

Методологии процесса разработки программного обеспечения:  
Водопадная модель, спиральная модель, итеративная модель (agile, scrum, xp), RUP, MSF.

Урок 3



## ВЫУЧИТЬ:

Контроль качества - это совокупность действий, проводимых над продуктом в процессе разработки, для получения информации о его актуальном состоянии в разрезах: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта"

Верификация (verification) - это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

Валидация (validation) - это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

Баг (bug) – ошибка в программе или системе, которая выдаёт неожиданный или не правильный результат.



# Вспомним:

## Фазы процесса разработки программного обеспечения.

Существуют 6 основных фаз разработки ПО :

- Анализ требований (requirements analysis). На этом этапе мы анализируем, что хочет заказчик, чтобы на этапе проектирования иметь четкое представление о будущем продукте.
- Проектирование (design). На этом этапе команда разработчиков должна написать, как будем реализовывать поставленную заказчиком задачу, что будем делать, какие методы будем использовать, на каком языке программирования будем писать программу.
- Реализация (coding). После того как все требования зафиксированы и решения приняты на этапе (coding) программисты пишут код.
- Тестирование и отладка (testing and debug). После того как часть кода была написана, тестировщики начинают тестировать написанную программистами программу, сопоставляют требования, заводят баги, отправляют их назад на исправление (fixing) программистам. Программисты исправляют баги, тестировщики опять проверяют.
- Внедрение (deployment). После того как отладка завершена мы можем выпустить часть этой программы, т.е. внедрить программу, если конечно этот продукт делает, то что он должен делать.
- Сопровождение (support). На этом этапе пользователи могут обращаться в службу поддержки сайта или программы с просьбами, например: добавить новые фичи (функциональности), либо же исправить то, что по их мнению должно работать не так и т.д. Команда на такие обращения реагирует и сопровождает ПО.



## План

1. Введение.
2. Водопадная модель.
3. V- образная модель.
4. Спиральная модель.
5. Итеративная модель (agile, scrum, xp)
6. RUP, MSF.



# Введение.

Методология — это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения.

Модель жизненного цикла программного обеспечения — это структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

Все модели можно разделить на 3 основных группы:

1. Инженерный подход
2. С учетом специфики задачи
3. Современные технологии быстрой разработки.


# Водопадная модель.

Водопадная модель (WATERFALL MODEL)— модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.


В водопадной модели , фазы разработки ПО идут в таком порядке:

1. Определение требований
2. Анализ
3. Проектирование
4. Кодирование
5. Тестирование и отладка
6. Эксплуатация
7. Поддержка





Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «определение требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того как реализация и интеграция завершены, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок.



Переход от одной фазы разработки к другой происходит только после полного и успешного завершения предыдущей фазы, и что переходов назад либо вперёд или перекрытия фаз — не происходит.

Водопадная модель- это тот процесс, который можно сертифицировать. Используют эту модель для разработки ПО, которое используется в работе медицинского оборудования, космических кораблей и т.д.


Одном из главных недостатков водопадной модели является «предрасположенность» к возможным несоответствиям полученного в результате продукта и требований, которые к нему предъявлялись. Основная причина этого заключается в том, что полностью сформированный продукт появляется лишь на поздних этапах разработки, но так как работу на разных этапах обычно выполняли различные специалисты и проект передавался от одной группы к другой, то по принципу испорченного телефона оказывалось так, что на выходе получалось не совсем то, что предполагалось вначале.



# V- образная модель.


V-образная модель. Была предложена для того, чтобы устранить недостатки каскадной модели, а название – V-образная, или шарнирная – появилось из-за ее специфического графического представления (рис. 2).





V-образная модель дала возможность значительно повысить качество ПО за счет своей ориентации на тестирование. Эта модель во многом разрешила проблему соответствия созданного продукта выдвигаемым требованиям благодаря процедурам верификации и аттестации на ранних стадиях разработки (пунктирные линии на рисунке указывают на зависимость этапов планирования/постановки задачи и тестирования/приемки).

Однако в целом V-образная модель является всего лишь модификацией каскадной и обладает многими ее недостатками. В частности, и та и другая слабо приспособлены к возможным изменениям требований заказчика. Если процесс разработки занимает продолжительное время (иногда до нескольких лет), то полученный в результате продукт может оказаться фактически ненужным заказчику, поскольку его потребности существенно изменились.



