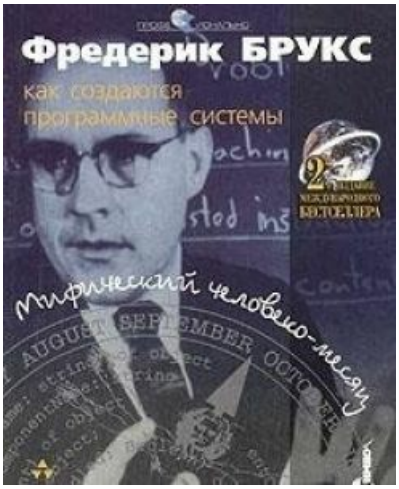


# Книга Мифический человеко-месяц

## Как создаются программные системы

Фредерик Брукс
Символ-плюс, 2010
Год первого издания:1975
Также есть на следующих языках: Английский

---

## Рецензия BooksInShort

Эта книга заслуженно считается классическим произведением в своей области. Все работы Фредерика Брукса о создании компьютерных программ представляют огромную ценность для людей, интересующихся историей и современным состоянием программирования. *BooksInShort* рекомендует книгу этого поразительно эрудированного автора не только программистам, но и всем тем, кто занимается планированием и организацией крупных проектов. Рассказывая о программировании, Брукс одновременно успевает говорить и о многом другом – о жизненных трудностях и умении их преодолевать, об искусстве планировать и организовывать работу. Своим выходом в свет настоящее издание отмечает двадцатилетний юбилей книги Брукса и содержит новые эссе этого автора, в которых он дает оценку идеям и принципам, изложенным в его ранних работах. Этот своеобразный ответ критикам – поистине уникальное и предельно честное размышление незаурядного профессионала.

## Основные идеи

- Концептуальная целостность – самое важное требование к проектированию систем программного обеспечения.
- Чтобы добиться этой целостности, комплексными проектами должен руководить один человек или группа единомышленников.
- Два обязательных условия успеха крупных проектов по созданию ПО – четкая организация работы и наличие полной стандартизированной документации.
- Добавление в проект новых сотрудников только замедляет работу.
- Программисты подобны архитекторам: первая версия их творения всегда проста, вторая – слишком сложна, а третья – лучше, чем вторая.
- Как правило, сроки работ срываются не из-за серьезных проблем, а из-за небольшого отставания от графика, которое накапливается ежедневно.
- Планируйте создание нескольких версий продукта.
- Компьютерные программы никогда не будут эволюционировать с той же скоростью, с какой эволюционируют сами компьютеры.
- Смените метафоры, которыми вы описываете программирование: программы не создаются, а нарабатываются.
- Используйте хорошие инструменты и воспитывайте искусных программистов.

## Краткое содержание

### Радости и сложности программирования

Почему людям нравится создавать компьютерные программы? Потому что они любят заниматься полезными делами, им нравится решать сложные задачи, приобретать опыт и работать с податливым материалом. Программисты, как поэты, работают с чистой мыслью. В отличие от поэзии, однако, программирование ведет к реальным изменениям в физическом мире. Программист подобен волшебнику, ошибки в работе которого недопустимы. Впрочем, какой бы безукоризненной ни была компьютерная программа, в ней всегда хватает сбоев. Создавая программу, программист знает, что она очень скоро устареет с появлением новых технологий и новых программ.

> “Стоимость действительно меняется в зависимости от количества людей и месяцев работы, но результативность – нет. Оценка объема работы в человеко-месяцах – опасное и пагубное заблуждение”.

Программирование крупных систем связано с решением уникальных по сложности задач. Большинство людей имеет неверное представление о том, как оценивается объем какой-нибудь работы. Скажем, использование такой единицы, как “человеко-месяц” (число людей, необходимое для решения задачи в течение заданного количества месяцев), содержит опасное в своей ошибочности допущение. А именно предполагается, что менеджеры могут произвольно делить заданный объем работы между исполнителями, причем любой из них может делать любую ее часть и нет никакой необходимости в коллективном взаимодействии. Возможно, эта тактика применима в таком виде деятельности, как сбор урожая, но только не для программирования или любого другого вида интеллектуального труда. Есть задачи, которые невозможно разделить между исполнителями, а есть и такие, которые дроблению поддаются, однако при этом требуется сложная организация процесса взаимодействия исполнителей. Эти люди должны узнать, как именно выполнять работу, а результаты этой работы нужно суметь свести воедино. Однако привлечение к проекту дополнительной рабочей силы всегда ведет к дополнительным сложностям и затратам времени. Чем больше людей участвует в проекте, тем больше (а не меньше) времени требуется на его выполнение.

