ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ЛЕКЦИЯ № 1

ПРЕПОДАВАТЕЛЬ: ХУСТОЧКА А.В.

МОДЕЛИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- Процесс жизни любой системы или программного продукта может быть описан посредством модели жизненного цикла, состоящей из стадий.
- Модели могут использоваться для представления всего жизненного цикла от замысла до прекращения применения или для представления части жизненного цикла, соответствующей текущему проекту.
- Модель жизненного цикла представляется в виде последовательности стадий, которые могут перекрываться и (или) повторяться циклически в соответствии с областью применения, размером, сложностью, потребностью в изменениях и возможностях. Каждая стадия описывается формулировкой цели и выходов.
- Процессы и действия жизненного цикла отбираются и исполняются на этих стадиях для полного удовлетворения цели и результатам каждой стадии. Различные организации могут использовать различные стадии в пределах жизненного цикла.
- Однако каждая стадия реализуется организацией, ответственной за эту стадию, с надлежащим рассмотрением информации, имеющейся в планах жизненного цикла и решениях, принятых на предшествующих стадиях. Аналогичным образом организация, ответственная за текущую стадию, ведет записи принятых решений и записи допущений, относящихся к последующим стадиям данного жизненного цикла

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- Под моделью жизненного цикла ПО понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Модель ЖЦ зависит от спецификации, масштаба и сложности проекта и спецификации условий, в которых система создается и функционирует.
- Модель ЖЦ ПО включает в себя: стадии, результаты выполнения работ на каждой стадии, ключевые события – точки завершения работ и принятия решений. Модель ЖЦ любого конкретного ПО определяет характер процесса его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям.
- Под стадией понимается часть процесса создания ПО, ограниченная определенными временными рамками и заканчивающаяся выпуском конкретного продукта (моделей, программных компонентов, документации), определяемого заданными для данной стадии требованиями.

КАСКАДНАЯ МОДЕЛЬ

Особенности:

- **Последовательность действий**. В каскадной модели проекта все этапы идут друг за другом: на следующий этап проекта переходят только после того, как сделаны все работы на предыдущем. После завершения этапа вернуться к нему нельзя. Например, при строительстве дома не получится переделать фундамент, если в нем нашли проблемы на стадии возведения стен и крыши. Поэтому этот подход сравнивают с каскадом и иногда называют водопадной моделью или waterfall-методологией. Так как вернуться на предыдущую фазу проекта невозможно, перед переходом на следующий этапа результат должен пройти проверку и приемку.
- **Регламентация процесса**. Все планы, требования и задачи проекта описывают в документах. Все участники следуют формальным правилам и не могут их менять во время работы. Так как нельзя вернуться к предыдущему этапу, требования к проекту после утверждения не меняются.

КАСКАДНАЯ МОДЕЛЬ

ПРЕИМУЩЕСТВА

- **Проект не зависит от конкретных исполнителей**. Все процессы зарегламентированы и описаны. Поэтому в течение жизненного цикла проекта члены команды могут приходить и уходить без вреда для сроков и качества работ.
- **Исполнители работают по четкому плану**. Участники знают свои задачи, в какой последовательности их выполнять и когда сдавать работу. Это делает ход проекта предсказуемым.
- **Сроки и бюджет зафиксированы**. Стоимость и длительность проекта заранее рассчитывают и утверждают, а в ходе работы их не меняют.
- **Требования не меняются во время работы**. Так как нельзя вернуться к предыдущему этапу, требования к проекту после утверждения не меняются. Но если до начала работ у заказчика изменилась ситуация, то есть время, чтобы пересмотреть концепцию проекта и изменить требования.

