1 слайд – Представление

Добрый вечер!

Меня зовут Шарипов Саит - я бакалавр 4 года обучения. Совместно с Полевым Антоном и Смирновым Валерием мы подготовили реферат на тему “Управление проектом разработки ПО” состоящий из 3 основных частей:

- Agile - о котором сегодня расскажу я

- RUP, PMBOK

- ГОСТы  
  
О которых на следующих семинарах Вам расскажут мои товарищи.

2 слайд – Содержание

Итак, сначала мы обсудим кризис ПО – причина из-за которой задумались над планированием управления разработки. Рассмотрим известные методы, зарекомендовавшие себя во времени. Далее обсудим сущность Agile и разберем некоторые из его методологий.

3 слайд – Кризис ПО

В связи со стремительным ростом вычислительной мощности компьютеров увеличился и масштаб задач, решаемых с помощью них. В связи с этим на конференции НАТО «Инженерия программного обеспечения» в 1968 году, Фридрих Бауэр ввел понятие “кризис программного обеспечения”.

Причины кризиса связаны со сложностью разработки ПО и выражены в следующем:

* Стоимость проектов превышает бюджет.
* В проектах превышаются сроки выполнения.
* Проекты неуправляемы
* Итоговое ПО не соответствует поставленным требованиям.

**Фридрих Людвиг Бауэр** – профессор математики и информатики в Мюнхенском техническом университете. Он полагал, что управление проектом разработки ПО позволит решать проблемы, связанные с его высокой стоимостью и сложностью создания. Причем для каждого проекта должна подбираться своя модель разработки, так как единственного решения позволяющего эффективно разрабатывать программное обеспечения нет и построено оно быть не может, так как возникающие при разработке проблемы могут быть самыми разнообразными.

На сегодняшний день есть несколько наиболее устоявшихся методологий разработки программного продукта. Выбор каждой из них зависит от многих факторов, таких как: специфика продукта, бюджет проекта, предпочтения команды разработчиков, представление о результате….

4 слайд – Каскадная модель / Waterfall

В 1970 году в статье *Managing the Development of Large Software Systems* Уинстон Ройс предложил свою модель разработки ПО. Ее интерпретировали как **каскадную модель** или **модель Waterfall**. Сегодня она считается одной из самых известных и базовых.

Данная модель подразумевает последовательное выполнение разработки, начиная с выработки требований и заканчивая сопровождением, с четким определением границ между этапами, на которых результат, полученный на предыдущей стадии, передается в качестве входных данных для следующей.

5 слайд – Итеративная модель

На самом же деле Ройс подразумевал каскадную модель с возможностью возвращения на предыдущий шаг, чтобы при необходимости изменить итоговый результат. Такой подход называется **итеративным**.

6 слайд – Сравнение Waterfall и итеративной модели

На данном слайде представлено сравнение этих моделей, а именно их плюсы и минусы.  
  
Линейность Waterfall позволяет легко управлять проектом, быстро разрабатывать ПО, и заблаговременно прогнозировать затрачиваемые цену и время. Однако главным минусом этого подхода является то, что нельзя вернуться на предыдущий шаг, тестирование продукта осуществляется в самом конце, поэтому в случае неудачи переделывать придется практически все и дорого.  
  
Итеративный подход решает минусы Waterfall, но из-за большей гибкости появляется проблема с прогнозированием затрат и бОльшей сложностью работы.

Обе модели применяются лишь когда требования к результату понятны и подробно описаны. Waterfall для небольших, а итеративный подход для достаточно больших проектов.

7 слайд – Спиральная модель: главная особенность и основные этапы

Развитие идеи итеративной каскадной модели стала спиральная модель. Ее предложил Барри Боэм в 1986 году. Она сочетает в себе идеи итеративного и поэтапного подхода.

Главная особенность спиральной модели – это большое внимание к возможным рискам и их подробный анализ.

**Основные типы рисков**, которые могут возникнуть в процессе разработки ПО:

* Несоответствующий бюджет.
* Несоответствие срокам.
* Несоответствие квалификации/количества специалистов для данной задачи.
* Частые изменения ожиданий от проекта.
* Погоня за несущественной оптимизацией.
* Малая производительность системы.