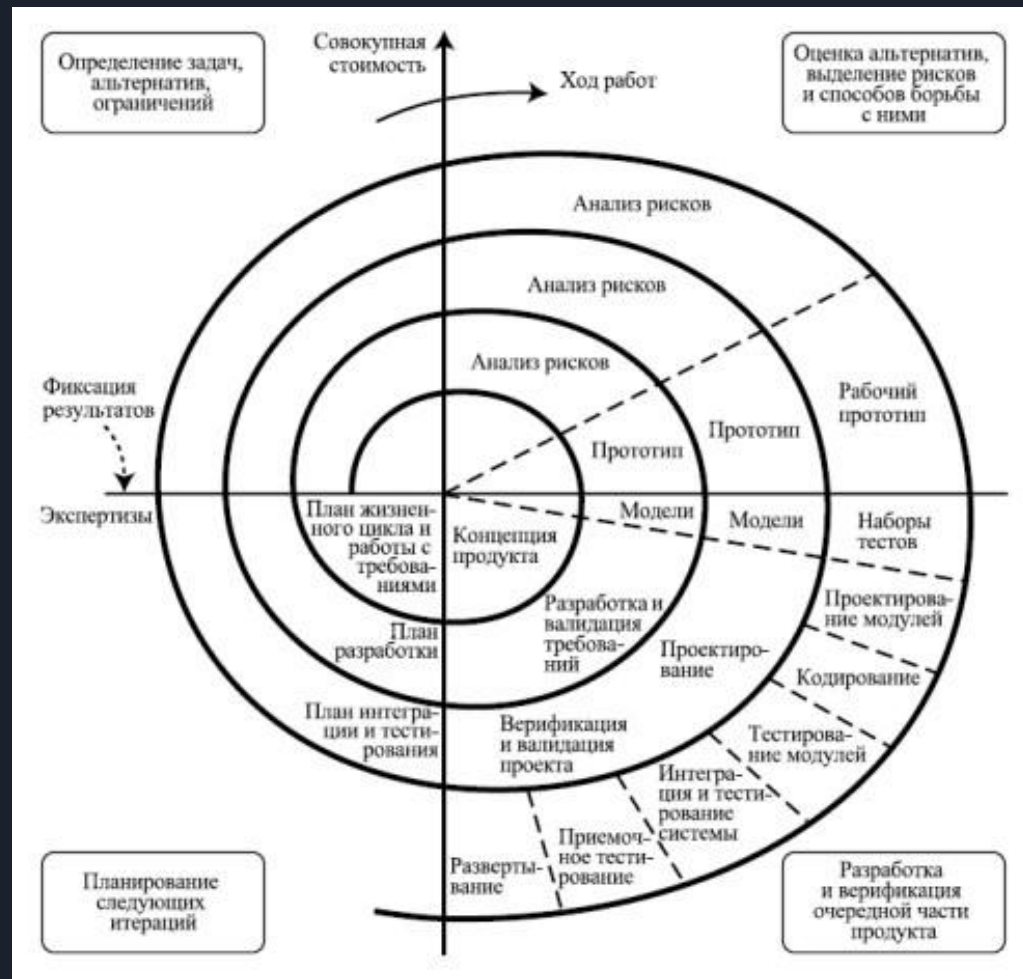
В той же мере актуален и вопрос влияния научно-технического прогресса: требования к ПО выдвигаются с учетом текущего состояния научных и практических достижений в области аппаратно-программного обеспечения, однако IT-сфера развивается очень быстро, и затянувшийся процесс разработки способен привести к созданию продукта, который базируется на устаревших технологиях и оказывается неконкурентоспособным еще до своего появления.


Важен также вопрос планирования показателей ожидаемой функциональности, поскольку в этих моделях он является не более чем допущением: в частности, определить, какую скорость обработки данных обеспечит создаваемый продукт либо сколько он будет занимать памяти, на этапе постановки задачи практически невозможно. Если подобные требования четко зафиксированы в условиях договора между заказчиком и исполнителем, то вполне вероятно, что полученное решение не будет им удовлетворять, хотя известно это станет только на завершающих этапах разработки, когда основные ресурсы уже израсходованы.

В итоге заказчик будет вынужден либо мириться с ограничениями созданного на основе рассмотренных моделей решения, либо дополнительно инвестировать средства, чтобы получить действительно то, что необходимо.

# Спиральная модель

Спиральная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и поэтапное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции.






В квадранте 1 – анализ требований, альтернативных вариантов и ограничений – определяются рабочие характеристики, выполняемые функции, стабильность (возможность внесения изменений), аппаратно/программный интерфейс.

