Применение Agile Kanban для автоматизации работы городской больницы (Часть 1)

Мартынов Михаил Васильевич

Аннотация: в статье описывается автоматизация работы больницы на основе методологии Agile Kanban для следующих ключевых бизнес процессов: назначить диагностику, расшифровать результаты диагностики и составить курс терапии. Разработка программы позволит автоматически выбрать необходимую диагностику, сформированную на основе жалоб пациента; автоматически расшифровать результаты диагностики и вывести на экран сообщения «Норма» или «Отклонение» и сформировать курс терапии с учетом диагноза и возраста пациента.

Введение

Условия развития современного мира, сопровождающиеся ростом потока информации и количества предоставляемых услуг, оказывают существенное влияние на внедрение информационных технологий во все сферы жизнедеятельности человека.

Важнейшей областью в современном обществе является сфера здравоохранения. Увеличение динамики заболеваний среди населения, сопровождающееся ростом количества пациентов, способно затруднить деятельность медицинских учреждений и их сотрудников.

В связи с чем, возникает необходимость создания медицинской информационной системы, способной повысить качество медицинского обслуживания и минимизировать риск возникновения врачебной ошибки, путем систематизации данных и автоматизации процессов. Наличие вышеприведенных проблем в медицинских учреждениях является доказательством актуальности представленной в данной работе темы.

Цель и задачи

Целью данной работы является автоматизация на основе методологии Agile Kanban следующих ключевых бизнес процессов: назначить диагностику, расшифровать результаты диагностики и составить курс терапии. Особенность работы состоит в разработке программного обеспечения, которое будет способно обеспечить выполнение ключевых бизнес-процессов:

- автоматически выбрать необходимую диагностику, сформированную на основе жалоб пациента;

- автоматически расшифровать результаты диагностики и вывести на экран сообщения «Норма» или «Отклонение»;
- сформировать курс терапии с учетом диагноза и возраста пациента.

Для достижения вышеуказанной цели, необходимо реализовать ряд следующих задач:

- детально проанализировать методологию внедрения Agile Kanban;
- идентифицировать и приоритизовать требования;
- спроектировать процессы и оргструктуру в моделях AS-IS и TO-BE на базе нотации UML Activity Diagram до 3-го уровня детализации;
- смоделировать разрабатываемые пользовательские формы;
- спроектировать структуру данных и нормализовать таблицы данных;
- реализовать ключевые бизнес-процессы в среде MS Access.

1. Детальный анализ методологии внедрения Agile Kanban

1.1 Гибкая методология разработки Agile

Методология разработки Agile – это комплекс «гибких» подходов к разработке программного обеспечения. Главные принципы методологии были отображены в Манифесте Agile (Agile Manifesto), принятом в феврале 2001 года. Данный документ провозглашает двенадцать наиболее значимых принципов создания программного обеспечения (ПО) [1]. Четыре из них являются основополагающими и звучат следующим образом:

- Люди и коммуникация превыше процессов и инструментов.
- Рабочий продукт превыше полной документации.
- Взаимодействие с клиентом превыше согласования условий договора.
- Готовность адаптироваться к переменам превыше следования первичному плану.

Специфика методологии Agile состоит в итеративности применяемых в процессе создания ПО подходов. Исходя из сказанного, планирование на высшем уровне осуществляется в отношении всего цикла реализации проекта, после чего весь цикл разработки делится на итерации – последовательность этапов, в процессе которых планомерно разрабатывается и тестируется программное обеспечение. Итогом завершения итерации является инкремент, посредством которого расширяются возможности создаваемого программного обеспечения.

Основное преимущество данной методологии состоит в максимальном снижении рисков, поскольку на каждой итерации процесс создания программного обеспечения контролируется заказчиком, который может одобрить полученные результаты, внести новые запросы или изменить существующие.

1.2 Подход Kanban

Одним из фреймворков методологии Agile является подход Kanban. Kanban применяет итеративный подход к созданию программного обеспечения и оперативную адаптацию к изменениям в ходе создания продукта. Специфика подхода заключается в том, что ее можно описать выражением «начните с того, что вы делаете сейчас» [2]. Основными принципами подхода Kanban являются [3]:

- Kanban основывается на действующих подходах к созданию продукта и по ходу осуществления оптимизирует, дополняет и совершенствует их.
- Осуществление значимых преобразований оговаривается заблаговременно: непрерывные преобразования являются путем к оптимизации действующего процесса создания программного обеспечения, при этом кардинальные трансформации кроют в себе немалые риски.
- Почтительное отношение к установленным правилам, ролям и выполняемым обязанностям.
- Стимулирование инициативы.

1.3 Использование подхода Kanban

Для того, чтобы внедрить подход Kanban в процесс разработки программного обеспечения, необходимо соблюдение особых правил, называемых практиками. Основными являются следующие практики [4]:

- визуализация процесса разработки;
- ограничение количества одновременно выполняемых задач;
- измерение и регулирование потока.

В Kanban применяется процедура визуального мониторинга осуществления стадий процесса разработки [5]. Главным средством визуализации служит Канбан-доска, необходимая для того, чтобы продемонстрировать команде, на каком этапе находится реализация каждой промежуточной цели, из которых затем складывается весь проект в целом. Доска расчерчена и поделена на столбцы, которые отображают этапы процесса разработки программного обеспечения. Карточки, содержащие задачи, передви-

гаются по этим столбцам по ходу реализации проекта. На рисунке 1.1 приведен пример Kanban доски, применяемой в процессе разработки программного обеспечения.

