Требования к системе в виде модели. Область применения и ограничения

Ирина Сурова на опыте отдела системного анализа Kaspersky Lab

Irina.Surova@Kaspersky.com

Обо мне

- Системный аналитик в Лаборатории Касперского
- Ресурсный менеджер инфраструктурных аналитиков
- Занимаюсь методологией и инструментальной поддержкой системного анализа

Disclaimer: это мой субъективный анализ и выводы,
математической проверки на контрольной группе – не было.

О чем будем говорить :

- Контекст и терминология
- Состояние на начало отсчета
- Текущая метамодель
- Инструментальная поддержка
- Анализ и прогноз: приживется ли модель на проекте
- Планы на будущее

Контекст и вводная терминология

Модель это

В Wikipedia:

- абстрактное представление реальности в какой-либо форме (например, в математической, физической, символической, графической или дескриптивной), предназначенное для представления определённых аспектов этой реальности и позволяющее получить ответы на изучаемые вопросы[3]:80.

У нас это

- набор связанных элементов, который описывает систему
- с конкретной точки зрения (требований)
- с определенным уровнем детализации
- Отображается в виде визуальных диаграмм с участием элементов, содержащих текстовое описание
- Поддерживается в актуальном состоянии после изменений

Возможные сценарии использования элементов модели

- Сбор информации (Выявление требований)
- Анализ и синтез решения
- Представление информации (Документирование требований)
- Учет в целях управления (управление требованиями)

И под всем этим лежит служебный сценарий:

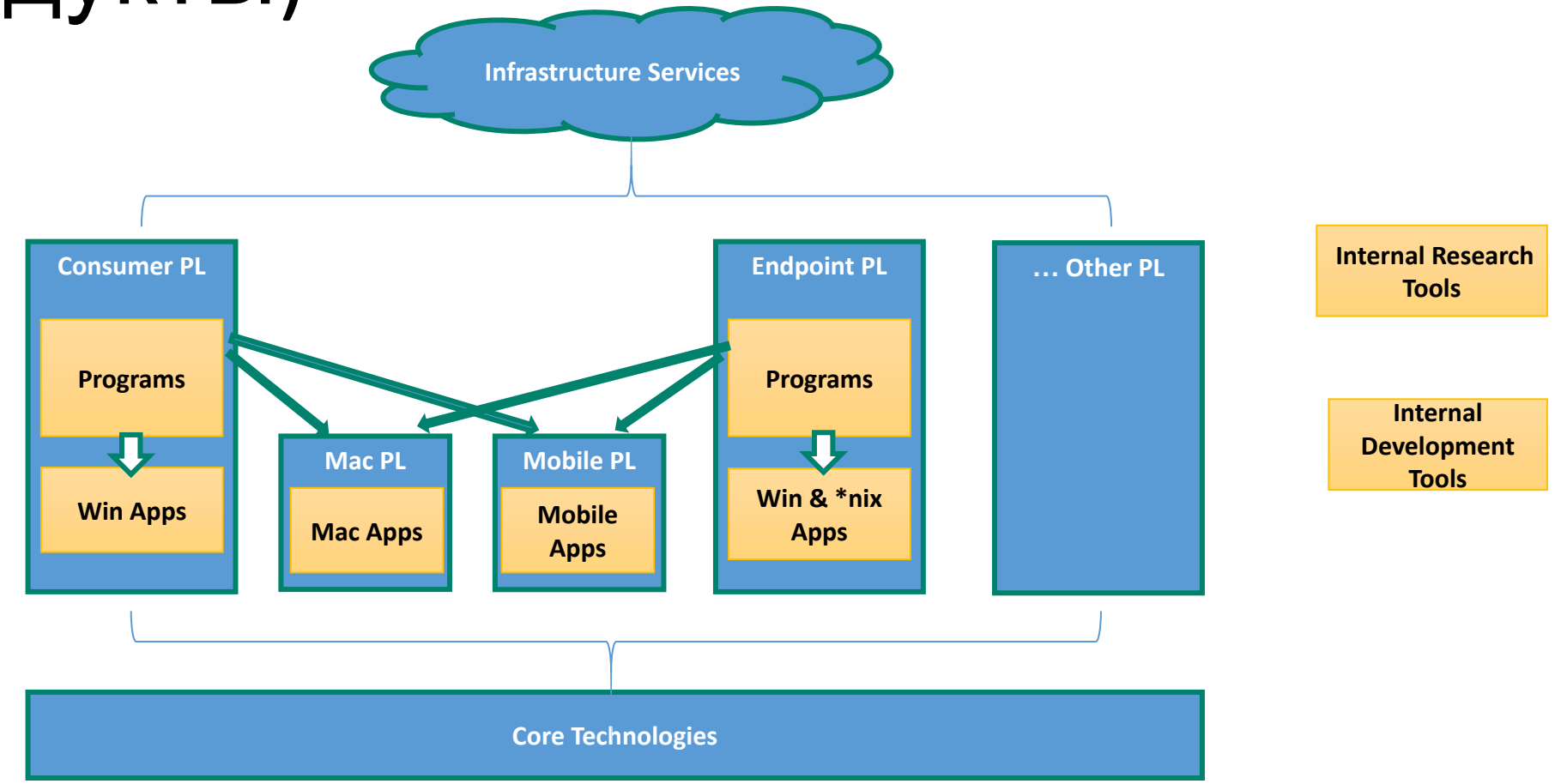
- Хранение требований и отношений между ними

