## Разница между разработкой и производством ПО

**Разрабо́тка програ́ммного обеспе́чения** (*software development*) — это род деятельности и процесс, направленный на создание и поддержание работоспособности, качества и надежности программного обеспечения, используя технологии, методологию и практики из информатики, управления проектами, математики, инженерии и других областей знания.

**Произво́дство**, в экономическом смысле — процесс создания какого-либо продукта. Понятие производства характеризует специфически человеческий тип обмена веществами с природой, или, более точно, — процесс активного преобразования людьми природных ресурсов в какой-либо продукт.

Как отмечает Философская энциклопедия, процесс производства всегда носит общественный характер: производство обособленного одиночки вне общества представляет собой, по выражению Маркса, такую же бессмыслицу, как развитие языка без совместно живущих индивидов.

Важные замечания:

1. Производство программного обеспечения (ПО) предполагает активное преобразование энергии, знании и опыта людских ресурсов в конченый продукт, то есть, само ПО.
2. Производство – одна из стадий (завершающая) разработки ПО.
3. Следует понимать, что разработка отличается от производства своей глобальностью и общностью. Разработка предоставляет каскад для команды / компании / корпорации, на который они могут навешивать различные парадигмы, по-разному распределять ресурсы и просчитывать риски в зависимости от желаемого продукта. Производство, как было сказано, это процесс, то есть, движение от точки А к точке Б. Можно провести такое сравнение: разработка это механизм в выключенном состоянии, который можно дополнять или урезать, а производство это тот же механизм (или некоторые его частные аспекты) во включённом состоянии.

Разработка программного обеспечения имеет дело с проблемами качества, стоимости и надёжности, об этом уже неоднократно говорилось. Эти аспекты характеризуют вроде бы исключительно техническую задачу с экономической точки зрения. Аналогичным образом строится и производство ПО. В данном случае, проблемы затратности и оптимизации рисков целиком попадают под производственный вопрос.

Разработка программного обеспечения может быть разделена на несколько разделов.

1. **Требования к программному обеспечению**: извлечение, анализ, спецификация и ратификация требований для программного обеспечения.
2. **Проектирование программного обеспечения**: проектирование программного обеспечения средствами [Автоматизированной Разработки Программного Обеспечения (CASE)](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%9E%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и стандарты формата описаний, такие как Унифицированный Язык Моделирования ([UML](https://ru.wikipedia.org/wiki/UML)), используя различные подходы: [проблемно-ориентированное проектирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и т. д.
3. **Инженерия программного обеспечения**: создание программного обеспечения с помощью языков программирования.
4. **Тестирование программного обеспечения**: поиск и исправление ошибок в программе.
5. **Сопровождение программного обеспечения**: программные системы часто имеют проблемы совместимости и переносимости, а также нуждаются в последующих модификациях в течение долгого времени после того, как закончена их первая версия. Подобласть имеет дело с этими проблемами.
6. **Управление конфигурацией программного обеспечения**: так как системы программного обеспечения очень сложны и модифицируются в процессе эксплуатации, их конфигурации должны управляться стандартизированным и структурированным методом.
7. **Управление разработкой программного обеспечения**: управление системами программного обеспечения имеет заимствования из управления проектами, но есть нюансы, не встречающиеся в других дисциплинах управления из-за специфики ресурсов и конечного продукта. Во многом, конечный продукт неосязаем. Об этом очень хорошо было сказано у Тарасова (тема SaaS).
8. **Процесс разработки программного обеспечения**: процесс построения программного обеспечения горячо обсуждается среди практиков, основными парадигмами считаются [agile](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) или [waterfall](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B2%D0%BE%D0%B4%D0%BE%D0%BF%D0%B0%D0%B4%D0%B0).
9. **Инструменты разработки программного обеспечения**, см. [CASE](https://ru.wikipedia.org/wiki/CASE): методика оценки сложности системы, выбора средств разработки и применения программной системы.
10. **Качество программного обеспечения**: методика оценки критериев качества программного продукта и требований к надёжности.
11. **Локализация программного обеспечения**, ветвь языковой промышленности.

На протяжении нескольких десятилетий стоит задача поиска повторяемого, предсказуемого процесса или методологии, которая бы улучшила продуктивность, качество и надёжность разработки. Одни пытались систематизировать и формализовать этот, по-видимому, малопредсказуемый процесс. Другие применяли к нему методы управления проектами и методы программной инженерии. Третьи считали, что без постоянного контроля со стороны заказчика разработка ПО выходит из-под контроля, съедая лишнее время и средства.

Опыт управления разработкой программ отражается в соответствующих руководствах, обычаях и стандартах.

Производство ПО не начинается, пока не будут определены все стандарты, на которые должно опираться производство. Т.е., механизм без всех гвоздиков не заработает.

Однако, это в теории всё так хорошо. На практике же всегда процесс производства, а именно цикл жизни проекта, начинается тогда, когда ещё не все стандарты могут быть определены.

К тому же, стоит заметить, что производство ПО не такой уж и закрытый объект, каким его хотелось бы видеть. Почему? Да потому что на проекты берут новых людей, которые не имеют понятия о том, с каких нулей стартовало производство. Но зато им понятно, какой кусок работы необходимо сделать. Однако, на мой взгляд, нельзя данную «слепоту нового разработчика» считать ужасной и неприемлемой: напротив, в некоторых ситуациях подробное знание всего, что происходило на проекте, может отвлечь разработчика от сути проекта и выполнения своих должностных обязанностей. А это стоит времени и денег, а это затягивает производственный процесс.

## Участники процесса разработки ПО

* Пользователь
* Заказчик
* Разработчик
* Руководитель проекта
* Аналитик
* Тестировщик
* Поставщик

Меняется ли как-то этот список на этапе производства ПО? Незначительно. Уходят из списка тестировщики. Во многом, аналитика во время производства уходит куда-то на задний план, тем не менее, оставаясь одним из самых важных столпов, обеспечивающих положительную обратную связь с пользователем.

В самом определении производства ПО смыты рамки настолько, что трудно понять: входит ли сам цикл работы над проектом внутрь производства или нет? В каких-то методиках он входит, а в каких-то производство ограничивается лишь настройкой конвейера, который поставляет ПО пользователю и следит за тем, что бы можно было добавить такого, чтобы и пользователь был рад, и прибыли было побольше.

Если следовать строгой логике данного ранее определения термина «производство», то цикл разработки проекта входит в него практически весь, лишь с оговорками, которые носят весьма субъективный характер в силу размытого определения производства ПО и его проекции на реалии разработки ПО.

Можно даже такую параллель провести: разработка – набор спецификационных документов и тестов, которым должна следовать и которые должна пройти определённая программа. А эта программа и есть производство ПО.

