Natalia Bobrova

Системный аналитик за неделю

Natalia Bobrova

System analyst in a week

http://www.litres.ru/pages/biblio_book/?art=71377831
SelfPub; 2025

Аннотация

В книге представлены наиболее важные концепции и инструменты, необходимые начинающему аналитику. Она позволит структурировать знания для дальнейшего и более глубокого погружения в данную сферу, сформирует понимание того, чем должен заниматься системный аналитик и какими базовыми знаниями должен обладать.

Книга построена по принципу разбора основных вопросов задаваемых на собеседовании на позицию младшего системного аналитика, что позволит вам не только уверенно чувствовать себя на интервью, но и успешно справляться с профессиональными задачами в дальнейшем.

Содержание

Natalia Bobrova
System analyst in a week

– это мощный инструмент, который помогает видеть прогресс, находить проблемы и налаживать работу. Чтобы улучшить производительность и создать оптимальную рабочую среду, важно понимать и применять эти показатели. Рассмотрим основные Agile метрики.

Скорость команды (Velocity)

Ключевая метрика в Scrum Framework и Agile. Помогает измерить, насколько продуктивна команда, и определяет объем работы, с которым участники команды справляются за спринт. Позволяет понимать, сколько задач они смогут закрыть в будущем. Как правило, скорость измеряют в Story Points или часах.

Преимущества Velocity.

Прогнозирование. Метрика Velocity позволяет предсказывать, сколько задач получится закрыть в будущем. Если команда знает свою среднюю скорость, она может планировать объем работы на предстоящие спринты.

Оценка производительности. Анализ скорости на протяжении нескольких спринтов помогает командам увидеть, выросла производительность или, наоборот, снизилась.

Управление ожиданиями. Когда специалисты знают свою скорость, они могут называть заказчикам более реалистич-

ные сроки, за которые справятся с задачами.

Метрика Capacity (емкость).

В Agile и Scrum Framework помогает измерить объем работы, который команда может сделать за обозначенный период времени, например, за спринт. С этой метрикой проще учитывать реальное количество доступных ресурсов, чтобы планировать свои задачи эффективнее.

Преимущества Capacity

Планирование на основе реальной загрузки. Capacity позволяет планировать задачи с учетом фактической доступности участников и при этом избегать перегрузки или недогрузки.

Прогнозирование выполнения. Метрика помогает команде определить, сколько задач реально можно закрыть в течение спринта.

Гибкость в планировании. Учитывать Capacity особенно полезно, когда меняется состав команды или индивидуальная загруженность участников. Это позволяет оперативно корректировать объем задач.

Прозрачность и управляемость. Capacity дает возможность команде и заинтересованным лицам четко понимать, сколько ресурсов доступно, и эффективно ими управлять.

Время выполнения задачи (Cycle Time).

Метрика процесса, которая измеряет, сколько времени

уходит на задачу. Cycle Time в системе управления разработкой показывает командам, насколько быстро они могут работать, и находить проблемные зоны в процессе.

Преимущества Cycle Time.

Анализ производительности. Метрика показывает, насколько эффективно получается справляться с задачами.

Выявление узких мест. Благодаря этой метрике участники могут обнаружить задержки и проблемные места, а это помогает оптимизировать рабочий поток.

Улучшение планирования. Когда команда понимает, сколько в среднем нужно времени, чтобы выполнять задачи, она может правильно планировать свои спринты и устанавливать более реалистичные сроки.

Метрика Wasted Time.

Wasted Time используют, чтобы измерять время, которое команда тратит на неэффективные или непродуктивные задачи – когда они не вносят прямого вклада в достижение целей проекта. Это может быть время на ожидание ресурсов, частые переключения между задачами, участие в слишком долгих и неструктурированных совещаниях.

Преимущества Wasted Time.

Выявление проблемных зон. Метрика помогает команде найти области проекта, которые приводят к потерям времени. Это может быть неоптимальная организация встреч, частые изменения требований или медленная коммуникация с

заказчиками.

Повышение эффективности. Когда команда понимает, на что тратит время впустую, у нее появляется возможность оптимизировать свои процессы. Например, можно сократить длительность и количество встреч .

Улучшение планирования. Анализ временных потерь помогает точнее определять, сколько времени нужно, чтобы выполнить задачи. Будущие спринты удобнее планировать с учетом этой информации.