НЕДОСТАТКИ

- **Проект сложно адаптировать под изменения среды**. Проект начинается с плана, в котором пытаются учесть все возможные события. Но заранее предугадать все проблемы невозможно из-за высокой неопределенности, поэтому многие решения будут ошибочными, а менять проект нельзя. Например, заказчик утвердил высотные жилые комплексы, но за время строительства покупатели захотели малоэтажные кварталы. Отменить стройку или переделать проект не получится.
- **Проект растягивается во времени**. Работы идут строго последовательно, поэтому исполнители на следующих этапах не могут начать работу, пока на предыдущем не выполнят все задачи. Чем дольше идет проект, тем быстрее он устаревает.
- **Поздно находят проблемы**. Тестирование – один из последних этапов, на котором ищут все ошибки проекта, не только изготовления. Из-за последовательной работы фундаментальные проблемы проекта находят слишком поздно. На их исправление не хватит времени и бюджета. Остается «сглаживать углы»: заделывать дыры в доме монтажной пеной, исправлять простые баги, заклеивать щели в ракете монтажной лентой.
- **Заказчик поздно дает обратную связь**. Заказчик видит результат в конце проекта и если у него изменились требования или условия, то исполнители поздно об этом узнают. Новые требования приводят к новому проекту.

КАСКАДНАЯ МОДЕЛЬ С ОБРАТНЫМИ СВЯЗЯМИ

- Чтобы не находить ошибки слишком поздно и адаптировать проект под изменения обстоятельств, каскадной модели добавили несколько элементов гибких подходов. Такую методологию называют гибридной.
- Каскадная модель с обратными связями. Обратные связи добавили, чтобы решить проблему позднего тестирования. Они срабатывают, когда во время работ находят ошибки, что позволяет их исправлять, не дожидаясь проверки. Но в этом случае сложнее планировать проект, распределять бюджет и укладываться в срок.

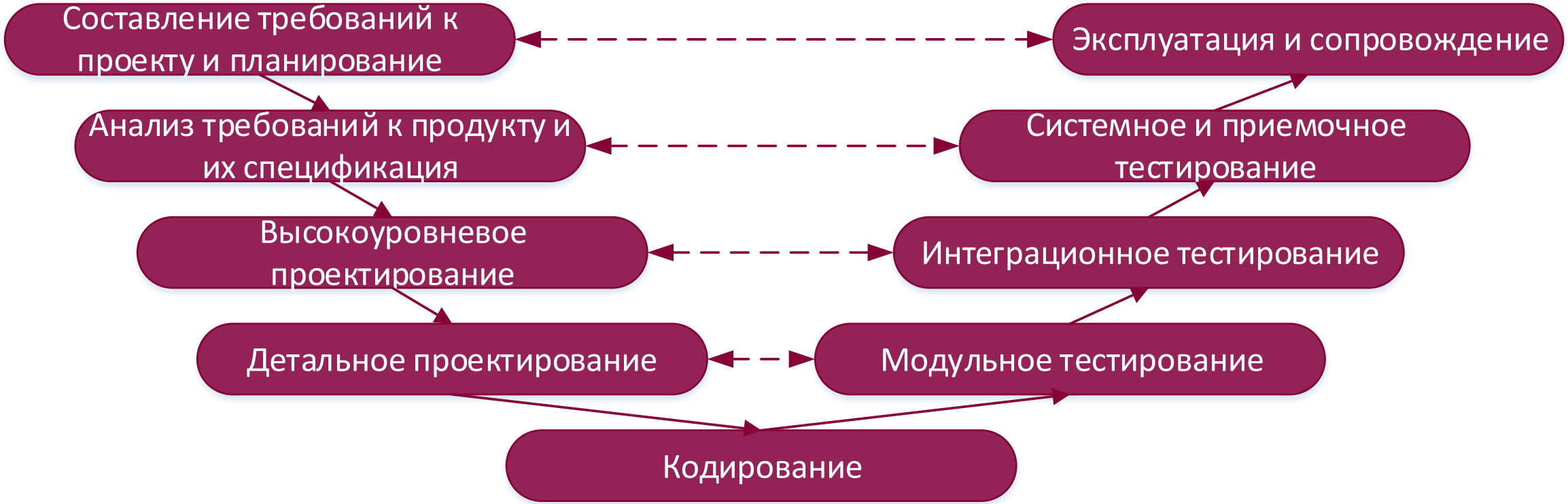
ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Когда использовать каскадную методологию?

