УДК 004.4

МОДЕЛИРОВАНИЕ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ СБОРА ТРЕБОВАНИЙ ДО ВНЕДРЕНИЯ НА ОСНОВЕ ПРИМЕНЕНИЯ UML-ДИАГРАММЫ

К. И. Бушмелева, Л. Р. Зарипова
Сургутский государственный университет
bkiya@yandex.ru, zlr23.07@gmail.com

В данной работе представлена модель сети Петри жизненного цикла программного обеспечения от сбора требований до внедрения посредством языка UML. Была дана характеристика бизнес-процессов жизненного цикла программного обеспечения, построены Case-модель и маркированная модель сети Петри с подробным описанием, после чего было реализовано дерево достижимости. Авторами проведен анализ сети Перти на безопасность и ограниченность, в результате которого было выявлено, что программное обеспечение является надежным.

Ключевые слова: сети Петри, программное обеспечение, UML-диаграммы, жизненный цикл.

SOFTWARE LIFE CYCLE MODELING FROM REQUIREMENTS GATHERING TO IMPLEMENTATION BASED ON THE USE OF UML DIAGRAMS

K. I. Bushmeleva, L. R. Zaripova
Surgut State University
bkiya@yandex.ru, zlr23.07@gmail.com

The paper presents a Petri net model of the software life cycle from requirements gathering to implementation via UML. The characteristic of the business processes of the software life cycle is given. The Case-model and the marked Petri net model with detailed description are constructed. Afterwards, the reachability tree is implemented. The authors analyzed the Petri net for security and limitations resulting in the reliability of software.

Keywords: Petri nets, software, UML diagrams, life cycle.

На сегодняшний день проекты по разработке программного обеспечения (далее – ПО) нуждаются в грамотном управлении требованиями, потому что с помощью четко сформулированных требований от заказчика можно получить наглядное представление о выпускаемом продукте, т. е. от полноценного сбора требований зависит качественное описание предметной области и предложенные технические решения.

Управление требованиями к ПО – это процесс, который включает идентификацию, выявление потребностей, документирование концепции, анализ требований, отслеживание ошибок, изменение требований [1].

Основными этапами выработки требований являются:

- подготовка требований – это документирование жизненного цикла (далее – ЖЦ) требований при разработке ПО;
- управление требованиями – это базовая версия управления требованиями, которая подразумевает собой набор функциональных требований, которые разработчики выполнят в определенной итерации программы.

Основная причина, по которой проваливаются проекты, связана с управлением требований от заказчика, прежде всего это неполнота требований, которая в свою очередь влияет на результат работы, на тестирование программы и, как результат, приводит к дополнительным затратам со стороны заказчика [2]. Этапы управления требованиями включают в себя:

- документирование проекта: описание концепции проекта, цели, способов решения, функциональных требований заказчика, составление технического задания;
- контроль над статусом требований;

- изменение требований по результатам тестирования ПО на стороне заказчика.

Шаблон состава требований содержит информацию о дате создания, номере версии ПО, авторе требований (аналитик), заказчике требований, техническом задании (далее – ТЗ).

Требования от заказчика собираются с помощью интервью посредством задавания вопросов по требованиям. Также необходимо детальное изучение цепочки бизнес-процессов и понимание того, в чем заключается ожидаемый заказчиком результат. Следует провести мозговой штурм с разработчиками ПО и получить информацию о возможности реализации нескольких вариантов, сроках выполнения разработки. Далее производится окончательное согласование требований с заказчиком. От данного этапа зависит рентабельность проекта в целом, и тогда можно сделать вывод, будет ли он успешным [3].

Таким образом, для решения указанной проблемы в данной работе вначале было произведено описание бизнес-процессов ЖЦ программного обеспечения от сбора требований до внедрения ПО. Далее была проанализирована надежность ЖЦ ПО с использованием дерева достижимости. Ниже приведена более детальная характеристика происходящих бизнес-процессов (табл. 1) и рассмотрена Case-модель (рис. 1) жизненного цикла программного обеспечения от сбора требований до внедрения на стороне заказчика.

