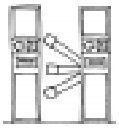
Школа современного программирования

Баранов Сергей Николаевич

ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММ КАК ОСНОВА ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ ПРОГРАММИСТОВ



1. ВВЕДЕНИЕ

Современное программирование уже давно стало инженерной дисциплиной со своим научно-техническим аппаратом и методологией. Однако ее отличительной особенностью является чрезвычайно быстрая, по сравнению с другими инженерными дисциплинами, сменяемость используемых методов и средств. Давно ли возник язык «Ява» (Java), а уже сейчас, спустя всего 5 лет после появления первых массовых публикаций по нему, он стал практически обязательным инструментом для всякого желающего вести профессиональные разработки в области современных информационных технологий!

Как и во всякой инженерной дисциплине, в программировании можно выделить сравнительно медленно меняющийся базис из фундаментальных понятий и быстро изменяющуюся надстройку из конкретных средств и методов их применения. Процесс разработки программных изделий является одним из таких фундаментальных понятий, которому до самого недавнего времени не уделялось должного внимания в сложившейся практике. Цель данной статьи – познакомить читателей с этим понятием и побудить их к дальнейшему освоению этого обязательного элемента современной программистской культуры.

Следует сразу оговориться, что речь идет о разработке программного продукта, которая всегда связана со следующими тремя ограничениями: ресурсами различного типа, временем разработки и ожидаемым качеством изделия (рисунок 1).

Время выделяется отдельно, потому что, в отличие от других ресурсов, это невозобновляемый ресурс! Если работу нужно было закончить к определенному сроку, то вернуть упущенное время уже невозможно.

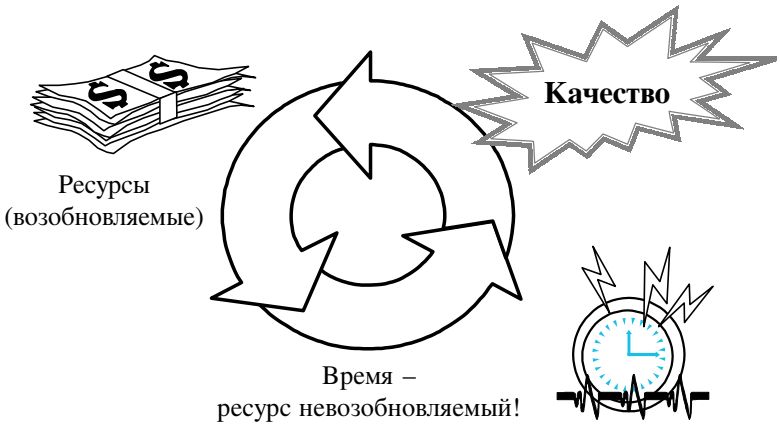
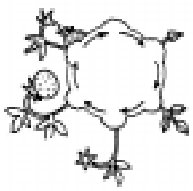


Рисунок 1



2. ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ

Процесс разработки программных изделий строится на понятии жизненного цикла, состоящего из нескольких фаз, как правило, плавно переходящих одна в другую. Приступая к новой разработке, программисты-профессионалы всегда знают, по какой модели жизненного цикла они собираются работать. На практике применяются свыше десятка различных моделей жизненного цикла, которые могут комбинироваться между собой в рамках одной разработки. Правильный выбор модели – одно из существенных условий успеха.

Отвлекаясь от особенностей конкретных моделей, можно сказать, что в той или иной степени они содержат следующие основные фазы: планирование, сбор требований, проектирование, кодирование и отладку, системное тестирование, сопровождение. В этом процессе собственно программирование (составление правильно работающей программы) занимает примерно 15 % усилий, в то время как остальные 85 % тратятся на иные виды деятельности, столь же необходимые для успешного завершения проекта.

Рассмотрим для примера классическую V-образную модель жизненного цикла, состоящую из девяти фаз и названную по форме латинской буквы V, которую образуют составляющие этапы жизненного цикла (рисунок 2).

Сначала идет разработка спецификаций и планирование всей дальнейшей работы. На следующей фазе предложенные требования анализируются и по-разному изучаются и проверяются. Если и эта фаза пройдена успешно, то приступают к высокоуровневому проектированию системы на основе утвержденных требований и спецификаций, которые к этому моменту уже сформулированы и проверены. После успешного завершения этой фазы переходят к детальному или низкоуровневому проектированию, а затем и к кодированию. Это как бы спуск вниз по букве V.

