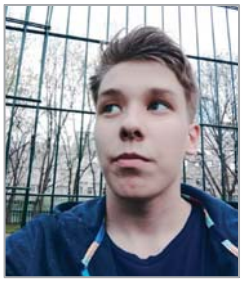
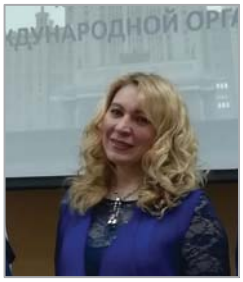
РАЗДЕЛ V. МОЛОДЫЕ УЧЕНЫЕ – ПОИСК САМООПРЕДЕЛЕНИЯ

ИЗУЧЕНИЕ МОДЕЛЕЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ В ОБЛАСТИ WEB-ТЕХНОЛОГИЙ

Сучков Ярослав Сергеевич
Студент 4-го курса Московского политехнического университета



Пучков Павел Владимирович
Студент 4-го курса Московского политехнического университета.



Бритвина Валентина Валентиновна
Кандидат педагогических наук, доцент кафедры «Инфокогнитивные технологии» Московского политехнического университета

Аннотация: В статье изучены и проанализированы предлагаемые модели разработки ПО, чтобы сравнить их и разработать указания к применению этих моделей к разным типам проектов.

Ключевые слова: Информационные технологии, Web-технологии, модель, приложение, ПО, проект.

Abstract: The paper studies and analyzes the proposed software DEVELOPMENT models to compare them and develop guidelines for the application of these models to different types of projects.

Keywords: Information technologies, Web-technologies, model, application, SOFTWARE, design.

Введение

В современном мире почти каждая команда разработчиков сталкивается с проблемой выбора между предлагаемыми моделями разработки программного обеспечения. Использование неверной модели усложняет разработку, ставит под угрозу успешную реализацию и поддержку проекта в дальнейшем. Логично, что у разработчиков и будущих владельцев приложений возникают вопросы касательно организации процесса разработки приложений.

Цель исследования

Изучить модели разработки программного обеспечения в области web-технологий.

Задачи исследования

1. Проанализировать модели организации процесса разработки приложений в области web-технологий.
2. Проанализировать эффективность рассмотренных моделей.
3. Разработать рекомендации по выбору модели организации рабочего процесса с целью добиться максимальной эффективности команды и каждого из разработчиков в целом.

Каскадная (водопадная) модель – Waterfall

В такой модели разработка делится на большие этапы, которые следуют друг за другом. С её помощью можно создать приложение, не тратя

огромные средства на процесс разработки, однако, важно, чтобы требования были четко определены и не менялись на протяжении всего процесса, так как возврат к предыдущим этапам не предусмотрен до окончания разработки [4].

Как пример, можно представить такую последовательность этапов (рис. 1):

1. Анализ требований.
2. Проектирование.
3. Реализация.
4. Тестирование.
5. Внедрение.
6. Поддержка.

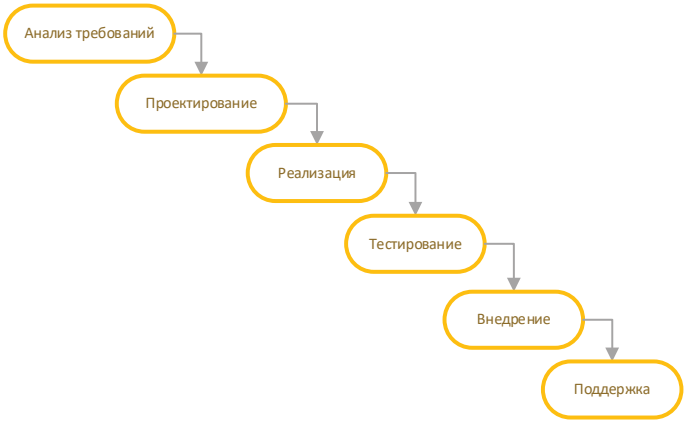


Рис. 1. Структура водопадной модели

Данная модель подвержена критике, в том числе за формализацию процесса в ущерб стоимости и срокам, что может быть критично для большинства web проектов. Однако, такая модель находит применение в сфере государственных заказов, где формальное управление может стать основным критерием пригодности исполнителя.

