Программирование как искусство

Когда в 1959 году началось издание журнала Communications ofthe ACM, члены Редакционного совета сделали следующее замечание: «Для того чтобы программирование стало важной частью исследования и разработки вычислительных машин, в этой области должен совершиться переход от искусства к науке» [2]. К этому замечанию не раз возвращались в последующие годы; например, в 1970 году можно было прочитать «о первых шагах превращения искусства программирования в науку» [26]. Между тем, мы действительно преуспевали в преобразовании нашей дисциплины в науку, к тому же необыкновенно простым способом.

Мы просто решили называть её вычислительной наукой, "computer science". В этих рассуждениях неявно присутствует мысль, что в той области человеческой деятельности, которая классифицируется как «искусство», содержится что-то нежелательное. Она должна стать Наукой, прежде чем приобретет какое-либо реальное значение. С другой стороны, я работаю более 12-ти лет над серией книг под названием «Искусство программирования» ("The *Art* of Computer Programming"). Люди часто спрашивают меня, почему я выбрал такое название; некоторые, очевидно, даже не верят, что я действительно назвал их так. Я видел, по крайней мере, одну библиографическую ссылку на мои книги под названием "The *Act* of Computer Programming".

Сейчас я попытаюсь объяснить, почему я считаю "Art" («Искусство») подходящим словом. Я рассмотрю вопрос о том, что означает быть искусством, а не наукой. Я попытаюсь выяснить, являются ли искусства хорошим или плохим занятием, и я попытаюсь показать, что правильная точка зрения в этом вопросе поможет всем нам повысить качество того, чем мы сейчас занимаемся.

Один из первых случаев, когда меня спросили о заглавии моих книг, произошел в 1966 году во время национальной конференции ACM, состоявшейся в Южной Калифорнии. Это было ещё до того, как была опубликована первая из моих книг. Я вспоминаю ланч со своим другом в отеле. Он знал, как я был тщеславен уже в то время, и спросил, не собираюсь ли я назвать книгу «Введением в Дона Кнута». Я ответил, что, напротив, я назову свою книгу *его* именем. Его звали Art Evans.

Из этой истории мы можем сделать вывод, что слово "art" имеет несколько значений. Действительно, одним из самых приятных свойств этого слова является то, что его можно использовать в различных значениях, каждое из которых вполне подходит к машинному программированию. Готовясь к этой лекции, я отправился в библиотеку, чтобы посмотреть, что же вообще написали люди о слове "art"; и после нескольких очаровательных дней, проведенных в книгохранилищах, я пришел к выводу, что слово "art" является одним из интереснейших слов английского языка.

Старинные искусства

Обратившись к латыни, мы находим корни "*ars*", "*artis*", обозначающие "skill" (мастерство, умение, искусство, ловкость). Существенно также, что от соответствующего греческого *τέχνη* произошли наши "technology", "technique". В наше время, когда говорят об "art", вы, вероятно, думаете прежде всего об «изящных искусствах», таких, как живопись и ваяние, но до 20-го века это слово употреблялось, главным образом, совсем в другом смысле. Поскольку это более старое значение слова "art" встречается во многих идиомах, особенно когда мы противопоставляем искусство науке, мне бы хотелось поговорить немного об искусстве в его классическом значении. В средние века университеты были созданы для того, чтобы обучать семи, так называемым, «гуманитарным наукам», а именно: грамматике, риторике, логике, арифметике, геометрии, музыке и астрономии. Заметим, что это совершенно отличается от программ современных гуманитарных колледжей, и что, по крайней мере, три из этих семи «гуманитарных наук» являются важными компонентами вычислительной науки. В то время «искусство» ("art") имело смысл чего-то придуманного человеческим разумом в противоположность деятельности, являющейся продуктом природы или инстинкта; «гуманитарные» науки были освобожденными или свободными, в отличие от ручных ремёсел как, например, пахота (ср. [6]). В средние века словом "art" называли логику [4], что обычно означало изучение силлогизмов.