Повышение прозрачности. Введение метрики помогает команде видеть реальную ситуацию и создать среду, в которой можно открыто обсуждать проблемы.

Фокус на ключевых задачах. Благодаря учету потерянного времени команда начинает фокусироваться на важных задачах и избегать лишних переключений и отвлечений.

Груминг в IT: уточнение и приоритизация задач

Груминг Бэклога – это процесс в Scrum, который состоит из регулярного обсуждения и уточнения задач. Обычно его проводит владелец продукта. От этого процесса зависит, насколько качественными и точными будут метрики эффективности проекта и команды.

Груминг в Scrum включает в себя:

Обсуждение задач – команда и владелец продукта обсуждают существующие задачи, уточняют детали и требования.

Приоритизацию – задачи сортируют по степени важности

и срочности, чтобы команда могла сосредоточиться на самых значимых.

Оценку – команда оценивает объем работы для каждой задачи, чтобы точнее рассчитывать метрики.

Kanban.

Канбан — это система постановки задач и организации рабочих процессов для эффективного достижения поставленных целей. Данная методология предполагает прозрачность продвижения работы и является одним из подходов Agile.

Канбан помогает отслеживать процесс выполнения поставленных задач и распределять нагрузку между сотрудниками. Этот метод позволяет эффективно управлять работой команды и следить за сроками ее выполнения. Визуализация всех этапов позволяет каждому сотруднику быть в курсе продвижения процессов.

Ценности метода.

1. Прозрачность – открытый обмен информацией;

2. Баланс – равновесие между нагрузкой и возможностями;

3. Сотрудничество – совместная работа участников команды и ее совершенствование;

4. Фокус на заказчике и его потребностях – создание про-

дукта, который нужен клиенту;

5. Поток – непрерывная работа;

6. Лидерство – вдохновление своим примером других участников. При этом нет иерархии, понятие применимо на всех уровнях;

7. Понимание – знание всеми участниками целей развития команды;

8. Согласие – совместное движение к целям и совершенствованию;

9. Уважение – понимание и положительная оценка всех участников команды.

Основные принципы методологии канбан.

1. Визуализация процессов. Важно, чтобы все поставленные задачи были добавлены в план. Их статус нужно обновлять по мере завершения каждого пройденного этапа. Такой подход позволяет шаг за шагом продвигаться вперед, следить за прогрессом и видеть задачи, решение которых требует большего времени и помощи.

2. Группировка задач. Это делают на основании статусов. Самый простой способ – разделить задачи на три колонки: «Надо выполнить», «Выполняется», «Выполнена». Такой подход предполагает перемещение поставленной задачи с одного этапа на другой и визуализирует рабочий процесс.

3. Внимание к незавершенным задачам. Если задачи подвисают на каком-то этапе, следует разобраться в причинах,

распределить ресурсы в случае необходимости или оказать нужную поддержку, чтобы завершить работу.

4. Постоянное совершенствование. Контроль за соблюдением сроков выполнения задач и их перемещением с одного уровня на другой в системе канбан помогает видеть слабые места в процессах. Поэтому вы можете четко определить, где нужно уделить больше времени работе, где меньше, а в каких ситуациях стоит скорректировать нагрузку.

Методология канбан подразумевает гибкий подход к организации работы, что позволяет легко добавлять новые задачи, изменять их приоритетность, увеличивать или сокращать сроки выполнения. Однако, несмотря на все явные преимущества такого подхода управления, у него все же есть и недостатки, о которых стоит знать. В следующем разделе вы узнаете о плюсах и минусах метода канбан.

Инструменты.

Главный инструмент канбана – доска с карточками. Это может быть физическая меловая доска, магнитная, со стикерами или электронная. К ней должны иметь доступ все участники команды в любой момент времени.

Колонки доски:

1. «Бэклог» – поле для всех карточек, пул задач, который может пополняться, сортироваться по приоритетности;

2. «В процессе» – включает несколько видов внутренних колонок, адаптированных под команду и обозначающих раз-

ные этапы работы над карточкой;

3. «Готово» – полностью выполненные задачи, которые не требуют от команды дальнейших действий.

На одной доске можно вести сразу несколько проектов, для этого используют карточки разных цветов или swimlanes – горизонтальные разделители. Каждая карточка в канбане может содержать дополнительную информацию с описанием задачи, именем того, кто над ней работает, ее приоритет, дедлайн. Задачи могут быть ежедневными, еженедельными, ежемесячными.

