https://orcid.org/0000-0001-9622-3871
https://orcid.org/0000-0003-1521-6958
https://orcid.org/0000-0002-3424-7293
https://orcid.org/0000-0003-3092-678X

УДК 004.054

В.В. ЛИТВИНОВ[*], И.В. БОГДАН[*], А.А. ЗАДОРОЖНИЙ[*], И.В. БЕЛОУС[*]

МЕТОДЫ ПРИОРИТИЗАЦИИ ЗАДАЧ В ГИБКИХ МЕТОДОЛОГИЯХ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

*Черниговский национальный технологический университет, г. Чернигов, Украина

Анотація. *У роботі розглядаються сучасні методи пріоритизації задач, що використовуються у гнучких методологіях розробки програмного забезпечення. Саме гнучкі методології розробки, такі як Scrum, Kunban та ін., на даний момент є найпопулярнішими, оскільки дозволяють фактично на будь-якому етапі створення проекту вносити до нього корективи, підвищувати якість створюваного продукту за рахунок щоденного контролю за його створенням та досить швидко випускати перші версії створюваного програмного забезпечення. Всі методи пріоритизації задач, що застосовуються при розробці програмних проектів із використанням гнучких методологій, поділяються на ті, що враховують точку зору команди розробників, та ті, що базуються на різних кількісних оцінках, серед яких можуть бути різноманітні метрики, експертні висновки, оцінки зацікавлених у проекті осіб, існуючі класифікації та багато іншого. Серед розглянутих методів пріоритизації, що враховують думку команди, такі популярні та активно використовувані на даний момент, як метод MoSCoW, карта історій (User story mapping) та проактивне удосконалення. До розглянутих методів, заснованих на кількісній оцінці, належать модель Кано, метод, заснований на створенні оціночних листів, запропонований Карлом Вігерсом метод оцінки відносних пріоритетів для набору функцій та метод структурування функцій якості (Quality Function Deployment). У залежності від особливостей проекту, вимог замовника, побажань команди та інших об'єктивних чи суб'єктивних факторів у проекті може бути використаний один або декілька методів пріоритизації одночасно або ж їх комбінація. Крім того, деякі з розглянутих методів можуть бути застосовані при короткостроковому плануванні, інші ж – при довгостроковому, а є й ті, які можуть бути використані на будь-якому етапі.*

Ключові слова: *метод пріоритизації задач, гнучка методологія розробки програмного забезпечення, Scrum, Kunban.*

Аннотация. *В статье рассматриваются современные методы приоритизации задач, которые используются в гибких методологиях разработки программного обеспечения. Именно гибкие методологии разработки, такие как Scrum, Kunban и другие, на данный момент являются самыми популярными, поскольку позволяют фактически на любом этапе создания проекта вносить в него коррективы, повышать качество создаваемого продукта за счет ежедневного контроля за его созданием и достаточно быстро выпускать первые версии программного обеспечения. Все методы приоритизации задач, которые применяются при разработке программных проектов с использованием гибких методологий, разделяются на те, которые учитывают точку зрения команды разработчиков, и те, которые основаны на различных количественных оценках, среди которых могут быть разнообразные метрики, экспертные мнения, оценки заинтересованных в проекте лиц, имеющиеся классификации и многое другое. Среди рассмотренных методов приоритизации, которые учитывают мнение команды разработчиков, такие популярные и активно используемые на данный момент, как метод MoSCoW, карта историй (User story mapping) и проактивное совершенствование. К рассмотренным методам, основанным на количественной оценке, принадлежит модель Кано, метод, основанный на создании оценочных листов, предложенный Карлом Вигерсом метод оценки относительных приоритетов для набора функций и метод структурирования функций качества (Quality Function Deployment). В зависимости от особенностей проекта,*

требований заказчика, пожеланий команды разработчиков и других объективных или субъективных факторов в проекте может быть использован один или же несколько методов приоритизации одновременно или же их комбинация. Кроме того, некоторые из рассмотренных методов могут быть использованы при краткосрочном планировании, другие же – при долгосрочном, но есть и те, которые могут быть использованы на каждом этапе.
Ключевые слова: *метод приоритизации задач, гибкая методология разработки программного обеспечения, Scrum, Kunban.*

