УДК 004.912
DOI: 10.25140/2411-5363-2018-1(1)-79-88

Алексей Кунгурцев, Наталия Новикова, Мария Решетняк, Яна Черепинина

УТОЧНЕНИЕ КЛАССИФИКАЦИИ И МОДЕЛЕЙ ПУНКТОВ СЦЕНАРИЕВ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Актуальность темы исследования. *Варианты использования являются одним из важных способов представления функциональных требований к проектируемым информационным системам. Их описание – достаточно сложный и трудоёмкий процесс. Поэтому исследования, направленные на автоматизацию этого процесса, являются актуальными.*

Постановка проблемы. *Для автоматизации описания вариантов использования необходима разработка классификации и моделирования пунктов сценариев. В данной работе решается проблема корректировки существующей классификации пунктов вариантов использования.*

Анализ последних исследований и публикаций. *Были рассмотрены последние публикации по теме применения вариантов использования. За основу принята классификация, предложенная в работе «Информационная технология автоматизированного составления вариантов использования».*

Выделение неисследованных частей общей проблемы. *Создание модели для пункта «Завершение прецедента».*

Постановка задачи. *Корректировка классификаций пунктов вариантов использования, создание и внесение изменений в модели пунктов сценариев.*

Изложение основного материала *Предложена усовершенствованная модель описания вариантов использования, отражающая как функциональные требования, так и концептуальные классы проектируемого программного продукта. Классификация пунктов сценария вариантов использования дополнена новым пунктом «Завершить вариант использования». Внесены изменения в ранее предложенные модели и созданы новые модели для пунктов сценария «Создать», «Запросить значение», «Действия пользователя, которые не укладываются в предложенную классификацию». Созданы соответствующие алгоритмы и выполнена модернизация программного продукта, поддерживающего технологию автоматизированного составления вариантов использования.*

Выводы в соответствии со статьей. *Реализация предложенных моделей и алгоритмов позволила сократить время на описание вариантов использования до 10 %, а также создать модели концептуальных классов для проектируемого программного продукта.*

Ключевые слова: *варианты использования; сценарии; модели; прецеденты; концептуальные классы.*
Рис.: 3. Библ.: 11.

Актуальность темы исследования. Варианты использования (use case, прецеденты) часто используются в проектировании информационных систем в качестве наиболее точного и полного способа формулировки функциональных требований [1; 2]. Описание варианта использования (ВИ) – трудоемкий и ответственный этап работы по формулировке требований [4], от качества выполнения которого во многом зависит успешность проекта. Поэтому исследования, направленные на автоматизацию процесса составления ВИ, позволяющие сократить время подготовки и повысить качества описания ВИ являются актуальными.

Постановка проблемы. Автоматизация процесса составления ВИ должна быть основана на классификации пунктов сценариев. Классификация позволит создать модели для описания различных пунктов, а также дает возможность предложить структуру концептуальных классов, поддерживающих сценарии ВИ.

Проблема – существующая классификация не перекрывает все варианты пунктов сценариев, не предлагает модели для некоторых пунктов, поэтому обеспечивает только частичную автоматизацию процесса описания ВИ.

Анализ последних исследований и публикаций. Не смотря на достаточно большое количество публикаций, в той или иной степени определяющих правила составления [3–5] и использования [5–7] ВИ, до настоящего времени не существовало классификации возможных пунктов их сценариев. Это приводило к тому, что в проектировании программных продуктов (ПП) прецедент рассматривался как заранее описанный элемент диаграмм [7–9] и вопросам автоматизации его составления не уделялось внимание.

Исходя из того, что ВИ – это сценарий работы пользователя с будущим ПП, системный аналитик при его описании обязательно продумывает архитектуру ПП. Однако эта информация никак не фиксировалось на этапе формулировки требований, что фактически приводило к дублированию работы на этапе реализации ВИ.