Наука — Искусство

В течение длительного времени слово «наука» употреблялось почти в том же смысле, что «искусство»; например, люди говорили о семи гуманитарных науках, которые совпадали с семью гуманитарными искусствами [1]. В 13-м веке Данс Скотус (Duns Scotus) называл логику «Наукой из Наук» и «Искусством из Искусств» (см. [12], стр. 34). По мере развития цивилизации и образования эти слова все более приобретали независимые значения, причем слово «наука» относилось к знаниям, а «искусство» — к применению знаний. Так, наука астрономия была основой искусства навигации. Ситуация была почти в точности такой, как сейчас, когда мы различаем «науку» и «технику». В 19-м веке многие авторы писали о соотношении между искусством и наукой, и я считаю, что самое лучшее рассуждение принадлежит Джону Стьюарту Милю (John Stuart Mill). В 1843 году он писал [28]: «Для того, чтобы сформировать фундамент некоторого искусства, часто необходимы несколько наук. Такова сложность человеческих дел: для того, чтобы *сделать* что-то одно, часто необходимо *знать* природу и свойства многих других вещей…

Вообще говоря, Искусство состоит из истин Науки, организованных таким способом, который наиболее удобен для практики, а не для мышления. Наука группирует и организует свои истины таким образом, чтобы мы могли одним взглядом, насколько это возможно, охватить общее устройство Вселенной. Искусство …формируется из отдельных фрагментов науки, далеких друг от друга, из истин, которые относятся к созданию различных и разнородных условий, вызванных насущными потребностями практической жизни».

Когда я просматривал эти высказывания о значении слова «искусство», я обнаружил, что различные авторы, на протяжении, по меньшей мере, двух веков, пытались установить переход от искусства к науке. Например, в предисловии к учебнику по минералогии, написанному в 1784 году, говорится следующее [17]: «Минералогия, которую до 1780 многие понимали как искусство, могла бы, в сущности, быть отнесена к Науке». В соответствии с большинством словарей, «наука» означает знания, которые были логически упорядочены и систематизированы в виде общих «законов». Преимущество науки заключается в том, что она избавляет нас от необходимости обдумывать вещи всесторонне в каждом конкретном случае; мы можем обратить наши мысли к понятиям более высокого уровня. Как писал в 1853 году Джон Раскин (John Ruskin) [32]: «Работа науки состоит в том, что она заменяет явления фактами, а впечатления — доказательствами». Мне кажется, что если бы авторы, которых я изучал, писали сегодня, они согласились бы со следующим определением: «Наука — это знания, которые мы настолько хорошо понимаем, что можем обучить им компьютер; если же мы не понимаем чего-нибудь до конца, то заниматься этим — искусство». Поскольку алгоритм или компьютерная программа обеспечивают нас чрезвычайно полезным тестом для проверки глубины наших знаний о любом заданном предмете, переход от искусства к науке означает, что мы научились кое-что автоматизировать.

Искусственный интеллект сделал значительные успехи, однако остается глубокая пропасть между тем, что компьютеры смогут делать в предвидимом будущем и что могут делать обычные люди. Таинственные «инсайты», которые используют люди, когда они говорят, слушают, пишут, и даже, когда они программируют, по-прежнему недостижимы для науки; почти всё, что мы делаем, это пока — искусство. С этой точки зрения, конечно, желательно превратить компьютерное программирование в науку, и мы, действительно, прошли немалый путь за эти 15 лет, прошедших с момента публикации замечаний, которые я привел в начале этого доклада.

Пятнадцать лет назад понимание программирования было настолько примитивным, что вряд ли кто-нибудь мог даже *думать* о доказательстве корректности программы; мы просто возились с программой до тех пор, пока, наконец, «знали», что она работает. В то время мы даже не умели сформулировать, каким-либо строгим путем, само понятие корректности программы. Лишь в последние годы мы узнали о процессах абстракции, которые позволяют писать и понимать программы. Эти новые знания дают сейчас важные результаты. Хотя лишь немногие программы оказываются полностью корректными, мы начинаем понимать принципы структуры программ. Дело в том, что если мы пишем программы сегодня, то мы знаем, что могли бы, в принципе, построить формальное доказательство их корректности, поскольку сейчас мы понимаем, как формулируются такие доказательства. Существование такого научного базиса позволяет составлять программы, которые значительно более надежны, чем те, которые мы писали в прежние дни, когда интуиция была единственной гарантией корректности.

