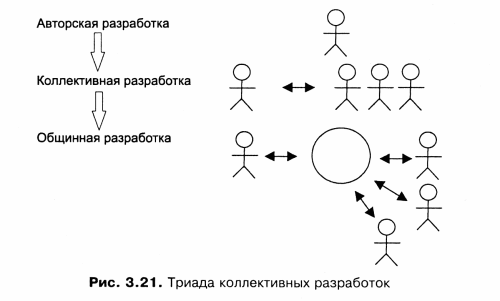
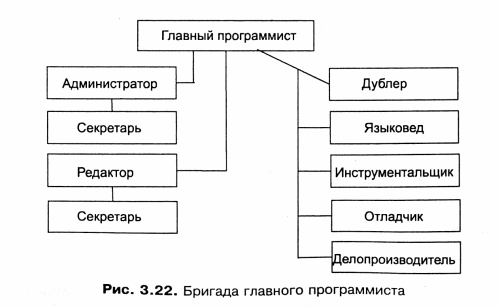
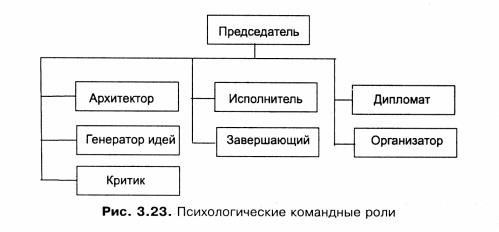
**Технологии коллективной разработки**

* **[1. Авторская разработка](http://oddler.ru/i/242/" \l "242_1)**
* [**2. Коллективная разработка**](http://oddler.ru/i/242/#242_2)
  + [**2.1. Технические командные роли**](http://oddler.ru/i/242/#242_21)
  + [**2.2. Психологические командные роли**](http://oddler.ru/i/242/#242_22)
  + [**2.3. Типы совместной деятельности**](http://oddler.ru/i/242/#242_23)
* [**3. Общинная модель разработки**](http://oddler.ru/i/242/#242_3)
* Все множество разработок в зависимости от количества участников и типов взаимоотношений между ними может быть сведено к триаде разработок, приведенной на рис. 3.21.
* 
* **1. Авторская разработка**
* **Люблю одиночество, даже когда я один.  
  *Ж. Ренар***
* Авторская разработка - принцип создания программных продуктов, при котором весь жизненный цикл разработки поддерживается одним единственным человеком.
* Этот принцип был достаточно широко распространен в 70-80-е годы XX века. Сейчас он применяется редко.
* Наиболее интересен принцип авторской разработки с точки зрения применения в области наукоемких приложений. Для таких приложений характерна необходимость многолетнего изучения предметной области, практически полное отсутствие начального финансирования проекта, малая рентабельность, определяемая узким кругом пользователей.
* Иногда авторская разработка может выигрывать по производительности в тридцать и более раз у коллективной разработки, что достигается за счет:
* исключения межличностных коммуникаций, связанных с необходимостью порождения и изучения большого количества технологической документации;
* исключения работ по разбиению проекта на составляющие, по распределению их между исполнителями, по координации деятельности исполнителей и контролю за их работой.
* Объем программного продукта, выполненного методом авторской разработки, в 5-20 раз меньше по сравнению с индустриальными аналогами.
* Авторская разработка предполагает достижение профессионального успеха, известности и славы в одиночку. Такое вполне реально, следует только правильно выбрать профессиональную "нишу", область ведения разработки.
* **2. Коллективная разработка**
* **Собрать стадо из баранов легко, трудно собрать стадо из кошек.  
  *С. П. Капица***
* Одним из основных вопросов коллективной разработки является разделение труда - от равноправных соисполнителей до организации в виде жесткой иерархии (например, бригады главного программиста).