Спиральная модель

Разработка в этой модели представляется в виде спирали, в которой поворот спирали будет итерацией.[6]

Обычно виток спирали представляет 4 этапа (рис. 2):

1. Планирование.
2. Анализ рисков.
3. Реализация.
4. Оценка.

Рис. 2. Структура спиральной модели

На этапе анализа рисков, происходит проектирование решений, которые помогут устранить эти риски как можно раньше.

Такая модель чрезвычайно дорогостоящая и её применение почти не оправдано в таких системах, в которых выявление и устранение рисков не является приоритетной задачей.

Итеративная модель

Аналогично спиральной модели, делится на этапы. Каждый из этапов именуется итерацией и может включать в себя почти все основные этапы водопадной модели (рис. 3):

1. Анализ требований.
2. Проектирование.
3. Реализация.
4. Тестирование.

Таблица 1

Преимущества, недостатки и показания к применению Водопадной модели

Преимущества	Недостатки	Показания к использованию
Понятная и простая архитектура разработки	Не гибкий процесс разработки	Требования четко определены и фиксированы
Удобная отчетность	Нестабильность (при необходимости изменений)	Требования не противоречат друг другу
Стабильность задач	Недостаточное тестирование	Небольшие проекты
Оценка стоимости и сроков сдачи проекта	Консервация в отношении изменений	Государственные проекты со строгой отчетностью

Таблица 2

Преимущества, недостатки и показания к применению спиральной модели

Преимущества	Недостатки	Показания к использованию
Возможность увидеть проект на ранних этапах разработки	Сложности в определение дальнейших целей в разработке	Анализ рисков имеет огромное значение
Выявление непреодолимых рисков	Усложненная архитектура	Высокий бюджет
Активное участие пользователей	Нескончаемость	Большой проект
Гибкое прототипирование	Дорогостоящая модель	Требования могут быть модифицированы

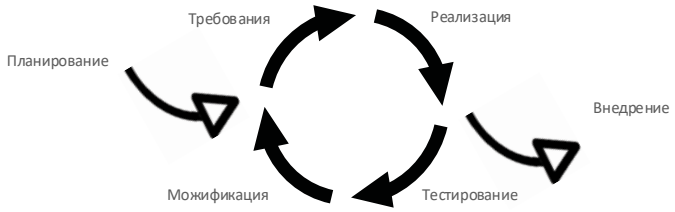


Рис. 3. Структура итеративной модели

Хоть данная модель обходится дороже, чем водопадная, она имеет очевидное преимущество: ошибки, возникающие во время одного этапа, можно исправить в другом. На основе этой модели построены почти все гибкие модели разработки.

Scrum

В данной методологии основной упор делается на конечный результат. Определяются наиболее важные для заказчика функции, реализация которых становится основной задачей команд. Процесс разработки разбивается на итерации, называемые спринтами. Спринты не должны длиться дольше 4 недель.

Таблица 3

Преимущества, недостатки и показания к применению Итеративной модели

Преимущества	Недостатки	Показания к использованию
Акцентирование внимания на слабых\важных местах проекта	Долгое время отсутствует понимание возможностей и ограничений проекта	Определена основная задача, но реализация может меняться с течением времени
Оценка текущего состояние проекта	При итерации часть работы отбрасывается	Большой проект
Обнаружение конфликтов между требованиями, моделями и реализацией проекта	Качество разработки снижается из-за сложностей с архитектурой кода	Требования четко сформулированы
Непрерывное тестирование		
Минимизация затрат на устранение рисков		
Обратная связь с потребителями		
Распределение задач среди участников проекта		
Распределение затрат по всему проекту		

В начале каждого из спринтов, на собрании определяются основные цели и задачи данной итерации, которые не должны изменяться на протяжении всего спринта.

Должны проводиться ежедневные 15-минутные собрания, с целью определить, что член команды успел сделать, чтобы приблизить достижение цели спринта, выявить основные проблемы и решить, необходимо ли привлечение другой команды к проекту для успешного выполнения спринта.