Abstract. *The modern task prioritization methods that are used in flexible software development methodologies are discussed in the paper. Very flexible development methodologies such as Scrum, Kunban and others are currently the most popular because they allow making adjustments to it at any stage of a project, to improve the quality of the created product through daily monitoring of its creation and quickly release the first versions of the software. All task prioritization methods that are used in software projects development including flexible methodologies are divided into those that take into account the point of view of the development team and those that are based on various quantitative assessments, among which various metrics, expert opinions, points of view of those who are interested in the project, available classifications etc. Among the considered prioritization methods, which take into account the opinion of the development team, there are such popular and actively used nowadays methods, as MoSCoW, story map (User story mapping) and proactive improvement. Among the considered methods, based on quantitative assessment, is Kano's model, the method based on the creation of evaluation sheets, the method for estimating relative priorities for a set of functions offered by Carl Wigers and the method of structuring quality functions (Quality Function Deployment). Depending on the features of the project, customer requirements, the wishes of the development team and the other objective or subjective factors, the project can use one or several prioritization methods at the same time or the combination of them. In addition, some of the considered methods can be used in short-term planning, the others – in the long-term, but there are those that can be used at each stage.*
Keywords: *task prioritization method, flexible software development methodology, Scrum, Kunban.*

DOI: 10.34121/1028-9763-2020-2-70-78

1. Введение

Гибкие методологии разработки программного обеспечения, такие как Scrum, Kunban и другие, на данный момент являются одними из самых популярных при разработке программного обеспечения [1]. В классическом варианте каждая из данных методологий предвидит приоритизацию задач, необходимых к выполнению в рамках проекта: какие пользовательские истории будут выпущены в первой версии продукта, какие – в последующих, а какие могут быть оставлены напоследок. В большинстве случаев определение приоритетности задач, подлежащих реализации, выполняет сам заказчик или его представитель в зависимости от своих желаний и предпочтений или же потребностей рынка. Однако достаточно часто данная задача ложится на плечи команды разработчиков, и в этом случае оптимальным решением является применение методов приоритизации задач.

На данный момент существует множество различных методов и подходов к приоритизации задач, которые используются отдельно или в комплексе. Практически все из них относятся к одной из двух возможных категорий:

– методы, основанные на использовании входных данных внутри команды разработчиков или данных, полученных от заказчика.

– методы, основанные на использовании количественной оценки (экспертное мнение, метрики, классификации и т.д).

2. Методы приоритизации задач, основанные на входных данных, получаемых со стороны команды или заказчика

Метод MoSCoW (Must have, Should have, Could have, Won′t have) [2] принадлежит к методам, основанным на использовании данных, получаемых исключительно от членов коман-

ды исполнителей проекта или (и) заказчика. Суть его состоит в том, чтобы разделить все задачи, относящиеся к спринту, на четыре группы:

– Must have – задачи, которые обязательны к выполнению в рамках данного спринта. Если хотя бы одна из задач этой группы не выполнена в рамках текущего спринта, то спринт можно считать проваленным.

– Should have – задачи этой группы также очень важны для выполнения, однако не являются критичными для спринта.

– Could have – к данной группе принадлежат задачи, которые желательны к выполнению, однако не обязательны.

– Won′t have – относительно задач этой группы было принято решение, что необходимости выполнять их в данном спринте нет, однако в будущем их нужно будет выполнить.

Самыми первыми выполняются задачи из группы Must have, после них – из группы Should have, далее Could have и самыми последними задачи, относящиеся к группе Won′t have.

Решение о том, к какой группе принадлежит та или иная задача, принимает либо заказчик, либо группа разработчиков.

Данный метод чаще всего используется при высокоуровневом планировании и ограниченном числе заинтересованных лиц, то есть, когда заказчик является частью команды.

Среди основных достоинств данного метода является простота использования и понимания.

Основные недостатки метода MoSCoW состоят в том, что не всегда ясно, по каким критериям относить задачи в различные группы и как приоритизировать задачи в рамках одной и той же группы.