Проектные роли и их отношение к модели

- Системный аналитик – создает и поддерживает модель
- Архитектор – использует модель при архитектурном проектировании
- Разработчик - использует модель или требования из нее при разработке
- Тестировщик - использует модель или требования из нее при проектировании тестов и тестировании
- Менеджер проекта – использует требования при планировании и анализе текущего статуса проекта

Вводная часть закончилась, пошла конкретика

Первоначальный контекст (проекты и продукты)



Первоначальный контекст (проекты и продукты)

- Продуктовая разработка, поддерживаемая сервисами (инфраструктурой) и in-house разработка
- PMDD (разнообразие вариантов и стилей ведения проектов в демократическом стиле)
- Специфичная предметная область (очень **техническая** область (ОС, файлы, участки памяти, вирусы, инжекты и т.д.), часто исследовательская работа (**результат заранее неизвестен**), наукоемкая (мат.методы) – **мало информации в свободном доступе**)
- В продуктах фокус на обработку, а не на ввод/хранение/вывод информации (есть интерфейсы, обработка, нет БД, миграций данных и т.д.), в сервисах фокус может быть на потоковую обработку данных

Первоначальный контекст (аналитики)

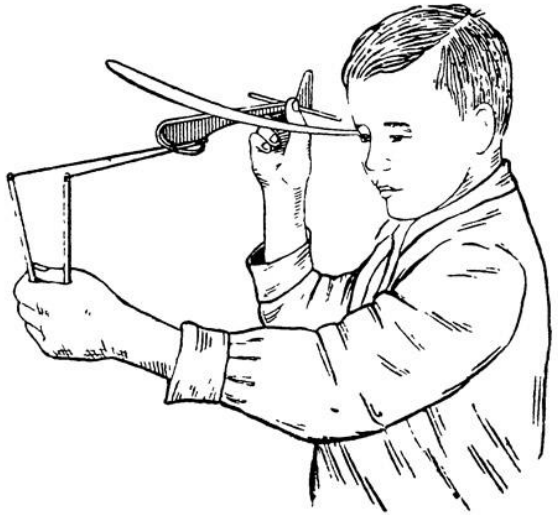
- Свежесформированный отдел анализа (раньше каждый сам на своей поляне, теперь все вместе, набор новых людей)
- Отсутствует общая методологическая и инструментальная база
- Разный уровень квалификации аналитиков и качества результатов, требуется единообразие для **управления аналитическими работами**
- Требуются практики **разработки требований** для очень разных систем в духе современных тенденций (mainstream – UML, Usecase)

Что было предложено

- Вот общий репозиторий требований в Enterprise Architect (общее место для хранения требований)
- Давайте писать требования в виде usecase’ов
- Вот методологическая группа, часть рабочего времени коллеги будут помогать (разрабатывать и описывать рекомендации по моделированию, готовить шаблоны отчетов для формирования SRS)
- Ходите на тренинги в Люксофт, читайте Вигерса, Коберна и т. д.