Должны проводиться ретроспективные совещания в конце спринта, на которых члены собрания высказывают свое мнение о прошедшей итерации и подводятся итоги работы.

Данная модель применяется в ситуациях, когда необходим результат в самые кратчайшие сроки. Благодаря строгим определениям критериев готовности и относительной гибкости, она завоевала широкую популярность как почти универсальная модель для ведения проектов. [2]

В процессе принято деление команды на определенные роли.

- Master - основной ведущий, следящий за соблюдением регламентов
- Product Owner – тот, для кого делается продукт, напрямую заинтересованный в успешной разработке приложения конечный пользователь
- Team - разработчики, между которыми распределены роли

SCRUM использует собственную систему баллов, «начисляемых» за успешное выполнение мини задач. Они используются для определения эффективности команды. В модели SCRUM также встречается интересный способ обсуждения вопросов. Его называют Покер Планирования.

Таблица 4

Преимущества, недостатки и показания к использованию Scrum

Преимущества	Недостатки	Показания к использованию
Быстрый запуск продукта с важными функциями и малым бюджетом	Отсутствие фиксированного бюджета	ТЗ не строго сформулировано или отсутствует в принципе
Ежедневный контроль	Сложности в заключении договоров	Большие проекты, растянутые по времени
Почти непрерывные демонстрации проекта заказчику	Узкая специализация методов самой методологии	Необходимо внесение множества изменений
Возможность изменять ТЗ в процессе разработки	Большое количество исключений в применении	

Product Owner представляет пункты повестки дня, а команда обсуждает их. Когда обсуждение конкретного пункта завершается, каждый участник кладет карточку определенного достоинства, которая может выражать сложность задачи или количество требуемых для ее выполнения дней/часов/т.д. Это обговаривается заранее.

Карточку вопроса используют, когда смысл высказывания не понятен.

Карточка «Чашка кофе» означает, что человек устал и ему требуется перерыв. [5]

Каждому участнику собрания перед итерацией выдается колода карточек, состоящая из бумаги с обозначениями (один из частых примеров 0, 1, 2, 3, 5, 8, 13, 20, 40, 100, «?», «Чашка кофе»).

XP (Extreme Programming)

В данной модели присутствуют 12 практик, которые должны обязательно применяться в течение разработки проекта. Считается, что данная модель позволяет максимально упростить внесение изменений в продукт. [5]

Вот описание 12 практик [3]:

1. Парное программирование.
2. Разработка через тестирование.
3. Рефакторинг.
4. Простая архитектура.
5. Игра в планирование.
6. Коллективное владение кодом.
7. Стандарты написания кода.
8. Непрерывная интеграция.
9. Частые релизы.
10. Метафора системы.
11. 40-часовая рабочая неделя.
12. Заказчик в команде.

Часто практикуется применение принципов XP в других моделях, чтобы обеспечить легкость изменений и поддержки продукта.

При использовании всех принципов XP, можно добиться ряда привлекательных преимуществ:

- Итоговый продукт - именно тот который нужен заказчику (благодаря правкам в работе).
- Функциональность.
- Рабочий код (за счет тестирования и интеграций).
- Единый стандарт написания кода.
- Быстрый темп (за счет парного программирования и т.д.).
- Высокое качество кода.
- Снижение рисков.

Таблица 5

Преимущества, недостатки и показания к использованию XP

Преимущества	Недостатки	Показания к использованию
Итоговый продукт - именно тот который нужен заказчику	Успех проекта зависит от активности заказчика	Заказчик должен быть вовлечен в проект
Функциональность	Зависимость от уровня программистов	Требования не строго сформулированы
Рабочий код (за счет тестирования и интеграций)	Сложно описать нефункциональные требования	Средние и маленькие по размеру проекты
Единый стандарт написания кода	Не подходит для крупных проектов	
Быстрый темп реализации	Нет определенных сроков и затрат	
Высокое качество кода		
Снижение рисков		

Kanban

В отличие от Scrum, в Kanban отсутствуют спринты и роли, однако, часто ее считают даже более гибкой, чем Scrum.