***О первых опытах коллективных разработок*  
Известно, что первые коллективные разработки программ велись примерно так. Начальник выполнял разделение большого проекта на меньшие части и передавал далее по иерархии. Через некоторое время, теперь уже снизу вверх, шла сборка программы из написанных фрагментов. Собрать работающий программный продукт удавалось не всегда.**

* **2.1. Технические командные роли**
* **Равноправные соисполнители**
* Бригада равноправных соисполнителей обычно состоит из специалистов, занимающихся примерно подобными задачами в рамках одного проекта. Естественно, специализаций в рамках одной бригады может быть несколько:
* инженеры-разработчики (специалисты по инженерии программирования и программисты);
* технические писатели;
* инженеры тестирования;
* инженеры качества;
* специалисты по сопровождению продукта;
* специалисты по продажам продукта.
* Тип работы определяет содержание и природу выполняемой работы. Приведем список типов работ и областей специализации на основе классификации Конгер [Conger 1994].
* Разработка приложений.
  + Программист.
  + Специалист по инженерии программирования.
  + Специалист по инженерии знаний.
* Работа с приложениями.
  + Специалист по приложениям.
  + Администратор данных.
  + Администратор базы данных.
* Техническая поддержка.
  + Системный администратор.
  + Сетевой администратор.
  + Администратор коммуникаций,
* Обеспечение качества продукта.
  + Технический писатель.
  + Инженер тестирования.
  + Инженер качества.
* Маркетинг.
  + Специалист по сопровождению продукта.
  + Специалист по продажам продукта.
* Системное интегрирование.
  + Системный интегратор.
* Из перечисленных специализаций очень интересна специализация системного интегратора. Основные задачи системного интегратора - предложить заказчику вариант решения его проблемы, выбрав наиболее приемлемый по цене и технике, и реализовать его. Таким образом, системный интегратор продает решения и несет ответственность за их реализацию. Системный интегратор как профессионал должен обладать знаниями из очень многих областей - прикладное и системное программное обеспечение, администрирование систем, аппаратура, сети, экономика и т. п.
* **Бригада главного программиста**
* **- Почему бригада скорой помощи состоит из двух врачей?  
  - Один знает - куда ставить клизму, а другой - зачем!  
  *Анекдот о специализации в команде***
* Харлан Миллз [Брукс 1999] предложил организовывать команды (бригады) главного программиста (chief programmer teams), подобные хирургическим бригадам. Лишь один участник команды занимается основной работой, остальные оказывают ему всевозможную поддержку. Бригада главного программиста включает десять человек, выполняющих специализированные роли в команде (рис. 3.22).
* 
* Основные члены бригады выполняют функции, перечисленные ниже.
* Главный программист. Лично выполняет анализ и проектирование, создание и отладку кода, написание документации. Должен обладать талантом, большим опытом работы и существенными знаниями.
* Дублер. Может выполнять любую работу главного программиста, но менее опытен. Подстраховывает главного программиста, может заниматься написанием кода, но не несет ответственности за проект.
* Администратор, он же - менеджер. Под его контролем - деньги, люди, помещения, машинные ресурсы, контакты с другими группами и руководством.
* Редактор. Фактически, это технический писатель. Его задача - критически переработать черновики документации (созданные главным программистом), снабдить их ссылками и библиографией и обеспечить публикацию или помещение в Интернете.
* Языковед. Эксперт в тонкостях языков программирования. Может найти эффективные способы использования языка для решения сложных задач. Обычно работает с несколькими бригадами.
* Инструментальщик. Разработчик специализированных инструментов - утилит и скриптов. Поддерживает основной инструментарий и оказывает по нему консультации. При необходимости может осуществлять администрирование операционной системы.
* Отладчик. Разработчик тестов и организатор тестирования программного продукта.
* Делопроизводитель. Отвечает за регистрацию всех технических данных бригады в библиотеке программного продукта. Благодаря делопроизводителю, активные программисты освобождались от рутинных работ. Заметим, что в настоящее время функции делопроизводителя автоматизированы и переданы репозиторию проекта.
* **2.2. Психологические командные роли**
* Роб Томсет (Rob Thomsett) [Thomsett 1990] предложил восемь ключевых ролей в проекте (рис. 3.23).
* 
* Председатель. Выбирает путь, по которому команда движется вперед к общим целям. Умеет обнаружить сильные и слабые стороны команды и обеспечить наибольшее применение потенциала каждого ее участника.
