СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Экстремальное программирование (XP) и унифицированный процесс разработки представляют собой два основополагающих подхода в методологии создания программного обеспечения. XP и унифицированный процесс выстраивают фундаментальные принципы, которые определяют ход процесса создания программных продуктов, однако, их фокус и стратегии существенно отличаются.

XP — это подход разработки программного обеспечения и ведения бизнеса в области создания программных продуктов, которая фокусирует усилия обеих сторон (программистов и бизнесменов) на общих, вполне достижимых целях. Данный подход ставит перед собой задачу обеспечить быструю разработку, акцентируя внимание на постоянном взаимодействии с заказчиком и оперативных реакциях на изменения требований. Этот подход способствует созданию гибких, адаптивных продуктов в условиях динамичной рыночной среды.

Унифицированный процесс стремится к созданию структурированной и комплексной системы управления проектом. Он ориентирован на документацию, строгие процессы и формализацию этапов разработки, что позволяет создавать крупные и сложные программные продукты с учетом широкого спектра требований и ограничений.

Обе эти методологии занимают центральное место в современной индустрии разработки программного обеспечения. XP предлагает гибкий и быстрый подход, позволяющий лучше адаптироваться к изменениям и быстрее вывести продукт на рынок. Унифицированный процесс, в свою очередь, обеспечивает структурированность, документирование и контроль, необходимые для разработки крупных и комплексных систем. Их значимость заключается в предоставлении разработчикам разнообразных инструментов и подходов для успешного создания качественного программного обеспечения в условиях постоянно меняющихся требований и ограниченных ресурсов.

ЧТО ТАКОЕ XP

Экстремальное программирование (Extreme Programming или XP) — это методология разработки программного обеспечения, которая была предложена в 1990-х годах Кентом Беком. XP была создана как ответ на проблемы, с которыми сталкиваются команды разработки при работе над большими и сложными проектами. Основная идея XP заключается в том, чтобы улучшить качество разработки программного обеспечения и повысить удовлетворенность заказчика через более гибкие и адаптивные практики.

XP — это упрощенный, эффективный, гибкий, предсказуемый, научно обоснованный и весьма приятный способ разработки программного обеспечения, предусматривающий низкий уровень риска. От других методик XP отличается по следующим признакам:

1. Благодаря использованию чрезвычайно коротких циклов разработки XP предлагает быструю, реальную и постоянно функционирующую обратную связь.

2. В рамках XP используется планирование по нарастающей, в результате общий план проекта возникает достаточно быстро, однако при этом подразумевается, что этот план эволюционирует в течение всего времени жизни проекта.

3. В рамках XP используется гибкий график реализации той или иной функциональности, благодаря чему улучшается реакция на изменение характера бизнеса и меняющиеся, в связи с этим требования заказчика.

4. XP базируется на автоматических тестах, разработанных как программистами, так и заказчиками. Благодаря этим тестам удается следить за процессом разработки, обеспечивать корректное эволюционирование системы и без промедления обнаруживать существующие в системе дефекты.

5. XP основана на обмене информацией, тестах и исходном коде. Три этих инструмента используются для обмена сведениями о структуре системы и ее поведении.

6. XP базируется на процессе эволюционирующего дизайна, который продолжается столь же долго, сколько существует сама система.

7. XP базируется на тесном взаимодействии программистов, обладающих самыми обычными навыками и возможностями.

8. XP основывается на методиках, которые удовлетворяют как краткосрочным инстинктам отдельных программистов, так и долгосрочным интересам всего проекта в целом.

В рамках XP существуют определенные вещи, которые необходимо делать чтобы использовать XP.

ИСТОРИЯ ПОЯВЛЕНИЯ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ

Экстремальное программирование как методология разработки программного обеспечения была предложена Кентом Беком в 1996 году. Эта методология была разработана в ответ на некоторые недостатки традиционных методов разработки программного обеспечения.

В начале 1990-х годов Кент Бек работал в области разработки программного обеспечения и сталкивался с ограничениями традиционных методологий, которые часто приводили к долгим циклам разработки, недостаточной гибкости и трудным сопровождением кода.

В 1996 году, работая над проектами, Бек начал применять набор практик, которые сочетали в себе лучшие элементы традиционных и нетрадиционных методологий. Он делал акцент на простоте, гибкости и обратной связи от заказчика.