Таблица 1

Характеристика бизнес-процессов ЖЦ ПО

№	Входные значения	Функция	Выходные значения
1	Требования заказчика	Анализ. Аналитик собирает данные у заказчика	Сбор требований
2	Сбор требований	Проектирование ПО. На основе полученного сбора требований аналитик формирует техническое задание	Техническое задание
3	Техническое задание	Кодирование. Аналитик передает ТЗ на разработку ПО программисту	Готовая программа
4	Готовая программа	Тестирование. Программист пишет программный код и производит первоначальную внутреннюю проверку выходных данных ПО, далее отдает ПО на тестирование тестировщику	Результат тестирования
5	Результат тестирования	Внедрение. Сотрудники отдела внедрения получают ПО для внедрения на стороне заказчика	Отчет о внедрении программы
6	Отчет внедрении программы	Тестирование на стороне заказчика. После этапа внедрения ПО у заказчика пользователи также тестируют полученную программу	Отчет о тестировании на стороне заказчика

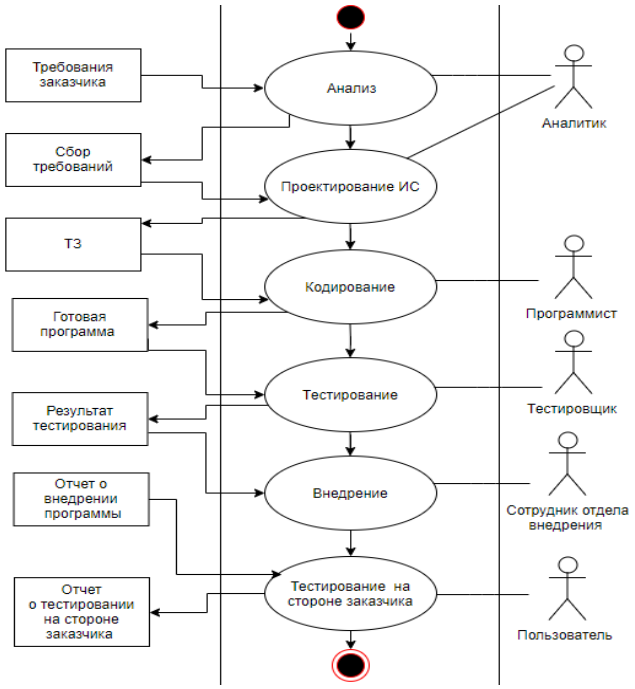

Рис. 1. Case-модель ЖЦ ПО

На следующем этапе была построена маркированная модель сети Петри ЖЦ ПО (рис. 2) и дано ее подробное описание (табл. 2) от сбора требований до внедрения на стороне заказчика. Модель включает возможные варианты перехода событий назад [4].