* Архитектор. Он же оформитель. Придает законченную форму действиям команды. Имеет четкое представление о проблемах и их возможных решениях.
* Генератор идей. Предлагает радикально новые идеи и стратегии, новые подходы к решению проблем, с которыми сталкивается группа. Особое внимание уделяет главным проблемам.
* Критик. Он же скептик, оценивающий проблемы с прагматической точки зрения. Ищет недостатки, изъяны и недоделки. Компенсирует оптимизм генератора идей.
* Исполнитель. Работник, собственно занимающийся написанием кода. Как правило, он не обладает широтой кругозора.
* Завершающий. Поддерживает в команде настойчивость в достижении цели. Играет доминирующую роль на завершающих стадиях разработки.
* Дипломат. Поддерживает силу духа в участниках проекта. Оказывает им помощь в трудных положениях. Пытается улучшить взаимоотношения в команде.
* Организатор. Обнаруживает и сообщает о новых идеях, разработках и ресурсах. Имеет много друзей и связей в своей организации, с помощью которых можно выпросить или одолжить необходимые ресурсы.
* В реальных командах программистов могут быть выделены не все из этих ролей. Роль исполнителя часто берут на себя сразу несколько членов команды.
* **2.3. Типы совместной деятельности**
* Коллективная разработка предполагает большое количество различных действий, причем степень совместной деятельности может существенно изменяться от одного действия к другому. Можно выделить четыре типа совместной деятельности [Robillard, Robillard 2000].
* Мандатная деятельность, обычно представленная формальными собраниями, проводимыми на регулярной основе. Обычно собрания планируются заранее, а присутствие на них обязательно. Статистика показывает, что программисты проводят около 4% своего рабочего времени на собраниях.
* Созываемая деятельность, которая имеет место в случае решения двух или более программистов собраться вместе для решения некоторого технического вопроса. Такие собрания обычно не планируются заранее и в них участвуют только действительно заинтересованные в решении проблемы программисты. На эту деятельность уходит около 14% рабочего времени.
* Естественная совместная деятельность имеет место, когда как минимум двое программистов работают над одной и той же задачей одновременно и обмениваются информацией о выполняемой работе. Эта деятельность занимает около 41% рабочего времени.
* Индивидуальная деятельность имеет место, когда программист работает над задачей, которая не выполняется в то же самое время никаким другим программистом и поэтому маловероятно его взаимодействие по этому предмету с любыми другими программистами группы. Эта деятельность занимает также около 41% рабочего времени.
* **3. Общинная модель разработки**
* **Совершенство в проекте достигается не тогда, когда нечего добавить,  
  а тогда, когда нечего убрать.  
  Антуан де Сент-Экзюпери**
* Идеология общинной ("базарной") модели разработки сформулирована в программной статье Эрика Раймонда (Eric Raymond) "Собор и Базар". Общинная модель характеризуется тремя основными факторами.
* Децентролизованность разработки. Не существует ограничения сверху на количество людей, принимающих участие в проекте. Как правило, разработки такого типа ведутся на базе сети Интернет и могут включать любого заинтересованного разработчика Сети.
* Разработка ведется на базе открытых исходных текстов. По ним можно разобраться с сутью задачи и в любой момент подключиться к разработке.
* Большое количество внешних тестеров (бета-тестеров), позволяющих быстро обнаруживать ошибки и проблемы в программе.
* Эрик Рэймонд сформулировал несколько уроков, которые позволяют лучше понять особенности общинной разработки.
* Каждая хорошая программа начинается с энтузиазма разработчика.
* Хорошие программисты знают, что можно написать, а великие - что можно переписать.
* При правильном отношении интересная проблема найдет вас сама.
* Когда вы теряете интерес к программе, ваша последняя обязанность - передать ее компетентному преемнику.
* Следует выпускать ранние и частые версии программ.
* Обнаружить проблему и исправить ее могут разные люди.
* Иногда использовать идеи пользователей лучше, чем свои идеи.