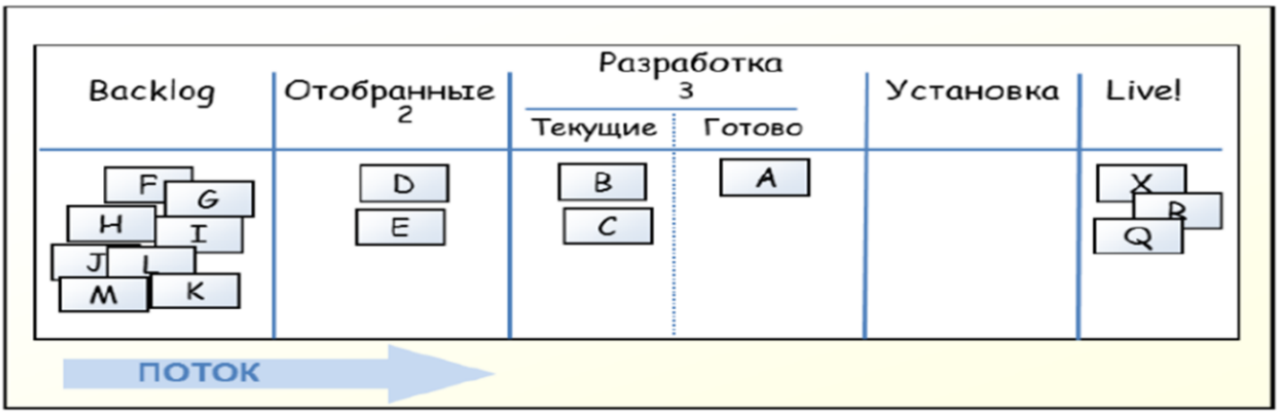

Рис. 1.1. – *Пример Kanban доски*

Для того чтобы использование подхода приносило ожидаемый результат, важно производить постоянный мониторинг, сокращение и оптимизацию незавершенной работы [2]. Соблюдение ограничения, позволяет сделать процесс разработки вытягивающим, в котором работа над новыми элементами начинается только по завершении предыдущих задач, что соответствует принципу «Точно в срок».

Следование данному принципу способствует снижению временных затрат на создание и увеличение качества продукта, поскольку большой объем выполненных не в полной мере задач снижает продуктивность и повышает необходимое время на разработку программного обеспечения, препятствуя своевременной реакции на изменения запросов клиента.

Лимит WIP (Work in progress) заключается в четком определении разрешенного количества заданий на каждой из стадий разработки. Когда команда разработчиков вводит ограничение на выполнение незавершенных работ, она добавляет цифру в колонку на доске, которая указывает на максимальное число рабочих элементов, разрешенных на данном этапе разработки программного обеспечения [4]. Это способствует освобождению места сбора большого количества рабочих компонентов и усовершенствованию процесса разработки.

Зависимость между ограничением числа задач в работе и временем доставки продукта описана математически в теории очередей и называется законом Литтла. Согласно данному закону, повышение количества выполняемых операций способствует

увеличению времени, необходимого для реализации каждой из них. Представим формулу (1.1), отражающую данный закон:

$$\text{Cycle Time} = \frac{\text{Work in Progress}}{\text{Throughput}}, \tag{1.1}$$

где Cycle Time – время, в течение которого определенное задание было на стадии выполнения, от начала работы над ним до окончания, Work in Progress – количество незавершённой работы, troughput – число, отображающее количество компонентов, которые коллектив разработчиков может производить за установленный временной промежуток.

Под измерением и управлением потоком предполагается измерение количества текущих задач и регулирование хода разработки для его максимального увеличения [4]. Главным средством, применяемым для измерения потока, является кумулятивная диаграмма потока – CFD. Она отображает ограничение на реализацию неоконченных задач (WIP-лимит), общее количество рабочих компонентов в процессе разработки, количество компонентов, которые присоединяются к списку каждый день, и среднее время, в течение которого рабочие компоненты остаются на стадии реализации [4]. Приведем пример CFD диаграммы на рисунке 1.2.

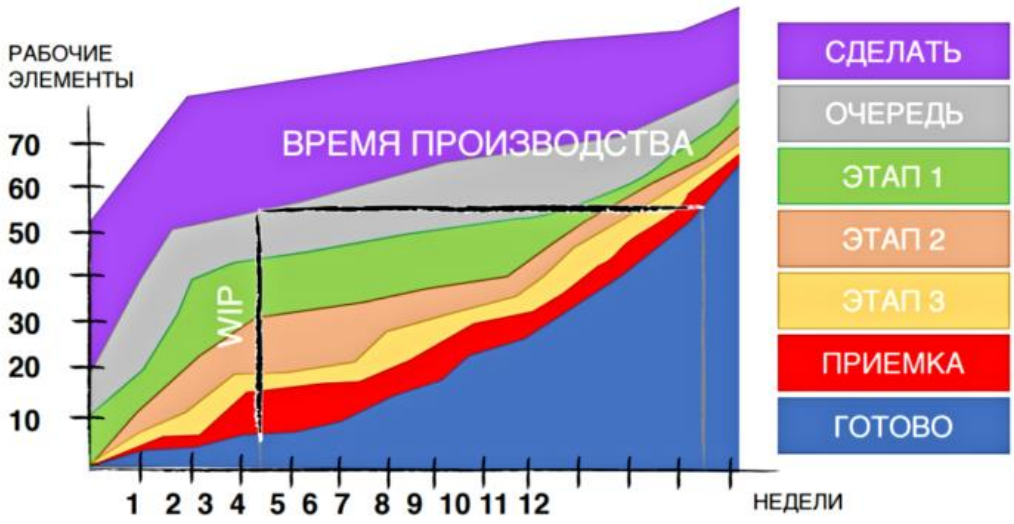

Рис. 1.2. – *Кумулятивная диаграмма потока CFD*

Данная диаграмма передает мониторинг количества рабочих компонентов на каждой стадии, проводимый изо дня в день. Ось X отображает время, а ось У – количество задач. Таким образом, можно наглядно заметить наличие узких мест в процессе разработки, если на графике возрастает какая либо область.