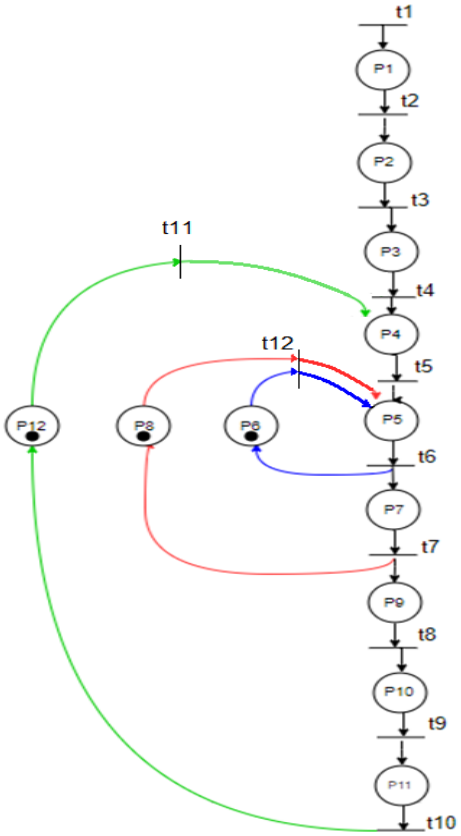

Рис. 2. Маркированная модель сети Петри ЖЦ ПО

Таблица 2

Описание маркированной модели сети Петри ЖЦ ПО

№	Предусловия переходов (T_N)	Позиция (P_{an})	Постусловия переходов (T_N)
1	Идентификация пользователя (t_1)	Пользователь распознан (логин, пароль) (P_1)	Формирование заявки (t_2)
2	Формирование заявки (t_2)	Заявка составлена и передана в службу технической поддержки (P_2)	Обработка заявки (t_3)
3	Обработка заявки (t_3)	Анализ заявки, статус «В разработку» (P_3)	Сбор требований от заказчика. Анализ предметной области (t_4)
4	Сбор требований от заказчика. Анализ предметной области (t_4)	Формирование ТЗ (P_4)	Начало разработки (t_5)
5	Начало разработки (t_5)	Состояние разработки (P_5)	Начало внутреннего тестирования (t_6)
6	Начало внутреннего тестирования (t_6)	Позиция P_6 используется для отображения состояний разработки ПО, если в P_6 имеется метка, то разработчик свободен и пришедшая заявка вызывает срабатывание перехода t_5. Пока программа не будет корректно работать, метки в P_6 не будет, следовательно, пришедшие в позицию P_5 запросы вынуждены ожидать срабатывания перехода P_6	Доработка программы (t_{12})

Окончание табл. 2

№	Предусловия переходов (T_N)	Позиция (P_{an})	Постусловия переходов (T_N)
7	Начало внутреннего тестирования (t_6)	Состояние тестирования (P_7)	Тестирование выполнено. (t_7)
8	Тестирование выполнено (t_7)	Позиция (P_8) используется для отображения состояний фиксирования багов тестировщиком, если в P_8 имеется метка, то тестировщик свободен и пришедшая заявка вызывает срабатывание перехода t_5. Пока баг не будет исправлен, метки P_8 не будет, следовательно, пришедшие в позицию P_5 запросы вынуждены ожидать срабатывания перехода	Доработка программы (t_{12})
9	Тестирование выполнено (t_7)	Состояние внедрения (P_9)	Внедрение выполнено (t_8)
10	Внедрение выполнено (t_8)	Передача программы заказчику (P_{10})	Начало тестирования на стороне заказчика (t_9)
11	Начало тестирования на стороне заказчика (t_9)	Пользователи тестируют программу (P_{11})	Тестирование на стороне заказчика выполнено. (t_{10})
12	Тестирование на стороне заказчика выполнено (t_{10})	Новые требования к программному продукту (P_{12})	Сбор требований от заказчика. Анализ предметной области (t_{11})
13	Сбор требований от заказчика. Анализ предметной области (t_{11})	Формирование ТЗ (P_4)	Начало разработки (t_5)
14	Исправление ошибок в разработке программы (t_{12})	Состояние разработки (P_5)	Начало внутреннего тестирования (t_6)

Далее в работе был произведен анализ маркированной модели сети Петри ЖЦ программного обеспечения от сбора требований до внедрения на стороне заказчика за счет разработанного дерева достижимости (рис. 3). Деревом достижимости называется ориентированный граф, вершинами которого являются разметки сети, а дуги взвешены переходами и связывают непосредственно достижимые маркировки [5].