Подъем начинается после кодирования с фазы модульного тестирования. И вот тут, если что-то оказывается не в порядке, то процесс возвращается к детальному проектированию, расположенному на том же уровне буквы V. То есть, высокоуровневое проектирование – это проектирование всей системы и ее компонентов в целом, в то время как детальное проектирование – это проектирование конкретных модулей. Модульное тестирование выявляет недостатки детального проектирования, поэтому приходится возвращаться и переделывать как эту фазу, так и все последующие.

Если же на фазе модульного тестирования все завершилось отлично, то происходит подъем на следующую фазу – на сборку и тестирование. Из готовых и теперь уже отлаженных модулей, которые удовлетворяют всем своим спецификациям, разработчики собирают систему и тестируют интерфейсы между модулями. Если выясняется, что что-нибудь не так, то происходит откат на параллельную фазу высокоуровневого проектирования. Разработчики что-то меняют в проекте, повторяют детальное проектирование, кодирова-

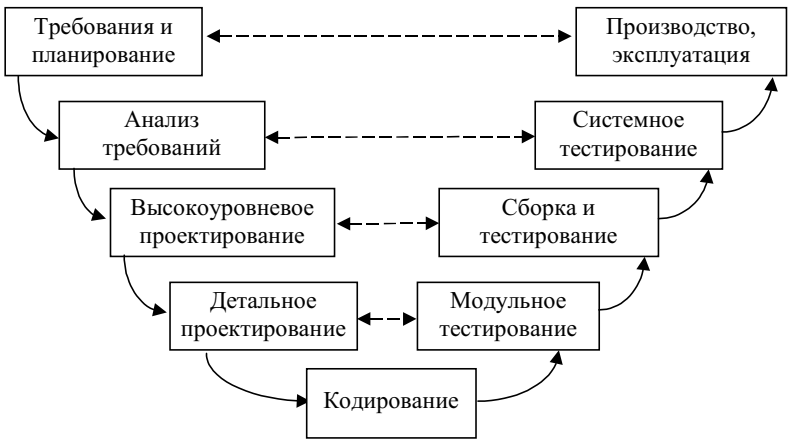


Рисунок 2

ние, модульное тестирование и вновь приступают к сборке и тестированию.

Дальше идет системное тестирование. Если проверены функции модулей (модульное тестирование) и сопряжения между модулями (сборка и тестирование), то системное тестирование проверяет всю собранную систему в целом, как она удовлетворяет исходным спецификациям. Параллельная деятельность на спуске, на которую может произойти откат, – это анализ требований и спецификаций, как они были заданы заказчиком и согласованы с ним. Если, наконец, и в системном тестировании все благополучно, то разработка переходит на последнюю фазу – запуск в производство и эксплуатацию данной системы, в результате чего могут всплыть новые требования к системе. Тогда вся разработка может быть повторена заново.

Уже из приведенного краткого описания данной модели видно, что из-за возможных откатов на предыдущие фазы она может потребовать много времени, поэтому одной из важных задач разработчиков в данном случае является снижение до минимума рисков, связанных с возможностью таких откатов.

3. МОДЕЛЬ ЗРЕЛОСТИ СПОСОБНОСТЕЙ CMM

В 1984 г. в США был создан Институт технологии программирования с задачей выяснить, почему программные проекты, которые делаются по заказам, либо не выполняются вообще, либо выполняются с огромным отставанием по срокам, либо требуют бюджетов, существенно превышающих первоначальные. Конечная цель этой правительственной инициативы – выработать рекомендации для того, чтобы разработки программных продуктов заканчивались в срок, с требуемым качеством и заданными ресурсами на их разработку. Как добиться улучшения качества систем, построенных исключительно на аппаратуре, считалось к тому времени известным. Примеры дают промышленность Японии, Кореи, Западной Европы. Но когда в системах появился относительно новый компонент – программное обеспечение, то качество таких систем стало резко снижаться; участились случаи, когда стало невозможно закончить разработку в срок и с заданными ресурсами.

Предложенная институтом в 1986 г. пятиуровневая модель называется по-английски Capability Maturity Model (модель зрелости способностей) или сокращенно CMM и нацелена прежде всего на создание программного обеспечения. Она не связана с материальным производством или с другими видами деятельности, хотя заложенные в ней общие принципы управления практически одинаковы для самых разных видов человеческой деятельности, начиная от создания программного обеспечения и кончая ремонтом квартиры. С точки зрения управления, здесь нет принципиальной разницы, поэтому управленческие методы, развитые для одного вида деятельности, годятся и для многих других.