2. Идентификация, формирование и приоритизация требований

2.1 Анализ требований

Требования к программному обеспечению – комплекс выдвигаемых запросов по поводу признаков, свойств и качеств программного обеспечения, которые необходимо реализовать [6]. Анализ требований – процесс классификации информации, касающейся требований, по различным категориям, оценка требований для определения желаемого качества, представление требований в различных формах, выделение детальных требований из требований более высокого уровня, а также обсуждение приоритетов требований [7]. Обработка требований включает в себя следующие этапы:

- установка основной идеи продукта, на этой стадии осуществляется формирование общего понимания, как должен выглядеть разрабатываемый продукт. Здесь принимается решение о необходимости разработки программного продукта;
- сбор требований, где большая часть работы посвящается коммуникации с клиентом и потенциальными пользователями разрабатываемого продукта. Основная цель состоит в четкой установке возможностей продукта и механизмов его внедрения в действующие процессы;
- анализ требований, здесь проходит структуризация собранных ранее требований. Требуется предоставить четкий список не дублируемых требований к системе, которые должны быть выделены из избыточных и частично дублирующихся сценариев и пользовательских историй [8].

2.2 Обработка требований

Пользовательские требования – это требования, выдвигаемые потенциальными пользователями к создаваемому программному обеспечению, озвученные в простой естественной форме. Для ключевых бизнес-процессов: «Назначить диагностику», «Расшифровать результаты диагностики», «Составить курс терапии», путем проведения интервью был определен ряд пользовательских требований, что послужило базисом проработки функциональных требований.

Функциональные требования – положение о фрагменте требуемой функциональности или поведения, которые система проявляет при определенных условиях [7]. Иными словами, функциональные требования устанавливают возможности создаваемого программного обеспечения – необходимые функции и операции, которые можно будет с помощью него производить.

В Agile терминологии под бэклогом подразумевается перечень пользовательских и функциональных требований к программному обеспечению. При формировании бэклога все требования необходимо выстроить в порядке их приоритетности. Для этого необходимо выполнить процесс приоритизации или распределения требований по итерациям разработки. Осуществление данного процесса позволяет понять разработчику, какие пользовательские требования необходимо реализовать первоочередно, а также на что следует обратить особое внимание, тем самым позволяя избежать излишних материальных и временных затрат на проектирование и разработку (таблица 2.1).

Таблица 2.1. – Бэклог продукта

№	Пользовательское требование	Функциональное требование	Программный компонент	Итерация
1.	Хранение данных: 1) Данные о пациентах. 2) Данные о специалистах. 3) Виды диагностики. 4) Список диагнозов гастроэнтерологической направленности 5) Список характерных симптомов гастроэнтерологической направленности. 6) Диапазон нормальных значений для каждого диагностического показателя 7) Результаты диагностики 8) Список лекарственных препаратов для диагнозов гастроэнтерологической направленности. 9) Данные о дозировках лекарственных препаратов. 10) Данные о первичном приеме пациента. 11) Данные о назначенном курсе терапии.	Таблицы: 1) Пациенты. 2) Специалисты. 3) Диагнозы. 4) Первичный прием. 5) Жалобы. 6) Виды диагностики. 7) Проведенная диагностика. 8) Показатели нормы. 9) Лекарства. 10) Диагнозы и лекарства. 11) Пациенты Жалобы. 12) Направление по жалобам.	Программа ввода данных.	2

№	Пользовательское требование	Функциональное требование	Программный компонент	Итерация
2.	Ввод и редактирование данных: 1) Регистрация нового пациента. 2) Редактирование данных о пациенте. 3) Редактирование данных о специалисте. 4) Поиск зарегистрированного пациента. 5) Ввод данных о первичном приеме пациента. 6) Запись результатов диагностики. 7) Ввод данных о назначенном курсе терапии. 8) Ввод данных о лекарственных средствах.	Формы для ввода данных.	Программа для ввода и редактирования данных.	2
3.	Вывод информации на экран.	Отчеты для отображения данных.	Программа для работы с интерфейсом.	3
4.	Автоматическое назначение диагностики на основе жалоб пациента.	Параметрический запрос.	Программа для создания запросов.	3
5.	Автоматическая расшифровка результатов диагностики: 1) Вывод на экран сообщений «Норма» или «Отклонение» для каждого диагностического показателя.	Параметрический запрос для автоматического сравнения полученных результатов диагностики с нормальными значениями. 1) Запись диапазона нормаль-	Программа для создания запросов.	3

№	Пользовательское требование	Функциональное требование	Программный компонент	Итерация
		ных значений для каждого диагностического показателя; 2) Вывод на экран сообщений «Норма», «Отклонение» в результате процесса сравнения.		
6.	Автоматическое назначение курса терапии: 1) Назначение лекарственных препаратов для определенного диагноза. 2) Подбор дозировки с учетом возраста пациента.	Параметрический запрос для определения курса терапии: 1) Автоматическое вычисление возраста пациента.	Программа для создания запросов.	3
7.	Обеспечение конфиденциальности.	Пароль при входе в главное меню.	Программа для работы с БД.	4
8.	Печать данных.	Передача данных на устройство печати.	Программа для работы с БД.	4

Согласно методологии Agile Kanban, жизненный цикл разработки программного обеспечения детализируется вплоть до итерации. Отобразим бэклог итераций и их детали в таблице 2.2, включая активности по проектированию и испытанию программного продукта.