**Четыре основных этапа спиральной модели:**

1. Планирование.
2. Анализ рисков.
3. Конструирование.
4. Оценка результата (в случае удовлетворительной оценки осуществляется переход к следующему витку спиральной модели).

8 слайд – Спиральная модель: картинка, плюсы и минусы

Каждый следующий виток спирали подразумевает этап создания фрагмента или версии программного обеспечения. На нём реализуются четыре вышеописанных этапа спиральной модели и планируются работы следующего витка спирали. В итоге углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который и будет представлен в реализации.

**Преимущества спиральной модели**: эффективный анализ рисков, хорошая документации разработки на каждом этапе, гибкость подхода и возможность внесения изменений при уточнении требований, создание прототипов результата на каждом этапе.

Однако данный подход имеет свои **минусы:** методология может приводить к чрезмерным расходам при её реализации, требуются высококлассные специалисты для анализа рисков, нецелесообразно использование в небольших проектах.

**В итоге** спиральную модель целесообразно использовать, когда очень важно учесть и минимизировать вероятность всех рисков, для больших проектов, в которых нет четких, подробных требований к итоговому результату, а также при разработки абсолютно новых продуктов на рынке.

9 слайд – Agile

**Agile** – это не какой-либо один подход к разработке ПО. Agile – это методология или семейство подходов управления процесса разработки программного обеспечения.

В переводе с английского agile – гибкий. В гибкости и заключается вся особенность Agile.

В 2001 году был разработан и принят **Agile Manifesto**, который не содержал каки-либо практических советов, а лишь общие идеи – ценности и принципы.  
Далее прочитать их с слайда.

Общая идея Agile методов – это гибкий процесс разработки, решения получаются благодаря сотрудничеству между самоорганизующимися функциональными командами. Это позволяет быстрее получать отдачу, повышая качество и предсказуемость, а также повышая способность команд реагировать на изменения.

10 слайд – Преимущества Agile

1. **Тесное и постоянное взаимодействие заказчика и всех заинтересованных в успехе сторон** – позволяет полностью понять клиента и способствует эффективному взаимодействию внутри команды.
2. **Прозрачность** – благодаря гибкому подходу клиент видит продукт на каждом этапе его разработки.
3. **Совещания по графику** – обычно промежутки между совещаниями от 1 до 4 недели. Это позволяет всегда держать руку на пульсе. В итоге повышается предсказуемость от результата.
4. **Спринты** – благодаря тому, что проект разделен на небольшие спринты, то команде легче сосредоточиться на кратковременной цели, а также провести тестирования той или иной законченной функции.
5. **Ориентированность на ценности бизнеса –** команда лучше понимает клиента и может предложить ему действительно то, что ему нужно.
6. **Внесение изменений –** гибкие подходы позволяют вносить изменение на следующей итерации в результаты, полученные на одном из предыдущих этапов.
7. **Пользователи на 1 месте** – гибкие методы позволяют работать с продуктом итеративно, что дает возможность проводить бета-тестирование и т.п. уже на ранних этапах, что позволяет получить фидбэк от пользователя.
8. **Качество проектов** – то, что проекты разбиты на управляемые единицы позволяет команде сосредоточиться на каждой функции продукта, за счет чего повысить его качество.
9. **Предвиденные расходы** – так как работа разделена на спринты, а совещания проходят по графику, то можно спрогнозировать затраты для создания конечного и принять корректирующие решения еще на раннем этапе.
10. **Единая цель** –гибкие подходы утроены так, что объединяют команду и дают цель каждому её участнику. Это повышает продуктивность и эффективность.

11 слайд – Методологии Agile

Agile – семейство гибких подходов, поэтому существует множество Agile ориентированных методов. Мы разберем некоторые из них. Стоит отметить, что некоторые из этих методов применялись еще до того, как возникло понятие Agile, после Agile Manifesto в 2001, но они также базируются на принципах и ценностях, которых придерживается Agile, поэтому их и относят к Agile методам.

12 слайд – Scrum - суть

Данный метод является самым популярным по результатам исследования Agile Development Survey. Scrum отличается тем, что делает основной упор на качественный контроль рабочего процесса.