Важно, что модель не говорит, как выполнять то или иное – она говорит только, что надо делать. Как это делать, остается на усмотрение конкретного процесса – реализации модели CMM. Модель ничего не говорит, с помощью каких инструментов надо выполнять разработку программного продукта – опять все зависит от конкретной задачи, конкретного коллектива, от самой разработки. Модель только требует, чтобы выбор инструментария был сделан заранее и был бы соответствующим образом обоснован и документирован. Уже существует большое количество стандартов (IEEE, ISO) в области технологии программирования. Модель сделана как дополнение к этим стандартам и во многом с ними согласована.

Модель CMM рассматривает процесс производства программного обеспечения и дает некоторую схему для его улучшения с тем, чтобы можно было, после того как принято решение работать по этой модели, сказать: «Был такой процесс производства до принятия модели, и вот теперь, спустя некоторое время виден уже

другой процесс. Вот так-то эти процессы отличаются друг от друга, и вот такое новое качество обеспечивает новый процесс». То есть, модель СММ выявляет, что надо улучшать в процессе производства, но не говорит, как надо это делать. Модель определяет базовые технологические действия, те умения, которые надо иметь, для того чтобы процесс производства программного обеспечения стал управляемым, предсказуемым и постоянно совершенствуемым.

Управляемость и предсказуемость процесса нужны для того, чтобы заранее, до начала работы, а также в любой момент времени в процессе разработки, все ее участники знали, чем эта работа закончится. Постоянная совершенствуемость процесса особенно важна в связи с рыночной экономикой. Если разработчики, находясь на достигнутом уровне, не будут постоянно работать в направлении улучшения своего процесса, то очень скоро конкуренты обгонят их, потому что они как раз думают об этом и работают над своим процессом.

Модель – это набор понятий и отношений между ними, в терминах которых описывается то или иное конкретное явление. В нашем случае явлением будет процесс разработки программного обеспечения. Пятиуровневая модель СММ отвечает на такой простой вопрос: в каких терминах описываются разработки программного обеспечения, и какие существуют между этими терминами соотношения. Разумеется, можно предложить и другие модели, однако, именно модель СММ на сегодняшний день прошла практическую проверку во множестве организаций. Она построена на проверенных принципах. Другие модели, скорее всего, этого качества еще лишены: они не построены на хорошо проверенных принципах и не прошли обкатку в организациях. Но это совершенно не значит, что других моделей изобретать не надо. Может быть, кто-нибудь придумает новую модель, еще более точную, еще более эффективную.

В модели СММ ведено четыре существенных понятия: уровни зрелости, ключевые области процесса, общие черты и ключевые приемы (рисунок 3).

С каждым из этих базовых понятий связано еще по одному, например, уровень зрелости характеризует способности организации-разработчика, то есть, что она может сделать. Когда говорят, что организация-разработчик программного обеспечения находится на таком-то уровне зрелости, одном из пяти в данной пятиуровневой модели, то заказчику, например, сразу понятно, чего можно ожидать от этой организации, то есть, уровень зрелости характеризует способности по разработке программного обеспечения в срок, с заданным качеством и при заданных ресурсах.

Когда рассматриваются ключевые области процесса, то для каждой из них существует конкретная цель или цели, так