Область «автоматического программирования» является сейчас одной из основных областей исследования проблем искусственного интеллекта. Её сторонники с удовольствием прочитали бы доклад под названием «Программирование как артефакт» (имея в виду, что программирование стало просто пережитком прошлого), так как их задача состоит в том, чтобы создать машины, способные составлять программы лучше, чем это делаем мы. Лично я не думаю, что такая цель будет когда-нибудь полностью достигнута, но я думаю, что их исследования чрезвычайно важны, так как всё, что мы узнаём о программировании, помогает нам повышать наше собственное мастерство. В этом смысле мы должны постоянно стремиться превращать *каждое* искусство в науку: таким образом мы совершенствуем искусство.

Наука и искусство

Наша дискуссия показывает, что программирование является сейчас и *наукой* и *искусством*, и что обе эти стороны прекрасно дополняют друг друга. По-видимому, большинство авторов, занимающихся исследованием таких вопросов, приходят к выводу, что их предмет является и наукой и искусством, каким бы ни был сам предмет (ср. [25]). Я отыскал книгу по элементарной фотографии, написанную в 1893 году, в которой утверждается, что «проявление фотографического изображения есть и искусство, и наука» [13]. Когда я впервые взял словарь, чтобы исследовать слова “art’ и “science”, я случайно бросил взгляд на предисловие редактора, которое начиналось словами «Составление словаря является и наукой, и искусством». Редактор словаря Funk & Wagnall [27] заметил, что тщательное накопление и классификация информации о словах имеет научный характер, в то время как хорошо подобранные изложения определений требуют умения писать экономно и аккуратно: «наука без искусства, вероятно, была бы неэффективной; искусство без науки определённо не точно».

В процессе подготовки этой лекции я просмотрел картотеку Стэнфордской библиотеки с целью узнать, как разные люди употребляли слова «искусство» и «наука» в названиях своих книг. Это оказалось очень интересным. Например, я нашел две книги, озаглавленные *«Искусство игры на фортепьяно»* [5, 15] и другие под названиями *«Наука пиано-форте техники»* [10], *«Наука игры пиано-форте»* [30]. Была также книга *«Искусство игры на фортепьяно: научный подход»* [22]. Затем я нашел прелестную книжечку *«Благородное искусство математики»* [31], которая заставила меня взгрустнуть о том, что я не могу честно изложить программирование как «благородное искусство».

Я знал ещё несколько лет назад о книге *«Искусство вычислений»*, опубликованной в Сан Франциско в 1879 году неким Ховардом (C. Frusher Howard) [14]. Это было практическое пособие по деловой арифметике, которое к 1890-му году было распродано тиражом свыше 400 тысяч экземпляров в нескольких изданиях. Меня позабавило предисловие, которое указывало на то, что философия Ховарда и смысл его заглавия совершенно отличались от моих. Он писал: «Знание науки о числах не имеет большого значения; овладение же искусством счёта абсолютно необходимо». Некоторые книги содержали оба слова — «наука» и «искусство» — в своих названиях, например: *«Наука существовать и искусство жить»* Mahesh Yogi [24]. Есть также книга *«Искусство научного открытия»* [11], в которой анализируется, каким образом были сделаны некоторые из великих открытий.

Как много информации в классическомзначении слова “art”! На самом деле, при выборе заглавия для своих книг я не думал об искусстве в этом смысле, а думал больше о его других сопутствующих значениях. Вероятно, самой интересной книгой, которую я нашел, была совсем недавняя работа Мюллера (Robert E. Mueller) под названием *«Наука Искусства»* [29]. Из всех книг, упомянутых мною, эта книга ближе всего выражает то, что я хочу сделать главной темой моей сегодняшней беседы. Мюллер замечает: «Когда-то думали, что поэтическая образная точка зрения художника является губительной для ученого. А логический научный подход означает смерть для всех возможных художественных взлётов фантазии». А затем он исследует достижения, которые поистине являются результатом синтеза науки и искусства.

Научный подход характеризуется, главным образом, такими словами как логический, систематический, объективный,спокойный, рациональный, в то время как художественный —

словами: эстетический, творческий, гуманитарный, беспокойный, иррациональный. Мне кажется, что оба этих явно противоречивых подхода имеют большое значение для программирования. Эмма Лемер (Emma Lehmer) писала в 1956 году, что для неё

программирование — «изнуряющая наука и, в то же время, увлекательное искусство» [23]. Коксетер (H.S.M. Coxeter) заметил в 1957 году, что он иногда чувствует себя больше художником, чем ученым [7]. Это было в то время, когда Сноу (C.P. Snow) начал выражать тревогу по поводу растущей поляризации между «двумя культурами» образованных людей [34, 35]. Он указывал, что нам нужно объединить научные и художественные ценности, если мы хотим достигнуть настоящего прогресса.