Правила работы с карточками.

Основные правила Kanban при работе с карточками направлены на непрерывное течение процесса, регулирование сроков и внимание к задачам, которые по каким-то причинам не движутся по потоку:

1. WIP-лимит может быть разным для конкретных специалистов или отделов в зависимости от их ресурсов. Цель применения лимита – направить фокус сотрудника на одну задачу, вместо того, чтобы он пытался делать несколько сразу.

2. Максимальным лимитом регулируется количество карточек в каждом столбце. Лимит основывается на реальных возможностях команды, в него входят все карточки, которые находятся в работе.

3. Нельзя начинать новую карточку, если не сделана предыдущая. Если задача по каким-то причинам не может

быть завершена, ее нужно перенести в колонку Blocked и искать другие способы ее завершения.

Главный закон эффективности канбана – «прекращайте начинать, начните заканчивать»

Приоритетность задач в канбане зависит от их важности для бизнеса или клиента, размера недополученной прибыли или издержек в случае, если они не будут сделаны в срок. Чтобы участникам команды было понятнее, какая работа важнее, внедряют классы обслуживания, на карточках их обозначают символами:

–срочный – нельзя откладывать;

–с фиксированной датой – нужно сделать к определенному сроку;

–стандартный – издержки растут пропорционально задержке, желательно сделать вовремя;

–нематериальный – стоимость задержки растет медленно, задача несрочная, делать ее сейчас необязательно, если есть более важные.

Преимущества Kanban.

Гибкость планирования. Система канбан построена таким образом, что команда концентрируется на одной конкретной задаче, несмотря на то, что их может быть несколько. При этом, руководитель может изменять приоритетность работы, не затрагивая рабочий процесс. По завершению одной задачи команда приступает к следующей.

Контроль сроков выполнения. Методология канбан позволяет отслеживать рабочий процесс, оптимизировать его длительность и прогнозировать время, которое потребуется для решения будущих задач.

Повышение эффективности работы. Многозадачность снижает качество работы и замедляет продвижение вперед. Чем больше зависает нерешенных задач, тем чаще приходится между ними переключаться. Метод канбан минимизирует застои, поскольку помогает быстро выявлять слабые места. Это позволяет сокращать время работы над задачей и повышать качество результата.

Наглядность продвижения работы. Одно из главных преимуществ управления процессами по методу канбан. Когда все члены команды имеют доступ к поставленным задачам и результативности продвижения, то легче выявить проблемы и устранить их.

Минусы Kanban.

Не подходит для долгосрочного планирования. Метод канбан рассчитан на достижение краткосрочных целей. Работа выстраивается на решении актуальных задач, при этом их приоритетность может меняться в зависимости от обстоятельств.

Не подходит для больших команд. Чем больше человек задействовано в рабочем процессе, тем сложнее контролировать выполнение задач. Поэтому, лучше всего, чтобы в одной

команде было не больше десяти человек, в идеале – пять.

Когда и кому нужен канбан.

Выделяют несколько характерных сигналов, которые указывают на возможность и даже необходимость внедрения канбана:

–команда выполняет много однотипных задач, и важным улучшением было бы делать это быстрее;

–участники команды постоянно перегружены – нет времени на улучшение, им бы справиться с имеющейся нагрузкой;

–регулярно срываются дедлайны;

–руководителю кажется, что вокруг хаос – непонятно, кто чем занят и когда поставленные задачи будут выполнены;

–исполнителю непонятно, кто ставит задачи и чьи распоряжения приоритетнее.

Если в команде имеются две и более проблемы из списка – канбан может стать эффективным способом усовершенствовать работу. Что касается бизнеса, то метод применим к любой сфере, где можно выделить этапы и типы работ.

LeSS.

Это фреймворк, позволяющий применить принципы скрама в больших проектах.

Одним из принципов LeSS является «получать большее за счёт меньшего», что значит не создавать бюрократии и об-

ходиться без лишних ролей, процессов и артефактов.

LeSS предлагает два различных фреймворка масштабного Scrum. Большая часть элементов LeSS, связанных с масштабированием, фокусирует внимание всех команд на всем продукте нежели на “своей части” продукта. Глобальный и сквозной (“end-to-end”) фокус, наверное, самые важные и требующие решения проблемы при масштабировании. Вот два фреймворка, которые по сути представляют собой масштабируемый однокомандный Scrum:

–LeSS: до 8 команд (каждая из которых до 8 человек).