- Только тогда, когда требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
- Нет проблем с доступностью программистов нужной квалификации.
- В относительно небольших проектах.

V-ОБРАЗНАЯ МОДЕЛЬ

- Это усовершенствованная каскадная модель, в которой заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе.
- История этой модели начинается в 1980-х.



ПРЕИМУЩЕСТВА

- **Особое значение придается верификации и аттестации** программного продукта, планированию всех действий, начиная с ранних стадий его разработки;
- Предполагается **аттестация и верификация** не только самого программного продукта, но и **всех полученных внутренних и внешних данных**;
- **Ход выполнения работы может легко отслеживаться**, так как завершение каждой фазы является контрольной точкой.

НЕДОСТАТКИ

- **В модели не предусмотрено внесение требования динамических изменений** на разных этапах жизненного цикла;
- **Тестирование требований в жизненном цикле происходит слишком поздно,** вследствие чего невозможно внести изменения, не повлияв при этом на график выполнения проекта;
- В модель **не входят действия, направленные на анализ и управление рисками**.

ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Когда использовать V-модель?

- Если требуется тщательное тестирование продукта, то V-модель оправдает заложенную в себя идею: validation and verification.
- Для малых и средних проектов, где требования четко определены и фиксированы.
- В условиях доступности инженеров необходимой квалификации, особенно тестировщиков.

ИНКРЕМЕНТНАЯ МОДЕЛЬ

- Идея, лежащая в основе инкрементной модели, состоит в том, что программную систему следует разрабатывать по принципу приращений, так, чтобы разработчик мог использовать данные, полученные при разработке более ранних версий ПО. Новые данные получаются как в ходе разработки ПО, так и в ходе его использования, где это возможно. Ключевые этапы этого процесса – простая реализация подмножества требований к программе и совершенствование модели в серии последовательных релизов до тех пор, пока не будет реализовано ПО во всей полноте.
- В ходе каждой итерации организация модели изменяется, и к ней добавляются новые функциональные возможности. Для организации инкрементной разработки обычно выбирается характерный временной интервал, например, неделя. Затем в течение этого интервала происходит обновление проекта: добавляется новая документация как текстовая, так и графическая, расширяется набор тестов, добавляются новые программные коды и т. д. Теоретически шаги разработки могут выполняться и параллельно, но такой процесс очень сложно скоординировать.

ИНКРЕМЕНТНАЯ МОДЕЛЬ

Пример создания социальной сети на основе инкрементной модели:

- Реализация страницы с личной информацией каждого пользователя
- Реализация функционала обмена текстовыми сообщениями
- Продукт показывают заказчику и выпускают на рынок. Если всех социальная сеть устраивает – работа над ней продолжается.
- Реализация функционала обмена фотографиями и документами.
- Обновление версий у пользователей
- Реализация функционала для прослушивания музыки
- Обновление версий у пользователей
- … и т.д. до конечного результата.

ПРЕИМУЩЕСТВА

- **Не нужно вкладывать много денег на начальном этапе**. Заказчик оплачивает создание основных функций, получает продукт, «выкатывает» его на рынок — и по итогам обратной связи решает, продолжать ли разработку.
- **Можно быстро получить обратную связь от пользователей** и оперативно **обновить техническое задание**. Так снижается риск создать продукт, который никому не нужен.
- **Ошибка обходится дешевле**. Если при разработке архитектуры была допущена ошибка, то исправить её будет стоить не так дорого, как в «водопаде» или V-образной модели.

НЕДОСТАТКИ

- **Каждая команда программистов разрабатывает свою функциональность и может реализовать интерфейс продукта по-своему**. Чтобы этого не произошло, важно на этапе обсуждения техничего задания объяснить, каким он будет, чтобы у всех участников проекта сложилось единое понимание.
- **Разработчики будут оттягивать доработку основной функциональности и «пилить мелочёвку».** Чтобы этого не случилось, менеджер проекта должен контролировать, чем занимается каждая команда.

ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Когда использовать инкрементную модель?

- Когда основные требования к системе четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени.
- Требуется ранний вывод продукта на рынок.
- Есть несколько рисковых фич или целей.

ИТЕРАТИВНАЯ МОДЕЛЬ

- Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований.
- Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, мы стремимся к ней так, чтобы каждый шаг был результативен, а каждая версия — работоспособна

ИНКРЕМЕНТНАЯ VS ИТЕРАТИВНАЯ

- На диаграмме показана итерационная «разработка» Мона Лизы. Как видно, в первой итерации есть лишь набросок Джоконды, во второй — появляются цвета, а третья итерация добавляет деталей, насыщенности и завершает процесс.
- В инкрементной же модели функционал продукта наращивается по кусочкам, продукт составляется из частей. В отличие от итерационной модели, каждый кусочек представляет собой целостный элемент.

1 2 3

Инкрементная модель

1 2 3

Итеративная модель

ПРЕИМУЩЕСТВА

- **Быстрый выпуск минимального продукта** даёт возможность оперативно получать обратную связь от заказчика и пользователей. А значит, фокусироваться на наиболее важных функциях программного обеспечения и улучшать их в соответствии с требованиями рынка и пожеланиями клиента.
- Постоянное тестирование пользователями позволяет **быстро обнаруживать и устранять ошибки**.

НЕДОСТАТКИ

- **Использование на начальном этапе баз данных или серверов** — первые сложно масштабировать, а вторые не выдерживают нагрузку. Возможно, придётся переписывать большую часть приложения.
- **Отсутствие фиксированного бюджета и сроков**. Заказчик не знает, как выглядит конечная цель и когда закончится разработка.

ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Когда использовать итеративную модель?

- Требования к конечной системе заранее четко определены и понятны.
- Проект большой или очень большой.
- Основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.

СПИРАЛЬНАЯ МОДЕЛЬ

- «Спиральная модель» похожа на инкрементную, но с акцентом на анализ рисков.
- Она хорошо работает для решения критически важных бизнес-задач, когда неудача несовместима с деятельностью компании, в условиях выпуска новых продуктовых линеек, при необходимости научных исследований и практической апробации.

ПРЕИМУЩЕСТВА

- **Большое внимание уделяется проработке рисков**.
- **Ускорение разработки** (ранее получение результата за счет прототипирования)
- Постоянное **участие заказчика в процессе** разработки
- **Разбиение большого объема работы** на небольшие части

НЕДОСТАТКИ

- Основная **проблема** спирального цикла **– определение момента перехода на следующую стадию**
- **Есть риск застрять на начальном этапе** — бесконечно совершенствовать первую версию продукта и не продвинуться к следующим.
- **Разработка длится долго и стоит дорого**.

«AGILE MODEL» (ГИБКАЯ МЕТОДОЛОГИЯ РАЗРАБОТКИ)