> “Вавилонская башня стала, видимо, первым (но отнюдь не последним) архитектурным фиаско. Взаимопонимание и порядок, который держится на этом взаимопонимании, обязательны для успеха”.

Следующая сложность связана с масштабами задачи: необходимо найти такой способ организации работы, при котором персонал был бы задействован наиболее эффективно. Один из таких способов предполагает назначение каждому работнику конкретной задачи в помощь “главному программисту”, подобно тому как в госпитале бригада медперсонала ассистирует оперирующему хирургу. При составлении рабочей группы выбирайте людей под конкретные задачи и организуйте их работу так, чтобы вам всегда было ясно, чем они занимаются. Кроме “главного программиста” это может быть менее опытный “второй пилот”, который выполняет функцию его заместителя. “Администратор группы” отвечает за организацию работ и общий порядок. “Редактор” создает и редактирует итоговый документ, над которым работает команда. “Секретари” и “клерки” подчинены “редактору” и вместе с ним тесно сотрудничают с “языковедом”, который следит за соблюдением правил и особенностей используемого языка программирования. “Инструментальщик” создает нужные инструменты (например, специализированные утилиты); “отладчик” занимается тестированием программы.

> “Концептуальная целостность является наиболее важным требованием при проектировании систем”.

При взгляде на многие соборы в Европе часто бывают видны несоответствия между отдельными архитектурными элементами. Они возникли из-за того, что эти сооружения возводили разные поколения зодчих, которые по-разному понимали свою задачу. Тем не менее собор в Реймсе обладает удивительной архитектурной гармонией. Совершенная компьютерная программа требует такой же концептуальной целостности – пожалуй, это важнейшее требование к разработке любой системы. Для достижения концептуальной целостности за проект должен отвечать один человек или небольшая группа программистов-единомышленников – своего рода проектная “аристократия”. Разумеется, это правило не абсолютно: творческая идея может потребовать для своего воплощения широкого круга специалистов. В целом построение компьютерных систем состоит из трех этапов: создание архитектуры, разработка и реализация. В большинстве областей искусства эти три этапа идут один за другим, а в программировании их возможно осуществлять параллельно.

## Организационные проблемы

Работа архитектора строится по определенным правилам, идет ли речь о создании мраморной колоннады или о разработке компьютерной системы. Первая версия любой программы, как правило, скромна и отличается простотой. При ограниченном бюджете и отсутствии ясного представления о том, как решить поставленные задачи, архитекторы действуют осторожно, поэтому на начальном этапе все хорошие идеи сберегаются про запас. Вторая версия программы, как правило, получается громоздкой и трудоемкой. Лишь к третьей версии программисты составляют достаточное представление о том, что работает, а что нет, и действуют с учетом этих знаний.

> “Первый проект архитектора, как правило, получается скромным и простым”.

Важнейшую роль в разработке компьютерных программ играют документация и наличие каналов взаимодействия. Прежде всего разработчикам требуется руководство по пользовательскому интерфейсу, в котором описывается все, что должен видеть пользователь на экране монитора. Стиль описания должен быть ясным, а определения – предельно четкими, исключающими любую двусмысленность. Для обсуждения проекта рабочая группа проводит совещания двух типов. Во-первых, это еженедельные встречи для разбора текущих дел. На такие совещания лучше всего собирать непосредственных участников проекта без консультантов и наблюдателей, раздав им до начала встречи материалы с информацией о плановых изменениях. Второй тип совещаний – ежегодное расширенное собрание, посвященное обсуждению накопившихся второстепенных вопросов и разногласий. Не забывайте фиксировать все принятые решения, а по окончании оповестите участников, где можно ознакомиться с обновленной версией проектной документации. Ведите журнал телефонных звонков и оперативно передавайте группе всю важную информацию, полученную от главных участников проекта.