–LeSS Huge: до нескольких тысяч человек, работающих над одним продуктом.

Составляющие LeSS :

–Единственный Бэклог Продукта (потому что он для Продукта, а не для команды),

–Единые критерии готовности для всех команд,

–Один Потенциально пригодный к использованию и выпуску Инкремент Продукта в конце каждого Спринта,

–Единственного Владельца Продукта,

–Множество полностью кросс-функциональных команд (без узкоспециализированных команд),

–Общий Спринт.

В LeSS все команды работают в одном общем спринте для создания общего готового к выпуску Продукта каждый спринт.

Отличия LeSS от Scrum.

Первая часть планирования Спринта: В этом мероприятии помимо единственного Владельца Продукта участвуют представители всех команд. Участники команд самостоятельно решают, как распределить элементы Бэклога между командами. Участники команд также обсуждают и выявляют возможности для совместной работы над связанными элементами Бэклога.

Вторая часть планирования Спринта: Она проводится независимо (и зачастую в параллель) каждой из команд, однако иногда две и более команд могут проводить данную часть планирования в одном помещении (в разных частях) в целях простой координации и обучения.

Ежедневный Скрам: Это мероприятие также проводится независимо каждой из команд. Однако, для более полного обмена информацией между командами, участник команды А может быть наблюдателем на Ежедневном Скраме команды Б.

Координация: Техники: “Просто поговорить”, “Коммуникации в коде”, “Путешественники”, “Открытое пространство” и “Сообщества”.

Общее уточнение Бэклога Продукта: Возможно проведение короткой и опциональной встречи по общепродуктовому уточнению в бэклога, в котором участвуют Владелец Продукта и представители всех команд. Основное назначение данного мероприятия – это решить каким командам пред-

почтительнее работать над какими элементами Бэклога, и, соответственно, выбрать эти элементы бэклога для дальнейшего более глубокого уточнения на командной сессии Уточнения Бэклога Продукта.

Уточнение бэклога продукта: Также как и в однокомандном Скраме, LeSS явно требует только проведение однокомандного Уточнения Бэклога Продукта. Но общепринятой и полезной практикой является проведение многокомандного Уточнения Бэклога Продукта. Когда две или более команд проводят уточнение в одном помещении, чтобы усилить обучение и координацию.

Обзор спринта: Помимо единственного Владельца Продукта в этом мероприятии участвуют члены всех команд, потребители/пользователи и другие заинтересованные лица. Для фазы инспекции инкремента продукта и новых элементов бэклога, можно использовать техники “базар” или “научная выставка”: большое помещение с несколькими зонами, в каждой из которых происходит презентация и обсуждение разработанных элементов Бэклога одной из команд.

Общая ретроспектива: Это новое мероприятие, которого нет в однокомандном Скрам, и его цель – улучшить систему в целом, вместо фокусировки на одной отдельной команде. Максимальная длительность этого мероприятия 45 минут на каждую неделю Спринта. В нем участвуют Владелец Продукта, Скрам-Мастера и ротирующиеся представители каждой из команд.

SAFE.

Scaled Agile Framework (сокращенно SAFe) – это методология для работы с командой. Она помогает, используя методы Agile, управлять сложными проектами в больших командах – от 50 человек.

Три уровня управления разработкой в SAFe.

1. *Командный уровень*

Единица управления на данном уровне – команда, реализующая пользовательские истории из бэклога. Роли на данном уровне такие же, как и в классическом Scrum: Владелец продукта, Scrum-мастер, член команды разработки.

2. *Программный уровень*

На этом уровне все ресурсы, команды, заинтересованные лица кооперируются вокруг одной важной цели, чаще всего представляющую собой поток создания ценности (Value Stream) или продукт. Синхронизируют свою работу команды при помощи совместных сессий планирования (Program Increment Planning) в начале каждого квартала и демонстрации интегрированного инкремента продукта (System Demos) каждые 2 недели и ретроспективы (Inspect & Adapt) в конце каждого квартала.

Соответственно, все роли и процессы ориентированы на поставку ценных элементов функциональности. Метафорой

этого процесса является “Поезд” (Agile Release Train).

ART – это долгоживущая группа команд, заинтересованных лиц и других участников, объединённых общей целью, создающая в едином ритме общее решение или его часть. Длина итераций и частота общего планирования внутри поезда фиксирована.

