов, ни с точки зрения научных принципов. Если учесть, что, как я доказывал в начале этого раздела, компьютерные науки больше не занимаются только «вычислениями» и при этом представляют собой и плохую математику, и плохую науку, то это не науки, а одна видимость, и их основания являются достаточно бутафорскими.

Пора уже компьютерным наукам разобраться в том, что они собой представляют. Если они хотят сохранить верность своему математическому происхождению, то должны найти новые средства, подтверждающие законность притязаний на такое наследство. Если же они хотят считаться наукой, то должны и вести себя подобающим образом. А если они будут заявлять, что занимаются «вычислениями», то должны, по крайней мере, признать, что их название сохраняется по историческим причинам, а не потому, что описывает сферу их интересов.

Все же, возвращаясь к первоначальному вопросу: насколько связь с математикой важна для компьютерных наук? Боюсь, что правильный ответ – «в очень значительной мере».

4.3. Роль интуиции в принятии решений

Современное общество склонно отвергать интуицию как метод принятия решений. Из имеющихся методов принятия решений большинство из нас в первую очередь выбрало бы количественный подход, затем рациональный, а к интуиции прибегло бы лишь в последнюю очередь.

Надо признать, что мы плохо понимаем, что такое интуиция. Мы приписываем ее женщинам и в соответствии с сексистскими традициями воспринимаем с иронией. Мы рассматриваем ее как знание, источник которого неизвестен, и не вызывающее особого доверия.

Но странное дело: когда действительно приходится принимать решения в сложных областях, все другие методы часто оказываются бесполезными. Сначала отказываются действовать количественные методы. Они прекрасны и безупречны, если задачи достаточно понятны или формулируются математически, но в нашей грубой действительности часто оказываются непригодными. Упрощая модель реального мира, чтобы иметь возможность применить эти методы, мы часто получаем приличные на вид, но не заслуживающие доверия результаты. Будем ли мы называть свои количественные методы принятия решений математическими, статистическими, научным управлением или станем исходить из числовых результатов, полученных каким-то другим способом, применение данных методов редко приносит пользу.

Рациональные методы готовы служить нам гораздо дольше. Мы собираем всю относящуюся к проблеме информацию, извлекаем из нее факторы, влияющие на решение, применяем к этим факторам критерии решения и получаем готовое решение. Звучит убедительно, четко и ясно. И действительно, если все проделать достаточно тщательно, создается полное впечатление количественного подхода. Числа, полученные в результате умозаключений, могут выглядеть в точности, как числа, полученные количественными методами! (Разумеется, здесь таится опасность, поэтому к числам нужно относиться так же критически, как к другим формам ответа.) Но даже рациональные методы могут подвести – и чаще, чем нам кажется или хотелось бы того. Когда картина осложняется политическими требованиями или крайней сложностью задачи, мы часто не ограничиваемся одними лишь рациональными методами.

Рассмотрим, например, задачу оценки сроков разработки программного обеспечения. Если полистать учебники или применяемые на практике инструкции, то можно найти очень разумные предложения относительно того, как осуществлять оценку. Эта разумность может основываться на мнениях экспертов, историческом опыте или некотором алгоритме, и во всех случаях вам предложат точное, методичное и рассудительное описание того, как нужно проводить оценку проекта.

Однако в реальном мире такой рациональный подход редко оказывается жизнеспособным. Действительные оценки – те, которые докладываются руководству, предоставляются заказчикам и отражаются в графике работ, – основываются на политических соображениях. Есть переговорный процесс, в ходе которого заказчик продукта сообщает производителю работ, к какому сроку он хочет получить результат, а производитель приводит свои разумные соображения, в результате чего (в большинстве случаев) график слегка корректируется, а затем изменяется, исходя из потребности. Не из разумных соображений, основанных на объеме работы, а из потребности – независимо от необходимого для изготовления продукта времени. Этот вывод сделан в исследовании, проведенном Ледерером [Lederer 1990], подтвержден в отчете о применении CASE-средств [HCS 1990] и многократно проверен большинством практикующих программистов.

Конечно, такие политически обусловленные графики тоже можно в известном смысле считать рациональными. Скажем, отдел маркетинга может решить, что программный продукт должен быть готов к очередной крупной компьютерной выставке, вроде бывшей COMDEX, и никакие другие соображения не влияют на дату поставки.

Но на каком основании отдел маркетинга решает, что продукт должен быть готов к следующей COMDEX? В конце концов, выставки COMDEX одно время проводились довольно регулярно – в лучшие времена дважды

в год. На чем основываются маркетологи, требуя, чтобы продукт был готов к очередной COMDEX?

Думаю, на интуиции. Исходя из своего долгого опыта работы на рынке, отдел маркетинга считает, что анонсирование продукта именно на этой очередной важной компьютерной выставке дает максимальные шансы выйти на желаемый рынок. Конечно, можно привести какие-то цифры и рациональные соображения в поддержку такой позиции, но если припереть их к стене, то они скажут, что *чувствуют* необходимость выпустить продукт к следующей COMDEX.