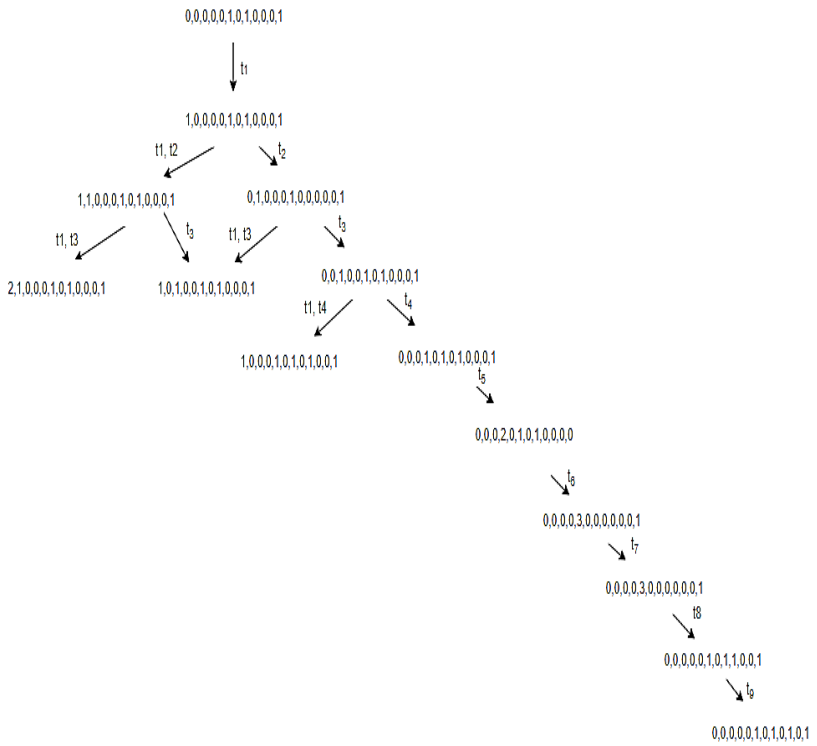

Рис. 3. Дерево достижимости

Для анализа дерева достижимости была использована прикладная теория сетей Петри. Известно, что сеть Петри – это математический аппарат, используемый для моделирования динамических дискретных систем [6].

Структура сети Петри описывается ее позициям и переходами, соединенными между собой дугами (при этом множество позиций и переходов не пересекаются), а также входной и выходной функциями [7]. Событием в данной сети называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно либо разновременно при выполнении некоторых условий. Сеть Петри (С) может быть представлена четверкой параметров (1):

$$C = (P, T, I, O), \quad (1)$$

где $P = \{p_1, p_2, p_3, \dots, p_n\}$ – конечное множество позиций, $n \geq 0$;

$T = \{t_1, t_2, t_3, \dots, t_m\}$ – конечное множество переходов, $m \geq 0$;

$I: T \rightarrow P$ – входная функция, представленная отображением переходов в комплекты позиций;

$O: P \rightarrow T$ – выходная функция, представленная отображением из комплектов позиций в переходы.

Состав сети Петри определяется по укрепленной входной функции I и выходной функции O (рис. 4).

C = (P, T, I, O)
P = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12}
T = {t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12}

I(p1) = {t1},	O(p1) = {t2},	I(t1) = {},	O(t1) = {p1},
I(p2) = {t2},	O(p2) = {t3},	I(t2) = {p1},	O(t2) = {p2},
I(p3) = {t3},	O(p3) = {t4},	I(t3) = {p2},	O(t3) = {t3},
I(p4) = {t4, t11},	O(p4) = {t5},	I(t4) = {p3},	O(t4) = {p4},
I(p5) = {t5, t12},	O(p5) = {t6},	I(t5) = {p4},	O(t5) = {p5},
I(p6) = {t6},	O(p6) = {t12},	I(t6) = {p5},	O(t6) = {p6},
I(p7) = {t6},	O(p7) = {t7},	I(t7) = {p7},	O(t7) = {p8, p9},
I(p8) = {t7},	O(p8) = {t12},	I(t8) = {p9},	O(t8) = {p10},
I(p9) = {t7},	O(p9) = {t8},	I(t9) = {p10},	O(t9) = {p11},
I(p10) = {t8},	O(p10) = {t9},	I(t10) = {p11},	O(t10) = {t12,
I(p11) = {t9},	O(p11) = {t10},	I(t11) = {p12},	O(t11) = {p4},
I(p12) = {t10}.	O(p12) = {t11}.	I(t12) = {p8}.	O(t12) = {p5}.