Одним из первых проектов, где применялись принципы экстремального программирования, был проект C3. Применение XP на этом проекте

позволило существенно улучшить качество кода, сроки поставки и вовлеченность заказчика в процесс разработки.

В 1999 году Бек опубликовал книгу под названием "Extreme Programming Explained: Embrace Change" (Экстремальное программирование: Обними изменения). В этой книге он подробно описал принципы и практики экстремального программирования.

После выпуска книги сообщество, интересующееся XP, начало активно развиваться. Профессионалы по всему миру начали внедрять методологию в свои проекты, а также вносить собственные изменения и улучшения. В последующие годы экстремальное программирование продолжило эволюцию, а практики, такие как парное программирование, тестирование на первом месте, постоянное объединение кода, стали более широко принимаемыми в индустрии разработки программного обеспечения.

Сегодня XP остается одним из множества методов гибкой разработки, используемых командами по всему миру, и вносит свой вклад в создание высококачественного ПО.

ГЛАВНЫЕ ПРИНЦИПЫ И МЕТОДИКИ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ

Методика XP предназначена для работы над проектами, над которыми может работать группа программистов, которые не зажаты в жесткие рамки существующего компьютерного окружения и в которых вся необходимая работа, связанная с тестированием, может быть выполнена в течение одного дня. XP, или экстремальное программирование, является методологией разработки программного обеспечения, которая стремится улучшить качество и эффективность процесса разработки. Вот некоторые из главных принципов и методик XP:

Простота: разработчики должны следовать принципу максимальной простоты при создании кода. Это включает в себя минимизацию сложности кода, избегания избыточности и лишних деталей, а также выбор наиболее простых и понятных решений. Основной вектор идет на создании наименьшего объема кода, который выполняет необходимые задачи.

Континуальная поставка - в экстремальном программировании принято за правило видеть результат своих действий настолько быстро, насколько это вообще возможно. Или, говоря техническим языком, обеспечить максимально быструю интеграцию нового кода в общую кодовую базу. Короткие циклы обратной связи являются ключевым элементом XP. Это означает, что изменения в коде должны тестироваться и интегрироваться в систему как можно быстрее. Это помогает выявлять проблемы, делать корректировки на ранних этапах разработки, уменьшить время на ввод новой функциональности в проекте.

Обратная связь: данный метод необходим чтобы получать обратную связь от заказчика и других участников процесса разработки. Это помогает убедиться, что продукт соответствует требованиям и может быстро реагировать на изменения.

Изменчивые требования: Требования могут меняться с течением разработки. XP предполагает, что требования могут меняться, и разработка должна быть способной быстро адаптироваться к этим изменениям.

Сопровождение: работать необходимо в устойчивом темпе, чтобы избежать перегрузки и усталости команды. Устойчивый темп способствует долгосрочной продуктивности. Работа над проектом должна вестись через короткие временные интервалы, называемые итерациями. Каждая итерация привносит новые функции и улучшения в проект, что позволяет быстро адаптироваться к изменяющимся требованиям.

Программирование в парах: разработчики работают в парах, совместно решая задачи. Один пишет код, а другой следит за процессом, предлагает

идеи и обеспечивает обратную связь. Это способствует повышению качества кода, снижает вероятность ошибок и обмену знаниями.

Общение: В командах, работающих по методу XP, всегда приветствуется общение - самое быстрое средство обмена информацией и опытом. Это очень важно, когда требуется максимальная скорость разработки. Необходимость стимулировать активное и открытое общение между участниками процесса разработки. Эффективная коммуникация считается ключевым элементом успешной разработки.

Общее владение кодом: каждый разработчик может вносить изменения в любую часть кодовой базы. Это способствует распределению знаний и повышению гибкости команды.

Тестирование: тестирование играет ключевую роль в XP. Тесты пишутся до написания кода, и разработчики регулярно запускают все тесты, чтобы удостовериться в работоспособности системы.

Рефакторинг: разработчики регулярно улучшают структуру кода, делая его более понятным и поддерживаемым, без изменения его функциональности. Рефакторинг проводится в тесном взаимодействии с тестированием.

Качество: необходимо поддерживать высокое качество кода через практики тестирования, рефакторинг и другие техники, чтобы минимизировать количество ошибок и упростить поддержку кода.