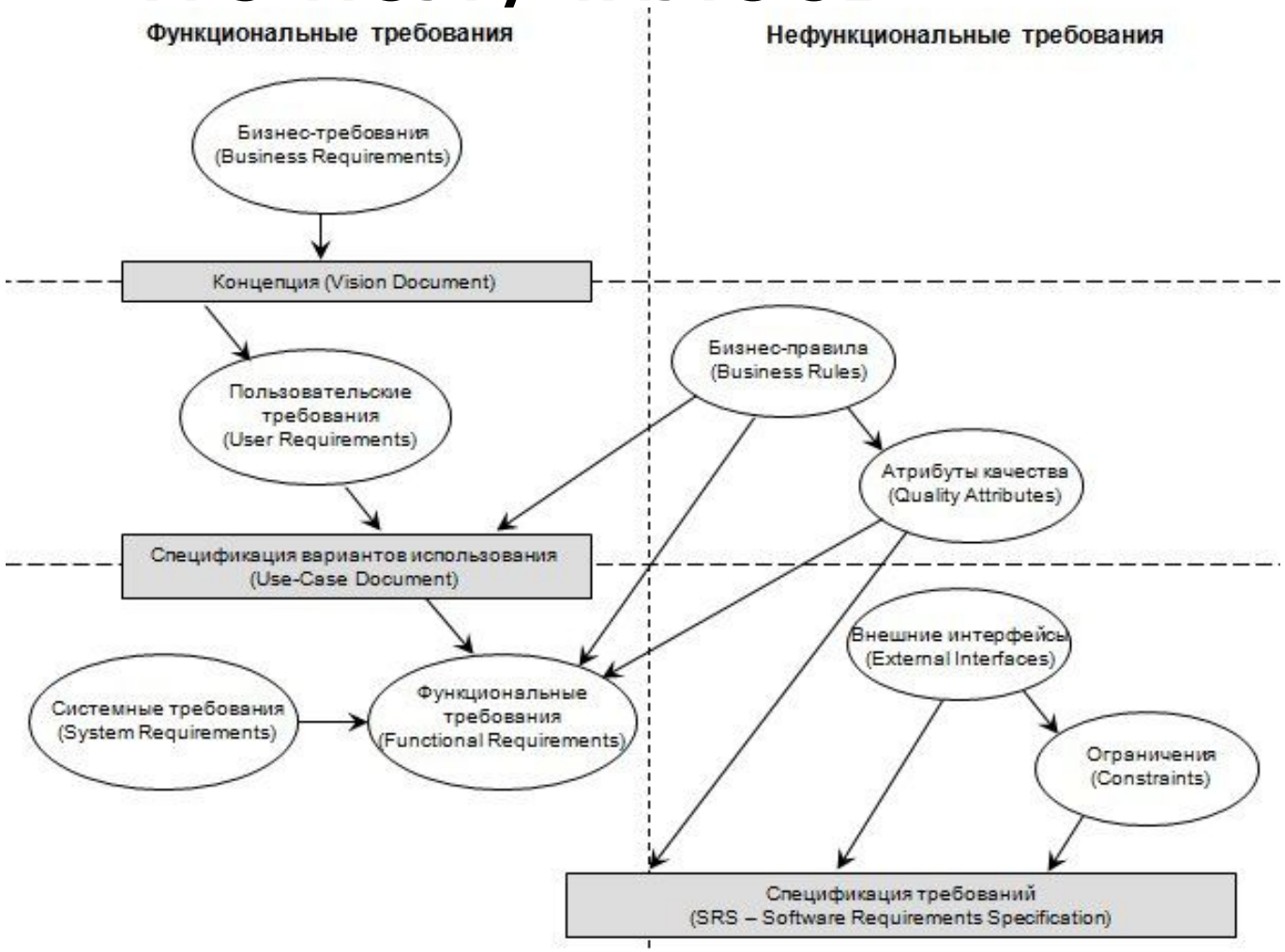
Общая идея

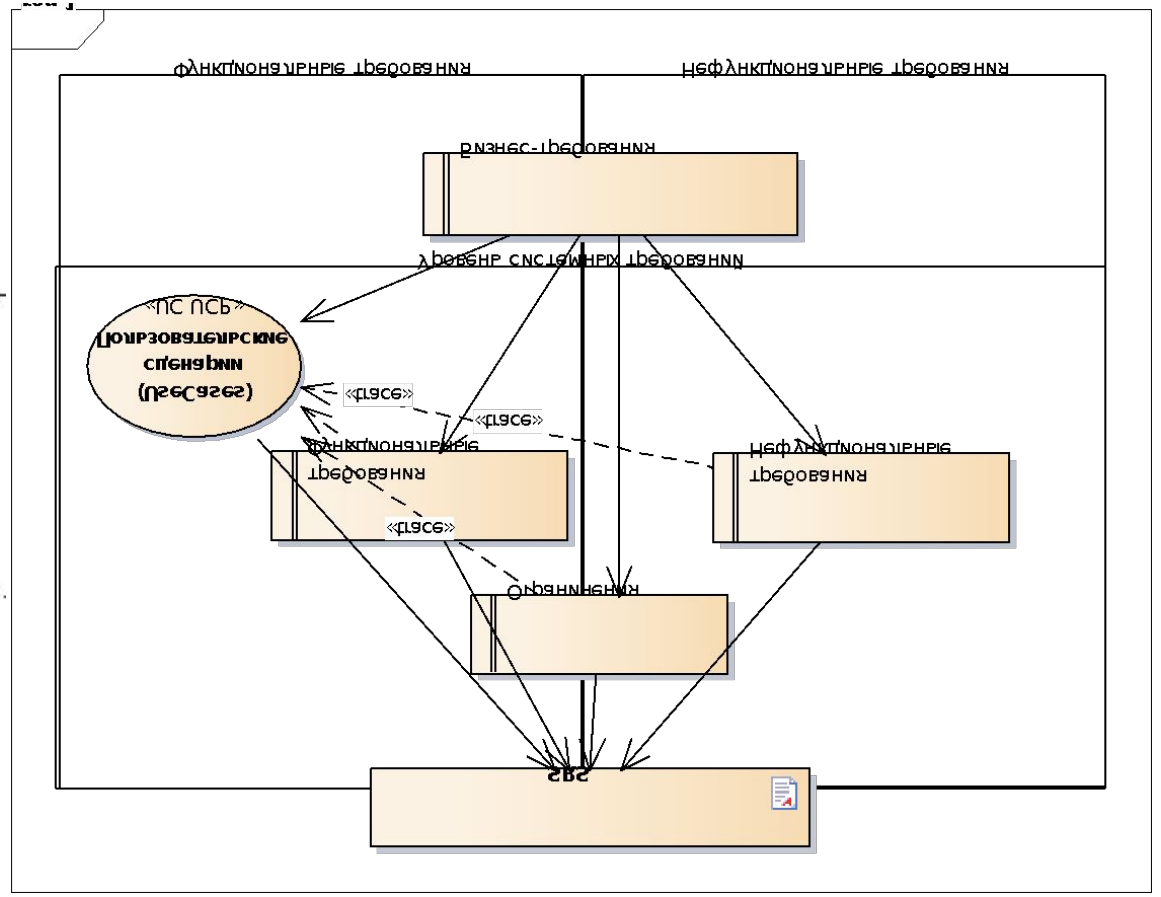
- Легкий старт
- Минимальные затраты на инструментальную поддержку
- Думайте сами, решайте сами, иметь или не иметь



Фиг. 69. Запуск модели планера при помощи рогатки.