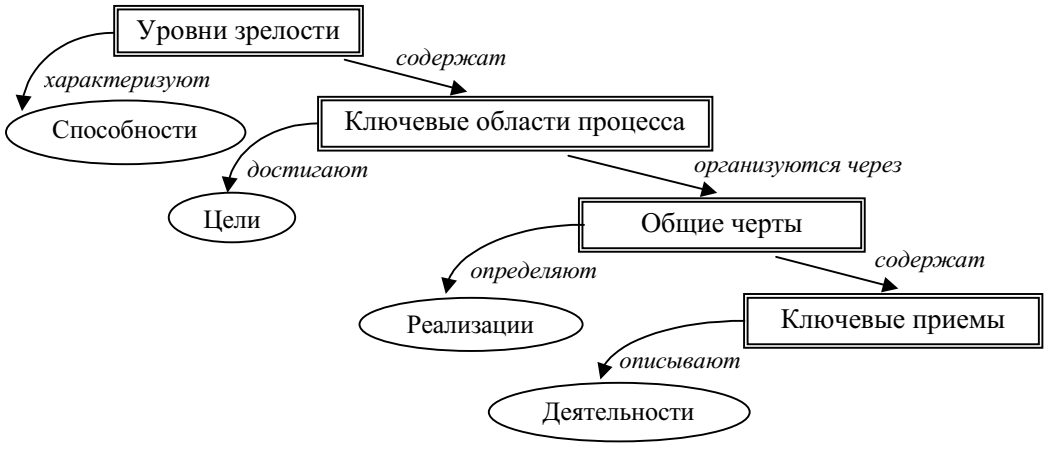


Рисунок 3

что реализация этих ключевых областей должна указанные цели достигать. Общие черты определяют те реализации ключевых областей процесса, которые могут быть выбраны. Совершенно необязательно, чтобы у всех были одинаковые реализации; более того, модель CMM такова, что не может быть одинаковой реализации у всех разработчиков. Совершенно неправильно брать процесс из одной организации и механически переносить его в другую. Из этого ничего не получится. Все, что можно сделать, – познакомиться с процессом в каких-либо организациях, проанализировать его и построить для себя свой собственный процесс, где-то похожий, где-то отличающийся от того, что есть у других, но наиболее подходящий для данной организации. И, наконец, ключевые приемы описывают разные деятельности, которые должны выполняться в рамках той или иной ключевой области (рисунок 4).

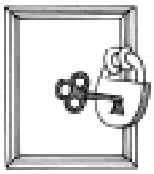
Зрелость процесса – это степень, в которой данный процесс определен, управляется, измеряется, контролируется и осуществляется. Ключевым словом является «измерение процесса». Нельзя ни о чем говорить, ни о каких свойствах, ни о каких качествах в сравнительной степени, если мы не умеем эти свойства или эти качества измерять в том или ином виде. Когда говорят, что такой-то программный продукт лучше как-то другого программного продукта, то это несерьезный разговор, если нет конкретных данных, по которым мы сравниваем эти продукты с соответствующими числовыми характеристиками. Пятиуровневая модель CMM построена так, что происходит постоянное измерение построенного в соответствии с ней процесса. Он описывается рядом характеристик, допускающих числовое выражение, и через них можно видеть, как процесс меняется и совершенствуется.

Рассмотрим структуру модели CMM. Она определяет пять уровней зрелости организации-разработчика, показанных прямоугольниками снизу-вверх. Первый уровень называется начальным. По определению, на нем находится любой разработчик программного продукта. Началь-

Рисунок 4

ный уровень не требует ничего. Второй уровень называется повторяемым. Дальше в каждом квадратике перечислены все ключевые области процесса данного уровня. Модель устроена так, что ключевые области каждого следующего уровня включают все ключевые области предыдущего уровня. То есть третий уровень включает перечисленные шесть и прибавляет к ним еще семь. Потом добавляются еще две и потом еще три на самой высокой пятой стадии.

Повторяемый процесс отличается от начального своим главным свойством, почему ему и дано название «повторяемый»: если мы закончили некоторую работу, то мы легко можем повторить эту же или близкую к ней работу с примерно таким же результатом и примерно такими же затратами. Начальный же процесс характеризуется тем, что про него просто ничего не известно. Если на начальном уровне сделана хорошая работа (а такое бывает), то это совершенно не значит, что следующая работа будет сделана так же хорошо. В этом принципиальное отличие начального уровня от повторяемого. Если про первый уровень можно сказать что-либо только после того, как разработка закончена, деньги и время потрачены: «Да, вот получилось, – или, – нет, извините, не получилось, давайте еще время и деньги, тогда, может быть, получится» – то второй уровень не такой. Второй уровень говорит заранее, до начала: «Работа примерно такая же, как была раньше, значит, будет стоить столько-то и займет столько-то времени». И действительно, по истечении этого времени, если все финансирование проделано так, как договаривались, то результат будет такой же. Поэтому, с точки зрения заказчика, второй уровень, конечно, предпочтительнее. По крайней мере, до начала работ можно сравнить несколько разработчиков, находящихся на втором уровне, и выбрать того, кто окажется предпочтительнее. При этом каждый из них гарантирует выполнение работ в указанные сроки за вполне определенные затраты.



4. КЛЮЧЕВЫЕ ОБЛАСТИ ПРОЦЕССА

Управление требованиями. Здесь под словом «требование» понимаются два понятия: требования и спецификации. Требованиями называются часто расплывчатые пожелания заказчика, что должен делать конечный продукт, в то время как спецификация – это производное от требования, но оно уже более четко и точно формулирует конкретное свойство продукта. Здесь целый куст деятельностей, связанных с переходом от требований к конкретным и точным спецификациям на данный программный продукт.