Термин **«scrum»** пришёл из регби, где он означает схватку. Проекты, над которыми тесно, будто в схватке работают небольшие команды из специалистов различного профиля, обычно систематически производят лучшие результаты - «регбийный подход».

Scrum состоит из нескольких элементов: роли, артефакты, процессы.

13 слайд – Scrum – роли

Scrum подход подразумевает распределение участников проекта на три группы: Product owner, Scrum Master, Scrum team.

* Product owner – “владелец продукта”, человек, который будет принимать конечное решение относительно продукта. То есть он руководит продуктом в течении проекта, но не командой.
* Scrum Master – работает с командой. Но так как в Scrum методологии команда – это самоорганизующийся объект, то Scrum Master не руководит командой, а скорее выполняет роль администратора для команды.
* Scrum team – обычно команда из 5-9 человек, специалисты разного профиля. Scrum team действует как единое целое, результат её деятельности – конечный продукт.

14 слайд – Scrum – артефакты

* Беклог продукта (Product Backlog) – приоритезированный список требований с оценкой трудозатрат. Обычно он состоит из бизнес-требований, которые приносят конкретную бизнес-ценность и называются элементы беклога.
* Беклог спринта (Sprint Backlog) – часть беклога продукта, с самой высокой важностью и суммарной оценкой, не превышающей скорость команды, отобранная для спринта.
* Инкремент продукта – новая функциональность продукта, созданная во время спринта спринта.

15 слайд – Scrum – процессы

Разработка проекта разделяется на спринты. Для каждого спринта устанавливают четкие временные рамки – от 2 до 4 недель.

1. Изначально есть **идея** для реализации у команды.
2. Далее её сопоставляют с тем, **что желает в конечном итоге клиент**.
3. Составляется **Беклог продукта** – список требований к проекту.
4. Из него выбираются самые важные и актульные темы.
5. Далее на этапе планирование составляется **Беклог Спринта** – список требований к спринту.
6. 2-4 недели длиться спринт.
7. У продукта появляется **новая функциональность**.
8. Результаты спринта **показываются владельцу продукта** с целью получения обратной связи.
9. **Ретроспектива** – обсуждают итоги спринта, чтобы сделать соответствующие выводы.

16 слайд – Scrum плюсы, минусы и итог

Итак, основная идея Scrum – это тесная командная работа в строгих временных рамках спринтах, на которые был поделен весь процесс разработки.

**Плюсы и минусы** прочитай со слайдов – там все логично.

В итоге Scrum – это эффективный способ организации работы, многократно повышающий продуктивность команд, позволяющий сделать продукт соответствующий видению заказчика.

17 слайд – Lean Software Development

Lean в переводе с английского **бережливый**. Соответственно это гибкая структура, основанная на оптимизации времени и ресурсов разработки, устранении потерь и, в конечном итоге, предоставлении только того, что нужно продукту. Подход Lean также часто называют стратегией минимального жизнеспособного продукта, в которой команда выпускает минимальную версию своего продукта на рынок, узнает от пользователей, что им нравится, что не нравится и что они хотят быть, добавляется, а затем повторяется на основе этого отзыва.

Сильные стороны ЛСД включают в себя:

1. Оптимизированный подход позволяет предоставлять больше функциональности за меньшее время.
2. Устраняет ненужную деятельность, и в результате может снизить затраты.
3. Позволяет команде разработчиков принимать решения, что также может повысить моральный дух.

Слабые стороны ЛСД включают в себя:

1. Сильно зависит от вовлеченной команды, что делает ее не такой масштабируемой, как другие структуры. (так как на ней все)
2. Зависит от сильной документации, и неспособность сделать это может привести к ошибкам разработки.

18 слайд – Kanban

**Kanban** – это метод управления созданием продуктов с акцентом на непрерывную доставку, не перегружая команду разработчиков. Как и SCRUM, Kanban призван помочь разработчикам работать более эффективно.

**Kanban основан на 3 основных принципах:**

1. Визуализируйте, что вы делаете сегодня (рабочий процесс): просмотр всех элементов в контексте друг друга может быть очень информативным
2. Ограничьте объем незавершенной работы (WIP): это помогает сбалансировать основанный на потоке подход, чтобы команды не начинали и не выполняли слишком много работы одновременно
3. Непрерывность: когда что-то закончено, следующая отстающая функция втягивается в игру.