Что получилось

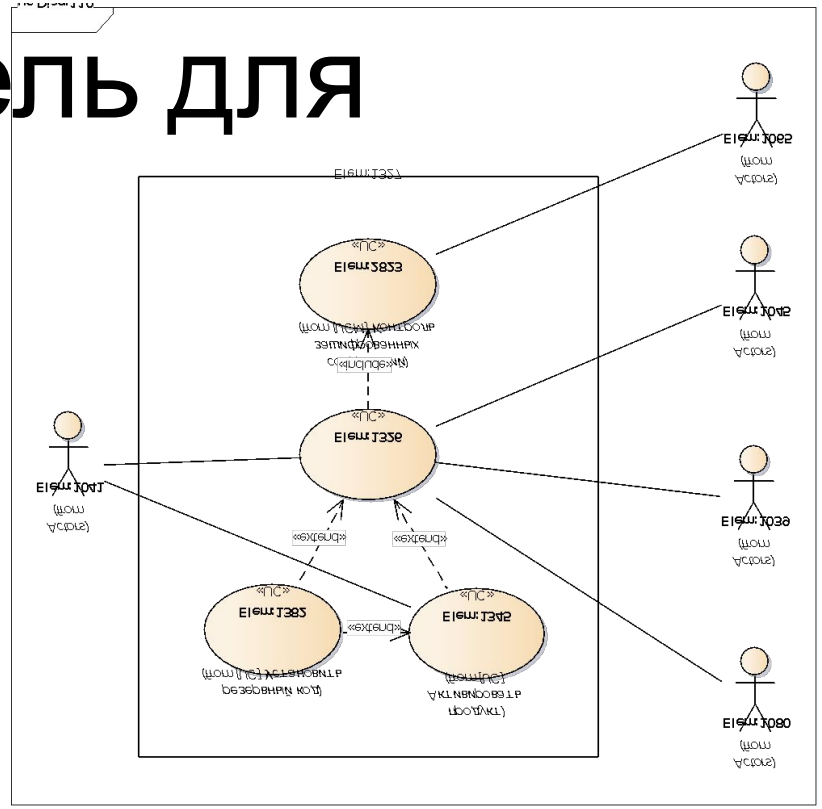


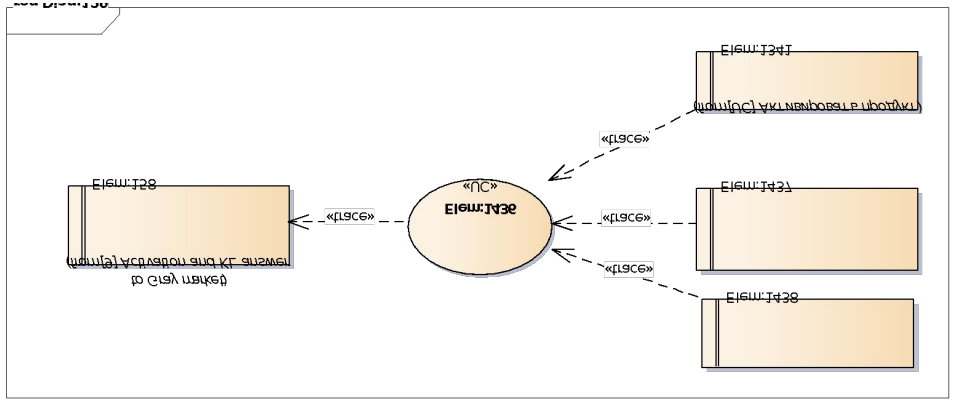


Слева – по Вигерсу, справа – используемые в САО ПК

Что получилось – метамодель для разработки требований

- Основные элементы – usecase, requirement, связь trace, диаграмма, actor
- 2уровневая иерархия требований в 1 проекте: бизнес-требование (BRQ) – Usecase (UC)+ функциональные и нефункциональные требования (SR)
- Основные диаграммы: диаграмма usecase, диаграмма трассировок UC-SR, BRQ-UC
- Дополнительные диаграммы: activity, DataFlow, StateChart, Classes, все, что захотите еще…





Структура пакетов проекта и виды SRS

- По конкретному бизнес требованию
- По конкретной функциональной области
- По всей версии целиком

Что получилось – вокруг метамодели

- Единым источником информации о требованиях является модель требований системы
- Демонстрация/представление новой функциональности для всех ЗЛ: отчет по модели по диаграмме трассировок конкретного бизнес-требования BRQ-UC+SR
- Демонстрация/представление требований на систему: отчет по модели по полному набору требований (пакет, группирующий требования по функциональной области, в нем UC+SR
- Элементы учета: либо SRS по BRQ (документ), либо отдельные UC, SR (отдельные элементы в учетной системе), либо только UC

Что получилось – метамодель для разработки требований – light вариант

- Основные элементы – usecase (текст a-la user story/Usecase 2.0), requirement, связь trace, диаграмма, actor
- 2уровневая иерархия требований в 1 проекте: бизнес-требование (BRQ) – Usecase (UC)+ функциональные и нефункциональные требования (SR)
- Основные диаграммы: диаграмма usecase, диаграмма трассировок UC-SR, BRQ-UC
- Дополнительные диаграммы: activity, DataFlow, StateChart, Classes…

Что получилось – метамодель для разработки требований - megalight

- Основные элементы – usecase (без текста, только название и уникальный Id), диаграмма, actor
- 2уровневая иерархия требований в 1 проекте: бизнес-требование (BRQ) – Usecase (UC)
- Основные диаграммы: диаграмма use case, диаграмма трассировок BRQ-UC
- Дополнительные диаграммы: activity, DataFlow, StateChart, Classes…

Преимущества модели

- Возможность работать над требованиями одной системы совместно, не мешая друг другу
- Наличие актуальных требований на всю версию (а не только на изменения)
- Легкий поиск требований в проекте при impact-анализе
- Скорость подготовки документов нужного формата (как отчетов по модели)
- Снижение затрат на обучение при переходах из проекта в проект
- Прозрачность требований для аналитиков смежных команд