Управляет «Поездом», соответственно, «Машинист» (Release Train Engineer). Он коучит весь «экипаж» поезда для повышения эффективности работы, взаимодействует с заинтересованными сторонами, фасилитирует общие собрания поезда поставки. Release Train Engineer больше похож на Scrum-мастера, но отвечающего не за одну, а за много команд и их взаимодействие между собой. В терминах PMI «Машинист» соответствует Менеджеру программы по уровню ответственности и исполняемым функциям (The PM role in a lean and agile world).

У поезда также есть главные «пассажиры» – Представители бизнеса (Business Owners). Это основные заинтересованные лица, которые отвечают за финансовые показатели Поезда. Business Owners – это руководители, которые будут получать выгоду от создаваемого решения, пользоваться им или продавать его.

Управляет продуктом и владеет бэклогом поезда команда Продуктового менеджмента (Product Management). В неё входят Владельцы продуктов отдельных команд, а также Продуктовые менеджеры. Они выявляют потребности кли-

ентов, формируют дорожную карту и приоритезируют элементы функциональности. Говоря простыми словами, Представители бизнеса ставят цели, а команда Продуктового менеджмента стараются их достичь силами команд.

Также на программном уровне есть роль Системного архитектора (System Architect). Архитектор определяет архитектуру будущего решения, направляет работу команды с технической стороны системы, взаимодействия подсистем и нефункциональных требований к системе.

3. *уровень портфеля*

Цель портфельного уровня – согласование стратегии компании с реализацией портфеля за счёт организации процессов вокруг потоков создания ценности. Этот уровень появляется, если у организации несколько потоков создания ценности.

Портфель в SAFe состоит из Потоков создания ценности. Это могут быть продукты или направления деятельности организации. На этом уровне определяется стратегия инвестирования, бюджеты и показатели эффективности. Также на этом уровне реализуется функция принятия решения самого высокого уровня – портфельное управление (Lean Portfolio Management)

V-модель.

V-модель – это тип модели SDLC, в которой процесс выполняется последовательно в V-образной форме. Модель основана на объединении фазы тестирования с каждой соответствующей стадией разработки. Разработка каждого шага напрямую связана с этапом тестирования. Следующая фаза начинается только после завершения предыдущей. Каждый этап разработки, напрямую связан с тестированием этого этапа.

Этап проектирования V-модели.

Анализ требований – этап включает общение с заказчиком с целью выделить его требования и ожидания от проекта. Другое название этого этапа – сбор требований.

Проектирование системы – этап включает проектирование системы и настройку оборудования и коммуникаций для разработки продукта.

Архитектурный дизайн – системный дизайн, разбивка на модули, выполняющие различные функции. Передача данных и связь между внутренними модулями и с другими системами.

Разработка модулей- система разбивает на небольшие модули. Определяется детальный дизайн модулей, также из-

вестный как Low-Level Design (LLD).Низкоуровневое проектирование (LLD) – процесс проектирования на уровне компонентов, который следует за пошаговым процессом уточнения. Предоставляет подробные сведения и определения фактической логики для каждого компонента системы.

Этапы тестирования V-модели.

Модульное тестирование – модульного тестирования разрабатываются на этапе проектирования модуля. Эти планы модульного тестирования выполняются для устранения ошибок на уровне кода или модуля.

Интеграционное тестирование – после завершения модульного тестирования выполняется интеграционное тестирование. При интеграционном тестировании модули интегрируются, и система тестируется. Интеграционное тестирование выполняется на этапе проектирования архитектуры. Этот тест проверяет связь модулей между собой.

Системное тестирование – тестирование тестирует все приложение с его функциональностью, взаимозависимостью и связью. Оно проверяет функциональные и нефункциональные требования разработанного приложения.

Пользовательское приемочное тестирование (UAT): UAT выполняется в пользовательской среде, напоминающей производственную среду. UAT проверяет, что поставленная система соответствует требованиям пользователя и готова к использованию в реальном мире.

Принципы V-модели.

От большого к малому: в V-модели тестирование выполняется в иерархической перспективе, например, требования, определенные командой проекта, создают этапы проекта высокого уровня и детального проектирования. По мере того, как каждый из этих этапов завершается, требования, которые определяются, становятся все более и более уточненными и подробными.

Целостность данных / процессов: этот принцип гласит, что успешное проектирование любого проекта требует интеграции и согласованности как данных, так и процессов. Элементы процесса должны быть идентифицированы для каждого требования.