Легкая для понимания модель, состоящая из трех принципов [1]:

1. За процессом следят с помощью канбан-досок (в их роли может выступать простая магнитная доска с прикрепляемыми на нее стикерами, либо специализированные приложения).
2. Постоянное измерение производительности.
3. Ограничение на количество задач в итерации.

Таблица 6

Преимущества, недостатки и показания к применению Kanban

Преимущества	Недостатки	Показания к использованию
90% успеха при использовании зависит от команды	Требуются специалисты высокого уровня	Когда какие-либо серьезные решения касательно проекта нужно будет откладывать
Позволяет перераспределять нагрузку между членами команды	После итерации может не быть MVP	Необходимость в разгрузке рабочего потока

RAD

Концепция организации процесса разработки ПО, ориентированная на получение максимально качественного результата в очень сжатые, четко установленные сроки.

Основные свойства модели:

1. Длительность проекта 60-90 дней.
2. Нет требований, или они нечетко сформулированы.

3. Бюджет строго ограничен.
4. Интерфейс стоит на первом месте.
5. Проект можно разбить на компоненты.
6. Сложность ПО не высока.

Данная модель реализует максимально сосредоточенную на результате и сроках итеративную модель. Но можно представить основные этапы (рис. 4):

1. Планирование и разработка требований.
2. Пользовательское проектирование / пользовательский дизайн.
3. Реализация (разработка).
4. Переключение (включает в себя тестирование etc.).

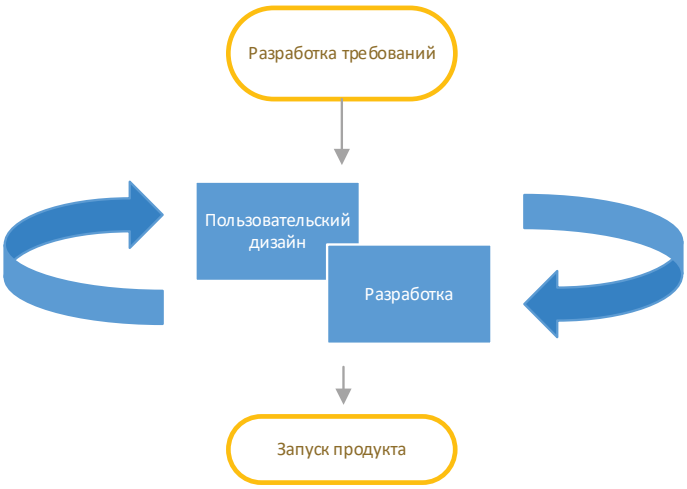


Рис. 4. Структура RAD-модели

На протяжении каждого этапа пользователи активно участвуют в процессе разработки.

Таблица 7

Преимущества, недостатки и показания к применению модели RAD

Преимущества	Недостатки	Показания к использованию
Благодаря мощным инструментальным средствам, время итерации разработки сокращается	Отсутствие обратной связи в разработке может снизить качество выходного продукта	При наличии высококвалифицированных программных архитекторов
Требуется меньшее количество работников	Необходима высокая квалификация разработчиков	Высокий бюджет
Благодаря участию заказчика в разработке, на выходе получается то, что ему нужно	Применение модели может оказаться провальным в случае отсутствия повторного использования кода и компонентов	Уверенное знание целевого бизнеса
Из-за принципов временного блока, сокращаются затраты и риск		Проект длится 60-90 дней
Позволяет в короткие сроки получить представление продукта		
В каждый временной блок входят этапы итеративной модели		

Алгоритм выбора модели разработки программного обеспечения представлен на рис. 5.

Создание алгоритмов для выбора модели было осложнено количеством работы, которую требуется провести для анализа всех преимуществ и недостатков проектов. Мы можем представить краткий алгоритм.