## Проблемы разработки ПО

Наиболее распространёнными проблемами, возникающими в процессе разработки ПО, считают:

* **Недостаток прозрачности**. В любой момент времени сложно сказать, в каком состоянии находится проект и каков процент его завершения. **В грамотно налаженном производственном механизме этой проблемы нет: механизм работает как часы / разработка подходит к звершению.**
* **Недостаток контроля**. Без точной оценки процесса разработки срываются графики выполнения работ и превышаются установленные бюджеты. Сложно оценить объём выполненной и оставшейся работы. **Встречается и в производстве (это уже перерастает в проблему халатности).**
* **Недостаток мониторинга**. Невозможность наблюдать ход развития проекта не позволяет контролировать ход разработки в реальном времени. С помощью инструментальных средств менеджеры проектов принимают решения на основе данных, поступающих в реальном времени. **(Аналогично прозрачности процессов в ПО).**
* **Неконтролируемые изменения**. У потребителей постоянно возникают новые идеи относительно разрабатываемого программного обеспечения. Влияние изменений может быть существенным для успеха проекта, поэтому важно оценивать предлагаемые изменения и реализовывать только одобренные, контролируя этот процесс с помощью программных средств. (**Встречается в производстве, из-за чего процесс = механизм приходится перенастраивать, а, следовательно, возвращаться к расчётам и т.д. и т.п.)**
* **Недостаточная надёжность**. Самый сложный процесс — поиск и исправление ошибок в программах на ЭВМ. Поскольку число ошибок в программах заранее неизвестно, то заранее неизвестна и продолжительность отладки программ и отсутствие гарантий отсутствия ошибок в программах. Следует отметить, что привлечение доказательного подхода к проектированию ПО позволяет обнаружить ошибки в программе до её выполнения. В этом направлении много работали Кнут, Дейкстра и Вирт. Профессор Вирт при разработке [Паскаля](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29) и [Оберона](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D1%80%D0%BE%D0%BD_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29) за счет строгости их синтаксиса добился математической доказуемости завершаемости и правильности программ, написанной на этих языках. **(В производстве вряд ли бывает, это больше проектирование ПО)**
* **Неправильный выбор методологии** разработки программного обеспечения. (Тоже больше проектирование, однако, способ налаживания конвейера в первую очередь как средства общения между разработчиком и заказчиком напрямую зависит от методологии).
* **Отсутствие гарантий качества и надежности программ** из-за отсутствия гарантий отсутствия ошибок в программах вплоть до формальной сдачи программ заказчикам.  
  Данная проблема не является проблемой, относящейся исключительно к разработке ПО. Гарантия качества — это проблема выбора поставщика товара (не продукта). (**Да, это проблема производства, над которой разработчикам надо беспокоится настолько, насколько их код должен проходить все необходимые тесты).**

Основные виды простых производств можно описать как:

* линеарное производство
* расходящееся производство
* сходящееся производство
* смешанное (из простых) производство

к сложным видам производствам можно отнести:

* цикличное производство
* смешанное (из простых и сложных) производство

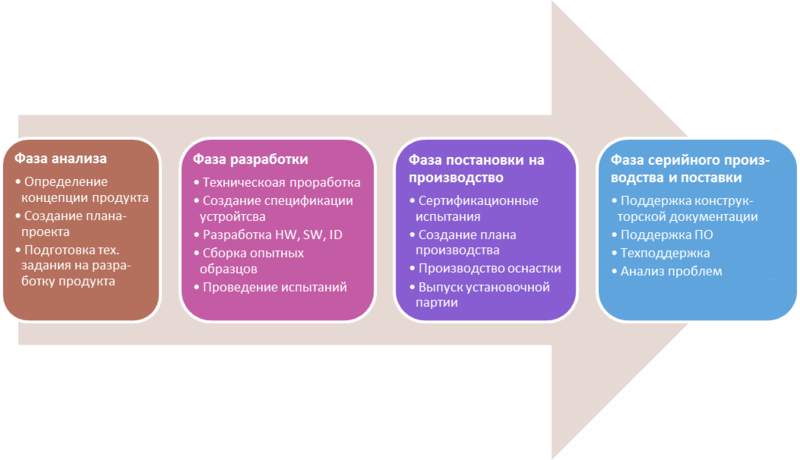
Реальное производство является зачастую смешанным производством, но для оптимизации производства или для расчёта расчётных цен необходимо понимание видов производства (организации структуры производственных факторов).

**Частные примеры**

Разработка продуктов в сфере электроники (взято: <http://habrahabr.ru/company/promwad/blog/172771/>)

Презентации новых гаджетов Apple, Samsung и других брендов — это только видимая часть айсберга, под которой скрывается человеко-десятилетия труда людей самых разных специализаций: инженеры, программисты, дизайнеры, логисты, руководители различных уровней, продавцы и так далее. Пока ты не погружаешься в эту внутреннюю кухню, может показаться, что процесс довольно простой и понятный: была бы идея, хорошая команда и достаточное финансирование. Однако не все так просто.

Упрощенно этапы создания нового устройства выглядят так:



#### Фаза №1. Анализ

У всех эта фаза проходит и называется по-разному: «Расчеты», «Проектирование», «Сбор требований». Одни компании сразу же стартуют с тех. задания, написанного инженером «в теме», другие уделяют чертовски много времени планированию и проработке. Почему же это может быть важно? Причины просты:

1. **Планирование бюджета.** Как правило, процесс разработки продукта довольно длинный и будет неприятно, если после 8—9 месяцев инвестирования у вас внезапно закончатся деньги. Матерые компании и продакт-менеджеры имеют собственные технологии планирования для новых продуктов, которые банки даже принимают в качестве гарантий по кредитам.
2. **Оптимизация себестоимости.** Если вы производите 50 тыс. устройств в год, будет полезно экономить каждый доллар в себестоимости устройства. – вот и производственная аналитика.
3. **Управление.** В разработку любого продукта, даже довольно простого, завязаны тысячи людей (прямо или косвенно: поставщики компонентов, инженеры, маркетологи, рабочие завода-изготовителя и т.д.). Задачи нужно будет делегировать правильным специалистам в правильное время.
4. **Цена ошибки**, она очень высока, и чем раньше мы ее выявляем, тем лучше.

Главная задача фазы анализа — определить четкую **концепцию продукта**, над которым предстоит работа на последующих этапах, и синхронизировать ее с остальными участниками по мере необходимости. Следует учитывать вопросы конечных клиентов, себестоимость, требования производственных площадок, планы проекта, контрактные обязательства, бюджеты, финансирование, исполнителей, разрешающие и сертифицирующие документы, выбор материалов и компонентов.   
  
Если говорить о **планировании нового продукта**, то это своего рода искусство, которым занимаются продакт-менеджеры.   
Финальный этап фазы анализа — **подготовка технического задания**, в нем мы определяем функциональные и нефункциональные требования к продукту, прорабатываем технический дизайн и создаем скетчи устройства:   
  
В электронике мы всегда говорим об ограниченности ресурса. Разработчики ПО вне темы embedded **уже давно забыли про дефицит, например, оперативной памяти**. А тем временем подобные ограничения делают работу инженера крайне увлекательной. Для инженеров-электронщиков технические параметры устройства — это рамки творчества.   
  