Определяются альтернативные способы реализации системы (разработка, повторное использование компонент, покупка, договор подряда и т.п.). Определяются ограничения, налагаемые на применение альтернативных вариантов (затраты, график выполнения, интерфейс, ограничения среды и др.).

Определяются риски, связанные с недостатком опыта в данной предметной области, применением новой технологии, жесткими графиками, недостаточно хорошо организованными процессами.

В квадранте 2 – оценка альтернативных вариантов, идентификация и разрешение рисков – выполняется оценка альтернативных вариантов, рассмотренных в предыдущем квадранте; оценка возможных вариантов разрешения рисков. Выполняется прототипирование как основа для работ следующего квадранта.




В квадрант 3 – разработка продукта текущего уровня – включаются действия по непосредственной разработке системы или программного продукта: проектирование системы и ее программных компонентов, разработка и тестирование исходных текстов программ, интеграция тестирование и квалификационные испытания продукта или системы и т.п.

В квадранте 4 – планирование следующей фазы – выполняются действия, связанные с разработкой планов проекта, управления конфигурацией, тестирования, установки системы.


Наиболее важным отличием этой модели от других является учет рисков. Риск – это вероятность того, что что-то может пойти не так. Из-за материализации рисков превышаются сроки, и происходит перерасход средств, поэтому необходимо учитывать риски и принимать меры по их смягчению.

Спиральная модель ориентирована на большие, дорогостоящие и сложные проекты. В условиях, когда бизнес цели таких проектов могут измениться, но требуется разработка стабильной архитектуры, удовлетворяющей высоким требованиям по нагрузке и устойчивости, имеет смысл применение Spiral Architecture Driven Development. Данная методология, включающая в себя лучшие идеи спиральной модели и позволяет существенно снизить архитектурные риски, что является немаловажным фактором



Каждый виток спирали соответствует созданию нового фрагмента или версии ИС, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Один виток спирали при этом представляет собой законченный проектный цикл по типу каскадной схемы. Такой подход назывался также «продолжающимся проектированием». Позднее в проектный цикл дополнительно стали включать стадии разработки и апробации прототипа системы.





Итерация - это законченный цикл разработки, приводящий к выпуску внутренней или внешней версии изделия (или подмножества конечного продукта), которое совершенствуется от итерации к итерации, чтобы стать законченной системой.

Каждый виток спирали соответствует созданию фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы на следующем витке спирали. На каждой итерации углубляются и последовательно конкретизируются детали проекта, в результате чего выбирается обоснованный вариант, который доводится до окончательной реализации.

Использование спиральной модели позволяет осуществлять переход на следующий этап выполнения проекта, не дожидаясь полного завершения текущего – недоделанную работу можно будет выполнить на следующей итерации. Главная задача каждой итерации – как можно быстрее создать работоспособный продукт, который можно показать пользователям системы. Таким образом существенно упрощается процесс внесения уточнений и дополнений в проект.



## Инкрементная модель



## Итеративная модель



# Итеративная модель (agile, scrum, xp)

Итеративная модель (iteration — повторение) — выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл: Планирование — Реализация — Проверка — Оценка





## Преимущества итеративного подхода:

- снижение воздействия серьезных рисков на ранних стадиях проекта, что ведет к минимизации затрат на их устранение;
- организация эффективной обратной связи проектной команды с потребителем и создание продукта, реально отвечающего его потребностям;
- акцент усилий на наиболее важные и критичные направления проекта;
- непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом;
- раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта;
- более равномерная загрузка участников проекта;
- эффективное использование накопленного опыта;
- реальная оценка текущего состояния проекта и, как следствие, большая уверенность заказчиков и непосредственных участников в его успешном завершении



# Agile-методы

Гибкая методология разработки (Agile software development) — серия подходов к разработке программного обеспечения, ориентированных на использование интерактивной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование (XP), Scrum.



# XP - Экстремальное программирование

- одна из гибких методологий разработки программного обеспечения.

Основные группы экстремального программирования:

- короткий цикл обратной связи;
- непрерывный процесс;
- понимание, разделяемое всеми;
- социальная защищенность программиста.

XP предполагает написание автоматических тестов.

Особое внимание уделяется двум разновидностям тестирования:


- юнит-тестирование модулей;
- функциональное тестирование.

Одним из приёмов XP является парное программирование. Парное программирование предполагает, что весь код создается парами программистов, работающих за одним компьютером.