Произведения искусства

Когда я сижу в аудитории и слушаю длинную лекцию, то моё внимание к рассматриваемому вопросу начинает ослабевать примерно через час. Я думаю, что вы тоже немного устали от моих разглагольствований о «науке» и «искусстве». Я, конечно, надеюсь, что вы сможете выслушать внимательно и остальную часть моей лекции, хотя бы потому, что сейчас речь пойдет о тех вещах, которые я чувствую наиболее глубоко. Когда я говорю о программировании как об искусстве, я думаю, в первую очередь, о нём как о художественной *форме* в эстетическом смысле.

Главная цель моей работы как педагога и автора состоит в том, чтобы научить людей составлять *красивые программы*. Вот почему я был особенно рад, узнав недавно [32], что мои книги имеются в Библиотеке Изящных Искусств при Корнельском университете. (Однако эти 3 тома, очевидно, спокойно лежат на полке без употребления, так что я боюсь, что библиотекари, может быть, совершили ошибку, поняв моё заглавие буквально). Я считаю, что составление программы похоже на сочинение стихов или музыки. Как сказал Андрей Ершов [9], программирование может давать нам и интеллектуальное и эмоциональное удовлетворение, так как овладение сложным и установление системы согласованных правил является истинным достижением.

Далее, когда мы читаем программы, составленные другими, то мы можем расценивать некоторые из них как подлинные художественные произведения. Я всё ещё помню, с каким волнением я читал в 1958 году описание ассемблера Поли (Stan Poley) SOAP II. Вы, вероятно, подумаете, что я сумасшедший, и, кроме того, с тех пор очень сильно изменились стили, но в то время для меня было огромным счастьем видеть, какой элегантной может быть системная программа, особенно, если сравнить её с другими тяжеловесными программами, которые я изучал в то же самое время. Возможность составления красивых программ, даже на языке ассемблера — это то, что заставляет меня придавать программированию первостепенное значение. Некоторые программы бывают элегантны, некоторые прелестны, а некоторые блестящи. Я утверждаю, что можно составлять *великолепные* программы, *превосходные* программы, поистине *изумительные* программы.

Вкус и стиль

Наконец-то, идея *стиля* в программировании выходит сейчас на первый план, и я надеюсь, что многие из вас видели замечательную маленькую книжку Кернигэна и Плоджер (Kernighan and Planger) *«Элементы стиля в программировании»* [16]. В этой связи, самое важное для всех нас — помнить, что не существует какого-либо «наилучшего» стиля. Каждый имеет свои собственные предпочтения, и было бы ошибкой, если бы мы пытались заставить людей придерживаться неестественного для них шаблона. Мы часто слышим: «Я ничего не знаю об искусстве, но я знаю, что мне нравится». Здесь важно, что на самом деле вам *нравится* тот стиль, который вы используете. Должно быть, это наилучший способ, который вы выбираете, чтобы выразить себя. Эджер Дейкстра подчеркнул это обстоятельство в предисловии к своей книге *«Краткое введение в искусство программирования»* [8]: «Моя цель — объяснить важность хорошего вкуса и стиля в программировании. Однако специфические элементы стиля, представленные здесь, служат только для иллюстрации в общем виде тех преимуществ, которые могут быть получены благодаря „стилю“. В этом отношении я чувствую сходство с преподавателем композиции в консерватории. Он не должен обучать своих студентов, как сочинить конкретную симфонию. Он должен помочь им найти свой собственный стиль и объяснить, к каким результатам это приводит. (Именно эта аналогия навела меня на мысль говоритьоб „Искусстве программирования“.)».

Теперь мы должны спросить себя: «Что такое хороший стиль, и что такое плохой стиль?». Мы не должны быть слишком строгими в этом отношении, оценивая работу других людей.