Я уже говорил про цену ошибки, внесение изменений в электронике — это долго и дорого. Для ПО **можно выпустить обновления**, а что можно выпустить для реального предмета? Только другой реальный предмет, в котором недостатки устранены.   
  
Некоторые продукты в сфере электроники «не взлетали» именно по причине нерабочего софта.   
  
Даже самое тщательное планирование не защитит от ошибок на 100%. Вы наверняка читали подобные заголовки новостей:

* Apple отзывает iPod nano: в плеерах найден дефект аккумулятора
* Nokia отзывает 14 миллионов потенциально бракованных зарядных устройств
* Intel отзывает партию чипсетов для процессоров из-за обнаруженного дефекта
* Garmin отзывает более миллиона GPS-навигаторов из-за перегрева аккумуляторов
* Lenovo отзывает свои компьютеры из-за риска внезапного возгорания

Итак, что мы получаем в итоге фазы проектирования? Набор документации, контрактных обязательств и команду, которая понимает продукт и заряжена на успех.

#### Фаза №2. Разработка

Как правило, электронное изделие состоит из следующих компонентов, которыми занимается инженерный отдел:

1. **Железо**
2. **Корпус**
3. **Софт**
4. **Технологии производства**

А вот эти компоненты остаются за границами инженерного отдела (мы к ним вернемся позже):

|  |  |
| --- | --- |
| * Упаковка * Брэнд * Логистика * Производство | * Каналы продаж * Сервисное обслуживание * Инфраструктура * Интеллектуальная собственность (патенты и т.д.) |

**Железо**. Благодаря современным САПРам разработка аппаратного обеспечения — это достаточно стандартизованный процесс, который при хорошо сформированных вводных не длится долго (от 2 недель до пары месяцев максимум). Самая сложная штука — это конструкция. Дело в том, что физическое расположение одних элементов относительно других играет большую роль и оказывает непосредственное влияние на работоспособность современного устройства.   
  
**Промышленный дизайн**. Внешниий вид корпуса плотно привязан к технологии производства. Технология — это материалы, а значит — стоимость. Дизайнер может нарисовать абсолютно волшебную штуку, но для ее воплощения в жизнь понадобится труд конструктора и технолога. Технологические лидеры ставят уникальный дизайн во главу угла, и потому непрерывно создают новые технологии производства.   
  
**Программное обеспечение** в электронике работает в условиях ограниченных ресурсов. Подход к его разработке кардинально отличается от подходов, принятых в индустрии разработки ПО для веба, десктопов и т.п. Используются другие инструменты и другая среда разработки: версии ОС для встраиваемых систем Windows CE, Linux Embedded и т.п.   
  
Программисты-электронщики сильно завязаны на железе. Если разработчик десктопных приложений не задумывается о корректности работы своей аппаратной платформы, то инженер-программист сталкивается с этим постоянно, особенно при работе с новыми компонентами. У меня были проекты, в которых баги в программе возникали из-за ошибок в работе процессора либо от плохого монтажа. К слову последнее — это действительно серьезная проблема для дизайн-центров.   
  
Еще пример: программист и дизайнер могут создать интерфейс пользователя, который будет отлично работать на тестовой платформе, но вызовет перегрузку оперативной памяти на реальном устройстве. Причем такие ошибки не всегда можно заметить сразу. Сутки устройство работает корректно, а потом перезагружается.   
  
Все эти особенности существенно усложняют жизнь программистов, которые работают со встраиваемым ПО, хотя многие специалисты выбирают электронику как раз ради таких увлекательных задач.   
  
Как правило, промежуточным результатом фазы разработки являются **опытные образцы устройства**. Процесс их сборки — это вообще отдельная история. Со всего мира в одну точку на карте съезжается сотня компонентов, каждый упакован в отдельную посылку. На специализированном предприятии по специальным технологиям делаются рабочие образцы.  
  
Ваше положение на карте мира практически не имеет значения при разработке софта. А вот при разработке железа сроки производства образца могут различаться в десятки раз в зависимости от положения разработчика и изготовителя. Компании в Китае собирают образец за 2 недели, в России этот срок может доходить до пары месяцев.   
  
После сборки образцы уходят на **тестирование**. По его результатам мы можем повторить весь процесс с самого начала, чтобы исправить найденные косяки либо понять, что в таком исполнении продукт никому не нужен. Такой вот суровый scrum c циклом в год.

#### Опытные образцы и испытания

Очевидно, что цель проведения всех видов тестирования — это снижение риска неприятных неожиданностей на следующих этапах.   
  
Итак, у нас завершился этап разработки, и мы имеем готовые опытные образцы. Что же это такое и для чего нужно? Опытный образец (или прототип) — это устройство, которое было собранно с применением технологий прототипирования согласно нашей конструкторской документации.   
  
Чтобы убедиться в том, что опытные образцы соответствуют всем необходимым требованиям, мы проводим самые разные испытания, их никогда не бывает мало, они зависят от компании и продукта. Основные тесты включают в себя предсертификационные испытания, функциональное тестирование, ЭМС, электробезопасность, климатику и др. Если продукт работает в какой-то системе, проводятся испытания «в поле», обычно они дают намного больше ценных данных, чем синтетические тесты устройства. Можно до бесконечности тестировать роутер на синтетических тестах, но пока не включишь его в реальную сеть и не начать решать реальные задачи, нельзя понять, работает он нормально или нет.  
  
В рамках одного проекта может быть сделано огромное количество опытных образцов — для выбора материалов корпуса и комплектующих. Есть вещи, которые нельзя cмоделировать даже в самых современных САПРах: тактильные ощущения, насколько устройство приятно держать в руках, маркость пластика, его блеск, ощущение веса устройства. Только живое сравнение разных вариантов позволяет сделать оптимальный выбор.  
  
В наших реалиях опытные образцы делаются на оборудовании для серийного производства, но по другим технологиям. Например, пластик не отливается, а фрезеруется или выращивается, поскольку создание литьевой пресс-формы — это длительный и затратный этап.  
  
Главная цель всех испытаний — убедиться в том, что продукт готов и что именно его хотят покупатели. На этом завершается фаза разработки. Следующий шаг — **постановка изделий на серийное производство.**

#### Фаза постановки на производство

Теперь наша цель — создать технологию производства устройства в определенных тиражах с заданными сроками, себестоимостью и показателями качества.  
  
Для начала нужно определиться с производственной площадкой. Как правило, крупные компании выбирают себе производственных партнеров заранее. Для стартапов выбор производства — отдельная история, зачастую именно на этой фазе они задумываются о процессе производства как таковом, не раньше. Поэтому можно увидеть множество прототипов, разработанных в различных небольших компаниях и при этом очень мало изделий, которые продаются сериями.  
  
В электронике, как правило, устройства разрабатываются под конкретное производство, т.к. на нем завязаны технологические процессы, поставка компонентов, логистика, контракты и т.д. В противном случае на этапе постановки на производство нам предстоит определить техпроцессы, цепочку поставок, согласовать логистику, связать все это в ясный план.  
  