Таблица 2.2. – Бэклог проекта

Итерация	Бэклог итерации
Итерация 1	1) Моделирование ключевых бизнес-процессов в моделях As-Is и To-Be в нотации UML Activity Diagram. 2) Моделирование пользовательского интерфейса. 3) Моделирование структуры данных разрабатываемого приложения.
Итерация 2	1) Реализация требований 1-3. 2) Тестирование и отладка на промежуточном этапе. 3) Демонстрация инкремента.
Итерация 3	1) Реализация требований 4-6. 2) Тестирование и отладка на промежуточном этапе. 3) Демонстрация инкремента.
Итерация 4	1) Реализация требований 7-8. 2) Тестирование и отладка на промежуточном этапе. 3) Демонстрация инкремента.
Тестирование	1) Функциональное тестирование. 2) Интеграционное тестирование. 3) Нагрузочное тестирование.

3. Первая итерация - проектирование бизнес-процессов

3.1 Нотации моделирования процессов

Проектирование – процесс разработки планов и чертежей, технических спецификаций и операционных характеристик, необходимых для создания концепций разработки производства и маркетинга новых изделий и процессов [9]. В ходе проектирования бизнес-процессов происходит их декомпозиция. Декомпозиция – это метод, который дает возможность разложить сложную многоуровневую структуру на ряд простых взаимозависимых элементов [10]. Глубина осуществляемой декомпозиции зависит от целей, которые преследуются в процессе проектирования и, следовательно, определяет уровень подробности характеристики процесса.

Основной целью проектирования бизнес-процессов является описание реального хода бизнес – процессов предприятия. В процессе проектирования важно установить, что представляет собой итог реализации процесса, какие операции и кем производятся, их последовательность, какой осуществляется документооборот в процессе реализации, а также степень надежности рассматриваемого процесса и его потенциал для

оптимизации. Таким образом, существуют следующие стадии проектирования бизнес-процессов:

- идентификация процессов и построение модели As-Is. Модель As-Is (как есть) представляет собой модель фактического состояния. Она дает возможность провести систематизацию происходящих в данное время процессов и применяемых информационных объектов;
- анализ и уточнение исходной модели. На данном этапе выявляются противоречия и дублирование действий в процессе, устанавливаются взаимосвязи между процессами, определяется необходимость изменения процесса;
- разработка модели To-Be. Данная модель, формируемая на базе итогов исследования модели As-Is, характеризует будущие свойства процессов, принимая во внимание требования клиента, а также результаты исследований и усовершенствования действующих процессов [11];
- испытание и применение модели To-Be. Происходит введение сформированной модели в эксплуатацию в процессе деятельности предприятия. По ходу использования модель тестируется в реальных условиях и, если того требует ситуация, ее подвергают правкам и дополнениям.

Проектирование бизнес-процессов состоит в наглядном отображении этих процессов посредством определенной нотации. Нотация – упорядоченная совокупность условных символов и знаков, используемая в определенном языке для наглядной демонстрации [15]. Унифицированный язык моделирования (Unified Modeling Language) UML – это нотация для визуализации, специфицирования, конструирования и документирования систем, в которых главная роль принадлежит программному обеспечению [12]. Одним из видов UML диаграмм является Activity Diagram – диаграмма деятельности. При ее помощи можно смоделировать последовательности действий, которые выполняются различными элементами, входящими в состав системы.

3.2 Проектирование ключевых бизнес-процессов в модели As-Is

Проектирование ключевых бизнес-процессов на первом уровне детализации заключается в верхнеуровневом описании данных процессов. Верхнеуровневое описание процессов – описание, в котором система процессов представлена деревом, процессы как минимум идентифицированы и определено их первоначальное взаимодействие [9]. Результаты проектирования бизнес-процессов на первом уровне детализации даны на рисунке 3.1.

Рис. 3.1. – *Описание бизнес-процессов на первом уровне детализации*

Для того, чтобы детально проанализировать процессы, представленные на первом уровне и выявить наличие узких мест в рамках деятельности больницы, необходимо провести их декомпозицию. На рисунке 3.2 дано описание процессов «Идентифицировать пациента» и «Провести первичный прием» на втором уровне детализации.