Согласно другому методу – User story mapping [3], изначально формируется представление о потенциальном пользователе создаваемого программного обеспечения, а также о тех действиях, которые он сможет выполнять над ним. Эти действия составляют так называемый хребет карты историй. Сформулированные действия описываются на отдельных карточках и выстраиваются в последовательность в соответствии с тем, как чаще всего происходит их выполнение. Под каждым действием по приоритету размещаются связанные с ним пользовательские истории. Далее слева направо рисуется горизонтальная линия и все те истории, которые оказались над линией, попали в текущий релиз, а те, которые под линией, не попали. По такому принципу можно выполнять разбиение сразу на несколько релизов. В рамках отдельного релиза пользовательские истории лучше всего выполнять слева направо, сверху вниз.

Данный метод чаще всего используется при долгосрочном планировании. Основными его достоинствами являются визуализация и простота понимания, формирование у команды общего представления о проекте и о ходе его реализации. Среди недостатков метода User story mappin – это значительные временные затраты на построение бэклога и сложность его поддержки в динамической среде, а также нецелесообразность использования для больших проектов по причине большого количества пользовательских историй.

В том случае, если приоритизацию задач в рамках спринта выполняют только члены команды разработчиков, возможно также применение метода Джиро Кавакита под названием Проактивное совершенствование [4].

Для использования данного метода необходимо наличие двухцветных стикеров, а также одного большого помещения со свободными стенами. Изначально для всех членов команды определяется и озвучивается актуальный вопрос, после чего применяется мозговой штурм. Все возможные идеи и варианты решения этого вопроса записываются на стикеры. Далее каждая идея озвучивается, а соответствующий стикер клеится на стене. Затем

стикеры с похожими по сути идеями переклеиваются рядом, обозначая группу, а рядом с ними клеится стикер с названием группы. После чего каждый участник команды получает бумажные точки в количестве от трех до пяти и голосует наклеиванием этих точек на стикеры за наиболее актуальные идеи. Чем больше точек собирает идея, тем выше ее приоритет.

Данный метод чаще всего используется для принятия быстрых решений внутри команды разработчиков.

Среди достоинств данного метода – это простота использования для решения конкретного возникшего вопроса. В то же время данный метод сложен в использовании, если актуальных вопросов, подлежащих решению, множество.

3. Методы приоритизации задач, основанные на количественной оценке

К методам, основанным на количественной оценке, принадлежит модель Кано [5]. Модель Кано была разработана в 80-х годах профессором Нориаки Кано, и суть ее состоит в анализе удовлетворённости пользователей продуктом. Согласно данному подходу, все функции программного продукта можно разбить на пять основных групп: базовые или же ожидаемые, основные или же желаемые, восхищающие или же воздействующие, нейтральные и нежелательные.

К базовым функциям принадлежат те, которые должны присутствовать в программном обеспечении обязательно, то есть их наличие подразумевается как само собой разумеющееся (цифра 3 на рис. 1).