[Выбор производственной площадки](http://promwad.ru/uslugi/postanovka-na-proizvodstvo) — это как выбор невесты. Я несколько раз проводил эту процедуру в Юго-Восточной Азии и отсеивал неподходящих производителей в поисках «того самого», который и качество обеспечит, и приемлемые коммерческие условия.   
  
Когда производство выбрано и все нюансы согласованы, начинается реальная работа. Одна из задач — создание оснастки.

#### Производство оснастки

Важная веха в постановке на производство — разработка и производство оснастки. Самая распространенная — пресс-форма для литья пластиковых деталей. Может производиться и любая другая оснастка, которая позволяет реализовать различные технологические операции: гибочная, штамповочная и т д.  
  
В общем, оснастка — это важно, качество литьевой формы напрямую скажется на качестве отлитого пластика.   
  
Еще один интересный момент — **проверка качества оснастки**. Один корпус можно рассмотреть, изнюхать, измерить и убедиться: это то, что нужно. А как же быть, если их производятся тысячи? Как правило, делается golden sample — тот самый Золотой Образец, с которым необходимо сравнивать все остальные образцы в случае сомнений. Этакий эталон килограмма в палате мер и весов.

#### Контроль качества на производстве

В процессе производства тестируется всё и на всех этапах, т.к. любой косяк может стоить много денег и лучше основательно подготовиться с проведением испытаний. Во всяком случае, так должно быть.   
  
Что тестируется:

* Каждый компонент у производителя компонентов. Когда эти компоненты прибывают на фабрику, они тестируются повторно непосредственно перед монтажом (выборочно или все подряд). Большинство производств тестирует всё подряд, чтобы обеспечить максимальное качество.
* Печатная плата (и у производителя, и перед запайкой).
* Оптический контроль качества монтажа.
* Внутрисхемное тестирование уже собранной платы.
* Функциональное тестирование. Для сложных устройств создаются стенды, способные быстро и качественно полностью протестировать функционал устройства и убедиться в том, что все работает.

И все равно после всех этих манипуляций средний показатель качества по индустрии — 0,5% процента брака. Для Китая приемлем 1%, т.е. каждое сотое устройство будет с косяком. Это довольно много для серии, поэтому хорошие компании устанавливают планку в 0,1% или еще ниже.   
  
Как-то мы общались с представителем китайской торговой братии по поводу приобретения одного электронного компонента. На открытом рынке его цена была приблизительно 15 долларов за штуку, а этот парень предлагал за 2$. На вопрос «а почему собственно так дешево» он отвечал: «Потому что там ровно 30% брака». «Так уж ровно 30%? А если больше?» — спросили мы и в ответ получили заманчивое предложение: «Тогда я вам еще насыплю».  
  
К слову контрафактные компоненты — это серьезная опасность для любого производства. Количество поддельных, не рабочих и бракованных компонентов на рынке Юго-Восточной Азии поражает воображение.

#### Выпуск установочной партии

Итак, у нас готова оснастка, мы привезли нужные компоненты, понимаем свой план тестирования и готовы стартовать производство.  
  
Все начинается с маленькой серии, например, с 10 устройств. На них мы отлавливаем ошибки в технологических цепочках. Могут быть проблемы с компонентами, с тестами, с рабочими, с фазой луны. Все эти вещи необходимо отследить и сделать так, чтобы это не влияло на качество и работоспособность продукта. Часто вносятся необходимые поправки, чтобы все работало как часы.  
  
После того как все проблемы устранены, выпускается чуть большая партия. Скажем, 100 штук.   
  
Разные рынки накладывают разные требования к сертификации и проверке. Сертификацию часто делают на устройствах из первой партии, после чего появляется возможность легально ставить заветные CE, FCA, и так далее.

#### Фаза серийного производства и поставки

Серийное производство в управленческой теории — это хороший и понятный процесс: где-то есть «черная коробочка», в которую загружаешь кэш, а она выдает готовые изделия, которые можно грузить в корабли/самолеты и отправлять покупателям.   
  
**На практике производство — это каждодневное решение проблем, не только технологических.** Например, компонент подорожал. Или приходит заявка на замену компонента без подвижек в сроках выпуска. Или, что хуже, в результате «оптимизации себестоимости» устройство перестало работать.  
  
В итоге на фазу постановки и самого производства необходимо держать инженерный штат, который будет разбираться в проблемах, допиливать документацию и, конечно, допиливать ПО.  
  
Но все проблемы решаемы, рано или поздно все начинает работать и после этого фокус перемещается на измерение качества готовой продукции.

#### Поддержка и обслуживание

Про это следует задуматься уже на этапе разработки продукта. Нужно ответить на вопросы: будет ли устройство обслуживаться вообще или после поломки оно должно отправиться в мусорное ведро? Кто его будет обслуживать? На территории каких стран/регионов будет обеспечиваться поддержка?   
  
Крупные компании зачастую формируют собственные центры техподдержки близко к центрам продаж. Компании поменьше передают это на аутсорсинг.   
  
Стоимость поддержки, как правило, изначально включена в стоимость товара. У компаний есть статистика по своему продукту, средние показатели по отрасли, а также собственные целевые показатели. Например, задана планка по затратам в 2$ на каждое устройство. И потом менеджеры ломают голову, как это организовать. Таким образом, поддержка — это чисто затратная часть, а, значит, ее качество мало у кого является хорошим.

#### Что осталось вне инженерного отдела?

Часть компонентов процесса разработки продукта остается за границами инженерного отдела:

|  |  |
| --- | --- |
| * Упаковка * Брэнд * Логистика * Производство | * Каналы продаж * Сервисное обслуживание * Инфраструктура * Интеллектуальная собственность (патенты и т.д.) |

