Программная инженерия. Лекция 3
Фаза анализа и четыре «П»

Михаил Пожидаев

28 февраля 2024 г.

Назначение

Зачем нужны варианты использования?

Вариант использования определяет последовательность действий, которые может выполнить система, приносящих ощутимый и измеримый результат, и значимый для некоторого актанта.

1. Варианты использования дают систематический и интуитивно понятный способ определения функциональных требований, ориентированный на получение необходимых пользователю результатов.
2. Варианты использования управляют всем процессом разработки. Все основные виды деятельности (анализ, проектирование и тестирование) выполняются на основе определенных вариантов использования.

Варианты использования, приносящие результаты с отрицательной ценностью, не считаются вариантами использования. Мы будем называть их «вариантами запрещенного использования».

Оценка критичности

Когда применять варианты использования?

1. В системе преобладают функциональные требования.
2. В системе много типов пользователей, которым она предоставляет разные функциональные возможности (много актеров).
3. В системе много интерфейсов.

Фокусируйтесь на том «что», а не «как».

Определение use cases

Кто разрабатывает варианты использования?

Определение вариантов использования проводится при участии:

- пользователей (экспертиза требований);
- заказчиков;
- разработчиков (экспертиза требований, облегчение обсуждения и помощь пользователям и заказчикам в изложении своих пожеланий).

Модель вариантов использования применяется для достижения соглашения с пользователями и заказчиками о функциях будущей системы.

Мир вокруг

Актанты и роли

Актанты — это среда, в которой существует система.

1. Актанты — это не обязательно люди. Актантами могут быть другие системы или внешнее оборудование компьютера, взаимодействующее с системой.
2. Каждый актант, взаимодействуя с системой, исполняет согласованный набор ролей.
3. Физический пользователь может играть роль одного или нескольких актантов, выполняя их функции по взаимодействию с системой.

Модель использования

Актанты $\leftrightarrow$ варианты использования

Определение модели вариантов использования

Модель вариантов использования — это полный набор актантов и вариантов использования.

Каждая категория пользователей представлена отдельным актантом. Актанты используют систему, взаимодействуя с вариантами использования. Диаграмма использования описывает часть модели вариантов использования, показывая набор вариантов использования с актантами и с ассоциациями между каждой взаимодействующей парой актант — вариант использования.

Три модели

Варианты использования в трёх моделях

1. Модель вариантов использования.
2. Модель анализа.
3. Модель проектирования.

В ходе анализа и проектирования модель вариантов использования преобразуется через модель анализа в модель проектирования, то есть в структуру, состоящую из классификаторов и реализаций вариантов использования.

Классы

Стереотипы анализа

В модели анализа используется три стереотипа классов :

- граничный класс;
- управляющий класс;
- класс сущности.

Пример классов

Вариант для банка

1. Устройство выдачи и интерфейс кассира — это граничные классы. Они в основном используются для моделирования взаимодействия между системой и ее актантами.
2. Управление выдачей наличных — управляющий класс. Классы такого типа в основном используются для представления координации, последовательности, взаимодействия и управления другими объектами.
3. Банковский счет — класс сущности;. Такие классы в основном используются для моделирования инкапсуляции.

Назначение класса

Обязанности и роли

Что такое обязанность класса?

Обязанности класса — это просто сумма всех ролей, которые он исполняет во всех реализациях вариантов использования.

Сложив все роли и удалив перекрывающиеся части ролей, мы получим список всех обязанностей и атрибутов класса.

Кто отвечает?

За определение ролей класса отвечают разработчики, выполняющие анализ и реализацию.

Подсистемы

Группировка классов

Классы группируются в подсистемы!

Для большой системы, состоящей из сотен или тысяч классов, невозможно реализовать варианты использования, пользуясь только классами. Такая система будет слишком велика для того, чтобы в ней можно было обойтись без группировки высшего порядка.

Подсистема — это осмысленная группировка классов или других подсистем. Подсистема имеет набор интерфейсов, которые обеспечивают ее существование и использование.

Четыре «П»

Четыре ключевых понятия

1. *Персонал:* архитекторы, разработчики, тестеры и их руководство, а также пользователи, заказчики и другие заинтересованные лица.
2. *Проект:* организационная сущность, при помощи которой происходит управление разработкой ПО. Результатом проекта является выпущенный продукт.
3. *Продукт:* артефакты, создаваемые в течение жизни проекта, такие как модели, тексты программ, исполняемые файлы и документация.
4. *Процесс:* определение полного набора видов деятельности, необходимых для преобразования требований пользователя в продукт.