Наша модель требований это

- набор связанных элементов, который описывает **систему**
- с конкретной точки зрения (требований)
- с **определенным уровнем детализации**
- Отображается **в виде визуальных диаграмм** с участием элементов, **содержащих текстовое описание**,
- Поддерживается в актуальном состоянии после изменений

Требования и сценарии использования инструмента (Enterprise Architect)

- Единое место работы для всех аналитиков с централизованным управлением (бекап, зеркалирование, управление доступом и т. д.)
- повторное использование элементов (соответствие: 1 сущность – 1 элемент в модели и использование его на всех диаграммах для всех связей)
- написание текстов и таблиц в элементах
- рисование диаграмм (диаграмма – SQL запрос по модели),
- генерация настраиваемых отчетов (word, html не подошел)
- Возможность экспорта в другие системы (TFS самописный, Confluence?)
- Версионирование

Инструментальная поддержка модели

- Поддержка репозитория (БД – передано в IT, у нас почти не занимает время)
- Операционная поддержка пользователей (вместе с базой) – 0,05 FTE
- Допиливание (переход на новые версии, оптимизация отчетов и прочие маленькие улучшалки) – 0,2 FTE
- Итоговые затраты – покупка и обновление лицензий, сервер для репозитория

Где прижилось (примеры проектов)

- Продукты (флагман, жесткий time-driven с фиксацией скоупа, много юридических согласований, много аналитиков на 1 проекте, много смежных команд, с которыми надо синхронизироваться)
- Порталы и сервисы (несколько человек на 1 проекте + все как выше)
- Набор проектов Core Technologies (одинаковые процессы разработки, много клиентов, 1 аналитик на несколько проектов)

Где не прижилось (примеры проектов)

- продукты (ограниченный доступ к экспертам и пользователям, новые технологии (долго-непонятно, что будет в результате), нет заморозки Scope/сроков, основной упор на добавление новой функциональности, мало связей с другими командами, мало аналитиков, умеющих моделировать)
- инфраструктурные сервисы (малое количество аналитиков, 1 аналитик на 1-2 проектах, уникальные сложные системы, специфичная предметная область, программисты-эксперты, мало аналитиков, умеющих моделировать)
- Сервисы (сильная воля руководства на создание полного справочника в текстовом виде)
- Продукт (часть многоплатформенной программы, долгий срок жизни бизнес-требований)

Везде: маленькие команды, короткие итерации, над требованиями работает вся команда, постепенная детализация требований

Прогноз приживаемости модели - 1

Модель приживется			**Модель не приживется**
Жесткие сроки релизов			
Жесткие обязательства по набору функциональности			
От системы требуется соответствие регламентам и сертификация			
Много поставок от смежных команд			Команда большую часть системы делает автономно
Понятные технологии			Исследование технологии для новой функциональности
Много обратной связи от пользователей			Мало обратной связи от пользователей
Изменение текущей функциональности			разработка новой функциональности

Прогноз приживаемости модели - 2

Модель приживется			**Модель не приживется**
			Высококвалифицированная команда разработки
Активная команда тестирования			
			Срок жизни бизнес-требования больше 1 релиза
			Короткие итерации в проекте
Команда получает от аналитика готовые требования			Над требованиями работает вся команда
Требования либо готовы, либо их нет			Принята постепенная детализация требований
Несколько аналитиков на проекте			1 аналитик на проекте
Несколько похожих проектов у аналитика			Непохожие проекты у аналитика
Аналитики с навыками моделирования			
Аналитики понимают и хотят моделировать			
Моделирование поддерживается руководством			
Есть инструментальная поддержка			

Планы на будущее

- По текущим целям
  - Версионирование
  - Связка с confluence – для отображения и согласования
  - Повышение эффективности (снижение временных затрат на подготовку документов, кастомизация уровня детализации под задачу)
- Новые вызовы:
  - Описание взаимодействия разных систем:
    - Сценарии использования в программах и трасссировки между сценариями разных систем
    - Описание архитектурных решений во взаимосвязанных системах при реализации требований программ
    - Карта сервисов
  - Прозрачность и новые законы о пользовательских данных
    - Требуется описывать, учитывать и анализировать данные и их потоки
  - Последствия ускорения и agile – команды хотят п**онятные** и **полные** требования **на 1 страницу**
    - Шаблон описания сервисов (частично связано с моделями)
    - Для продукта?