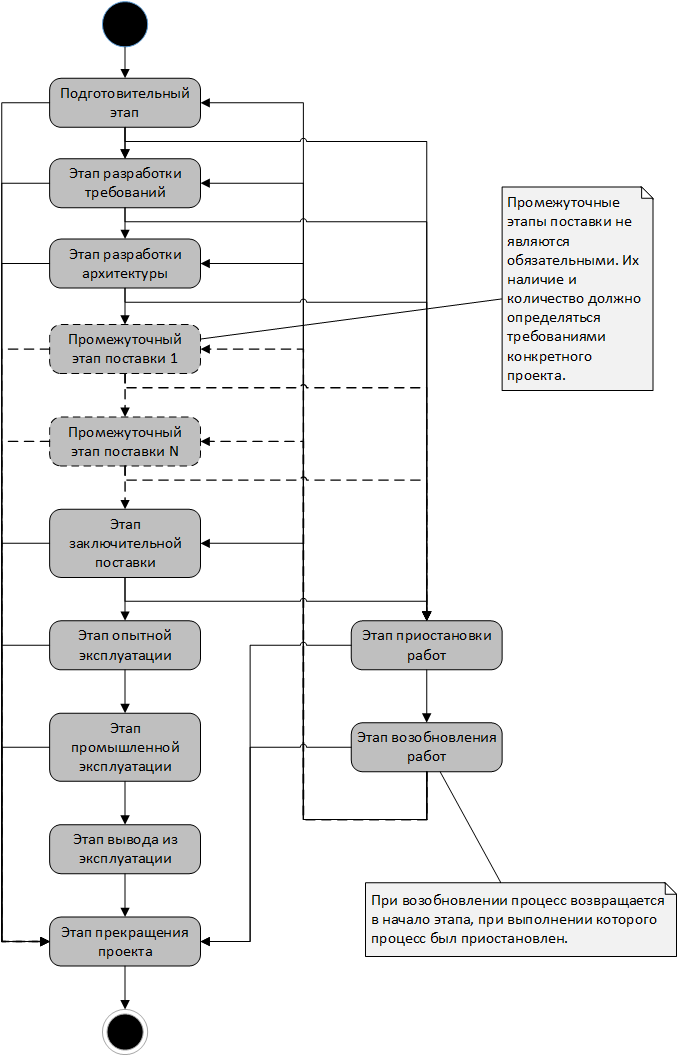
Большинство этих вопросов находятся в компетенции **продакт-менеджера**, который, по сути, отвечает за то, чтобы продукт был успешен. Каким должен быть процесс планирования, как считать продукт «на коленке», как составлять требования и взаимодействовать с подрядчиками, чтобы получилось хорошо? Ответам на эти вопросы хотелось бы посвятить отдельный пост про продакт-менежмент в электронике. Традиционно, при наличии интереса со стороны читателей.

# Обзор процесса разработки программного обеспечения

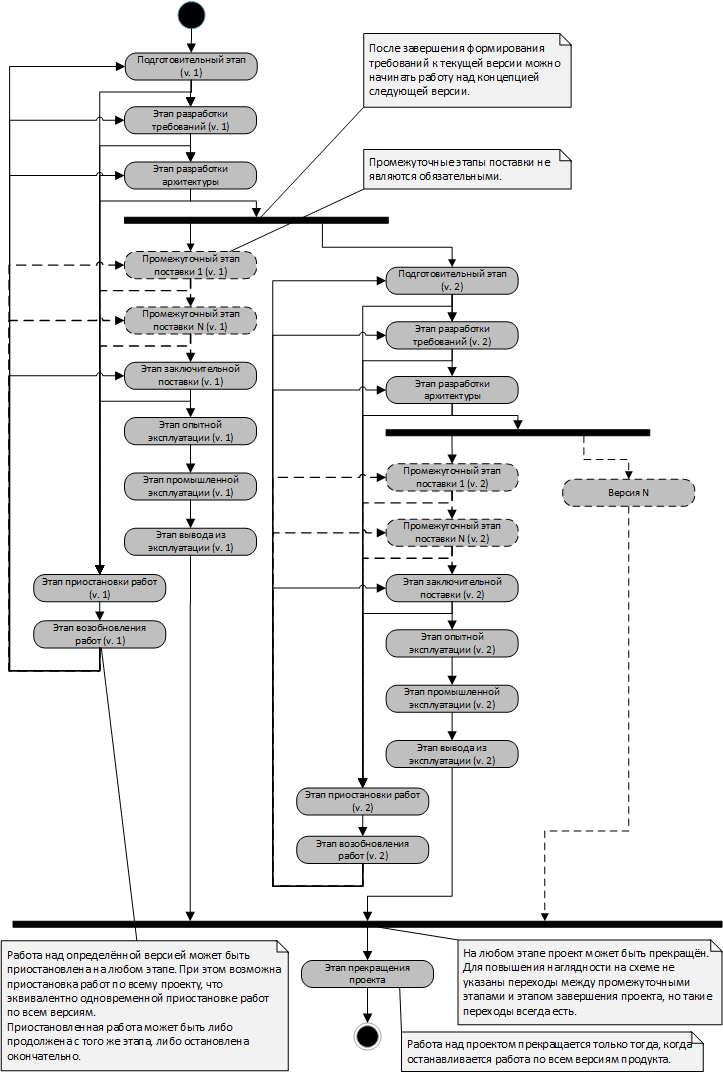
#### Введение

Прежде, чем предложить обзор процесса разработки, сложившегося в результате накопления опыта за последние годы, я хотел бы сделать несколько общих пояснений, которые мне кажутся существенными.  
  
Я работаю в IT последние 15 лет, хотя программированием начал заниматься значительно раньше. Основное направление моей деятельности как системного архитектора была организация разработки программ, разработка концепций и верхнеуровневой архитектуры и контроль выполнения концепции на протяжении проекта. Кроме управления разработкой ПО и создания архитектуры, я время от времени занимаюсь решением сложных технических проблем и написанием некоторых критически важных участков кода, где необходимо не только знание самого языка и среды разработки, но и их внутренней организации, иногда преподносящей неприятные сюрпризы.  
  
Проекты, над которыми я работаю, чаще всего связаны с разработкой заказного или инвестиционного программного обеспечения. Также мне приходилось работать с встроенным ПО и программами, ориентированными на выпуск «хитов» (что, с лёгкой руки Джоэля Спольски, я называю далее игровым ПО, хотя на самом деле некоторые игровые проекты ближе к инвестиционным).  
  
Заказное программное обеспечение может быть предназначено для внутреннего или внешнего заказчика. Эксклюзивные права на разработанную систему получает заказчик, и работа над развитием системы в дальнейшем может быть передана другому исполнителю.   
  
В отличие от заказного ПО, работа над инвестиционным программным обеспечением ведётся самим исполнителем на деньги внутреннего или внешнего инвестора. Как правило, права на код системы остаётся у исполнителя, что стимулирует непрерывную работу по улучшению своего продукта и последовательный выпуск версий с более развитой функциональностью.  
  
Встроенное программное обеспечение поставляется вместе с аппаратной частью и, грубо говоря, не подлежит сопровождению, поскольку отзыв партии устройств производителем – дело очень затратное и потому исключительное.   
  
Разработка игровых хитов также практически не содержит фазы сопровождения. Кроме того, пользователи игровых программ, даже столкнувшись с ошибкой в игре, очень редко загружают обновлённую версию. Поэтому разработка игр, как правило, имеет свою экономику и свой процесс разработки.  
  
Нашими заказчиками являются органы власти, крупные государственные и коммерческие организации и, конечно, мы сами. Поэтому в смысле заказного ПО в нашем процессе часто присутствует некоторая разница между процессами разработки продуктов для внутреннего и для внешнего заказчиков. Некоторые нюансы я укажу в этой статье. Уровень формализации отношений с заказчиком у нас варьируется от проекта к проекту очень широко. В целом, чем больше бюджет проекта, тем выше формальность. Государственный заказчик или крупные коммерческие предприятия (особенно с государственным участием) обычно имеют законодательные ограничения на формирование, размещение заказа и приёмку результатов работ. Ещё одним ограничением крупных организаций является тот факт, что их персонал, являющийся источником требований и основным пользователем наших систем, имеет очень ограниченную доступность для исполнителей, хотя бы вследствие своей занятости. Однако для небольших организаций уровень формализации падает и иногда уходит в противоположную крайность, где возникает недостаточный уровень ответственности заказчика в рамках проекта.  
  