Джереми Бентам (Jeremy Bentham), философ начала 19-го века, говорит об этом следующее [3, Кн. 3, Гл. 1]: «Те, кто судит об элегантности и вкусе, считают самих себя благодетелями человеческой расы, в то время как они, в действительности, только мешают людям получать удовольствие...Не существует вкуса, который заслуживает эпитета *хороший,* разве только вкус к таким занятиям, которые доставляют удовольствие благодаря тому, что в них содержится какая-то доля ожидаемой в будущем пользы. Не существует вкуса, который заслуживает оценки *плохого*, если только это не вкус к некоторым занятиям, имеющим вредные свойства».

Когда мы используем наши собственные предубеждения для того, чтобы «исправить» чей-либо вкус, мы, может быть, невольно лишаем его некоторого вполне законного удовольствия. Здесь важно то, что они создают нечто такое, что *они сами* находят красивым. В отрывке, который я только что процитировал, Бентам дает нам некоторый совет относительно надежных принципов эстетики, которые лучше других, а именно — полезность результата. Мы имеем определенную свободу устанавливать наши личные стандарты красоты, но особенно приятно, если вещи, которые мы считаем красивыми, в то же время рассматриваются другими людьми как полезные. Я должен признаться, что люблю писать компьютерные программы, но я получаю особое удовольствие, когда пишу программы, которые служат, в каком-то смысле, на общее благо. 0Конечно, программа может быть «хорошей» в различном смысле. В первую очередь, очень хорошо иметь программу, которая правильно работает. Во-вторых, часто бывает полезно иметь программу, которую было бы нетрудно изменить, если наступила пора адаптировать ее. Обе эти цели достигаются, если человек, знающий соответствующий язык, легко читает и понимает эту программу.

Другой важный способ создавать хорошие программы — дать им возможность элегантно взаимодействовать с пользователями, особенно при исправлении ошибок во входных данных. Это настоящее искусство — сочинять содержательные сообщения об ошибках или разрабатывать гибкие форматы ввода, не подверженные ошибкам. Ещё один важный аспект качества программ это эффективность использования ресурсов компьютера. К сожалению, я должен сказать, что в наши дни многие программисты отрицают эффективность программ, утверждая, что это признак плохого вкуса.

Причина этого заключается в том, что мы сейчас испытываем реакцию того времени, когда эффективность была единственным общепризнанным критерием качества, и программисты были настолько озабочены эффективностью, что они производили неоправданно сложные коды. Результатом этой излишней сложности было снижение общей эффективности за счет трудностей отладки и эксплуатации. Настоящая проблема заключалась в том, что программисты тратили слишком много времени в заботах об эффективности в неподходящих местах и в неуместное время. Преждевременная оптимизация — это корень всех ошибок в программировании (или, по крайней мере, большинства). Мы не должны быть на пенни мудрыми и на фунт глупыми, мы не должны всегда думать об эффективности в том смысле, сколько процентов времени или пространства мы выиграли или потеряли. Когда мы покупаем автомобиль, то многие из нас не обращают внимания на разницу в 50 или 100 долларов в его цене. Но в то же время мы способны сделать специальный рейс к определенному магазину, чтобы купить какую-нибудь мелочь стоимостью 50 центов всего лишь за 25 центов. Я считаю, что для эффективности есть время и место. Истинную роль эффективности я рассматриваю в своей статье о структурном программировании, которая выходит в текущем номере журнала *Computing Surveys.*

Меньше возможностей — больше удовольствия

Одна довольно любопытная вещь, которую я заметил относительно эстетического удовлетворения, заключается в том, что наше удовольствие значительно увеличивается тогда, когда мы создаем что-либо при ограниченных средствах труда. Например, программа, которая мне лично доставила больше всего удовольствия и гордости, это компилятор, который я когда-то написал для примитивной минимашины, имевшей запоминающее устройство ёмкостью 4096 16-разрядных слов. Человек чувствует себя настоящим виртуозом, когда он достигает чего-то в условиях серьёзных ограничений.

Подобное явление имеет место во многих других сферах. Например, часто люди кажутся влюбленными в свои «Фольксвагены», но редко в свои Линкольны “Continental” (которые, по-видимому, гораздо лучше). Когда я изучал программирование, было популярным развлечением сделать с программой всёвозможное, чтобы уместить её на одной перфокарте.