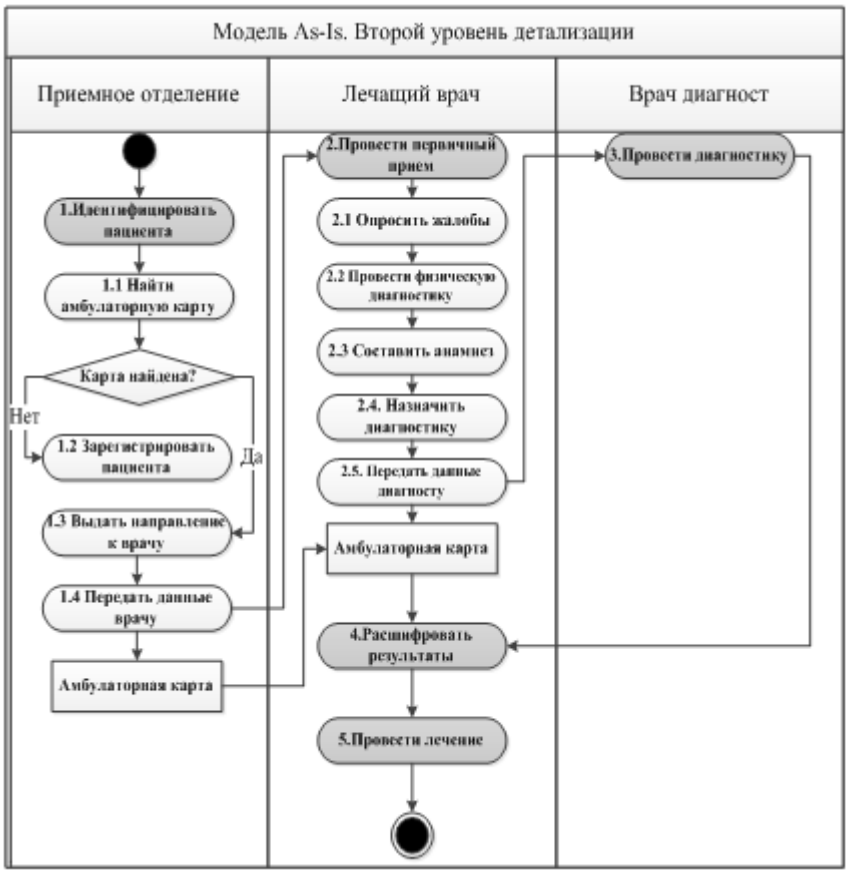

Рис. 3.2. – *Проектирование процессов «Идентифицировать пациента», «Провести первичный прием» на втором уровне детализации*

В результате декомпозиции процесса «Провести первичный прием» на втором уровне детализации была установлена необходимость автоматизации процесса 2.4 –

«Назначить диагностику». Результаты моделирования этого процесса на третьем уровне приведены на рисунке 3.3.

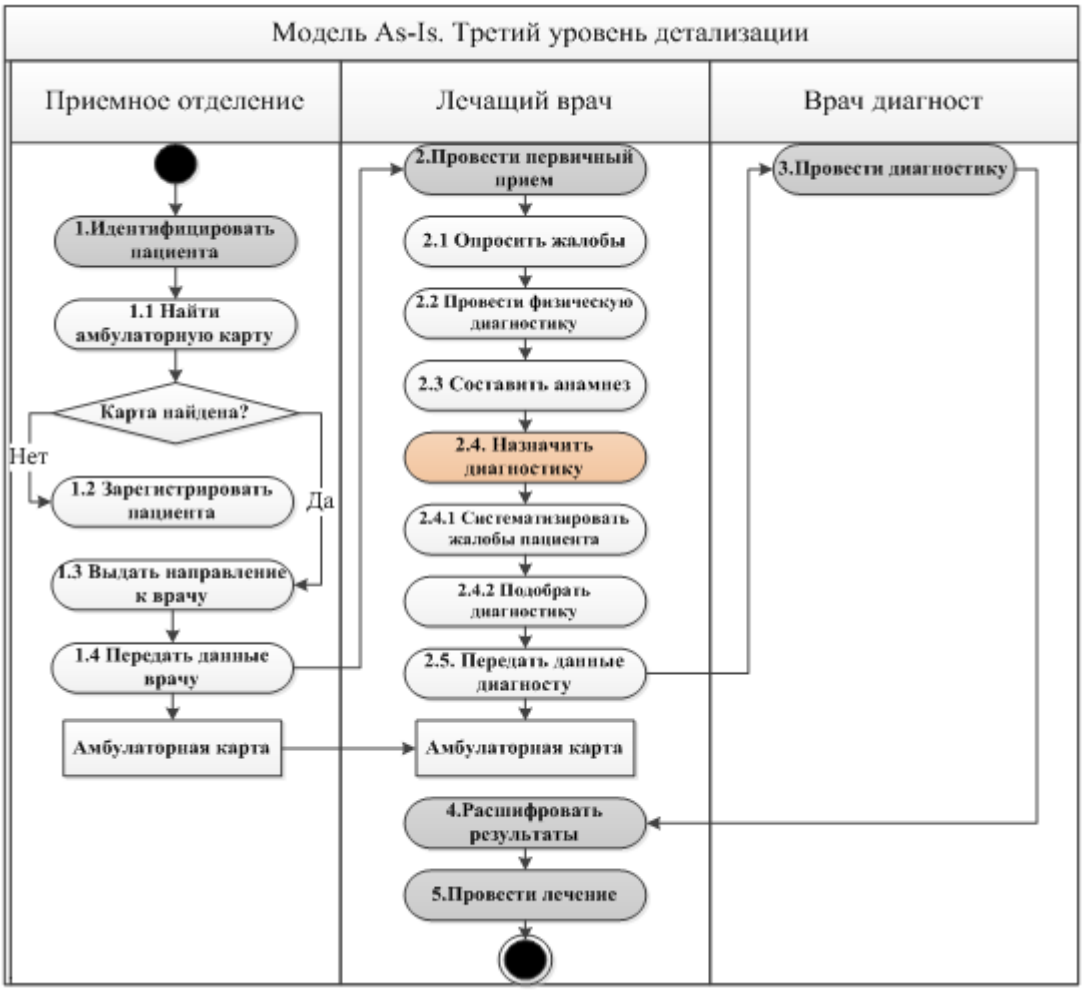

Рис. 3.3. - *Детализация процесса «Назначить диагностику»*

В ходе проектирования бизнес-процессов на нижних уровнях детализации была проанализирована текущая деятельность больницы и установлены следующие недостатки:

- хранение амбулаторной карты пациента в бумажном виде;
- долгий поиск амбулаторной карты в регистратуре;
- длительное заполнение бланка диагностики в бумажном виде;
- расшифровка результатов диагностики вручную на основе медицинского справочника;
- субъективное назначение курса терапии для установленного диагноза и подбор дозировки пациенту.