Другая сторона наших заказных проектов – высокие требования к функциональности. Это и высокая нагрузка на все системы, и большая географическая распределённость, и высокие требования к точности вычислений при очень ограниченных временных рамках. Часто в наших проектах появляются элементы исследовательской работы и творческого поиска, направленного на решение нетривиальных проектных задач. Иногда нам приходится комбинировать в рамках одного процесса разработки разные методологии, например, вставляя в общий процесс, близкий к RUP, один или несколько этапов почти чистого scrum, порождая что-то вроде проекта в проекте. Это позволяет нам сохранять невысокий уровень вовлеченности пользователей, связанный с природой проекта, с гибкостью разработки в условиях высокой неопределённости требований. В этом плане для меня важен именно подготовительный этап, во время которого можно выбрать необходимую методологию и выстроить оптимальный процесс разработки. Один из примеров применения гибкой методологии я описал в статье [«Применение agile при разработке проекта для государственного заказчика»](http://megamozg.ru/post/14554/).  
  
В качестве примера работы над инвестиционным проектом я могу привести разработку комплексной системы безопасности, которую мы создавали как «коробочный» продукт. Под моим руководством было выпущено последовательно четыре версии этой системы, пользователями которой стали самые разные коммерческие и государственные организации, включая мэрию Москвы, АФК «Система», банки, бизнес-центры и, конечно, наш собственный офис. Первая версия была не очень успешной, но у нас была стратегия развития, которая позволила нам успешно захватить рынок и пережить сложные времена кризиса. Опыт работы над этим и ещё несколькими инвестиционными проектами тоже был учтён при формировании используемого мной процесса разработки.  
  
Наш процесс представляет собой последовательность определённых этапов. Приведённая мной классификация ПО сделана только, чтобы показать возможную разницу в организации разработки различных программных средств. Делая обзор процесса разработки, я остановлюсь только на различиях именно самого процесса касаемо разных видов ПО. Однако надо помнить, что различия между процессами разработки разных видов ПО гораздо глубже, поэтому при планировании каждого этапа необходимо учитывать эти нюансы.  
  
Важно понимать, что переход процесса от одного этапа к другому не имеет чёткой границы. Как правило, работы следующего этапа начинаются по мере выполнения 80-90% работ по предыдущему этапу. Особенно это касается разработки требований, когда в ряде случаев снятие неопределённости происходит лишь к концу проекта. Безусловно, наличие такой неопределённости в проекте является существенным риском и должно находиться под постоянным контролем.

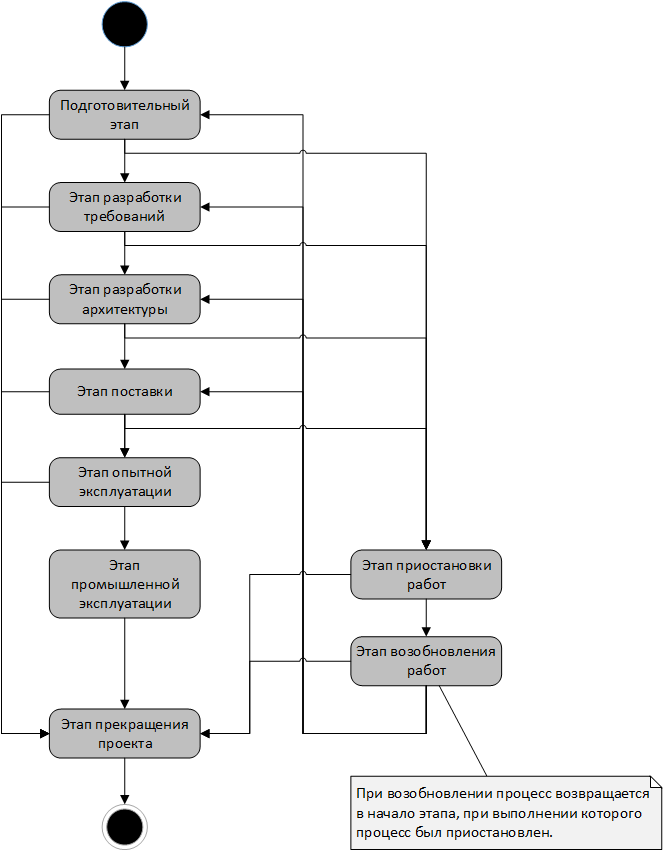
#### Процесс разработки заказного ПО

Обзор процесса разработки начнём с наиболее общего случая – разработки заказного программного обеспечения. Схема процесса приведена на рисунке 1.  
  
  
*Рисунок 1. Процесс разработки заказного программного обеспечения.*  
  
Работа над проектом начинается с подготовительного этапа. Цель этапа состоит в том, чтобы на основе предложений заказчика создать некоторую концепцию будущей системы и, отталкиваясь от этой концепции, провести оценку востребованности и реализуемости проекта. Если решение о привлечении исполнителя принимается заказчиком на конкурсной основе, то предварительный этап фактически является стадией подготовки потенциального исполнителя к конкурсу, включая формирование необходимой документации.  
  
Не нужно тратить время и ресурсы на проект, чья концепция признаётся невостребованной или нереализуемой. Такой проект должен быть завершён. В ряде случаев требуется некоторая итеративная работа с заказчиком по коррекции концепции проекта, пока либо не будет достигнут приемлемый баланс требований заказчика и затрат исполнителя, либо не будет принято решение о сворачивании работ.  
  
Проект, концепция которого выглядит приемлемой для реализации, выходит на этап разработки требований. На этом этапе исполнитель должен сформировать перечень всех явных и скрытых потребностей заказчика. Часто оказывается, что заказчик либо не определился со своими потребностями, либо его потребности вступают в противоречие между собой, с возможностями заказчика или с возможностями исполнителя. Целями этапа являются выявление всех скрытых потребностей, решение конфликтов требований, формирование целостного технического решения и анализ реализуемости подготовленного решения.   
  
Иногда уточнение требований приводит к пересмотру концепции проекта. Если после уточнения всех требований не удаётся найти приемлемого технического решения, проект приходится сворачивать либо откладывать на некоторое время в ожидании более приемлемых обстоятельств.  
  
Если техническое решение найдено, исполнитель приступает к разработке архитектуры будущей системы. Цель этапа – определение верхнеуровневой логической и физической архитектуры, полностью покрывающей все требования заказчика. При разработке архитектуры проводится рецензирование и уточнение концепции, требований и предварительного технического решения, что даёт возможность предупредить наиболее опасные риски.  
  
После завершения проектирования архитектуры необходимо снова провести ревизию основных параметров проекта и решить, в состоянии ли исполнитель завершить проект. Полезно на стадии разработки архитектуры отказаться от излишних и слишком громоздких функций. Оптимизация архитектурного решения часто помогает вписаться в приемлемые параметры проекта. В иных случаях требуется более радикальное сокращение функционала разрабатываемой системы. Однако даже остановка проекта на этой стадии, если она происходит по веским причинам, должна восприниматься как победа: продолжение работ в таком случае может привести только к ещё большим потерям.  
  