В работе [10] впервые предложена классификация пункты сценариев прецедентов. На её основе построены математические модели, позволившие автоматизировать процесс составления ВИ путем предоставления системному аналитику шаблонов для каждого типа пункта. Кроме этого, была сделана попытка зафиксировать минимально необходимое количество классов (концептуальные классы), необходимых для обеспечения реализации ВИ.

Ниже приведена классификация пунктов ВИ.

- Создать. Пользователь приказывает системе создать некоторый документ (объект), который в зависимости от положения соответствующего пункта в сценарии может играть роль некоторого шаблона для накопления информации, либо отчета о выполненной работе.

- Ввести данные. Пользователь вводит в систему ряд данных, для которых система обычно должна проверить возможность их использования для дальнейшей работы.

- Запросить значение. Пользователь запрашивает у системы некоторое данное. Обычно после этого следует оценка данного пользователем.

- Запросить список. Пользователь заказывает список (например, данных, услуг или документов) для дальнейшего выбора из него некоторых элементов.

- Выбрать из списка. Пользователь выбирает из списка нужное данное или услугу (документ).

Ввести услугу (документ). Пользователь вводит необходимую услугу или документ, который определяет дальнейшую последовательность действий. Например, способ оплаты по банковской карточке.

Повторение действий. Пользователь имеет возможность перейти к расположенным выше пунктам сценария, либо отказаться от их повторения.

Свободно конструируемый пункт. Очевидно, возможна ситуация, когда пользователь не обнаружил нужной ему операции среди предложенных действий с системой. В этом случае ему предложено создать свободно конструируемый пункт.

Выделение неисследованных частей общей проблемы и постановка задачи. Анализ приведенной классификации, а также опыт использования программного продукта UseCaseEditor, реализующего предложенную технологию, показали необходимость решения следующих задач:

- составление общей модели пунктов сценариев;
- корректировка пункта «Создать», поскольку заложенные в нем проверки условий практически никогда не используются;
- корректировка пункта «Запросить значение» для определения источника получения данных;
- составление математической модель для пункта «Свободно конструируемый пункт», что позволит автоматизировать его составление;
- добавление в классификацию нового пункта «Завершить вариант использования».

Основной материал.

Модели пунктов сценария. На рис. 1 представлена структура модели пункта сценария. Модель описания пункта входит в описание прецедента. Модель проектирования пункта не имеет представления в требованиях и является начальным этапом проектирования.