## Фактор взаимодействия

Коммуникация участников – важнейший элемент комплексных программных разработок. Проект подобен Вавилонской башне: чем лучше люди понимают друг друга, тем выше можно строить здание, но как только связь ухудшается, вся постройка рушится.

> “За мастерством стоит изобретательность, благодаря которой на свет появляются экономичные и быстро работающие программы”.

Неопытному менеджеру подготовительная бумажная работа чаще всего кажется пустой тратой времени и помехой на пути к настоящему делу. Понимание важности документации приходит с опытом. Благодаря тщательному документированию всего происходящего иногда удается

обнаружить ту или иную непоследовательность в ходе проекта. Письменная фиксация мыслей помогает превратить интуитивные ощущения в четкие формулировки. При ведении записей вы, вероятно, обнаружите, что ряд элементов процесса разработки универсален и относится к любым проектам вообще, занимаетесь ли вы созданием ПО или организуете университетскую кафедру.

> “Общая тенденция заключается в перегруженности второго варианта системы идеями и разного рода излишествами, благоразумно отложенными в сторону при работе над первым вариантом”.

Менеджеру проекта прежде всего понадобится рабочая тетрадь для фиксации основных элементов проекта – целей, стандартов, распоряжений и тому подобного. Эта рабочая тетрадь может выступать одновременно и справочным пособием для участников проекта, и сводом технических характеристик. Любому проекту требуется график и бюджет. Кроме того, ему необходима организационная схема, описывающая пути взаимодействия персонала, а также схема распределения рабочего пространства, чтобы люди знали, где будут работать и какие места им отведены. Начиная создание новой программы, сделайте прогноз затрат, оцените сроки работ и определитесь с предварительной ценой для конкретного рынка. Основные факторы формирования цены – это продолжительность эксплуатации программы и ее размер. Представление данных в виде таблиц и схем облегчит принятие решений в ситуациях, требующих компромисса.

> “Чтобы управлять крупным проектом в условиях острой нехватки времени, прежде всего необходимо составить график работ”.

Иногда проекты не укладываются в срок или поставленные сроки срываются. Чаще всего это происходит не из-за крупных ошибок, а из-за незаметного накопления мелких недочетов. Чтобы избежать отставаний от графика и скопления невыполненной работы, нужно четко обозначить этапы работы над проектом. Чем яснее и конкретнее они очерчены, тем легче контролировать ход проекта. Проанализируйте эти этапы с помощью “Техники оценки и анализа программ” (PERT) или метода критического пути, чтобы получить представление о том, какие шаги или стадии могут вызвать задержку. Задача руководителя – поощрять информационную открытость своих подчиненных и учить их решать проблемы своими силами. Регулярно проводите совещания и обновляйте график работ с учетом текущих данных. При формировании команды в первую очередь ищите тех, кто готов к тому, что в спорте называется “жесткой игрой”. Небольшие задержки графика сполна могут компенсироваться готовностью команды работать в более плотном режиме, когда этого требует ситуация.

## Планирование изменений и преобразований

Первая версия любого продукта, как правило, получается громоздкой, малоэффективной и практически непригодной к эксплуатации, поэтому лучше заранее рассчитывать на то, что первая версия программы или ее модели пойдет на выброс. Создание программы с последующей ее браковкой нужно учитывать в масштабе организации в целом. Вы не просто видоизменяете программу или создаете новые версии – вся система должна быть построена таким образом, чтобы ее можно было легко адаптировать и видоизменять, и этому требованию следует подчинить структуру всей работы.

> “Создать систему, которую можно изменять со временем, гораздо легче, чем создать организацию, структура которой допускает изменения”.

Менеджер проекта должен думать о том, как загрузить работой программистов, как удовлетворить потребность сотрудников в профессиональном росте и повышении квалификации, а также о том, как обеспечить признание их профессиональных заслуг. В частности, чтобы решить эти вопросы, в Bell Labs полностью отказались от названий рабочих должностей. В IBM принята “двойная” служебная лестница: сотрудники сами выбирают, строить им карьеру в технической или управленческой области. Структура организации должна легко адаптироваться к выполнению любых задач на любых этапах – от создания продукта до его поддержки. В среднем стоимость поддержки какой-нибудь популярной программы на 40 процентов превышает стоимость ее создания.