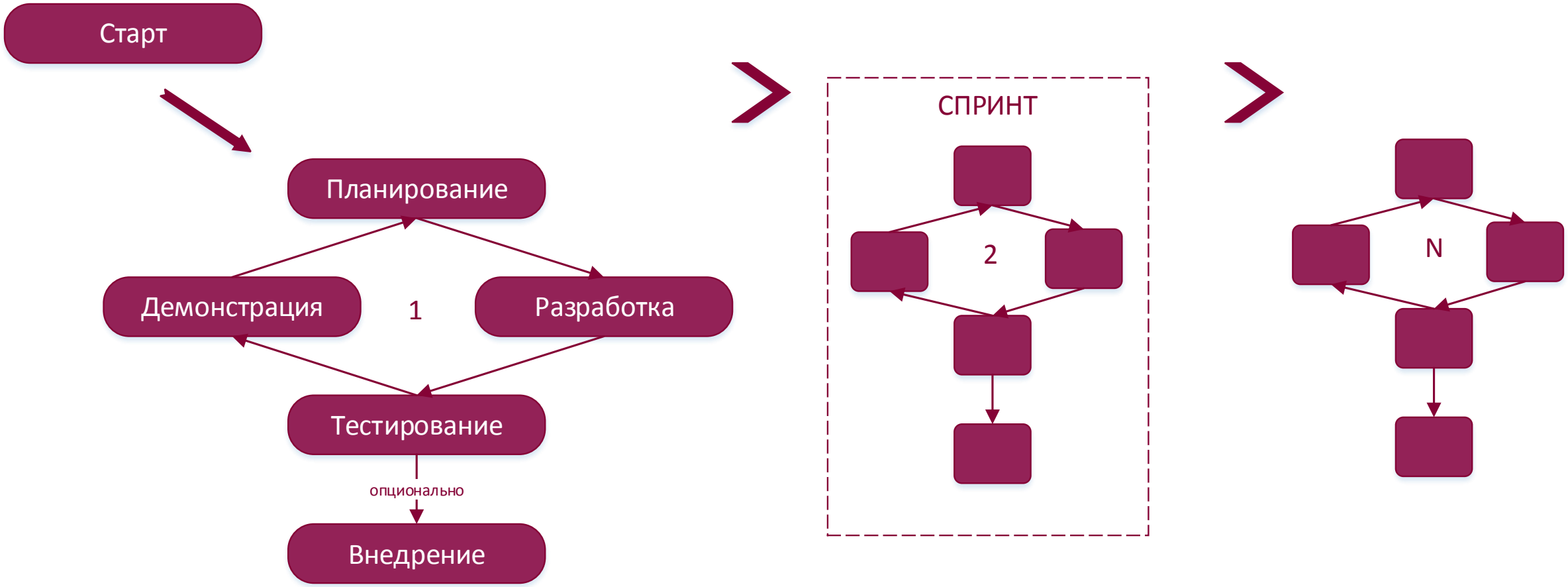
В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет. Это одно из преимуществ гибкой модели. К ее недостаткам относят то, что из-за отсутствия конкретных формулировок результатов сложно оценить трудозатраты и стоимость, требуемые на разработку. Экстремальное программирование (XP) является одним из наиболее известных применений гибкой модели на практике.

В основе такого типа — непродолжительные ежедневные встречи — «Scrum» и регулярно повторяющиеся собрания (раз в неделю, раз в две недели или раз в месяц), которые называются «Sprint». На ежедневных совещаниях участники команды обсуждают:

- отчёт о проделанной работе с момента последнего Scrum'а;
- список задач, которые сотрудник должен выполнить до следующего собрания;
- затруднения, возникшие в ходе работы.

Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка. Соответственно, в процессе реализации требования изменяются.

«AGILE MODEL»


Старт
Планирование
Демонстрация
1
Разработка
Тестирование
опционально
Внедрение
СПРИНТ
2
N

ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Когда использовать Agile?

- Когда потребности пользователей постоянно меняются в динамическом бизнесе.
- Изменения на Agile реализуются за меньшую цену из-за частых инкрементов.
- В отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

ТРАДИЦИОННЫЙ VS ГИБКИЙ ПОДХОДЫ РАЗРАБОТКИ

Характеристика	Традиционные модели	Гибкие модели
Подход	Прогнозирующий	Адаптивный
Критерий успеха	Следование плану	Ценность для бизнеса
Риски	Риски определены	Риски не определены
Контроль	Легко контролировать	Зависит от профессионального уровня специалистов
Заказчики	Низкая вовлечённость	Высокая вовлечённость
Документация	Детальная с начала проекта	Доработка по мере развития проекта
Требования	Известны заранее, стабильны	Не всегда известны заранее, легко изменяемы
Команда проекта	Включение новых специалистов на любом этапе	Опытные специалисты, стабильный состав
Рефакторинг/изменение кода	Дорого	Недорого