**Преимущества Kanban:**

1. Более короткое время цикла может предоставить функции заказчику быстрее.
2. Отзывчивость к изменениям.
3. Когда приоритеты меняются очень часто, Kanban идеален.
4. Соотношение спроса и пропускной способности гарантирует, что большинство функций, ориентированных на клиента, всегда работают.
5. Для начала работы требуется меньше изменений в структуре организации / помещения.
6. Сокращение отходов и устранение действий, не приносящих пользы команде / отделу / организации
7. Быстрые петли обратной связи увеличивают шансы более мотивированных, уполномоченных и более эффективных членов команды.

19 слайд – Экстремальное программирование

Экстремальное программирование стало одной из самых популярных и противоречивых гибких методологий. Экстремальное программирование – это дисциплинированный подход к быстрой и непрерывной поставке высококачественного программного обеспечения. Он способствует высокой вовлеченности клиентов, быстрым циклам обратной связи, постоянному тестированию, постоянному планированию и тесной командной работе для предоставления рабочего программного обеспечения с очень частыми интервалами, обычно каждые 1-3 недели.

**Основано на 4 принципах** - простота, общение, обратная связь и смелость

**Экстремальное программирование опирается на следующие практики:**

1. Планирование игры.
2. Небольшие релизы.
3. Простой дизайн.
4. Парное программирование.
5. Разработка через тестирование.
6. Рефакторинг кода.
7. Непрерывная интеграция.
8. Стандарты кодирования.
9. Устойчивый темп.

20 слайд – Crystal

Методология Crystal – один из самых легких и адаптируемых подходов к разработке программного обеспечения.

Семейство Crystal направлено на осознание того, что для каждого проекта может потребоваться слегка адаптированный набор политик, практик и процессов, чтобы соответствовать уникальным характеристикам проекта. Некоторые из ключевых принципов Crystal включают командную работу, общение и простоту, а также умение часто корректировать и улучшать процесс – то есть все характерное для Agile, но даже более гибкий, можно учитывать команду, заказчика, продукт, процесс. Как и другие методологии гибких процессов, Crystal способствует скорейшей, частой поставке рабочего программного обеспечения, высокой вовлеченности пользователей, адаптируемости.

21 слайд – Функционально-управляемая разработка FDD

FDD - ориентирована на клиента, известна короткими итерациями и частыми выпусками.

Выпуская новые функции поэтапно, разработчики могут расставлять приоритеты по запросам клиентов, своевременно отвечать на запросы и удовлетворять клиентов. Чтобы достичь этого, разработчики планируют, какие функции они могут создавать, разбивают сложные запросы на ряд меньших наборов функций, а затем создают план того, как выполнить каждую цель с течением времени.

FDD был разработан, чтобы следовать пятиэтапному процессу разработки, построенному в основном на отдельных «функциональных» проектах. Жизненный цикл проекта выглядит следующим образом:

1. Разработать общую модель.
2. Создать список возможностей.
3. Спланировать все функции.
4. Создать дизайн для функций.
5. Реализовать все по функции.

22 слайд – FDD плюсы и минусы

Сильные стороны FDD включают в себя:

1. Простой пятиступенчатый процесс обеспечивает более быстрое развитие.
2. Позволяет большим командам продвигать продукты с постоянным успехом.
3. Использует заранее определенные стандарты разработки, поэтому команды могут. быстро двигаться.

Слабые стороны FDD включают в себя:

1. Не работает эффективно для небольших проектов.
2. Меньше письменной документации, которая может привести к путанице.
3. Сильно зависит от ведущих разработчиков или программистов.

23 слайд – Спасибо за внимание.

Итак, мы рассмотрели подходы к управлению процессом разработки ПО и причины, по которым их следует использовать. Были рассмотрены методы, которые зарекомендовали себя в прошлом и семейство гибких методов Agile. Несмотря на все преимущества Agile методов и того, что существует еще множество Agile ориентированных подходов – среди которых можно найти наиболее подходящий, следует помнить, что гибкий подход не является панацеей и его следует подбирать в зависимости от проекта, команды, заказчика и других факторов. Он требует отличной подготовки руководителя проекта и адаптации всей команды.