Планирование: планы разработки строятся на основе актуальных требований и возможностей команды. Планирование в XP более гибкое и адаптивное, чем в традиционных методологиях.

40-часовая рабочая неделя: Сверхурочная работа рассматривается как признак больших проблем в проекте. Не допускается сверхурочная работа 2 недели подряд — это истощает программистов и делает их работу значительно менее продуктивной.

Заказчик на рабочей площадке: основной проблемой разработки программного обеспечения является недостаток знаний программистов в

разрабатываемой предметной области. Экстремальное программирование предполагает, что заказчик должен принимать участие в процессе разработки.

Эти принципы и методики взаимодействуют друг с другом, создавая гибкую и адаптивную среду для разработки программного обеспечения, способствуя улучшению качества продукта и ускорению процесса разработки. Однако важно отметить, что использование конкретных практик может изменяться в зависимости от конкретных условий проекта и команды.

ДОСТОИНСТВА И НЕДОСТАТКИ ЭКСТРЕМАЛЬНОГО ПРОГРАММИРОВАНИЯ

К достоинствам экстремального программирования (если его удается применить) можно отнести:

1. Гибкость: XP позволяет быстро реагировать на изменения в требованиях и потребностях клиентов, возможность быстро и аккуратно вносить изменения в ПО в ответ на изменившиеся требования.
2. Быстрое развертывание: XP ставит основной упор на функциональность и быстроту выпуска программного обеспечения.
3. Качество кода: XP сосредотачивается на качестве и улучшении кода благодаря тестированию и интеграции.
4. Участие заказчика: методология включает заказчика в процесс разработки, что обеспечивает более точное соответствие продукта потребностям и ожиданиям.
5. Уклонение от излишних формальностей: вместо документации XP основывается на разработке и быстром обмене информацией в команде.

К недостаткам экстремального программирования можно отнести следующие пункты:

1. Недостаток структуры: некоторые разработчики могут столкнуться с нехваткой формализации, что приводит к хаосу в разработке.

2. Сложности в масштабируемости: XP может быть сложно масштабировать на крупные проекты и большие команды.

3. Возможное недостаточное внимание к архитектуре: из-за уклонения от излишней документации, архитектурный дизайн может страдать.

4. Не все проекты подходят для XP: некоторые проекты, особенно те, где требуются строгие процессы и документация, могут не подходить для XP.

5. Невозможность долгосрочного планирования: невозможность запланировать сроки и трудоемкость проекта на достаточно долгую перспективу и четко предсказать результаты длительного проекта в терминах соотношения качества результата и затрат времени и ресурсов

6. Отсутствие предварительных исследований: XP неприспособленна для проектов в которых возможные решения не находятся сразу на основе ранее полученного опыта, а требуют проведения предварительных исследований.

Как и любая методология, XP имеет как свои преимущества, так и недостатки, и применение метода на конкретном проекте должно осуществляться с учетом конкретных особенностей и контекста проекта.

УНИФИЦИРОВАННЫЙ ПРОЦЕСС РАЗРАБОТКИ

Унифицированный процесс разработки (УП) — это методология для построения процессов разработки программного обеспечения, позволяющий команде разработки преобразовывать требования заказчика в работоспособный продукт. В зависимости от требований и доступных ресурсов, процесс разработки может быть адаптирован путём включения или исключения определённых проектных активностей. Данная методика основана на объектно-ориентированном подходе к процессу разработки программного обеспечения, который обеспечивает методику управления жизненным циклом разработки ПО. УП обладает следующими основными характеристики:

1. Итеративность: УП поддерживает разработку через серию небольших итераций, в рамках каждой из которых происходит выполнение определенных задач. Этот подход позволяет быстрее получать обратную связь от заказчика и быстрее реагировать на изменения.

2. Инкрементальность: разработка в УП осуществляется путем добавления новой функциональности к системе постепенно и последовательно. Каждая итерация или инкремент добавляет новые возможности или улучшения к системе, что позволяет не просто создавать большие объемы кода, а давать конечный продукт клиентам на более ранних этапах разработки.

3. Артефакты: в рамках УП используются стандартные артефакты, такие как модели UML, документация требований, диаграммы, спецификации и т.д. Эти артефакты помогают визуализировать, документировать и управлять различными аспектами процесса разработки.