Если баланс был найден, и удалось создать приемлемую архитектуру системы, исполнитель может переходить к реализации и поставке системы. Реализация может проходить в один или несколько этапов. Для небольших проектов одноэтапная поставка всего функционала системы может быть вполне приемлемой. Однако, чем больше проект, тем выше зависимости подсистем внутри создаваемой системы. В этих условиях следует делить реализацию на несколько этапов так, чтобы в конце каждого этапа команда разработчиков имела готовый к поставке продукт. При этом самый важный, фундаментальный функционал должен разрабатываться на ранних этапах, а надстройки, работающие поверх этих основных компонентов, следует реализовывать позднее. В таком случае наиболее опасные для системы ошибки будут исправлены на первых этапах, и риск того, что прикладная функциональность системы будет основана на нестабильной основе, будет значительно снижен.  
После поставки полностью завершённой системы проект заказного ПО обычно переходит к этапу опытной эксплуатации. Цель этого этапа заключается в проверке качества работы разработанной системы в реальных условиях эксплуатации. Как правило, на этом этапе исполнитель совместно с заказчиком проводит измерение количественных метрик, позволяющих определить качество созданной системы. В первую очередь проверяются функциональные характеристики качества, затем – нефункциональные. При наличии несоответствий исполнитель корректирует код системы.  
  
Полностью отлаженная и настроенная система вводится в промышленную эксплуатацию. Как правило, исполнитель должен сопровождать систему, по крайней мере, в течение срока гарантии. Выявляемые несоответствия должны исправляться. Пользователи и обслуживающий персонал заказчика должны получат оперативную консультативную поддержку.  
  
Наконец, приходит момент, когда система перестаёт устраивать заказчика по какой-либо причине. Наступает этап вывода системы из эксплуатации. Впрочем, для заказного ПО этот этап не всегда актуален, поскольку заказчик может воспользоваться своими эксклюзивными правами на систему и отстранить исполнителя от дальнейших работ по сопровождению и развитию системы ещё до того, как она потеряет актуальность.   
  
Любой проект в конечном счёте приходит к своему завершению. Этап прекращения проекта имеет целью анализ результатов, внесение изменений в процесс разработки на основе полученного опыта и пополнение базы знаний разработчиков новыми эффективными решениями и предостережениями, а также новыми готовыми компонентами, которые можно будет использовать в следующих проектах.  
  
Осталось отметить ещё два этапа процесса разработки. Бывает, что обстоятельства не позволяют продолжать реализацию проекта, но результаты проделанной работы показывают, что у проекта может быть будущее. Закрывать такой проект преждевременно. Поэтому вместо полной остановки работ исполнитель может временно приостановить деятельность по проекту, зафиксировав достигнутые результаты. Как только обстоятельства позволят, проект можно буде возобновить, расконсервировав инфраструктуру, вернув в проект разработчиков и восстановив состояние проекта. Важно, однако, возобновлять работу с того этапа, на котором проект был прерван, повторно проведя ревизию достигнутых результатов.

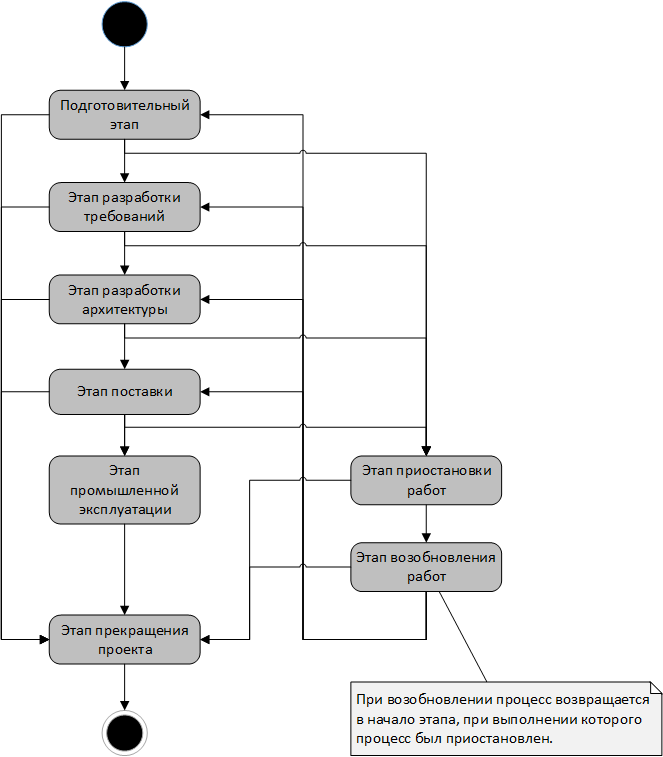
#### Процесс разработки инвестиционного ПО

Процесс разработки инвестиционного ПО отличается тем, что параллельно может идти работа сразу над несколькими версиями продукта: пока первая версия сопровождается, вторая уже реализуется, а для третьей формулируются требования. Процесс показан на рисунке 2.  
  
  
*Рисунок 2. Процесс разработки инвестиционного программного обеспечения.*  
  
Как нетрудно заметить, при разработке инвестиционного ПО имеют место те же этапы, которые были рассмотрены выше для процесса разработки заказного программного обеспечения. Но отличие состоит в том, что этапы относятся не ко всему продукту, а к отдельной версии продукта. Исключение составляет этап прекращения проекта: проект не может завершиться, пока идёт работа хотя бы над одной версией продукта.  
  
Обратите внимание на момент начала работ над следующей версией продукта. Этот момент настаёт, как только пройден этап создания архитектуры текущей разрабатываемой версии. До этого на этапах формирования требований и создания архитектуры, как правило, идёт обсуждение, какие функции следует реализовать в текущей версии, а какие перенести на будущее. И только тогда, когда требования к текущей версии сформулированы, рецензированы и подтверждены архитектурой системы, имеет смысл думать о следующей версии.  
  
Кроме того, после разработки архитектуры, как правило, у аналитиков и архитекторов проекта появляется некоторая свобода действий, поскольку на этапах поставки основная нагрузка ложится на программистов. Эту свободу можно использовать для проработки концепции и требований для следующей версии.  
  
В принципе, можно перенести начало работ над следующей версией на более поздний срок. Например, вполне допустимо сначала ввести текущую версию в опытную или даже промышленную эксплуатацию, и только после этого начать работу над следующей версией. Но нужно помнить, что такое решение неприменимо в случае высокой конкуренции: вас просто опередят и выдавят с рынка. Решение нужно принимать, исходя из всего комплекса обстоятельств, влияющих на ваш бизнес.  
  
Говоря о процессе разработки инвестиционного ПО, нужно понимать, что работа над несколькими версиями имеет ряд явных и скрытых взаимозависимостей между параллельными ветками процесса.   
  