3.3 Проектирование ключевых бизнес-процессов в модели To-Be

Для отображения изменений, привносимых разрабатываемым программным обеспечением в бизнес-процессы, описанные в модели As-Is, необходимо спроектировать данные процессы в To-Be. На рисунке 3.4 даны результаты проектирования процессов «Идентифицировать пациента» и «Провести первичный прием» на втором уровне детализации.

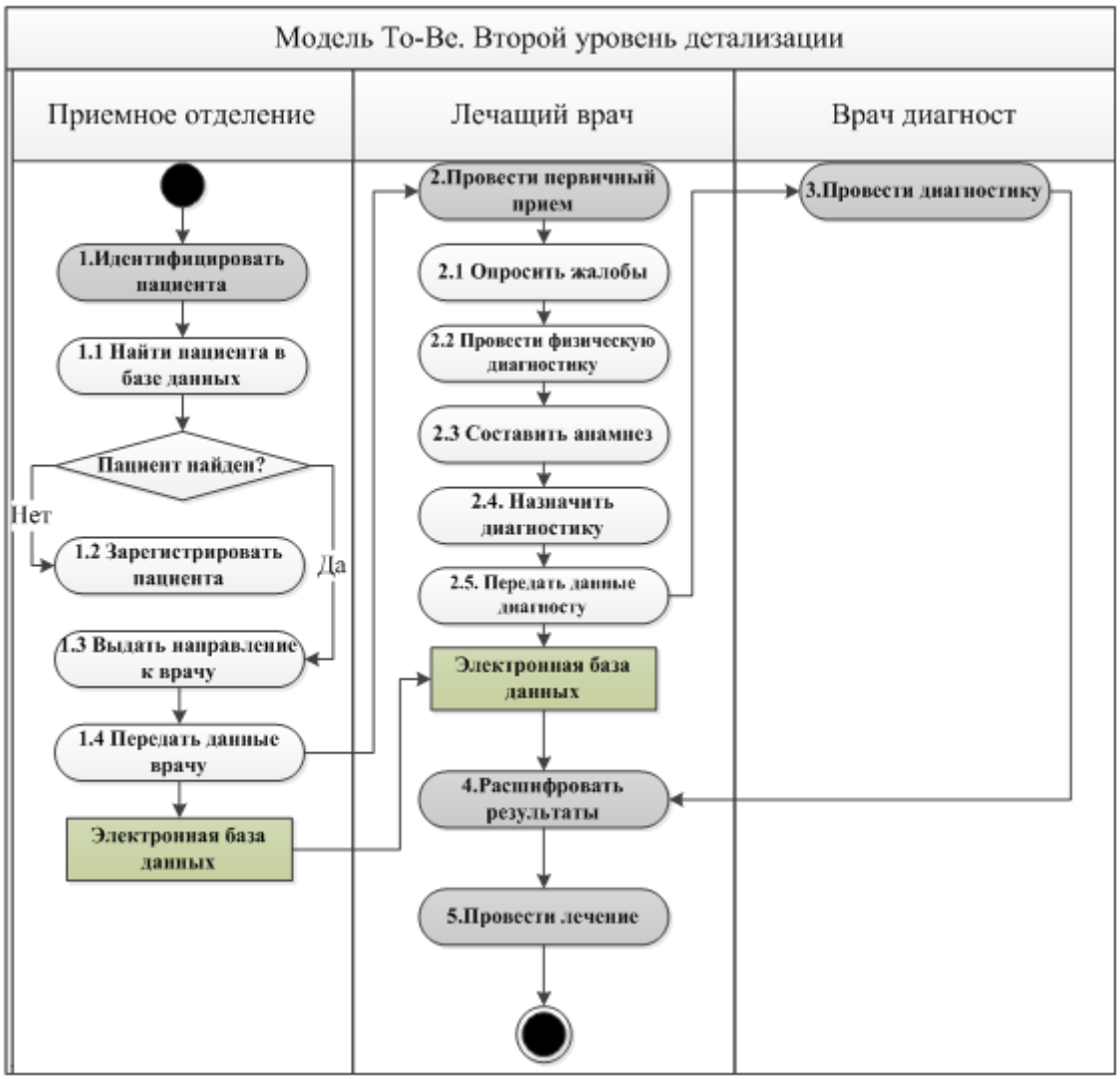

Рис. 3.4. – *Описание процессов «Идентифицировать пациента» и «Провести первичный прием»*

Аналогичным способом было проведено моделирование процессов «Провести диагностику», «Расшифровать результаты» и «Провести лечение» на втором уровне детализации. Для уточнения всех изменений, привносимых в бизнес-процесс «Прове-

сти первичный осмотр», проводилось проектирование на третьем уровне детализации, путем уточнения операции «Назначить диагностику» (рисунке 3.5).