Цель методики XP — справиться с постоянно меняющимися требованиями к программному продукту и повысить качество разработки. Поэтому XP хорошо подходит для сложных и неопределенных проектов

Методология XP строится вокруг четырех процессов: кодирования, тестирования, дизайна и слушания. Кроме того, экстремальное программирование имеет ценности: простоту, коммуникацию, обратную связь, смелость и уважение.





## 1. Вся команда

Все участники проекта с применением XP работают как одна команда. В нее обязательно входит представитель заказчика, лучше, если это будет реальный конечный пользователь продукта, разбирающийся в бизнесе. Заказчик выдвигает требования к продукту и расставляет приоритеты в реализации функциональности. Ему могут помогать бизнес-аналитики. Со стороны исполнителей в команду входят разработчики и тестировщики, иногда коуч, направляющий команду, и менеджер, который обеспечивает команду ресурсами.

## 2. Игра в планирование

Планирование в XP проводят в два этапа — планирование релиза и планирование итераций.

На планировании релиза команда программистов встречается с заказчиком, чтобы выяснить, какую функциональность он хочет получить к следующему релизу, то есть через 2-6 месяцев. Так как требования заказчика часто размытые, разработчики конкретизируют их и дробят на части, реализация которых занимает не более одного дня. Важно, чтобы заказчик разбирался в операционной среде, в которой будет работать продукт.

Задачи записываются на карточки, а заказчик определяет их приоритет. Дальше разработчики оценивают, сколько времени уйдет на каждую задачу. Когда задачи описаны и оценены, заказчик просматривает документацию и дает добро на начало работы. Для успеха проекта критично, чтобы заказчик и команда программистов играли на одном поле: заказчик выбирает действительно необходимую функциональность в рамках бюджета, а программисты адекватно сопоставляют требования заказчика со своими возможностями.

В XP если команда не успевает выполнить все задачи к дате релиза, то релиз не отодвигается, а режется часть функционала, наименее важная для заказчика.



### 3. Частые релизы версий

В ХР версии выпускаются часто, но с небольшим функционалом. Во-первых, маленький объем функциональности легко тестировать и сохранять работоспособность всей системы. Во-вторых, каждую итерацию заказчик получает часть функционала, несущую бизнес-ценность.

### 4. Пользовательские тесты

Заказчик сам определяет автоматизированные приемочные тесты, чтобы проверить работоспособность очередной функции продукта. Команда пишет эти тесты и использует их для тестирования готового кода.

### 5. Коллективное владение кодом

В ХР любой разработчик может править любой кусок кода, т.к. код не закреплен за своим автором. Кодом владеет вся команда.

### 6. Непрерывная интеграция кода

Это значит, что новые части кода сразу же встраиваются в систему — команды ХР заливают новый билд каждые несколько часов и чаще. Во-первых, сразу видно, как последние изменения влияют на систему. Если новый кусок кода что-то сломал, то ошибку найти и исправить в разы проще, чем спустя неделю. Во-вторых, команда всегда работает с последней версией системы.


### 7. Стандарты кодирования

Когда кодом владеют все, важно принять единые стандарты оформления, чтобы код выглядел так, как будто он написан одним профессионалом. Можно выработать свои стандарты или принять готовые.

### 8. Метафора системы

Метафора системы — это ее сравнение с чем-то знакомым, чтобы сформировать у команды общее видение. Обычно метафору системы продумывает тот, кто разрабатывает архитектуру и представляет систему целиком.





## 9. Устойчивый темп

XP команды работают на максимуме продуктивности, сохраняя устойчивый темп. При этом экстремальное программирование негативно относится к переработкам и пропагандирует 40-часовую рабочую неделю.

## 10. Разработка, основанная на тестировании

Одна из самых трудных практик методологии. В XP тесты пишутся самими программистами, причем ДО написания кода, который нужно протестировать. При таком подходе каждый кусок функционала будет покрыт тестами на 100%. Когда пара программистов заливают код в репозиторий, сразу запускаются модульные тесты. И ВСЕ они должны сработать. Тогда разработчики будут уверены, что движутся в правильном направлении.

## 11. Парное программирование


Представьте двух разработчиков за одним компьютером, работающих над одним куском функциональности продукта. Это и есть парное программирование, самая спорная практика XP. Старая поговорка «одна голова хорошо, а две лучше» отлично иллюстрирует суть подхода. Из двух вариантов решения проблемы выбирается лучший, код оптимизируется сразу же, ошибки отлавливаются еще до их совершения. В итоге имеем чистый код, в котором хорошо разбираются сразу двое разработчиков.