Такая же проблема, как с оценкой сроков, на самом деле встречается везде. Большинство людей, занимавшихся административной работой в этой отрасли, сталкивалось с проблемой следующего характера.

Начальник задает вам вопрос, ответ на который требует определенной исследовательской и аналитической работы. Вы проводите соответствующие исследования и анализ и приходите к начальнику с ответом. «Этот ответ никуда не годится, – заявляет начальник. – Идите и все переделайте». Ситуация повторяется несколько раз, пока вы не начинаете понимать, что начальник заранее знает, какой ответ ему нужен, и ваша задача в том, чтобы найти не точный ответ, а обоснования, подкрепляющие точку зрения начальника. (Конечно, сам начальник вам об этом не скажет, поскольку это выглядело бы неестественно и подозрительно.) И тогда вам приходится решать моральную проблему, с которой вы столкнулись, и либо а) дать начальнику тот ответ, который, очевидно, ему нужен, приведя доводы в его поддержку, либо б) найти выход из затруднительного положения, в котором вы оказались, – например, заставить начальника принять разумно обоснованный ответ (редкая удача!), отказаться от поставленной перед вами задачи или найти себе другое место работы.

Опять-таки, откуда ваш начальник взял свой «правильный» ответ? Из своей интуиции. Он столько времени командовал в своем бизнесе, что пришел к убеждению: он сам лучше разбирается в своей отрасли, чем все эти не от мира сего технари и аналитики, оказавшиеся в его подчинении.

Здесь возникают два важных вопроса:

1. Верен ли такой подход?
2. Что в действительности представляет собой интуиция?

Большинство из нас с подозрением отнесется к методам такого начальника. Его подход не соответствует тем образцам, на которых нас учили. Он противоречит тому, что мы знаем из учебников, и тому, что нам известно о морали. Общаясь между собой, мы высмеиваем такие задания и дивимся своему непреклонному старомодному начальнику. Как ему с такими методами вообще удалось стать начальником?

Но действительность такова, что на удивление часто начальник оказывается прав. Необычные, иррациональные способы принятия им решений в том или ином отношении приводят к успеху – ровному технологическому процессу, блестящему продукту или хорошим финансовым результатам. В чем тут дело?

Ответ на *этот* вопрос будет также ответом на второй из поставленных выше вопросов. Интуиция, как мне кажется, – это способность нашего разума использовать богатые залежи накопленной опытным путем информации (наличие которых мы можем не осознавать), применяя при этом не ясные нам методы.

Обратите внимание, что такое определение ведет нас к некоторым другим, еще более интересным выводам. Например, поскольку интуиция является функцией разума, то можно считать, что она тоже представляет собой рациональный подход. Если мы не понимаем, как она действует, то это потому, что мы не знаем, как ее объяснить, а не из-за ее иррациональности. Причина уверенности начальника в своем решении, очевидно иррациональном с виду, состоит в том, что где-то в глубине оно может оказаться рациональным.

На практике подобный способ принятия решений вызывает более позитивное отношение. Мы придумываем для него разные названия, стараясь скрыть его очевидную иррациональность за социально приемлемыми терминами. Например, мы говорим об «инстинктивном» решении, когда оно принимается на основании неких глубоких внутренних убеждений вопреки всей окружающей обстановке. По всей видимости, источником таких инстинктивных решений является наша старая знакомая – интуиция. В таком случае, «интуитивный» процесс принятия решений, обычно считающийся положительным мужским свойством, мало отличается от «женской интуиции», которую мы считаем несколько подозрительной!

В действительности, недавно (в 2005 году) вышла целая книга, посвященная принятию решений «в мгновение ока», которая весьма убедительно (с помощью приведенных результатов исследований и отдельных примеров) показывает, что моментальное решение может быть таким же эффективным, как более рациональные методы [Gladwell 2005]. Таким образом, эта идея в некоторой степени становится респектабельной!

Феномен интуитивного мышления хорошо известен практикам, но менее изучен в теории. Принятие решений на количественной и рациональной основе – это предмет, которому можно обучать и который допускает проверку; интуитивное принятие решений просто не вписывается в нашу научную систему. Поэтому я очень заинтересовался, когда несколько лет назад на ежегодной международной конференции по информационным системам докладчик из академических кругов предложил признать су-

ществование по крайней мере политических (и нерациональных) принципов принятия решений путем создания систем поддержки принятия решений, которые бы не занимались измерением, анализом и рационализацией, а принимали на входе ответ и давали на выходе обоснования в пользу такого ответа! Об осуществимости такой системы можно спорить и рассуждать часами. Однако в итоге, если учесть, что может потребоваться наделить компьютер интуицией, которая даже в человеке действует непонятным для нас образом, такая задача представляется едва ли разрешимой. Но все такой же захватывающей!

Так о чем же мы здесь говорили? И какой вывод я пытаюсь сделать? Выводов несколько, вот они:

1. Принятие решений на количественной основе, если оно возможно, обычно предпочтительнее, чем его альтернативы, но оно осуществимо не так часто, как нам того хотелось бы.
2. Принятие решений на рациональной основе – второй по значимости способ, но на практике его часто замещают чем-то другим.
3. Этим «чем-то другим» является интуиция, и даже с учетом нашего непонимания этого явления данный способ принятия решений допустим и часто применяется.

