

|  |
| --- |
| Development process  BookIt |
|  |

| Related Artifacts | |
| --- | --- |
| Ref. | Name |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Abbreviations and Acronyms | |
|  |  |
|  |  |

Contents

[1 Section 1 **Error! Bookmark not defined.**](#_Toc419294329)

[1.1 Section 2 5](#_Toc419294330)

[1.1.1 Section 3 5](#_Toc419294331)

[1.1.1.1 Section 4 5](#_Toc419294332)

[Appendix A. Example of 1st lvl appendix **Error! Bookmark not defined.**](#_Toc419294333)

[Appendix A.1. example of 2nd lvl appendix 7](#_Toc419294334)

# методология процесса разработки

Для управления процессами разработки используется методология Feature Driven Development (FDD).

FDD – это итеративная методология разработки программного обеспечения, которая принимает за основу важную для заказчика функциональность разрабатываемого программного обеспечения. Основной целью методологии является разработка реального, работающего программного обеспечения в поставленные сроки.

FDD построена на основе лучших практик в разработке программного обеспечения:

* Объектное моделирование предметной области.
* Разработка по функциям – любая функция, которая оценивается больше 2х недель, разбивается на подфункции.
* Индивидуальное владение классом(кодом) – над каждым участком кода одновременно работает только один разработчик, который отвечает за его согласованность с остальной системой и производительность.
* Команда по проектированию интерфейсов и функций – небольшая, динамически формируемая команда разработчиков, занимающаяся небольшой подзадачей. Позволяет нескольким разработчикам участвовать в разработке контрактов функций и повышает общий уровень знаний разработчиков о работе всей системы.
* Проверка кода – обеспечивают хорошее качество кода.
* Конфигурационное управление – помогает определять ответственность классов за функционал и поддерживать историю изменения кода при добавлении и изменении функционала.
* Регулярные сборки (билды) – гарантирует наличие актуальной версии системы для показа заказчику.
* Прозрачность процессов и планов разработки для заказчика.

Основные этапы разработки программного обеспечения по FDD:

1. Разработка архитектуры приложения.
2. Составление списка функций/задач.
3. Составление плана разработки функций – устанавливается приоритет, зависимости между задачами.
4. Проектирование функций/задач.
   1. Анализ.
   2. Дизайн.
   3. Проверка дизайна.
5. Реализация функций/задач.
   1. Написание кода.
   2. Проверка кода.

5.3. Включение в сборку.



Рисунок 1 - Основные этапы разработки Программного обеспечения по методологии FDD

## Разработка архитектуры системы

Разработка архитектуры системы проводится под руководством координатора и ведущего разработчика проекта.

Разработка архитектуры системы состоит из следующих этапов.

1. Формирование команды разработчиков, отвечающих за архитектуру системы. В команду должны входить постоянные участники данного проекта по разработке системы, координатор и ведущий разработчик проекта.
2. Обзор разрабатываемой системы и ее предметной области. Проводится координатором и ведущим разработчиком проекта.
3. Изучение документации. Команда изучает документы с требованиями.
4. Команда разбивается на небольшие группы не больше 3х человек.
5. Каждая группа предоставляет свой вариант архитектуры системы или подсистемы. Координатор и ведущий разработчик также предоставляют свои варианты. Выбирается наиболее подходящий вариант архитектуры системы.
6. Улучшение архитектуры. Повторяются шаги 5 и 6.
7. Документирование выбранной архитектуры, возможного развития архитектуры, альтернативных решений.
8. Проверка. Внутренняя оценка архитектуры проводится координатором проекта и ведущим разработчиком. Архитектура предоставляется заказчику на проверку соответствия будущей системы его ожиданиям. При необходимости пункты с 5 по 8 повторяются.

Архитектура считается удовлетворительной, если ее одобрили координатор, ведущий разработчик и заказчик.



Рисунок 2 – Разработка архитектуры системы

Результатом разработки архитектуры является:

1. Структура системы – основные модули и связи между ними. Контракты модулей.
2. Диаграмма основных классов бизнес-логики.
3. Диаграммы последовательностей.
4. Документация по выбранной архитектуре и альтернативным вариантам с объяснением выбора.

## Составление списка функций

Составлением списка функций занимаются координатор, ведущий разработчик и руководители групп из пункта 1.1.

Список функций системы строится на основании функциональных требований к системе. Основываясь на основных объектах системы и связях между ними, определенных в пункте 1.1, команда выделяет основные функциональные блоки. Блоки содержат наборы пользовательских и системных активностей, каждый шаг которых выделяется в отдельную функцию.

Функции приблизительно оцениваются так, чтобы их реализация занимала не больше 2х недель.



Рисунок 3 – Составление списка функций

Результатом является плоский список функций системы.

## План разработки функций

Координатор проекта и ведущий разработчик формируют команду, отвечающую за разработку плана реализация функционала.

Строится последовательность разработки функционала на основе зависимостей между функциональными блоками и сложностью реализации функционала. Расставляются приоритеты разработки. Устанавливаются приблизительные сроки выполнения каждой функции. Закладываются риски. Устанавливаются даты для критически важных функциональных блоков.

Назначаются ответственные за разработку основных функциональных блоков.

Формируются списки классов для функциональных блоков.



Рисунок 4 – План разработки функций

Результатом является:

1. Последовательность разработки функционала системы с ключевыми датами, сроками выполнения функций, приоритетом разработки.
2. Список ответственных за разработку основных функциональных блоков.
3. Список основных классов для функциональных блоков.

## Проектирование функций

Для каждой функции из плоского списка функций системы разработчик, ответственный за блок функции, определяет связанные с функцией объекты системы и классы. Из участников проекта выбираются разработчики, из которых формируется команда по разработке функции.

По запросу ответственного за блок разработчика координатор и ведущий разработчик разъясняют детали бизнес-логики функции и предоставляют необходимую документацию. Команда изучает документацию.

Команда разрабатывает детальную диаграмму последовательностей. Предлагаются альтернативные варианты решения, уточняются требования к функционалу. Все предложенные решения фиксируются в документации.

Ответственный разработчик создает ветку в системе контроля версий, в которой будет разрабатываться функция. Команда проектирует объектную модель. Ответственный разработчик проверяет объектную модель и дополняет при необходимости.

Основываясь на объектной модели, команда пишет контракты методов и классов для каждого объекта функции.

Проводится проверка кода остальными участниками проекта. По завершении этапа контракты методов и классов добавляются в основную ветку.



Рисунок 5 – Проектирование функции

Результатом проектирования функций является:

1. Диаграмма последовательностей для каждой функции.
2. Документация с описанием контрактов функции и возможных альтернативных решений.
3. Объектная модель функции с описанием классов, методов и атрибутов.
4. Список изменений в других функциональных блоках системы при разработке функции.

## Реализация функций

Когда для функции системы завершен этап проектирования, команда разработки функции из пункта 1.4 приступает к реализации функции.

Команда разрабатывает объекты функции в соответствии с контрактами классов и методов в 1.4. Этот этап включает разработку юнит-тестов.

Разработчик, ответственный за функциональный блок, к которому относится функция, проводит инспекцию кода. Возможно подключение других участников проекта.

Ведущий разработчик проверяет код на соответствие бизнес-требованиям.

Если проверки прошли успешно, ведущий разработчик включает реализацию функции в сборку.



Рисунок 6 - Реализация функции