Масштабируемость: этот принцип гласит, что концепция V-Model обладает гибкостью, позволяющей приспособить любой ИТ-проект независимо от его размера, сложности или продолжительности.

Перекрестные ссылки: прямая корреляция между требованиями и соответствующей деятельностью по тестированию называется перекрестными ссылками.

Материальная документация: этот принцип гласит, что каждый проект должен создавать документ. Эта документация требуется и применяется как группой разработки проекта, так и группой поддержки. Документация используется для поддержки приложения, когда оно становится доступ-

ным в производстве.

Преимущества V-модели:

–Каждая стадия имеет конкретные результаты;

–Более высокие показатели по сравнению с каскадной моделью по причине того, что тестирование начинается на ранних этапах;

–Экономия времени по сравнению с каскадной моделью может достигать 50%;

–Отлично подходит для небольших проектов, где все требования к продукту очевидны сразу;

–Полноценная реализация доступных ресурсов.

Недостатки V-модели:

–Отсутствие гибкости, как и в случае с каскадной моделью. Вносить изменения на поздних этапах будет трудно и дорого;

–Сама разработка начинается строго с началом соответствующей стадии, то есть, никаких прототипов на ранних этапах не разрабатывается;

–Контроль рисков затруднен: нет определённого способа решения критических проблем, обнаруженных на этапе тестирования.

Когда использовать V-модель.

Преимущественно в тех ситуациях, когда важно выпол-

нить проект быстро и с наименьшими затратами. В целом V-модель и каскадная модель очень похожи, но первая предоставляет ощутимо большую экономию времени.

Спиральная модель.

Спиральную модель можно описать как повторяющуюся последовательность циклов разработки с непрерывным контролем рисков.

Спиральная модель разработки программного обеспечения не так широко известна, как, например, Scrum или Kanban. Причина в том, что данный подход может оказаться довольно затратным в применении. Именно поэтому он не очень хорошо подходит для небольших проектов. В спиральной модели особое внимание уделяется управлению рисками. На практике это означает, что фаза оценки и разрешения рисков является критичной для успеха проекта. Контроль рисков, в свою очередь, требует проведения специфического анализа на каждой итерации. Для регулярного обзора и анализа текущего состояния проекта необходимы дополнительные навыки и ресурсы.

Спиральная модель состоит из четырех главных повторяющихся стадий. В ходе процесса разработки проект несколько раз проходит через все эти фазы. Каждая такая итерация

называется спиралью.

Четыре главные фазы :

1. *Определение целей, альтернатив, ограничений, или фаза планирования*. С этой стадии начинается работа над проектом. Команда разработчиков формулирует цели проекта, основные требования (такие как, например, Business Requirement Specifications, или BRS, System Requirement Specifications, или SRS), возможный дизайн и т.д. На последующих спиралях требования формируются согласно отзывам, полученным от заказчика. 2. *Анализ, определение и разрешение рисков является одной из самых значимых стадий разработки.* В данном контексте, риски – это возможные события и состояния проекта, препятствующие достижению командой разработчиков поставленных целей. Существует довольно обширный диапазон возможных рисков, от тривиальных и легко преодолимых, до крайне серьезных. Главной задачей для команды разработчиков является выявление всех возможных рисков и присвоение им определенного уровня приоритета на основе их значимости. Следующим шагом является разработка возможных стратегий преодоления этих рисков. В итоге этих действий возможны изменения в последующих стадиях разработки. В качестве результата работы на этом этапе создается прототип. 3. *Фаза разработки.* На этом этапе происходит разработка и последующее тестирование продукта. Во время первой итерации, ко-

гда общие требования еще не так четко сформулированы, разрабатывается так называемый концепция будущего продукта, которая необходима для получения отзыва заказчика. На последующих витках спирали рабочие версии продукта, или билды , отправляются заказчику. Это позволяет получить более детальный отзыв и четче сформулировать требования.

4. *Планирование следующей фазы.* На этом этапе вся полученная информация используется для планирования дальнейших этапов разработки.