Рис. 4. Структура сети Петри системы управления требованиями

Основные задачи исследования сетей Петри можно разделить на две группы: задачи анализа активности переходов и задачи анализа достижимости, при этом задача достижимости является основной и может быть представлена деревом достижимости. Дерево представляет все возможные последовательности запусков переходов. Всякий путь в дереве, начинающийся в корне, соответствует допустимой последовательности переходов. Для получения дерева достижимости, которое можно считать полезным инструментом анализа, необходимо найти средства ограничения его до конечного размера.

В данной работе анализ дерева достижимости произведен по свойствам: ограниченности и безопасности. Рассмотрим эти свойства сети Петри:

1. Безопасность – единичный риск ограниченности при К = 1. Это означает надежность сети Петри, когда ни при каких условиях не может появиться более одной метки в любой позиции [8, 9]. Очевидно, что безопасные сети могут быть реализованы логическими схемами с двоичными элементами памяти, представляющими отдельные позиции этой сети.

2. Ограниченность – величина меток в каждой позиции сети не может превышать некоторого значения К, т. е. не допускается превышения количества меток в данной позиции некоторого фиксированного числа [8, 9].

Подводя итог, можно отметить, что в работе была построена Case-модель жизненного цикла программного обеспечения от сбора требований до внедрения на стороне заказчика. На основе реализованной структуры данной модели была смоделирована маркированная сеть Петри, что позволило реализовать дерево достижимости. В результате получили выводы о том, что анализ безопасной и ограниченной модели ЖЦ ПО с помощью сети Петри доказывает надежность проектирования ПО.

Литература

1. Атисков А. Ю. Разработка технологии и программной системы автоматизированной трансформации диаграмм функционального проектирования в диаграммах UML. URL: http://www.dissercat.com/content/ (дата обращения: 02.07.2018).

2. Буч Г., Максимчук Р., Энгл М., Янг Б., Коналлен Дж., Хьюстон К. Объектно-ориентированный анализ и проектирование с примерами приложений. 3-е изд. / пер с англ. М. : ООО «И. Д. Вильямс», 2010. 720 с.

3. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри. Курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления. СПб. : Профессиональная литература ; АйТи-Подготовка, 2014. 400 с.

4. Воевода А. А. Разработка программного обеспечения с применением UML диаграмм и сетей Петри для систем управления локальным оборудованием : автореф. дис. … канд. техн. наук. 2012. URL: http://www.dissercat.com/content/razrabotka-programmnogo-obespecheniya-s-primeneniem-uml-diagramm-i-setei-petri-dlya-sistem-u (дата обращения: 20.01.2019).

5. Зарипова Л. Р., Бушмелева К. И. Модель мультиверсионной информационной системы управления изменениями программного обеспечения в информационных проектах // Вестник кибернетики. 2018. № 3 (31). С. 212–216.

6. Питерсон Дж. Теория сетей Петри и моделирования. М. : Мир, 1984. 264 с.

7. Мальков М. В., Малыгина С. Н. Сети Петри и моделирование // Сборник научных трудов 2010. URL: https://cyberleninka.ru/article/v/seti-petri-i-modelirovanie (дата обращения: 20.01.2019).

8. Марков А. В. Автоматизация проектирования анализа программного обеспечения с использованием языка UML и сетей Петри : автореф. дис. … канд. техн. наук. 2015. 24 с. URL: https://www.nstu.ru/files/dissertations/markov_abstract_142926270737.pdf (дата обращения: 20.01.2019).

9. Пашковская Е. С. Математическое и программное обеспечение мультиверсионной информационной системы с универсальной структурой интерфейсов на основе маркированной сети Петри : автореф. дис. … канд. техн. наук. Воронеж, 2016. 18 с.