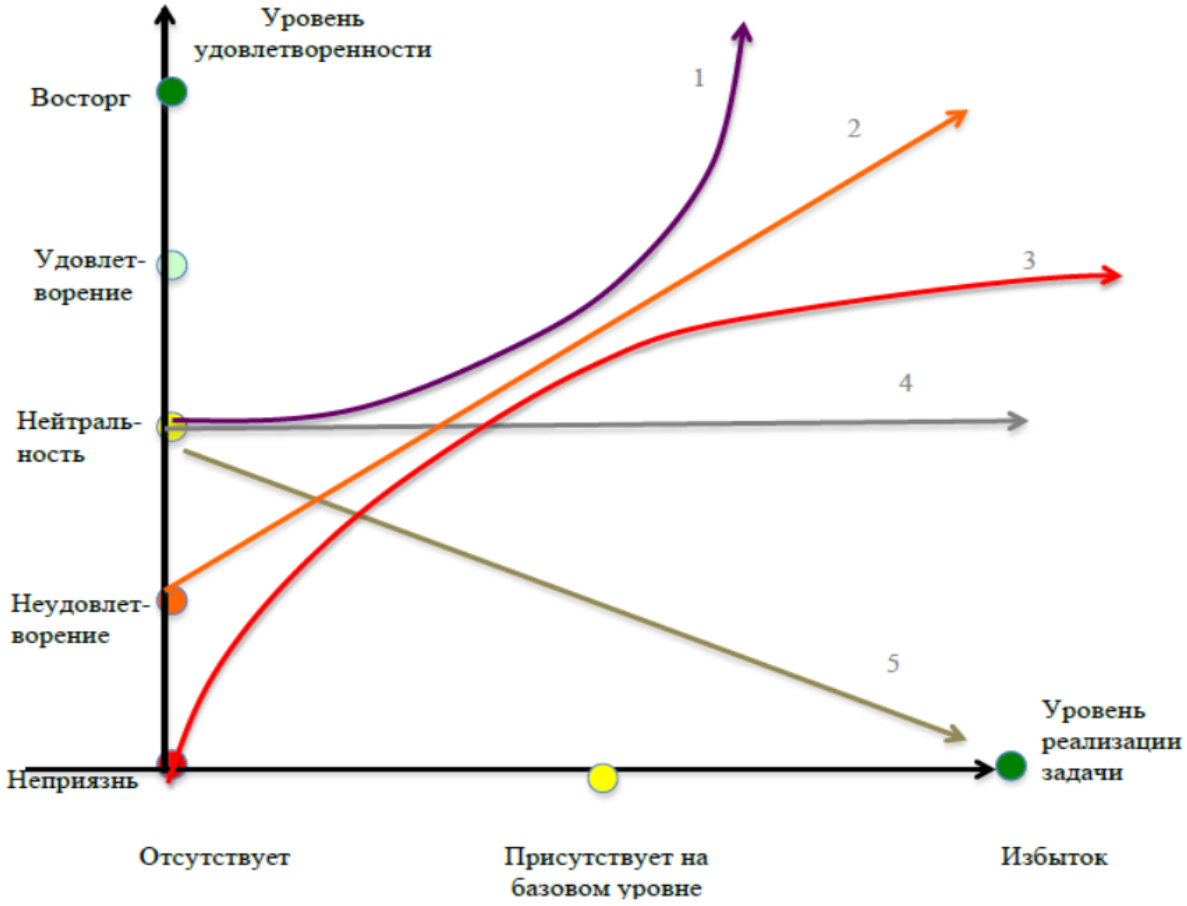

Рисунок 1 – Модель Кано

Основными же являются те функции, которые напрямую влияют на удовлетворенность пользователя (цифра 2 на рис. 1).

К восхищающим функциям относятся те, которые являются неожиданными для пользователя, такие как неожиданные дополнительные функциональные возможности (цифра 1 на рис. 1).

К нейтральным принадлежат те функции, наличие или отсутствие которых в программном продукте никак не влияет на удовлетворенность пользователей (цифра 4 на рис. 1).

Нежелательными являются функции, наличие которых в программном обеспечении отрицательно сказывается на удовлетворенность пользователей (цифра 5 на рис. 1).

Далее строится график, в котором по оси Y указывается уровень удовлетворенности пользователя продуктом, а по оси X – уровень реализации функции или задачи (рис. 1).

Принадлежность задачи или функции к той или иной группе определяется путем получения ответа от потенциальных пользователей на два основных вопроса:

– Насколько Вам понравится наличие такой функции в программном обеспечении?

– Как Вы отнесетесь к тому, что данная функция в конечном продукте будет реализована частично или вообще будет отсутствовать?

На оба вопроса возможны пять различных вариантов ответа: нравится, ожидаю, отношусь к этой функции нейтрально, могу терпеть наличие и не нравится указанная функция. Для того, чтобы определить, к какой группе отнести функцию, необходимо по оценочной таблице (табл. 1) найти ячейку, находящуюся на пересечении соответствующих полученным ответам строки и столбца.

Таблица 1 – Оценочная таблица для модели Кано

Положительный вопрос	Отрицательный вопрос					
		Нравится	Ожидаю	Нейтрально	Могу терпеть	Не нравится
	Нравится	-	Восхищающие	Восхищающие	Восхищающие	Основные
	Ожидаю	Нежелательные	Нейтральные	Нейтральные	Нейтральные	Базовые
	Нейтрально	Нежелательные	Нейтральные	Нейтральные	Нейтральные	Базовые
	Могу терпеть	Нежелательные	Нейтральные	Нейтральные	Нейтральные	Базовые
	Не нравится	Нежелательные	Нежелательные	Нежелательные	Нежелательные	-

Реализации изначально подлежат базовые функции, затем основные, восхищающие, после чего нейтральные, а потом нежелательные.

Данный метод чаще всего используется на этапе анализа требований в процессе создания нового продукта.