4. Архитектурное управление: УП уделяет особое внимание архитектуре системы, включая проектирование, документирование и поддержание соответствия архитектуры на протяжении всего процесса

разработки. Акцент делается на создание стабильной архитектуры системы как базы для последующего развития.

5. Управление рисками: Методология УП включает в себя управление рисками, что позволяет выявлять и управлять рисками на ранних этапах разработки, что в свою очередь снижает вероятность возникновения проблем в будущем.

6. Поддержка различных типов процессов: УП гибко подходит для различных типов проектов.

7. Управление изменениями: УП предусматривает механизмы контроля изменений в требованиях и архитектуре системы.

Эти характеристики делают УП эффективным инструментом для разработки программного обеспечения, который позволяет управлять сложными проектами и обеспечивать высокое качество конечных продуктов.

ИСТОРИЯ ВОЗНИКНОВЕНИЯ

Унифицированный процесс разработки (УП) был создан как результат совместной работы нескольких крупных компаний в области разработки программного обеспечения, включая Rational Software (позднее приобретенная компанией IBM) и Object Management Group (OMG). Главными разработчиками УП были Гради Буч, Ивар Якобсон и Джеймс Рамбо.

История Унифицированного процесса началась в конце 1980-х годов, когда Рациональное программное обеспечение независимо разрабатывало методику для управления разработкой программного обеспечения. В то время они называли свой подход "Rational Unified Process" (RUP), который позднее стал основой для Унифицированного процесса.

В 1994 году Гради Буч впервые опубликовал работу, в которой представил концепцию UML (Unified Modeling Language) и ООА/ООП

(Объектно-ориентированный анализ и проектирование), которые стали основой для УП. Далее, в 1997 году, книга "The Unified Software Development Process" была выпущена, где Гради Буч, Ивар Якобсон и Джеймс Рамбо впервые формализовали концепцию УП.

Затем в 1999 году, представители Rational Software стали работать с Object Management Group, чтобы интегрировать УП с их стандартами по разработке программного обеспечения, что привело к появлению Унифицированного процесса в настоящем виде.

С тех пор УП стал широко используемой и изучаемой методологией разработки программного обеспечения, и она продолжает развиваться и приспосабливаться к изменяющимся требованиям индустрии.

ФАЗЫ РАЗРАБОТКИ ПРИ УНИФИЦИРОВАННОМ ПРОЦЕССЕ РАЗРАБОТКИ

Каждый цикл разработки, при унифицированной разработки, состоит из четырёх фаз, представляющих собой промежуток времени между важными этапами проекта, позволяющими руководителям принять важные решения относительно продолжения процесса разработки, объёма работ, бюджета и расписания. Эти фазы включают:

Начальная фаза (Inception): позволяет очертить границы системы, определить предполагаемую архитектуру, выявить критические риски и принять решения относительно старта проекта, в зависимости от предположительных оценок его стоимости, вложенных усилий, расписания и качества продукта. С этой фазой связан важный этап проекта под названием «Цели жизненного цикла».

Задачи: определение общей цели проекта, оценка его технической и экономической осуществимости, выявление основных рисков.

Результат: создание начального бизнес-плана и определение основной архитектуры системы.

Разрабатывающая фаза (Elaboration): Создана для решения следующих задач: выяснение большинства функциональных требований, преобразование предполагаемой архитектуры в работающий прототип, сфокусированный на архитектуре, устранение критических рисков, принятие окончательного решения о начале проекта и создание достаточно детального плана. Завершает эту фазу этап под названием «Архитектура жизненного цикла».

Задачи: детальное проектирование архитектуры, уточнение требований, определение основных компонентов системы.

Результат: уточненная архитектура, план разработки, определенные риски и способы их управления.

Строительная фаза (Construction): включает в себя итеративную разработку системы, которая может успешно взаимодействовать с пользователями на бета-окружении. Наличие более-менее работоспособной системы знаменует собой успешное достижение соответствующего этапа.

Задачи: фактическая реализация и тестирование системы, разработка пошаговых инкрементов возможностей системы.

Результат: рабочая версия системы, продолжение доработки и тестирования.

Переходная фаза (Transition): полностью функционирующую систему передают на пользование пользователям. Является завершающей фазой цикла разработки. Этапом для неё считается выпуск продукта.

Задачи: предварительное развертывание системы, обучение пользователей, подготовка документации, поддержка и обслуживание.

Результат: успешное развертывание системы, обеспечение поддержки и обновлений.