Планирование проекта – тоже целый куст деятельностей, связанных с разработкой планов. Пусть есть заказчик, который выбирает себе исполнителя и некоторая организация-разработчик, которая говорит, что она на втором уровне и сделает данную работу за столько-то. Тогда заказчик спрашивает: «Хорошо, покажите, как у вас обстоит дело с планированием и что у вас с управлением требованиями, как вы его делаете?» Если всего этого нет, то какой же это второй уровень? Где гарантии завершаемости проекта?

Отслеживание проекта – это тоже куст деятельностей, связанных с постоянным контролем того, как идет разработка, насколько продвижение соответствует плану. Отслеживание имеет своей целью (у каждой ключевой области есть цель, одна или более) получить как можно более раннее оповещение об отставании. Потому что одно дело, когда мы скажем заказчику, что не получается, накануне сдачи, и совсем другое дело, когда мы скажем ему в самом начале проекта, что начинают накапливаться опасные отклонения, связанные с тем-то и тем-то; тогда у нас еще есть неизрасходованные ресурсы и время на перепланирование работ. То есть мы замечаем проблему или симптомы проблемы как можно раньше. Для того чтобы это все происходило, требуется отслеживание проекта. Реально в организациях-

разработчиках отслеживание выполняется еженедельно. Каждую неделю все разработчики пишут отчет. Не просто отчет, что конкретный человек сделал, а отчет о соответствии плану того, что сделано, и показывают определенные характеристики того, что сделано, из которых сразу видно, как процесс идет: по плану или же начинаются отклонения от него. А если видны отклонения, то тут же собирается совещание, на котором анализируются причины отклонения и принимаются определенные решения, как поправить ситуацию. Поскольку это делается каждую неделю, то заранее, задолго до обвала, разработчики начинают видеть его первые симптомы и, как правило, принимают вовремя какие-то решения, позволяющие этого обвала избежать.

Следующая ключевая область называется «управление субподрядчиками». Иногда ее может не быть, в том случае, когда проект выполняется только силами одного разработчика. Когда разработчик часть работы перепоручает кому-то другому, то такое перепоручение называется субподряд. Если есть субподрядчики, то надо правильно поставить работу с ними. Потому что все остальное может быть хорошо, но если работа с субподрядчиками поставлена плохо, то опять весь продукт будет плохого качества.

Обеспечение качества – это целый куст мероприятий по тому, как мы хотим добиться надлежащего качества нашего продукта через постоянный контроль самого продукта, всех промежуточных продуктов и контроль процесса исполнения. Без контроля нет знания о том, что процесс соблюдается.

И, наконец, управление конфигурацией – это целый куст деятельностей, связанных с поддержанием целостности создаваемого продукта и среды его разработки.

Второй уровень зрелости говорит о том, что если берется новый проект и он похож на прежний, то результаты, затраты и сроки окажутся примерно такими же. Отличие второго уровня от первого еще в том, что на первом уровне процесс вообще не имеет никакой структуры, а на втором он уже структурирован – разделен на фазы и, кроме самого процесса, есть еще и некоторое управление им.

Если на втором уровне фазы процесса замкнуты, то на третьем они уже имеют некоторую структуру и управление ими более сложное: оно имеет дополнительные элементы, адресующиеся к структурным элементам фаз. Благодаря этому обстоятельству, практически нет незакончившихся объектов. Большинство все же в срок не успевают по разным причинам.

Перечислим ключевые области третьего уровня.

- Нацеленность процесса, то есть, какие цели он должен преследовать, ясное понимание этих целей.
- Определение процесса – весь процесс должен быть определен, создается так называемая книга процесса, в которой описаны все связанные с ним деятельности.
- Повышение квалификации, то есть, имеется постоянно действующая программа повышения квалификации, обязательная для всех участников разработки.
- Интеграционное управление связано как со сборкой продукта, так и с тестированием и созданием уже всего сложного продукта.

Технология производства объединяет определенные деятельности в этом направлении.

- Межгрупповая координация. Как правило, существуют несколько проектов в организации-разработчике; каждая группа занимается своим проектом, и это могут быть внешне не связанные проекты, тем не менее, в организации должна существовать практика взаимодействия разных групп.
- Товарищеские обзоры – средство обязательного контроля каждого выходного и промежуточного документа.