Среди основных достоинств модели Кано – помощь в анализе удовлетворенности пользователей, а также в определении потенциальных типов пользователей создаваемого программного обеспечения. К недостаткам данного подхода относится необходимость постоянной перегруппировки функций для получения актуального результата, поскольку те функции, которые в один момент времени являются восхищающими, в следующий момент уже будут ожидаемыми, а ожидаемые со временем становятся базовыми. Также недостат-

ком является невозможность использовать модель Кано для системных требований и тот факт, что данный подход не учитывает мнение команды разработчиков.

Также к данной категории принадлежит метод, основанный на построении оценочных листов [6]. Данный подход состоит в приоритизации задач, подлежащих реализации путем согласования набора критериев оценки со всеми заинтересованными лицами. Согласно данному методу, изначально определяются от пяти до девяти критериев, которые помогут выяснить, какие задачи являются важными для реализации в текущем спринте. Среди возможных критериев:

– важность функции для пользователей;

– прибыль, которую можно получить от реализации функции;

– важность функции для решения последующих задач;

– риски, связанные с реализацией данной задачи.

Затем выбирается задача, которая вероятнее всего будет реализована в рамках данного спринта и которая является максимально понятной для всех членов команды. Данная задача считается базовой, и все рассматриваемые задачи будут оцениваться по каждому выбранному критерию относительно данной задачи.

В результате строится таблица (табл. 2), где по вертикали указываются критерии, а по горизонтали – задачи. В том случае, если важность критериев не одинакова, то в первом столбце в процентах указывается важность каждого критерия. Далее каждая задача оценивается по тому или иному критерию, и если она важнее базовой, то в соответствующей ячейке указывается +1, если менее важна, -1. Если важность задачи равна базовой, то устанавливается значение 0. После того, как все задачи оцениваются по всем критериям, происходит подсчет суммарной важности задачи по всем критериям. Полученное число и является приоритетом задачи.

Таблица 2 – Оценочный лист

Критерий	Вес	Задача 1	Задача 2 (Базовая)	Задача 3	Задача 4
Критерий 1	50%	1	0	-1	1
Критерий 2	30%	0	0	0	1
Критерий 3	20%	0	0	1	0
Критерий 4	10%	1	0	1	1
Сумма баллов		0,6	0	-0,2	0,9
Приоритет		2	3	4	1

Данный метод чаще всего используется при высокоуровневом планировании.

Среди достоинств этого метода будет масштабируемость и простота в использовании, недостатками же являются субъективность выбора критериев для оценки и высокая вероятность реализации в рамках одного спринта частей различных функций разрабатываемого программного обеспечения, что затрудняет дальнейшую разработку и не дает возможности заказчику получить готовую к эксплуатации промежуточную версию программного обеспечения.

На количественной оценке основывается также и метод, предложенный Карлом Вигерсом и позволяющий оценить относительные приоритеты для набора функций [7]. Согласно данному методу, привлекательность функции прямо пропорциональна ее полезному действию и обратно пропорциональна стоимости и риску, связанному с ее реализацией.

Для реализации данного метода необходимо прежде всего определить функции, которыми обязательно должно обладать создаваемое программное обеспечение. Далее заказчику или его представителю следует по шкале от 1 до 9 определить относительную выгоду, которую ему даст реализация каждой указанной функции. Затем необходимо опреде-

лить относительный урон, который может потерпеть бизнес в случае отсутствия функции в текущей или результирующей версии программного продукта. Для этой оценки также необходимо использовать шкалу от 1 до 9, причем 1 балл выставляется, если отсутствие функции никак не повлияет на результат, а 9, если это принесет существенный урон. Сумма выгоды и урона определит ценность функции.

Таким образом, те функции, которые будут иметь низкий рейтинг и выгоды, увеличивают стоимость программного продукта, но при этом имеют незначительную ценность.

На следующем этапе происходит оценка стороной заказчика относительной стоимости реализации имеющихся функций. Для данной оценки необходимо учесть сложность функций, возможность повторного использования их кода, объем тестирования и документации по ним.

Предпоследним шагом происходит оценка технического риска, то есть вероятности неудачной реализации функции с первого раза. При этом оценка 1 будет означать, что риск минимален, а 9, что риск максимален.