Я полагаю, что это как раз то самое явление, которое заставляет энтузиастов APL наслаждаться своими “one-liners” («программа в одной строке»). Когда мы сегодня учим программированию, то наблюдаем любопытный факт — нам редко удается увлечь студента вычислительной наукой до тех пор, пока он не проходит курс, который позволяет поработать с минимашиной. Использование наших больших вычислительных машин с их удивительными операционными системами и языками не вызывает особой любви к программированию, по крайней мере, поначалу.

Не совсем понятно, каким образом применить этот принцип, чтобы программисты получали большее удовольствие от своей работы. Конечно, программисты застонали бы, если бы их менеджер объявил неожиданно, что у новой машины объём запоминающего устройства в два раза меньше, чем у старой. И я не думаю, что кто-нибудь, даже из самых преданных «программистов-художников» приветствовал бы такую новость, так как никому не хочется терять имеющиеся возможности. Другой пример может помочь прояснить ситуацию: кинорежиссёры очень сильно сопротивлялись введению звукового кино в 20-х годах, так как они гордились тем, что могли передавать содержание без звука. Точно так же истинный программист-художник мог бы негодовать по поводу введения более мощной техники. Так, современные запоминающие устройства имеют тенденцию разрушить красоту наших старых методов сортировки на магнитных лентах. Однако сегодняшние режиссеры не хотят возвращаться к немым фильмам. И не потому, что они ленивы, а потому, что они знают, что можно создать прекрасные фильмы, используя новую технологию. Форма искусства изменилась, но вместе с тем появилась масса новых художественных средств. Как же они развивали своё искусство? Лучшие кинорежиссеры овладевали своим искусством в сравнительно примитивных условиях, часто в странах с отсталой кинопромышленностью.

И в последние годы кажется, что большинство важных вещей, которые мы узнаем о программировании, исходят от людей, которые не имели доступа к очень мощным машинам. Мораль, как мне кажется, заключается в том, что мы должны использовать идею ограниченных ресурсов для нашего собственного усовершенствования. Мы можем добиться больших успехов путём составления различных «игровых» программ при заданных искусственных ограничениях, так, чтобы нужно было выжимать свои способности до предела. Мы не должны жить всё время в роскоши, так как это ведет к апатии. Искусство решать с максимальной энергией минипроблемы разовьёт наши способности для решения настоящих больших задач, и опыт поможет нам получать больше удовольствия от работы при ограниченных возможностях. Точно так же мы не должны отказываться от «искусства ради искусства»; мы не должны стыдиться программ, которые служат только «для развлечения». Я однажды получил огромное удовольствие от составления АЛГОЛ-программы, которая реализовала процедуру скалярного произведения таким необычным способом, что вместо скалярного произведения она вычисляла m-ое простое число [19]. Несколько лет назад студенты Стэнфорда были взволнованы составлением самой короткой FORTRAN-программы, которая печатает сама себя в том смысле, что выходная информация идентична её собственному исходному тексту.

Та же самая задача рассматривалась для многих других языков. Я не думаю, что работа над этим была пустой тратой времени. Бентам не отрицал бы «полезности» такого приятного времяпрепровождения. Он писал: «Напротив, нет ничего, полезность которого является более неоспоримой. Чему же приписать характер полезности, если не тому, что является источником удовольствия? [3, Кн. 3, Гл. 1].

Создание прекрасных инструментов

Другим свойством современного искусства является акцент на творческие способности. Мне кажется, что сейчас многие художники не заботятся о создании прекрасных произведений; только новизна идеи имеет значение. Я не рекомендую, чтобы программирование было подобно современному искусству в этом смысле, но это приводит меня к наблюдению, которое я считаю важным. Иногда мы должны программировать безнадежно скучную, бестолковую задачу, не дающую нам никакого выхода, не говоря о каком-то творчестве. И в таких случаях человек вполне может прийти ко мне и сказать: «Значит, программирование прекрасно? Вам, конечно, хорошо говорить, что я должен получать удовольствие, создавая элегантные, красивые программы, но почему Вы предполагаете, что я могу превратить этот мусор в произведение искусства?».Да, это верно, не все задачи программирования кажутся приятными.