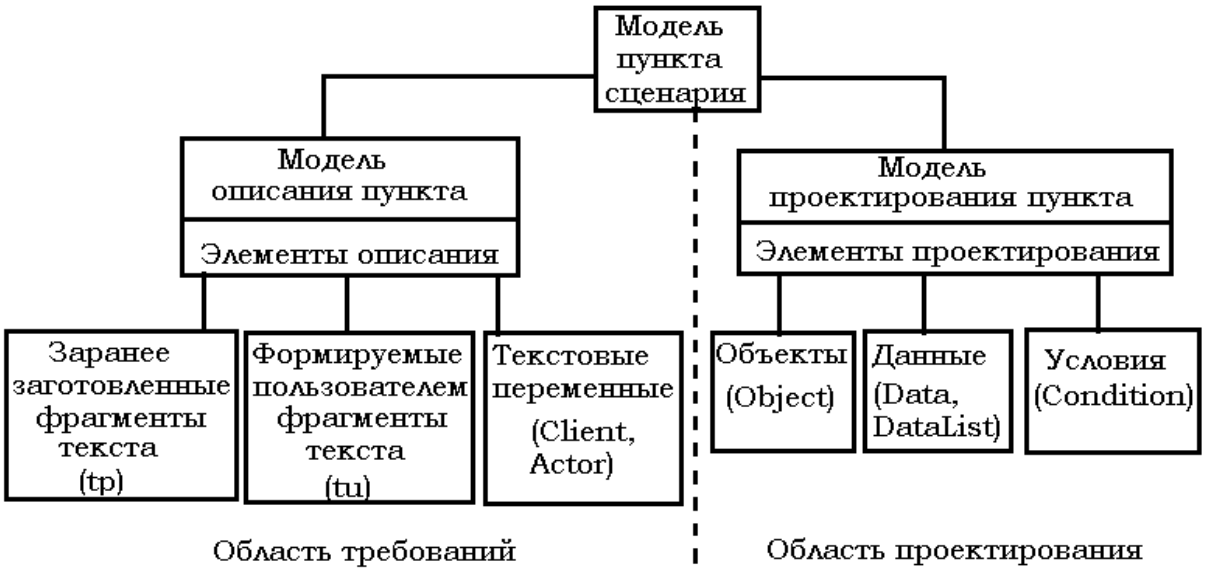

Рис. 1. Структура модели пункта сценария ВИ

В соответствии с [10] элементами описания могут быть заранее заготовленные фрагменты текста – *tpi*, фрагменты текста, формируемые в процессе составления прецедента – *tuj* , либо текстовые переменные, представляющие исполнителей и заинтересованных лиц – *Client*, *Actor*. Элементами проектирования являются объекты концептуальных классов – *Object* или ссылки на объекты – *addrKeep* , отдельные данные или их списки – *Data*, *DataList*, которые должны быть введены в ПП или получены из ПП, а также условия выполнения операций – *Condition.*

Значения данных и условий могут вводиться системным аналитиком для дальнейшего использования в процессе тестирования прецедента.

В модели используется ряд метасимволов. Отдельные элементы или группы элементов, заключенные в квадратные скобки – «[ ]», не обязательно должны присутствовать в описании пункта ВИ. Символ «+» обозначает конкатенацию строк. Символ «/» обозначает использование одного из двух элементов, разделенных этим символом.

Предлагается на основе ранее сформулированных требований и использованных для их составления документов создать словарь предметной области [11], который даст единое и согласованное толкование терминов.

Пункт «Создать». В [10] предполагалась следующая последовательность действий, описывающих этот пункт. Пользователь предписывает системе создать некоторый объект, который может содержать данные, используемые как в рамках данного прецедента, так и в других прецедентах. Вводимые данные в общем случае могут проходить проверку на корректность, для этого вводились соответствующие условия.

В данной редакции исходим из того, что, если объект создается в рамках пункта, то предварительно заданных условий нет. Проверять данные одновременно с вводом условий их проверки не имеет смысла. Поэтому из модели пункта исключается необязательное условие и соответствующий фрагмент текста $tp6$ =”система подтверждает”, а также возможная проверка атрибутов. Кроме этого, уточняем, что возможно повторение части описания. В результате описание пункта принимает вид:

$$Create =< N,[Client,tp1,Actor,tp2,tu1],Actor,tp3,Object[,tp5,< tu2,Data >]>, \quad (1)$$

где N – номер пункта прецедента;

- *Client* – необязательный элемент; вводится, если требуется определить, от кого исходит инициатива;

$tp1$ = “обращается к”;

$tp2$ = “по поводу”;

$tp3$ = “создает в системе”;

$tu1$ – текст, формируемый пользователем (например, «заказ на ремонт оборудования»);

- *Actor* – пользователь, взаимодействующий с системой (должен быть определен во введении к прецеденту);

- *Object* – имя создаваемого объекта; предусматривается проверка на отсутствие объекта с подобным именем.

При создании объекта с параметрами используется фрагмент из (1):

$tp5, < tu2, Data >$,

где $tp5$ = “с параметрами ”.

Кортеж $< tu2, Data >$ определяет операцию описания и ввода значений атрибутов объекта, которая может повторяться. Здесь

$tu2$ – название атрибута, формируемое пользователем (например, № телефона);

- *Data* – значение атрибута.