А также должны быть утилиты: программы, используемые для автоматизации определенных в процессе видов деятельности.

Персонал

Полная ясность для исполнителей

Для обеспечения эффективности работы членов команды необходимо обеспечить каждому из них:

- ясное понимание выполнимости работы;
- ясное управление рисками для чувства безопасности;
- понимание принципов организации команды;
- планирование проекта и его понятность.

Очевидно, немногие люди будут рады работать в проекте, если они не верят в достижимость целей и грамотное управление.

Команды

Всё для них!

Люди лучше всего работают группами по 6–8 человек. Процесс разработки должен быть ориентирован на комфорт отдельных исполнителей.

Вера в общее дело!

Если люди не верят в то, что план реален, мораль стремительно падает — люди не любят ходить на работу, зная, что как бы усердно они ни работали, им никогда не удастся получить ожидаемый от них результат.

Каждый исполнитель должен видеть и понимать архитектуру, а также свою роль в ней.

Продукт

Система артефактов

Проект в ходе своего выполнения должен привести к появлению продукта. Продуктом называют комплекс артефактов, множество которых существенно шире, чем просто программный комплект, передаваемый заказчику.

В понятие продукта входят все артефакты, существующие в понятном компьютеру или человеку виде и представляемые компьютерам, сотрудникам или заинтересованным лицам.

Артефакты

Примеры и типы

Артефакт — это общее название для любых видов информации, создаваемой, изменяемой или используемой сотрудниками при создании системы. Примерами артефактов могут служить:

- диаграммы Унифицированного языка моделирования и связанный с ними текст;
- наброски и прототипы пользовательских интерфейсов, компоненты;
- планы тестирования и тестовые примеры.

Можно выделить два основных типа артефактов:

- технические артефакты;
- артефакты управления.

Процесс

Как шаблон проекта

В Унифицированном процессе слово «процесс» относится к концепции, работающей подобно шаблону, который может быть многократно использован для создания его экземпляров. Это походит на идею класса, который может быть использован для создания объектов класса в объектно-ориентированной парадигме. Экземпляр процесса — это синоним проекта.

Утилиты

Максимальная автоматизация разработки

Правило!

Успешная разработка средств автоматизации процесса невозможна без параллельной разработки каркаса процесса, с которым должны работать эти средства.

Идея очень хорошо знакома линуксоидам:

- скрипты автоматизации выпуска nightly-релизов;
- автоматическая выкладка релизов для заказчика и пр.

Требования

Рабочий процесс определения требований

Примером рабочего процесса может быть рабочий процесс Определение требований. В нем участвуют следующие сотрудники:

- системный аналитик; архитектор;
- спецификатор вариантов использования; разработчик пользовательских интерфейсов;
- инженеры по компонентам; тестеры интеграции.

Он включает следующие артефакты:

- модель вариантов использования; реализации вариантов использования;
- классы, подсистемы и интерфейсы.

Проработка требований

Почему требования важны?

Основные причины провала проектов:

1. Неучастие пользователей в выработке требований.
2. Неполное выявление требований.
3. Неуделение должного внимания управлению рисками.

Определение требований преследует две цели:

- определение существенныхъ требований;
- представление их в форме, удобной для пользователей, заказчиков и разработчиков.

Под «существенными требованиями» понимаются те требования, реализация которых принесет пользователям ощутимый и значимый для них результат.
Под «представлением в форме, удобной для пользователей, заказчиков и разработчиков» понимается то, что итоговое описание требований должны понимать пользователи и заказчики.

Риски

Оценка и классификация рисков

Риск — нечто неизвестное или неопределенное, что может навредить успешному выполнению проекта. Может быть следующих типов:

- прямой риск: можно контролировать;
- косвенный риск: нельзя или почти нельзя контролировать;

Основные параметры риска:

- вероятность появления;
- степень воздействия на проект.

Минимизация рисков

Что делать с рисками?

1. *Уклонение от риска:* изменение проекта таким образом, чтобы он не был подвержен этому риску.
2. *Перевод риска:* изменение проекта таким образом, чтобы риску был подвержен кто-то другой или что-то другое.
3. *Принятие риска:* для проекта риск принимается как непредвиденное обстоятельство.

Контрактные обязательства могут быть приняты только при приемлемых рисках.

Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/engineering.html
E-mail: msp@luwrain.org
Канал в Телеграм: https://t.me/MarigostraRu