Следующий четвертый уровень называется управляемым. На нем все проекты заканчиваются в срок и близко к цели.

Здесь тоже внутренние элементы процесса имеют свою структуру, но важно то, что управление процессом взаимодействует активно и с ними. Поэтому четвертый уровень и называется управляемым. Если на третьем уровне мы только наблюдаем за ним, что-то измеряем, то на четвертом уровне мы активно воздействуем на процесс в ходе разработки. Две новые ключевые области деятельности, характеризующие четвертый уровень, называются «управление качеством» и «количественное управление».

Управление качеством. Если на втором уровне было обеспечение качества, то есть мы только добивались того уровня, который был задан, то здесь мы им уже управляем. Это принципиально новая возможность.

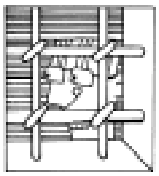
И вторая область – количественное управление. Это означает, что управление процессом, все воздействия на процесс осуществляются с применением метрик, которые немедленно показывают, как наше воздействие отзывается на процессе.

Пятый уровень идеален по части планирования. Если до начала проекта разработчики определили, что понадобится шесть месяцев и две недели, то именно так и будет, не больше и не меньше. Процесс пятого уровня называется оптимизирующимся, потому что в него встроено постоянное самосовершенствование процесса; три ключевые области, с которыми это связано, – предотвращение дефектов, управление изменением технологий и управление изменением самого процесса.

Предотвращение дефектов (не исправление ошибок, не техника выявления ошибок и дефектов в программах, а именно их предотвращение). Создание условий, при которых дефекты вообще не возникают.

Управление изменением технологий. Добиться хороших результатов нельзя, не совершенствуя постоянно и не внедряя постоянно все новые и новые технологии. Прогресс в технологиях совершается постоянно, поэтому надо следить за этим, схватывать это и внедрять в свой процесс.

И, наконец, управление изменением процесса. Поскольку процесс самосовершенствуется, то он сам себя меняет и тоже должен быть под управлением самого себя, преследуя определенную цель. Цель всегда задана – процесс постоянно движется в сторону все большей своей оптимизации.



5. ЗАКЛЮЧЕНИЕ

Суммируя все предыдущее, можно сказать, что на первом уровне зрелости результат не предсказуем. Разные находки для реализации сделаны для данного случая. Есть латинское выражение, это обозначающее, – ad hoc – для данного конкретного случая. Нет массовости, нет продуманности, первое решение, которое приходит в голову, и может оказаться реализуемым. Оно может быть удачным для данного случая, но может быть совершенно не годящимся для общего случая.

На втором уровне результат предсказуем и имеет место управление процессом. Это уже прекрасно, когда есть управление.

На третьем уровне технология разработки программного продукта и управление существуют, определены и согласованно объединены.

На четвертом уровне процесс и продукты, создаваемые в этом процессе, постоянно контролируются измерениями и управляются через эти измерения. Именно поэтому четвертый уровень называется управляемым. Здесь ключевое слово – измерения. Мы измеряем и процесс, и продукты; смотрим на результаты этих измерений, говорим, что нам нравится или не нравится, предпринимаем какие-то управляющие воздействия.

Наконец, на пятом уровне имеет место постоянное встроенное совершенствование процесса.

Литература.

1. *Boehm B. W.* Software Engineering Economics. Englewood Cliffs: Prentice Hall, 1981. Русский перевод: *Боэм Б. У.* Инженерное проектирование программного обеспечения: Пер. с англ. М.: Радио и связь, 1985.

2. *Brooks F. P. Jr.* The Mythical Man-Month. S.L.: Addison-Wesley, 1975. Русский перевод: *Брукс Ф. П. мл.* Как проектируются и создаются программные комплексы. (Серия: «Библиотечка программиста»). М.: Наука, 1979; второе издание: СПб: Символ, 2000.

3. *DeMarco T.* Controlling Software Projects. Englewood Cliffs: Prentice Hall, 1982. *Humphrey G.* Managing the Software Process. Reading: Addison-Wesley, 1989.

4. *Ruskin A.M., Estes W.E.* What Every Engineer Should Know about Project Management. New York: Marcel Dekker, Inc., 1994.

5. *Баранов С.Н., Домарацкий А.Н., Ласточкин Н.К., Морозов В.П.* Процесс разработки программных изделий. М.: Наука Физматлит, 2000.

Баранов Сергей Николаевич,
доктор технических наук,
профессор, главный конструктор
программного обеспечения ЗАО
«Моторола ЗАО».