Приятно думать, что когда все другие средства бессильны, мы можем положиться на свой внутренний природный механизм принятия решений, правда?

Ссылки

GLADWELL 2005 – «Blink: The Power of Thinking Without Thinking», Little, Brown, and Co., 2005; Malcolm Gladwell.

HCS 1990 – 1990 CASE/CASM Survey, HCS, Inc., Portland, OR.

LEDERER 1990 – «Information System Cost Estimating: A Management Perspective», MIS Quarterly, June 1990; Albert L. Lederer et al.

4.4. Тьма ловушек: числа бывают разные

Есть два противоположных взгляда на роль чисел в принятии решений.

Приверженцы первого считают, что «нельзя управлять тем, чего нельзя измерить», а принимать решения лучше на количественной основе, чем на какой-либо другой.

Приверженцы второго, выступающие под лозунгом «есть ложь, грязная ложь и статистика», полагают, что количественные методы принятия решений ничем не лучше остальных, а иногда и хуже.

Кто из них прав?

Как ни смешно, скорее всего, правы и те и другие. Безусловно, лучше всего принимать решения, исходя из количественных оценок на основе честно полученных и релевантных численных данных. Но столь же безусловно, что решения, принятые на основе подтасованных или нерелевантных численных данных, не только не лучше, но и, скорее всего, хуже.

Проблема в том, что на неверные числа очень часто полагаются, не ведая того. И принятие решений на количественной основе опасно тем, что неверные цифры выглядят ничуть не хуже верных. Если я вижу число 25,623, представляющее результат каких-то измерений, то полагаю, что оно достаточно точно представляет измеряемую величину (например, среднее количество строк кода в модуле). Однако если немного поразмышлять об измеряемой величине, возникает ряд вопросов:

1. Стоит ли вообще измерять данную величину? Например, есть ли реальная польза от знания «среднего количества строк кода в модуле»?
2. Насколько точно проведено измерение? Например, сколько модулей было взято для получения этого числа? Каким образом подсчитывались строки кода? Каково было качество этих модулей? Нужно ли учитывать как хорошие, так и плохие модули?
3. Насколько значимо это измерение? Например, добавляет ли ,623 в 25,623 что-нибудь для нас сверх просто 25? Или неоправданно усиливает наше доверие к этому числу?

Взглянув на число 25,623 в таком свете и в качестве примера среднего количества строк в модуле, можно ответить так:

1. Пока мы не установили, в чем цель нашего измерения, 25,623 остается просто числом.
2. Пока мы не установили, что измерения проведены правильно, число 25,623 ничем не лучше любого другого, выбранного случайно.
3. Пока мы не установили, что все цифры корректны и релевантны, число 25,623 ничем не лучше числа 25. На самом деле, в данном примере 3/5 всех цифр не нужны («,623» только добавляет путаницы к «25»).

Иными словами, числа бывают разные. Важно уметь каким-то образом различать хорошие и плохие числа.

Дело еще осложняется тем, что сторонники количественного подхода открыли лозунговую кампанию с целью убедить в своей правоте доверчивых руководителей программных проектов:

> «Измерить – значит понять». Джеймс Кларк Максвелл
>
> «Нельзя управлять тем, что нельзя измерить». Том Демарко
>
> «В невидимую цель трудно попасть». Том Гилб

Но война лозунгов – вещь обоюдоострая. Вот другие лозунги, взятые из [Leveson 1993]:

> «Данные оценки рисков можно сравнить со сведениями, полученными от пленного: если достаточно долго мучить его на допросе, он скажет все, что вы хотели бы услышать». Уильям Раккельсхауз, руководитель Национального агентства США по защите окружающей среды (два срока)
>
> Игры с числами при оценке рисков «можно проводить только закрытым образом между взрослыми лицами, достигшими взаимной договоренности, иначе они могут быть неправильно интерпретированы». И. А. Райдер, Комитет по здравоохранению и промышленной безопасности Великобритании
>
> «Наш энтузиазм в отношении измерений не должен распространяться на то, что измерений не допускает». Нэнси Ливсон и Кларк Тернер

Как это обычно бывает, приверженность к лозунгам и фанатичная пропаганда оказываются плохой базой для любых методов принятия решений, будь они количественные или нет.

Приведенные высказывания не следует рассматривать как неодобрение принятия решений на количественной основе. Сейчас появилось достаточно отличных книг по метрикам в программировании, написанных такими авторами, как Роберт Грэйди (Robert Grady), Билл Хетцел (Bill Hetzel) и Норм Фентон (Norm Fenton), и организациям, занимающимся программированием, стоит всерьез задаться вопросом: «Не следует ли нам внедрить подход на основе метрик, чтобы усилить количественный аспект нашей процедуры принятия решений?»

Но будьте осторожны. На этом пути вас поджидает множество опасностей.

Ссылки

LEVESON 1993 – «An Investigation of the Therac-25 Accidents», *IEEE Computer,* July 1993; Nancy G. Leveson and Clark S. Turner