Каждая из этих фаз имеет свои цели, активности и результаты, и представляет собой важный этап в цикле разработки программного обеспечения с использованием унифицированного процесса.

ДОСТОИНСТВА И НЕДОСТАТКИ УНИФИЦИРОВАННОГО ПРОЦЕССА РАЗРАБОТКИ

Унифицированный процесс, также как и другие методики программирования, имеет свои достоинства и недостатки. Можно выделить следующие достоинства унифицированного процесса разработки:

1. Итеративность и инкрементальность: унифицированный процесс поддерживает итеративное и инкрементальное развитие продукта, что позволяет быстрее реагировать на изменения в требованиях заказчика и вносить коррективы в процесс разработки.

2. Гибкость и адаптивность: унифицированный процесс гибок и может быть адаптирован к различным типам проектов, учитывая их размер, сложность и требования.

3. Ориентация на архитектуру: унифицированный процесс акцентирует внимание на архитектурных аспектах разработки, что способствует созданию более стабильных и поддерживаемых систем.

4. Управление рисками: методология унифицированный процесс включает в себя процессы управления рисками, что позволяет идентифицировать и решать проблемы на ранних стадиях разработки.

5. Фокус на качестве: унифицированный процесс предоставляет инструменты и методы для обеспечения высокого качества продукта.

Можно выделить следующие недостатки унифицированного процесса разработки:

1. Сложность: унифицированный процесс может казаться слишком сложным и трудным в освоении для небольших и менее сложных проектов.

2. Ресурсоемкость: применение унифицированного процесса может требовать значительных ресурсов (как по времени, так и по финансам).

3. Недостаточная конкретика: при использовании унифицированного процесса может быть недостаточно конкретных указаний

по ряду вопросов, что может потребовать дополнительных усилий от команды для их разрешения.

4. Не всегда подходит для малых проектов: унифицированный процесс может казаться избыточным для небольших проектов, где формализованный подход может быть чрезмерным.

5. Неудовлетворительно для некоторых видов проектов: для некоторых типов проектов унифицированный процесс может быть менее подходящим, поскольку требует достаточного уровня предварительной информации о требованиях.

Выбор методологии зависит от конкретных потребностей проекта, размера команды, характера требований и других факторов. Унифицированный процесс может быть успешным в определенных контекстах, но, как и любой подход, он имеет свои сильные и слабые стороны.

ЗАКЛЮЧЕНИЕ

Экстремальное программирование больше всего подходит для различных проектов, особенно в условиях переменных требований, быстрого изменения среды и необходимости частого взаимодействия с заказчиком. Вот несколько типов проектов, для которых XP может быть особенно подходящим:

1. Проекты с быстро меняющимися требованиями
2. Стартапы и инновационные проекты
3. Малые и средние проекты
4. Проекты с неопределенными или слабо определенными требованиями
5. Проекты с акцентом на качество кода
6. Проекты с активным участием заказчика
7. Проекты, требующие частых релизов

Унифицированный процесс разработки представляет собой гибридную методологию управления проектами и разработки программного обеспечения. Он может применяться в различных проектах, но чаще всего находит применение в средних и крупных проектах с достаточно сложными требованиями и потребностями. Унифицированный процесс разрабатывался, чтобы быть адаптивным и расширяемым, что позволяет его использование в разнообразных контекстах. Вот несколько типов проектов, для которых унифицированный процесс разработки может быть подходящим:

1. Средние и крупные проекты разработки программного обеспечения
2. Проекты с переменными или изменяющимися требованиями
3. Проекты, где важно поддерживать высокое качество
4. Проекты с акцентом на архитектуре
5. Проекты с требованиями к документации
6. Распределенные проекты

Данные методики имеют свои преимущества и недостатки при определенных условиях разработки, но важно отметить, что выбор методологии зависит от множества факторов, включая размер команды, характер проекта, требования заказчика и т.д.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

RUP: Руководство пользователя от Philippe Kruchten

Практика программирования (The Practice of Programming) Эндрю Ханта (Andrew Hunt), Дэвид Томас (David Thomas).

Экстремальное программирование в деталях (Extreme Programming Adventures in C#) Рон Джеффрис (Ron Jeffries).

Экстремальное программирование. Разработка через тестирование (Extreme Programming Explained: Embrace Change) Кент Бек (Kent Beck).