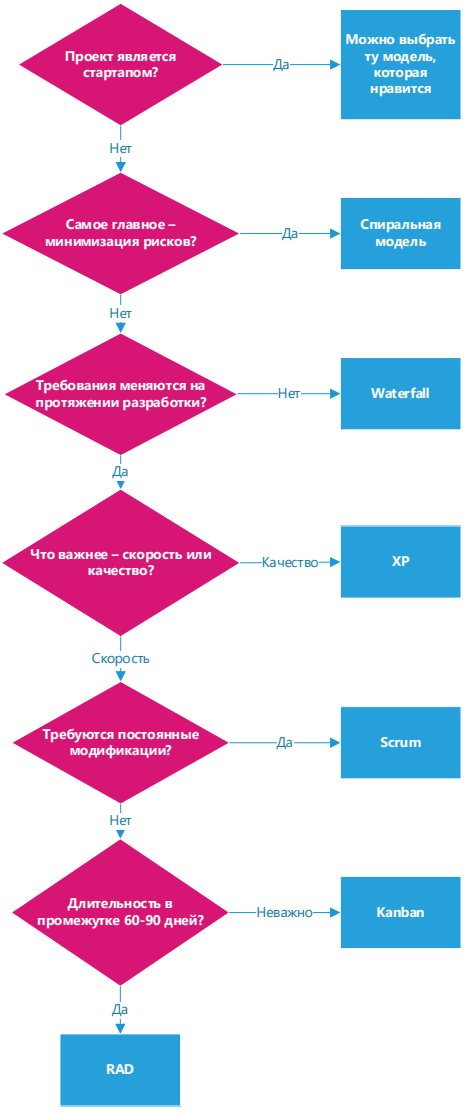


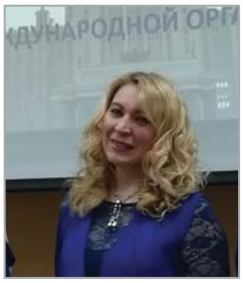
Рис. 5. Алгоритм выбора модели

Список литературы
1. **Книберг Х.**, **Скарин М.** Scrum и Kanban: выжимаем максимум. – InfoQ, 2010
2. **Stephens M.**, **Rosenberg D.** Extreme Programming Refactored: The Case Against XP. - APress, USA, 2003
3. **Кент Бек**: Экстремальное программирование. - Питер, 2002
4. **Royce**, Winston. Managing the Development of Large Software Systems. - 1970.
5. **Книберг Х.** Scrum и XP: заметки с передовой = Scrum and XP from the trenches. - C4Media, 2007
6. **Richard W.** Selby. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research. — John Wiley & Sons, 2007-06-04. — 834 с. — ISBN 9780470148730.

АНАЛИЗ ЭФФЕКТИВНОСТИ АНТИВИРУСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В СОВРЕМЕННЫХ ТЕХНОЛОГИЯХ ЗАЩИТЫ ИНФОРМАЦИИ

Югансон Анна Райвовна
Студент Московского политехнического университета

Овчинников Максим Владиславович
Студент Московского политехнического университета



Бритвина Валентина Валентиновна
Кандидат педагогических наук, доцент кафедры «Инфокогнитивные технологии» Московского политехнического университета

Аннотация. В данной статье проанализирована эффективность использования антивирусного ПО в современных способах защиты информации. Разработаны рекомендации по обеспечению безопасности хранения данных, находящихся на компьютерах. Сделан вывод о том, что антивирусные программы не всегда защищают от вирусных атак, что в свою очередь может привести к потере данных на устройстве.

Ключевые слова: антивирус, компьютер, защита, безопасность, вирус, вредоносная программа, уязвимость, информационная технология.

Annotation. This article analyzes the effectiveness of anti-virus SOFTWARE in modern methods of information protection. Recommendations to ensure the security of data storage on computers have been developed. It is concluded that antivirus programs do not always protect against virus attacks, which in turn can lead to loss of data on the device.

Keywords: antivirus, computer, protection, security, virus, malware, vulnerability, information technology.

Введение

Большинство пользователей при покупке нового компьютера сразу задаются вопросом установки антивирусной программы. Вредоносные файлы могут не только незначительно помешать работе устройства, но и украсть конфиденциальную ин-