Наконец, расчет приоритета функции производится по следующей формуле:

$$P = \%c(\%s * w\_s) + (\%r * w\_r), \quad (1)$$

где *%c* – выраженная в процентах ценность функции;

%s – выраженная в процентах стоимость функции;

w_s – вес стоимости функции;

%r – выраженный в процентах риск неудачной реализации функции;

w_r – вес риска.

Чаще всего данный метод используется при долгосрочном планировании. К его недостаткам принадлежит сложность точной оценки выгоды, урона, стоимости и риска каждой конкретной функции.

Еще одним методом количественной оценки является метод структурирования функции качества Йоджи Акао Quality Function Deployment (QFD) [8], который позволяет сопоставить пожелания заказчика с имеющимися аналогичными продуктами и рассмотреть функции создаваемого программного обеспечения как с точки зрения заказчика, так и с точки зрения потенциального пользователя. Суть данного метода заключается в построении таблицы специального вида, которая, благодаря своей форме, получила название «дома качества» (Quality House) (рис. 2).

Изначально от потенциальных пользователей создаваемого программного продукта получают информацию о том, какие их требования к программе. Далее для каждого из требований пользователя им же устанавливается приоритет. На следующем этапе команда разработчиков формулирует функциональные характеристики продукта, после чего эти характеристики вместе с требованиями пользователей заносятся в центральную часть таблицы (рис. 2): по столбцам указываются функциональные характеристики, а по строкам – пользовательские. Следующий шаг – определение зависимостей между пользовательскими требованиями и функциональными характеристиками продукта. В данном случае оптимально использовать цифровые значения в диапазоне от 1 до 9, где 1 будет означать наличие слабой связи, а 9 – сильного отношения, допускается также и отсутствие какого-либо значения, что говорит об отсутствии зависимости между соответствующей функцией и требованием пользователя. Для обозначения зависимостей возможно использование и таких характеристик, как «сильная связь», «средняя связь» и «слабая связь». Эти значения и указываются в соответствующих ячейках центральной части «дома качества».

«Крыша дома качества» представляет собой корреляционную матрицу, в которой указывается связь между пользовательскими требованиями и функциональными характеристиками: если они противоречат друг другу, то соответствующая ячейка помечается знаком «-», если же не противоречат – «+».