Подумайте о хозяйке в домашней «западне», которая должна каждый день убирать один и тот же стол: не всякая ситуация способствует творчеству или артистизму. Однако, даже в таких случаях есть возможность улучшения: даже рутинная работа может доставлять удовольствие, если мы имеем дело с красивыми вещами. Например, человеку может быть приятно день за днем вытирать обеденный стол, если это красивый стол из какого-нибудь высококачественного твердого дерева.

Поэтому я хочу адресовать мои заключительные замечания системным программистам и разработчикам машин, которые создают те системы, с которыми мы должны работать. *Пожалуйста,* дайте нам инструменты, которыми было бы приятно пользоваться, вместо таких, с которыми нам приходится бороться. Пожалуйста, дайте нам инструменты, которые стимулируют нас писать лучшие программы, увеличивая удовольствие, которое мы получаем при этом. Разработчики компьютеров могут сделать использование машин гораздо более приятным, если они, например, организуют арифметику с плавающей точкой, которая удовлетворяет простым математическим правилам. Средства, имеющиеся сейчас на большинстве машин, делают задачу точного анализа погрешностей безнадежно трудной. В то же время, соответствующим образом построенные операции могли бы помочь программисту создать хорошие подпрограммы, имеющие гарантированную точность (ср. [20, с. 204]).

Что может сделать разработчик софтвера? Один из лучших способов поддержать бодрость духа пользователя — обеспечить его программами, с которыми он может взаимодействовать. Мы не должны делать системы слишком автоматизированными, когда действие всегда происходит за сценой. Мы должны пользователю-программисту дать шанс направить свои творческие способности в соответствующие каналы. Есть одна особенность, общая для всех программистов: им приятно работать с машинами; так давайте держать их в цикле. Некоторые задачи лучше выполняет машина, другие могут быть лучше решены с помощью человеческой интуиции; хорошо разработанная система должна найти разумный баланс. (Я в течение многих лет пытался устранить излишнюю автоматизацию, см. [18]).

Хорошим примером являются средства оценки программ. В течение многих лет программисты не имели представления о том, как, в действительности, стоимость вычислений распределена в их программах. Опыт показывает, что почти все программисты имеют смутное представление об узких местах в их программах; неудивительно, что попытки оценить эффективность так часто терпят неудачу: программист никогда не знает распределения стоимости по строкам написанного им кода. Его работа в чем-то напоминает молодоженов, которые пытаются построить сбалансированный бюджет, не зная цен на продукты, одежду и жильё. Всё, что мы давали программистам — это оптимизирующий компилятор, который таинственным образом что-то делает с программами, которые он транслирует, но никогда не объясняет, что он делает.

К счастью, сейчас мы, наконец, наблюдаем появление систем, которые позволяют пользователю поверить в их разумность. Они автоматически анализируют программы и выдают информацию о реальной стоимости. Эти экспериментальные системы — большой успех, поскольку они позволяют оценить усовершенствование программы, и особенно потому, что с ними приятно работать. Таким образом, я уверен, что со временем использование таких систем станет стандартной процедурой. В моей статье в Computing Surveys [21] этот вопрос рассматривается подробнее.

В этой статье высказываются также другие идеи, которые могут способствовать удовлетворению программистов. Долг разработчиков языков — создавать такие языки, которые стимулируют хороший стиль, поскольку мы знаем, что стиль в значительной степени зависит от используемого языка. Нынешний подъём интереса к структурному программированию показывает, что никакой из наших существующих языков не является идеальным для работы с программами и данными, и вообще неясно, каким должен быть идеальный язык. Поэтому я надеюсь в ближайшие годы увидеть множество серьёзных экспериментов в области разработки языков.

Заключение

Мы видели, что программирование это искусство, потому что в нем применяются накопленные в мире знания, потому что оно требует умения и изобретательности, и особенно потому, что оно создает прекрасное. Программист, который подсознательно чувствует себя художником, будетнаслаждаться тем, что он делает и будет делать

это всё лучше. Поэтому, мы можем радоваться, что люди, которые выступают на конференциях по вычислительным машинам, говорят о *State of the Art*.