Пункт «Запросить значение». Пользователь запрашивает у системы некоторое данное. Обычно после этого следует оценка данного пользователем. В предыдущей версии не указывались данные, на основе которых определялось выводимое значение. В данной редакции пункта добавлено определение этих данных. Например, в предыдущей версии пункт был сформулирован так: «Определить стоимость корзины…». В настоящей редакции он формулируется так: «Определить стоимость корзины на основании количества и стоимости отдельных товаров….». Описание пункта принимает вид:

$inquiry =< N, [Client, tp1, tu1,] Actor, tp2, Data_q, tp3, Data_b [tp4, Condition, tp5,] tp6 >$

Client – необязательный элемент (вводится, если требуется определить, от кого исходит инициатива);

где $tp1$ = “желает получить”;

$tu1$ – фраза, формируемая пользователем, например, «стоимость услуги».

$tp2$ = “вводит в систему запрос на получение”;

$Data_q =< dataName_q, addrKeep_q >$ – запрашиваемое у системы данное; *addrKeep* – название объекта содержащего данное;

$tp3$ = “на основании”;

$Data_b =< dataName_b, addrKeep_b >$ – данные, на основании которых определяется запрашиваемое значение;

$tp4$ = “при условии”;

$Condition =< textCond, addrKeep >$ – условие получения данного, например, *textCond* может быть «за прошлый месяц»; *addrKeep* – название объекта, способного проверить и выполнить условие;

$tp5$ = “Система подтверждает корректность условия”- необязательный элемент;

$tp6$ = “Система выводит + $nameData$ + *Client* / *Actor*+ согласен”.

Пункт Завершить прецедент. Предложено ввести новый пункт, который предусматривает успешное завершение прецедента. Выполнение пункта может предусматривать прием некоторых данных, которые не требуют анализа (например, ввод фамилии и адреса заказчика некоторой услуги), регистрацию определенных данных (регистрация проданного билета), создание документа (чек, квитанция об уплате услуги).

Описание пункта имеет вид:

$Finish =< N, [Elem1], [Elem2], [Elem3], tp1 >$

где $tp1$ = ”Завершение работы прецедента”;

$Elem1, Elem2, Elem3$ - необязательные элементы, позволяющие принять данные, зарегистрировать транзакцию и сформировать документ соответственно.

Прием данных

$Elem1 =< Client / Actor, tp2, tu1, Data1, tp3, addrKeep1 >$,

где $tp2$ = “сообщает”;

$tu1$ – название данного предоставляемого клиентом;

$Data1$ – данное, которое сообщает клиент;

$tp3$ = “Система сохраняет данное в”;

$addrKeep1$ – объект, принимающий данное (объект выбирается из созданных ранее).

Операция приема данного может повторяться несколько раз.

Регистрация данных

$Elem2 =< tp4, Data2, tp5, addrKeep2 >$

$tp4$ = “Система регистрирует данное:”;

$Data2$ – регистрируемое данное (данное выбираются из списка созданных ранее);

$tp5$ = “в”;

$addrKeep2$ – объект, регистрирующий данное (возможно создание нового объекта).

Операция регистрации данного может повторяться несколько раз.

Создание документа

$Elem3 =< tp6, Object, tp7, DataList3 >$

$tp6$ = “Система выдает документ:”;

$Object$ – создаваемый объект (документ);

$tp7$ = “содержащий”;

$DataList3$ – список данных, включаемых в документ (данные выбираются из созданных ранее).

Свободно конструируемый пункт. В предыдущей версии модель пункта имела следующий вид:

$FreeConstr =< N, [Client, tu1,] Actor, tu2 >$

$tu1, tu2$ - фразы, формируемые пользователем.

Такое описание позволяло получить словесное определение пункта, но никак не определяло объекты и данные, необходимые для его реализации. Исходя из того, что пункты, входящие в классификацию, покрывают все возможные атомарные действия с проектируемой системой, в новой редакции пользователю предложено самостоятельно выбирать действия в произвольном порядке и количестве. В результате модель пункта принимает вид:

$FreeConstr =< N, [Client, tu1,] Actor, tu2 >, mAction, mFinishPhr ast >$,

где $mAction$ – мультимножество действий, где каждый элемент соответствует одному из возможных типов пунктов сценария;

$mFinishPhrast$ - различные варианты завершающих фраз пунктов.