> “Строгий контроль при тестировании – отличный метод отладки, с успехом применяемый в программировании”.

Необходимо запланировать цикл исправления ошибок и сбоев в программе, которые будут неизбежно возникать. Как правило, в новых программах выявляется большое число ошибок, которое затем сокращается, а потом снова увеличивается по причине появления новых сбоев на более высоких уровнях. Количество сбоев программы можно сокращать разными способами. Стремитесь к концептуальной целостности уже на уровне разработки. Создание программы аналогично возведению здания – оценивайте любые изменения с точки зрения функций, которые они затрагивают. Выстраивайте программу сверху вниз, начиная с самого “тонкого” ее уровня. Прежде чем писать очередной блок программного кода, проверяйте технические характеристики, а затем тестируйте уровни программы – тоже сверху вниз. Программирование и отладку следует осуществлять на языке высокого уровня, который позволяет устранять проблемы в интерактивном режиме и реальном времени. На отладку нужно отвести достаточный срок; в большинстве случаев на это требуется столько же времени, сколько на написание самой программы. Если это возможно, добавляйте в систему уже отлаженные компоненты один за другим. Для целей тестирования встройте в нее “фиктивный компонент” и “мини-файл”.

> “Самые пагубные и неуловимые системные ошибки возникают из-за несовпадения допущений, сделанных авторами различных компонентов”.

Любая программа требует ясной и исчерпывающе полной документации. Начните с описания того, для чего предназначена программа и каковы принципы ее работы. Перечислите аппаратные требования, укажите, что должен выбрать пользователь при ее установке и как правильно вводить данные. Сделайте краткий обзор работы программы. Также в документации можно привести схематическую “структуру решений” программы. При описании элементов программы используйте общепринятую терминологию.

## Трудности программирования: универсального решения нет

Люди уже привыкли к тому, что компьютеры работают все быстрее и быстрее. У некоторых в этой связи даже возникает надежда на появление своеобразной “серебряной пули”, которая наповал уложит монстра, создающего трудности для создания компьютерных программ. Одни усматривают это волшебное средство в искусственном интеллекте, другие – в автоматическом или визуальном программировании, третьи – в новейших экспертных системах. Однако сама суть программирования исключает возможность решать проблемы как по волшебству.

> “Программист, ломающий голову над проблемой нехватки памяти, поступит лучше всего, если оставит в покое свой код, вернется назад и внимательно посмотрит на данные. Наличие хорошего представления о данных – суть программирования”.

В прошлом прогресс в области программирования происходил благодаря созданию языков высокого уровня, появлению методики разделения времени и объединенной среды программирования. Сегодня стала актуальной возможность покупать ПО вместо самостоятельного его создания, а также повторно использовать компоненты уже созданных программ. В будущем производители станут еще внимательнее относиться к запросам клиентов и предлагать “быстрое макетирование”. Хотя “объектное” программирование пока не получило такого развития, как ожидалось, его потенциал далеко не исчерпан.

> “Закон Брукса: если проект не укладывается в сроки, то добавление новой рабочей силы замедлит его еще сильнее”.

Руководителю проекта следует использовать два подхода, учитывающих творческий характер программирования и влияние социальных факторов. Прежде всего, начните использовать новые метафоры. Когда-то программисты “писали” программы. Затем появилась концепция “строительства” программ и масса сопутствующих “строительных” метафор – например, “концептуальные леса” или “сборка компонентов”. Однако очень скоро и эти метафоры себя изживут. Приходит время говорить о “наращивании” программ, то есть пошаговом их развитии. Разработку программного обеспечения нужно понимать как искусство, плод творчества незаурядных мастеров, которых и следует воспитывать компаниям, занятым разработкой ПО. Важно как можно раньше найти талантливых людей и помочь им приобрести ценный опыт. Ваша задача как менеджера – организовать их взаимодействие с другими “растущими” программистами и подобрать для них опытных инструкторов, которые помогут им в профессиональном развитии.

## Об авторе

**Фредерик Брукс** – автор книги “Дизайн дизайна” и ряда других. В 1999 году стал лауреатом премии Тьюринга, вручаемой “Ассоциацией вычислительной техники” (США).