## 12. Простой дизайн

Простой дизайн в XP означает делать только то, что нужно сейчас, не пытаясь угадать будущую функциональность. Простой дизайн и непрерывный рефакторинг дают синергетический эффект — когда код простой, его легко оптимизировать.


## 13. Рефакторинг

Рефакторинг — это процесс постоянного улучшения дизайна системы, чтобы привести его в соответствие новым требованиям. Рефакторинг включает удаление дублей кода, повышение связности и снижение сопряжения. XP предполагает постоянные рефакторинги, поэтому дизайн кода всегда остается простым.



Преимущества экстремального программирования имеют смысл, когда команда полноценно использует хотя бы одну из практик XP. Итак, ради чего стоит стараться:


- заказчик получает именно тот продукт, который ему нужен, даже если в начале разработки сам точно не представляет его конечный вид
- команда быстро вносит изменения в код и добавляет новую функциональность за счет простого дизайна кода, частого планирования и релизов
- код всегда работает за счет постоянного тестирования и непрерывной интеграции
- команда легко поддерживает код, т.к. он написан по единому стандарту и постоянно рефакторится
- быстрый темп разработки за счет парного программирования, отсутствия переработок, присутствия заказчика в команде
- высокое качество кода
- снижаются риски, связанные с разработкой, т.к. ответственность за проект распределяется равномерно и уход/приход члена команды не разрушит процесс
- затраты на разработку ниже, т.к. команда ориентирована на код, а не на документацию и собрания



Несмотря на все плюсы, XP не всегда работает и имеет ряд слабых мест.

Итак, экстремальное программирование — недостатки:

- Успех проекта зависит от вовлеченности заказчика, которой не так просто добиться
- трудно предугадать затраты времени на проект, т.к. в начале никто не знает полного списка требований
- успех XP сильно зависит от уровня программистов, методология работает только с senior специалистами
- менеджмент негативно относится к парному программированию, не понимая, почему он должен оплачивать двух программистов вместо одного
- регулярные встречи с программистами дорого обходятся заказчиком
- требует слишком сильных культурных изменений, чтобы не контролировать каждую задачу
- из-за недостатка структуры и документации не подходит для крупных проектов
- т.к. гибкие методологии функционально-ориентированные, нефункциональные требования к качеству продукта сложно описать в виде пользовательских историй.



# Скрам (Scrum)

— это ещё одна из гибких методологий разработки ПО. Характеризуется набором принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО, с новыми возможностями, для которых определён наибольший приоритет. Основные роли в методологии Scrum:

Скрам мастер- проводит совещания, следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов.

Владелец продукта- представляет интересы конечных пользователей и других заинтересованных в продукте сторон.

Скрам команда- команда разработчиков проекта, состоящая из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т.д. Размер команды в идеале составляет 7-9 человек.



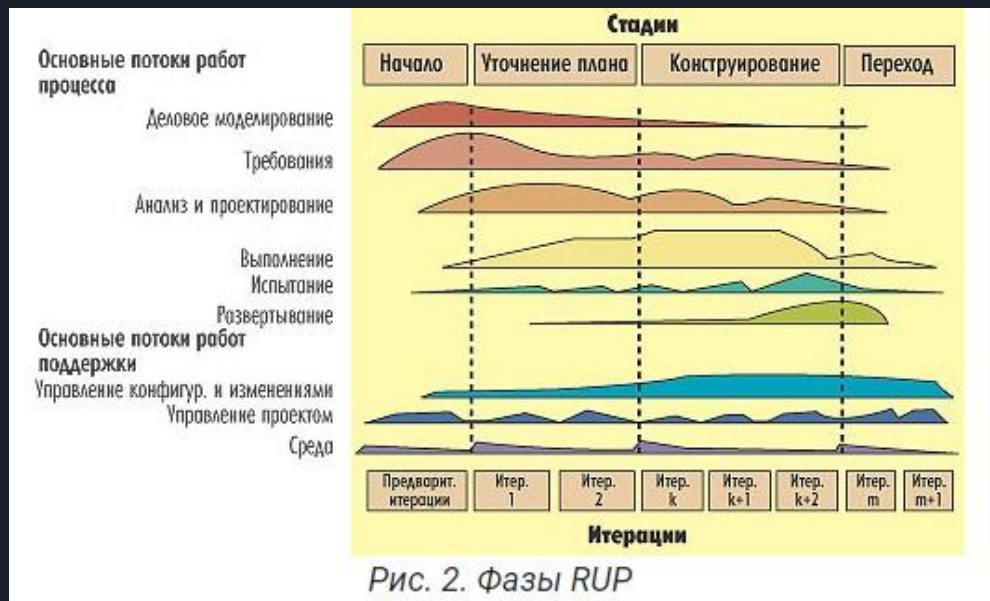
Спринт — итерация в скраме, в ходе которой создаётся функциональный рост программного обеспечения. Жёстко фиксирован по времени. Длительность одного спринта от 2 до 4 недель. Бэклог проекта — это список требований к функциональности, упорядоченный по их степени важности, подлежащих реализации. Элементы этого списка называются «пожеланиями пользователя» (user story) или элементами бэклога.