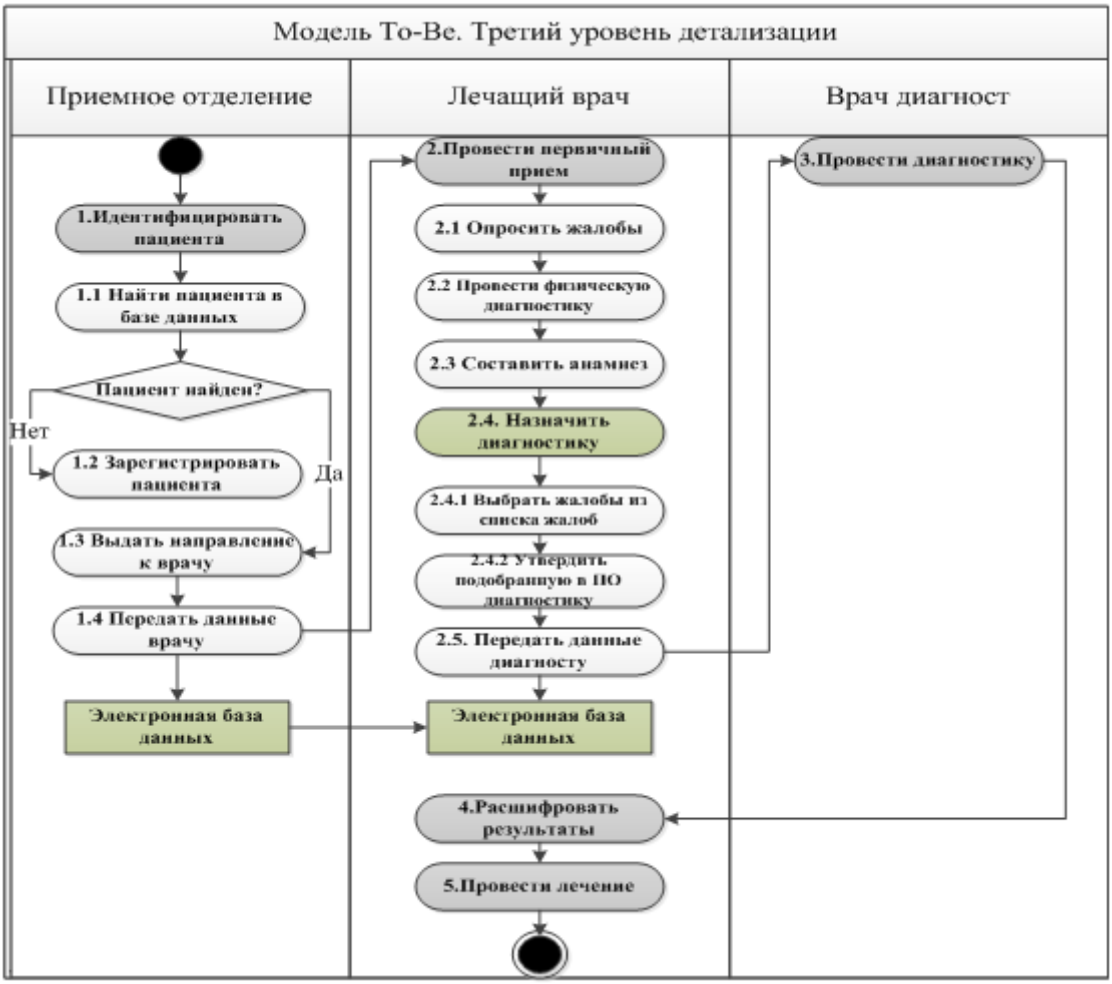

Рис. 3.5. – *Описание процессов «Идентифицировать пациента», «Провести первичный прием», детализация процесса «Назначить диагностику»*

3.4 Карта процессов

Картой бизнес-процессов называют графическое представление бизнес-процессов в виде иерархического дерева. На рисунке 3.6 показана полученная карту процессов в модели To-Be.