Во-первых, исправления несоответствий, выявленных в ранней версии, должны вноситься и в версию, где они были обнаружены, и во все более поздние версии, включая разрабатываемые. Это касается не только кода программы, но и всех остальных артефактов проекта: технической и пользовательской документации, справочной системы, оценок и планов работ и т.п. Причём исправления должны вноситься немедленно, поскольку уменьшить стоимость исправлений вам не удастся, но, если не внести исправления сразу, их стоимость на более поздних стадиях может увеличиться в десятки и даже сотни раз.   
  
Во-вторых, для параллельной работы над несколькими версиями нужна особая инфраструктура проекта, включая организацию контроля версий кода и документации, контроля заданий и несоответствий, утилит автоматической сборки и тестирования и т.п. Нельзя допустить, чтобы работа над одной версией продукта блокировала выполнение задач по другим версиям только из-за того, что инфраструктура проекта не позволяет запустить два процесса сборки одновременно для разных версий продукта.  
  
Особое внимание нужно уделить стендам, на которых проводится тестирование: на них должны быть развёрнуты все версии продукта, которые были выпущены ранее (по меньшей мере, те версии, которые сопровождаются), и все версии, разработка которых ведётся в настоящий момент.  
  
В-третьих, в работе над несколькими версиями могут быть одновременно задействованы одни и те же участники. Имеется большой риск, что ключевой сотрудник может погрязнуть в работе над одной версией программы и допустить существенное превышение сроков по задачам, связанным с другой версией.  
  
В-четвёртых, имеет место обратная ситуация, когда персонал, работающий над одной версией, ничего не знает о том, какие решения принимаются в рамках работ над другой версией. Частично проблема снимается, если исправления всей документации и кода будут немедленно распространяться на все более поздние версии, о чём я говорил выше. Но одними исправлениями дело не должно ограничиваться. Нужно, чтобы команда, работающая над одной версией, понимала, почему были приняты те или иные решения при работе над другой версией. Для этого нужна база знаний для разработчиков – специальная информационная система, в которой должны описываться все проблемы, с которыми столкнулись разработчики при работе над той или иной версией продукта, и способы решения этих проблем. База знаний должна рассылать всем участникам проекта уведомления о поступлении новых записей. Нельзя пускать на самотёк взаимодействие двух команд, работающих над разными версиями одного продукта.

#### Процесс разработки встроенного ПО

Как уже отмечалось выше, встроенное ПО отличается от заказного тем, что его крайне сложно сопровождать.  
  
Допустим, вы выпускаете программы для холодильников. После того, как ПО поставлено производителю, десятки тысяч устройств начинают расходиться по всему миру, и вы понятия не имеете, где они окажутся. И если один из холодильников выйдет из строя по вине вашего софта, то проще заплатить неустойку, чем возвращать холодильник на завод и проводить диагностику. Конечно, можно подготовить инженеров для дилерских центров, которые смогут провести диагностику на месте и заменить прошивку вашей системы, но это всё равно очень дорого.  
  
Таким образом, при разработке встроенного ПО возникает сразу несколько важных ограничений.   
  
Во-первых, поставка выполняется в рамках только одного этапа: никто не будет встраивать в устройства наполовину работающую программу.  
  
Во-вторых, при поставке вы должны уделить особое внимание качеству программы, поскольку с момента внедрения её внутрь железного ящика менять её будет очень сложно. Особое внимание нужно уделить этапу опытной эксплуатации, когда программа внедряется в ограниченную партию устройств, и эти устройства проходят комплексные испытания в различных режимах эксплуатации. Вы должны собрать максимум информации о динамике поведения вашей системы, проанализировать эту информацию и доработать ПО.  
  
В-третьих, когда устройство с вашим ПО ушло в серию, вы имеете очень мало возможностей для исправления ошибок. По факту, такие исправления возможны только в случае брака ПО, приводящего к неработоспособности всей партии устройств, из-за чего производитель будет вынужден отозвать эту партию, а вы получите большое чёрное пятно на свою репутацию.  
  
Наконец, в-четвёртых, этапа вывода из эксплуатации у встроенного ПО нет. Программу просто выбрасывают вместе с устройством. Поэтому, как только для партии устройств, в которых работает ваше ПО, истекает гарантийный срок, можно переходить к закрытию проекта.  
  
Процесс разработки встроенного ПО показан на рисунке 3.  
  
  
*Рисунок 3. Процесс разработки встроенного программного обеспечения.*

#### Процесс разработки игр

Игровое программное обеспечение было выделено мной по причине специфики их производства и эксплуатации. Бизнес игрового ПО основан на выпуске хитов. Один успешный хит оплачивает расходы на создание нескольких игр, которые остаются незамеченными пользователями. Поэтому процесс разработки одной игры взаимосвязан с процессами разработки других игр.  
  
Ещё одним фактором, выделяющим производство игр, является тот факт, что игра интересна пользователю либо пока он не прошёл последний уровень, либо пока у него не произошла фатальная ошибка. Это значит, что вторую версию игры он не будет покупать или даже бесплатно загружать только ради исправлений нескольких ошибок.  
  
Указанные факторы сказываются на процессе разработки игрового ПО. Процесс представлен на рисунке 4.   
  
  
*Рисунок 4. Процесс разработки игрового программного обеспечения.*  
  
Нужно отметить следующие особенности процесса разработки игрового ПО.  
  
Прежде всего, при производстве игр крайне важно качество концепции. Если концепция игры не позволяет создать хит, то дальнейшая работа бессмысленна. Ситуация, когда большинство проектов заканчиваются на подготовительном этапе, для разработки игрового ПО типична.  
  
При разработке требований и архитектуры для игрового ПО часто повторно используются наработки, полученные при работе над предыдущими проектами. В этом плане также дополнительный вес получает этап прекращения проекта, когда все полезные наработки должны быть зафиксированы в базе знаний разработчиков.  
  
Поставка игрового программного обеспечения происходит в рамках одного единственного этапа. Даже если сначала создаётся некое ядро, «движок» игровой системы, его работу невозможно проверить без реализации всего функционала системы.  
  
Для игрового ПО нет этапов опытной эксплуатации и вывода из эксплуатации. Игры сразу поступают в продажу, а после использования просто удаляются пользователем по мере утраты интереса к ним.

#### Заключение

В рамках статьи я попытался сделать обзор «верхнего уровня» процесса разработки прикладного программного обеспечения. Каждый этап процесса, безусловно, нуждается в отдельном обсуждении с обязательным учётом особенностей разрабатываемых программных средств.   
  
Отмечу, что рассматриваемая здесь схема процесса является результатом обобщения моего личного опыта разработки различных программных средств. Как любое обобщение, моя схема является абстракцией. И, как любая абстракция, у неё есть свои границы применимости. Нельзя бездумно применять эту схему к конкретному проекту. Важно понимать, что каждый проект имеет свои нюансы, влияющие на организацию процесса разработки. И поэтому для каждого проекта приведённую здесь схему нужно адаптировать, а в ряде случаев потребуется разработать принципиально другой подход.