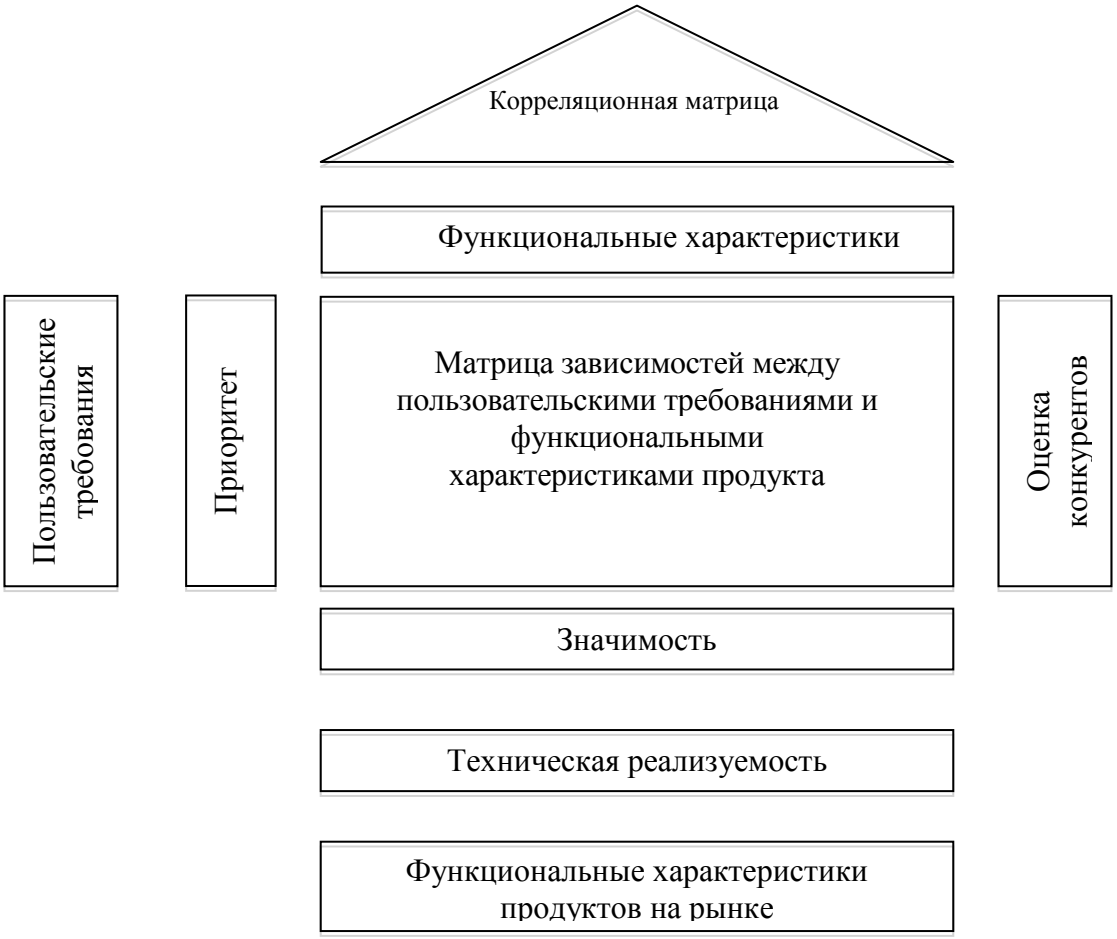

Рисунок 2 – «Дом качества»

На следующем этапе происходит формирование «подвала дома качества». В нем первой строкой указывается значимость каждой имеющейся функциональной характеристики. Для вычисления значимости определенный пользователем рейтинг пользовательского требования умножается на показатель зависимости между функциональной характеристикой и этим пользовательским требованием, затем результаты по всем строкам для одной и той же технической характеристики суммируются и получается конечное значение значимости технической характеристики. Первыми реализации подлежат характеристики с наибольшим значением значимости. Второй строкой в «подвале дома качества» указывается, насколько технически реализуемой является каждая характеристика, а третьей – результаты функциональных характеристик аналогичных продуктов на рынке.

Данный метод используется при долгосрочном планировании.

4. Выводы

Гибкие методологии разработки программного обеспечения с каждым днем становятся все более и более популярными, так как позволяют быстро подстраиваться под постоянно изменяющиеся требования рынка, осуществлять ежедневный контроль за ходом выполнения проекта и быстро запускать его в эксплуатацию с наиболее приоритетными функциями, но минимальным бюджетом.

Одним из важнейших этапов в любой гибкой методологии является приоритизация задач для реализации в рамках всего выполняемого проекта или его отдельного спринта,

что есть достаточно сложной задачей. Однако на данный момент существует большое количество различных методов приоритизации, часть из которых учитывает мнение команды разработчиков, другая часть – различные количественные показатели. Использование этих методов по отдельности или в комплексе поможет быстро и эффективно решить вопрос, связанный с установлением последовательности реализации задач, подлежащих разработке.

СПИСОК ИСТОЧНИКОВ

1. Книберг Х., Скарин. М. Scrum и Kunban: выжимаем максимум. Киев: C4Media, InfoQ.com, 2010. 78 с.
2. Приоритизация MoSCoW. URL: https://myalm.ru/news/Приоритезация-MoSCoW (дата обращения: 08.05.2019).
3. Patton Je., Economy P. User Story Mapping: Discover the whole story,build the right product. O'Reilly Media, Inc., 2014. 277 p.
4. Мартин Б., Ханингтон Б. Универсальные методы дизайна. СПб.: Питер, 2014. 208 с.
5. Kano N., Seraku N., Takahashi F., Tsuji S. Attractive Quality and Must-be Quality. J*ournal of the Japanese Society for Quality Control*. 1984. Vol. 14 (2). P. 147–156.
6. Theme scoring. URL: https://www.mountaingoatsoftware.com/tools/theme-scoring (дата обращения: 08.05.2019).
7. Вигерс К. Разработка требований к программному обеспечению / пер. с англ. М.: Издательско-торговый дом “Русская редакция”, 2004. С. 275–280.
8. Ryan N. Methods and QFD. Hows and Whys for Management. Dearborn, Michigan: ASI Press, 1988. 80 p.

Стаття надійшла до редакції 03.03.2020