Достоинства модели:

–Мониторинг рисков является одной из главных особенностей, делающих данную модель особенно привлекательной в том случае, если вам предстоит управление большим, сложным и дорогостоящим проектом. Более того, проект будет более прозрачным, поскольку спиральная модель изначально была спроектирована таким образом, чтобы каждая итерация тщательно анализировалась;

–Заказчик может увидеть работающую версию продукта уже на ранних стадиях жизненного цикла ПО;

–Изменения могут быть внесены на поздних стадиях разработки;

–Строгий контроль над документацией, как результат постоянного анализа рисков.Проект может быть разделен на несколько частей и те из них, которые, согласно анализу, ока-

жутся более рискованными, могут быть реализованы на ранних стадиях. Такой подход может снизить трудности, связанные с управлением проектом;

Недостатки модели:

–Мониторинг рисков требует дополнительных ресурсов, а значит, эта модель может оказаться весьма затратной. Каждая итерация требует отдельной экспертизы, что делает управление проектом сложнее. Именно поэтому спиральная модель плохо подходит для небольших проектов;

–Большое количество промежуточных стадий разработки. Как следствие большой объем документации;

–На самых ранних стадиях дата завершения работы над проектом может быть неизвестна, что также усложняет контроль над процессом разработки

Применение спиральной модели.

Спиральная модель может оказаться полезной, если предстоит работа над проектом со средним или высоким уровнем возможных рисков, заказчик не может предоставить достаточно четкий список требований к конечному продукту или эти требования достаточно сложные, а также в том случае, когда ожидаются значительные изменения в процессе разработки.

Требования.

Требования.

Требования – это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы.

IEEE Standard Glossary of Software Engineering Terminology (1990) определяет требования как:

1. Условия или возможности, необходимые пользователю для решения проблем или достижения целей;

2. Условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;

3. Документированное представление условий или возможностей для пунктов 1 и 2.

Это определение охватывает требования как пользователей (внешнее поведение системы), так и разработчиков (некоторые скрытые параметры). Термин пользователи следует распространить на всех заинтересованных лиц, так как не все, кто заинтересован в проекте – пользователи.

Классификация требований.

Бизнес-требования – это высокоуровневые формулировки целей, задач и потребностей предприятия. Они описывают причины, по которым был инициирован проект, задачи, которые будут достигнуты в ходе проекта, а также метрики, которые будут использоваться для измерения его успешности. Бизнес-требования описывают потребности организации в целом, а не группы или заинтересованных в нем сторон. Они разработаны и определены на основе анализа предприятия.

Требования заинтересованных сторон – это формулирование нужд конкретного заинтересованного лица или класса заинтересованных лиц. Эти требования описывают потребности, которые описывают что заинтересованная сторона имеет и как эта заинтересованная сторона будет взаимодействовать с решением. Требования заинтересованных сторон служит в качестве моста между бизнес-требованиями и другими классами требований к решению. Они разработаны и определены на основе анализа требований.

Требования к решению описывают характеристики к решению, которые соответствуют бизнес-требованиям и требованиям заинтересованных сторон. Они разработаны и определены на основе анализа требований. Их часто делят на подкатегории, особенно в случаях, когда требования описывают программное решение:

Функциональные требования описывают поведение решения и информацию о том, чем оно будет управлять. Эти требования описывают возможности системы, которые доступны к выполнению, в части режимов работы или операций – действия или реакции конкретного ИТ-приложения.

Нефункциональные требования фиксируют условия, которые напрямую не относятся к поведению или функциональности решения, а скорее описывают условия окружающей (внешней) среды, в условиях которой решение должно сохранять эффективность и качества, которыми система должна обладать. Они также известны как требования качества или дополнительные требования. Они могут включать в себя требования, связанные с мощностью (емкостью), скоростью, безопасностью, доступностью, информационной архитектурой и пользовательским интерфейсом.

Переходные требования описывают возможности, которыми решения должно обладать для осуществления перехода из текущего состояния предприятия к желаемому (требуемому) состоянию в будущем. Эти требования статут ненужными, как только этот переход завершится. Они отличаются от других типов требований, потому что они всегда носят временный характер и потому, что они не могут быть разработаны, пока оба решения, существующее и новое, не будут определены. Как правило, они охватывают преобразование данных из существующих систем, пробелы в знаниях и квалификации, которые должны быть решены, а также другие

соответствующие изменения, необходимые для достижения желаемого будущего состояния. Их разрабатывают и определяют посредством проведения оценки и валидации (проверки) решения.

Методы сбора требований:

– Мозговой штурм;
– Анализ документов;
–Фокус-группы;
– Анализ интерфейсов;

Конец ознакомительного фрагмента.