Диаграмма сгорания задач (Burndown chart)- это диаграмма, показывающая количество сделанной и оставшейся работы. Обновляется ежедневно с тем,

# RUP - RATIONAL UNIFIED PROCESS — методология разработки программного обеспечения, созданная компанией Rational Software.

В основе методологии лежат 6 основных принципов:

- компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта;
- работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам;
- ранняя идентификация и непрерывное устранение возможных рисков;
- концентрация на выполнении требований заказчиков к исполняемой программе;
- ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки;
- постоянное обеспечение качества на всех этапах разработки проекта





Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. Можно по окончании каждого этапа и каждой итерации создавать все требуемые документы и достигнуть максимального уровня формализации, а можно создавать только необходимые для работы документы.

За счет такого подхода к формализации процессов методология является достаточно гибкой и широко популярной. Данная методология применима как в небольших и быстрых проектах, где за счет отсутствия формализации требуется сократить время выполнения проекта и расходы, так и в больших и сложных проектах, где требуется высокий уровень формализма, например, с целью дальнейшей сертификации продукта. Это преимущество дает возможность использовать одну и ту же команду разработчиков для реализации различных по объему и требованиям.



# MICROSOFT SOLUTIONS FRAMEWORK — методология разработки программного обеспечения, предложенная корпорацией Microsoft

MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.


Базовые концепции и принципы модели процессов MSF:

- единое видение проекта — все заинтересованные лица и просто участники проекта должны чётко представлять конечный результат, всем должна быть понятна цель проекта;
- управление компромиссами — поиск компромиссов между ресурсами проекта, календарным графиком и реализуемыми возможностями;
- гибкость – готовность к изменяющимся проектным условиям;
- концентрация на бизнес-приоритетах — сосредоточенность на той отдаче и выгоде, которую ожидает получить потребитель решения;
- поощрение свободного общения внутри проекта;
- создание базовых версии — фиксация состояния любого проектного артефакта, в том числе программного кода, плана проекта, руководства пользователя, настройки серверов и последующее эффективное управление изменениями, аналитика проекта.



MSF предлагает проверенные методики для планирования, проектирования, разработки и внедрения успешных IT-решений. Благодаря своей гибкости, масштабируемости и отсутствию жестких инструкций MSF способен удовлетворить нужды организации или проектной группы любого размера. Методология MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов.





## 1. Выработка концепции (Visioning)

Команда вырабатывает единое видение проекта или его части. Совместными усилиями коллеги решают, какая именно функциональность будет разрабатываться в ходе итерации, определяют основные концепции, которые лягут в основу разработки. Этап завершается вехой «Концепция утверждена».

## 2. Планирование (Planning)

Задачи, которые необходимо выполнить в ходе итерации, разбиваются на подзадачи, определяется сложность их реализации, устанавливаются сроки и назначаются ответственные. В планы закладывается время на тестирование и исправление дефектов, а также предварительно намечается, как именно будет проходить тестирование.

## 3. Разработка (Developing)

На данном этапе MSF создается программный код новой функциональности в соответствии с концепцией и утвержденными планами.

## 4. Стабилизация (Stabilizing)

К делу подключаются тестировщики. После тестирования выявленные баги и недочеты возвращаются разработчикам для исправления.

## 5. Внедрение (Deploying)

Очередной релиз программного продукта передается заказчику и устанавливается на клиентских компьютерах.

В конце каждой итерации клиент должен получить работоспособную версию приложения. По завершении этапа внедрения и итерации в целом немедленно начинается новая итерация.



Существует множество различных методологий разработки программного обеспечения, они не универсальны и описываются различными принципами. Выбор методологии разработки для конкретного проекта зависит от предъявляемых требований.