Литература

1. *Bailey, Nathan.* The Universal Etymological English Dictionary. T. Cox, London, 1727. See “Art,” “Liberal,” and “Science.”
2. *Bauer, Walter F., Juncosa, Mario L., and Perlis, Alan.* J. ACM publication policies and plans. J. ACM, 6 (Apr. 1959) – PP. 121–122.
3. *Bentham, Jeremy.* The Rationale of Reward. Trans. from Theorie des peines et des recompenses, 1811, by Richard Smith, J. & H.L. Hunt, London, 1825.
4. The Century Dictionary and Cyciopedia 1. The Century Co., New York, 1889.
5. *Clementi, Muzio.* The Art of Playing the Piano. Tram. from .L'art de jouer le pianoforte by Max Vogrich. Schirmer, New York, 1898.
6. *Colvin, Sidney*. “Art.” Encyclopaedia Brilannica, eds 9, 11, 12, 13, 1875–1926.
7. *Coxeter, H. S. M.* Convocation address, Proc. 4th Canadian. Math. Congress, 1957. – PP. 8–10.
8. *Dijkstra, Edsger W.* EWD316: A Short Introduction to the Art of Programming. T. H. Eindhoven, The Netherlands, Aug. 1971.
9. *Ershov, A.P.* Aesthetics and the human factor in programming Comm. ACM (July 1972). – PP. – 501–505.
10. *Fielden, Thomas.* The Science of Pianoforte Technique. Macmillan, London, 1927.
11. *Gore, George.* The Art of Scientific Discovery. Longmans, Green, London, 1878.
12. *Hamilton, William.* Lectures on Logic 1. Wm. Blackwood, Edinburgh,1874.
13. *Hodges, John A.* Elementary Photography: The “Amateur Photographer” Library 7. London, 1893. Sixth ed, revised and enlarged, 1907. –P. 58.
14. *Howard, C. Frusher.* Howard's Art of Compulation and golden rule for equation of payments for schools, business college and self-culture …. C.F. Howard, San

Francisco, 1879.
15. *Hummel J.N.* The Art of Playing the Piano Forte. Boosey, London, 1827.
16. *Kernighan B.W., and Plauger, P.J.* The Elements of Programming Style. McGraw-Hill, New York, 1974.
17. *Kirwan, Richard.* Elements of Mineralogy. Elmsly, London, 1784.
18. *Knuth, Donald E.* Minimizing drum latency time. J. ACM 8 (Apr. 1961). – PP. 119–150.
19. *Knuth, Donald E., and Merner, J.N.* ALGOL 60 confidential. Comm. ACM 4 (June 1961). – PP. 268–272.
20. *Knuth, Donald E.* Seminumerical Algorithms: The Art of Computer Programming 2. Addison-Wesley, Reading, Mass., 1969.
21. *Knuth, Donald E.* Structured programming with go to statements. Computing Surveys 6 (Dec. 1974), pages in makeup.
22. *Kochevitsky, George.* The Art of Piano Playing: A Scientific Approach. Summy-Birchard, Evanston, Ill., 1967.
23. *Lehmer, Emma,* Number theory on the SWAC. Proc. Symp. Applied Math. 6, Amer. Math. Soc. (1956). – PP. 103–108.
24. *Mahesh Yogi, Maharishi.* The Science of Being and Art of Living. Allen & Unwin, London, 1963.
25. *Malevinsky, Moses L.* The Science of Playwriting. Brentano, New York, 1925.
26. *Manna, Zohar, and Pnueli, Amir.* Formalization of properties of functional programs. J. ACM 17 (July 1970). – PP. 555–569.
27. *Marckwardt, Albert H.* Preface to Funk and Wagnall's Standard College Dictionary. Harcourt, Brace & World, New York, 1963, vii.
28. *Mill, John Stuart.* A System of Logic, Ratiocinative and Inductive. London, 1843. The quotations are from the introduction, §2, and from Book 6, Chap. 11 (12 in later editions), §5.
29. *Mueller, Robert E.* The Science of Art. John Day, New York, 1967.
30. *Parsons, Albert Ross.* The Science of Pianoforte Practice. Schirmer, New York, 1886.
31. *Pedoe, Daniel.* The Gentle Art of Mathematics. English Univ. Press, London, 1953.
32. *Ruskin, John.* The Stones of Venice 3. London, 1853.
33. *Salton, G.A.* Personal communication, June 21, 1974.
34. *Snow, C.P.* The two cultures. The New Statesman and Nation, 52 (Oct. 6, 1956). – PP. 413–414.
35. *Snow, C.P.* The Two Cultures: and a Second Look. Cambridge University Press, 1964.