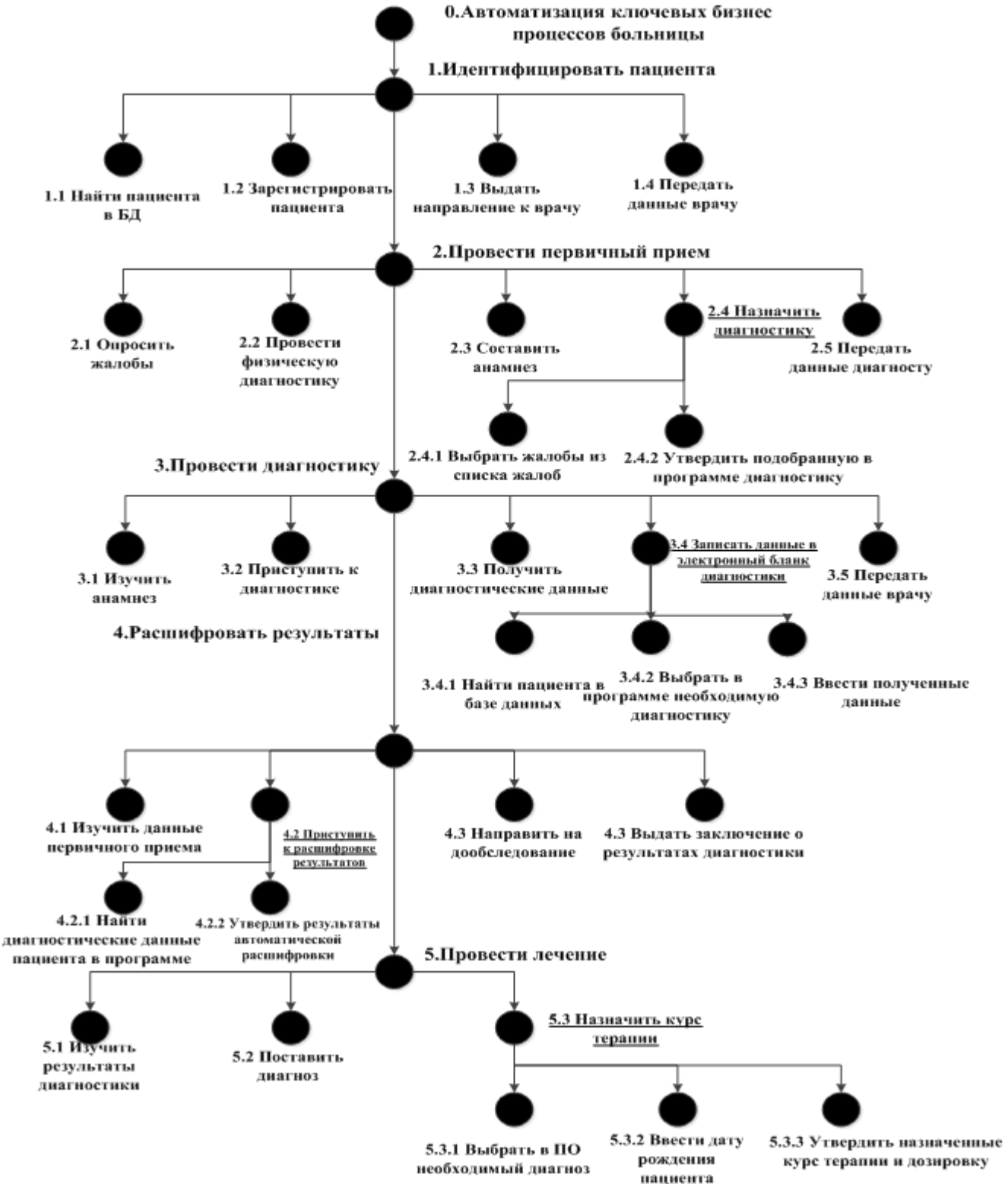

Рис. 3.6. – *Карта процессов в модели To-Be*

Литература

1. Agile-манифест разработки программного обеспечения, статья «Agile манифест»// [Интернет-ресурс]. Режим доступа: http://agilemanifesto.org (Дата обращения 21.03.2019).
2. Андерсон Д., Кармайкл Э. Канбан: краткое руководство [Текст] / Д.Андерсон, Э. Кармайкл – LeanKanban University – 2015. – 79с.
3. Свободная энциклопедия Википедия, статья «Канбан (разработка)»// [Интернет-ресурс]. Режим доступа: https://ru.wikipedia.org (Дата обращения 21.03.2019).
4. Грин Д. Постигая Agile [Текст] / Грин Д. – Манн, Иванов и Фербер –2016.–288 с.
5. Книберг Х., Скарин М. Kanban и Scrum: выжимаем максимум [Текст] / Х. Книберг, М. Скарин – InfoQ.com – 2010. – 76с.
6. Свободная энциклопедия Википедия, статья «Анализ требований» // [Интернет-ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Анализ_требований (Дата обращения 25.03.2019).
7. Вигерс К. Разработка требований к программному обеспечению [Текст] / К.Вигерс – М.: Русская редакция – 2014. — 736 с.
8. Химонин Ю. И. Сбор и анализ требований к программному продукту (Версия 1.03). – 2009, – 51 с.
9. Воронков, А.Н. Словарь по менеджменту [Текст]: учебное пособие/ А.Н. Воронков, Т.В. Колосова; Нижегород. гос. архит.-строит. ун-т. – Н. Новгород: ННГАСУ, 2013 – 125 с.
10. Статья «Описание бизнес-процессов» // [Интернет-ресурс] режим доступа: http://regcons.ru (Дата обращения 2.04.2019).
11. Онлайн библиотека Studbooks.net, статья «Модель AS-IS и Модель TO-BE» // [Интернет-ресурс]. Режим доступа: https://studbooks.net (Дата обращения 12.04.2019).
12. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS [Текст] / Г. Буч, А. Якобсон, Дж. Рамбо – СПб.: Питер – 2006. – 736с.
13. Официальный сайт компании Microsoft, статья «Описание основных приемов нормализации базы данных» // [Интернет-ресурс]. Режим доступа: https://support.microsoft.com/ru-ru/help/283878/description-of-the-database-normalization-basics (Дата обращения 19.04.2019).

14. Официальный сайт компании Microsoft, статья «Создание связей между таблицами в базе данных» // [Интернет – ресурс]. Режим доступа: https://support.microsoft.com/ru-ru/help/304466/how-to-define-relationships-between-tables-in-an-access-database (Дата обращения 19.04.2019).

Выходные данные статьи

Мартынов М.В. Применение Agile Kanban для автоматизации работы городской больницы (Часть 1) // Корпоративные информационные системы. – 2021. – №3 (15) – С. 25-42. – URL: https://corpinfosys.ru/archive/issue-15/180-2021-15-agilekanban.

Об авторе

Мартынов Михаил Васильевич – студент 4-го курса кафедры оптических и биотехнических систем и технологий физико-технологического института РТУ МИРЭА. Тема выпускной квалификационной работы бакалавра «Применение методологии Agile Kanban для автоматизации ключевых бизнес-процессов городской больницы». Электронная почта: mail@corpinfosys.ru.