Элемент, соответствующий пункту «Создать»

$elCreate =< tp1, Object, [tp2, ParamList,] >$

где $tp1$ = “создает в системе”;

- $Object$ – имя создаваемого объекта; предусматривается проверка на отсутствие объекта с подобным именем;

$tp2$ = ”с параметрами”;

- $ParamList$ – необязательный элемент; список атрибутов для создаваемого объекта.

Элемент, соответствующий пункту «Ввести данные»

$elInputDat\ a =< [Client, tp3, tu1,] Actor, tp4, DataList [, tp5] >$,

где $tp3$ = “предоставляет данные о”;

$tp4$ = “вводит в систему”;

$tp5$ = “Система проверяет данные”;

$tu1$ – текст формируемый пользователем, например, «место проживания»;

- $DataList = \{d_1, d_2, ..., d_n\}$ – список вводимых данных;

Каждое данное представляет собой кортеж

$d_i =< name, addrCheck, addrKeep >$,

где $name$ – наименование данного;

- $addrCheck$ – ориентировочный адрес объекта, осуществляющего проверку данного. Если адрес отсутствует, то значение данного не проверяется, $addrKeep$ – ориентировочный адрес объекта, сохраняющего значение данного.

Элемент, соответствующий пункту «Выбрать из списка»

$elSelectEl\ em =< [Client, tp6,] Actor, tp7, Elem, tp8, tp9, Object >$

где $tp6$ = “указывает на нужный элемент из списка”;

- $tp7$ = “выбирает из списка”;

- $Elem$ – элемент из списка;

- $tp8$ = “и вводит его в систему.”;

- $tp9$ = “Система фиксирует значение в”;

- $Object$ – имя объекта, который фиксирует $Elem$.

Элемент, соответствующий пункту «Ввести услугу»

$elInputSer\ vice =< [Client, tp10, tu1,] Actor, tp11, serviceNam\ e, tp12, Object >$

$tp10$ = “желает использовать”;

$tu1$ – текст, определяющий сервис (например, «осмотр ходовой части», «регулировка развала/схождения»);

$tp11$ = “вводит в систему”;

$serviceName$ – наименование сервиса. Название сервиса связано с Nm – номером пункта сценария, которому будет передано управление;

$tp12$ = “фиксируется в”;

- $Object$ – имя объекта, который фиксирует $serviceName$.

Элемент, соответствующий пункту «Повторение действий»

$elCyclActi\ on =< Client\ /\ Actor, tp13, Np >$

где $tp13$ = “желает повторить действия, начиная с пункта”

Np – номер пункта сценария, расположенный выше пункта N.

Элемент, соответствующий пункту «Запросить список»

$elListInqu\ iry =< [Client, tp14, tu2,] Actor, tp15, List, [tp16, Condition,] tp17 >$

где $tp14$ = “желает получить список”;

$tu2$ - фраза, формируемая пользователем, например, «предоставляемых услуг».

$tp15$ = “вводит в систему запрос на получение”;

$List =< listName, addrKeep >$ – запрашиваемый у системы список; $listName$ – название списка, $addrKeep$ - место хранения (формирования) списка;

$tp16$ = “при условии”;

Condition =< *textCond* , *addrKeep* > -условие получения данного, например, *textCond* может быть представлено текстом «за прошлый месяц»; *addrKeep* – название объекта, способного проверить и выполнить условие;

*tp*17 = "Система находит и выводит *listName*";

Варианты фраз, завершающие пункты сценария

*el*1*FinishPhrase* = "Система подтверждает корректность условия";

*el*2*FinishPhrase* = "Система проверяет и подтверждает возможность выполнения услуги".

*el*3*FinishPhrase* = "Система подтверждает корректность данных";

*el*4*FinishPhrase* = "Система подтверждает"

*el*5*FinishPhrase* = *Client* + "согласен."

*el*6*FinishPhrase* = *Actor* + "согласен".

Апробация

В соответствии с предложенными моделями пунктов ВИ были разработаны алгоритмы их реализации. Алгоритм формирования пункта «Завершить прецедент» представлен на рис. 2.

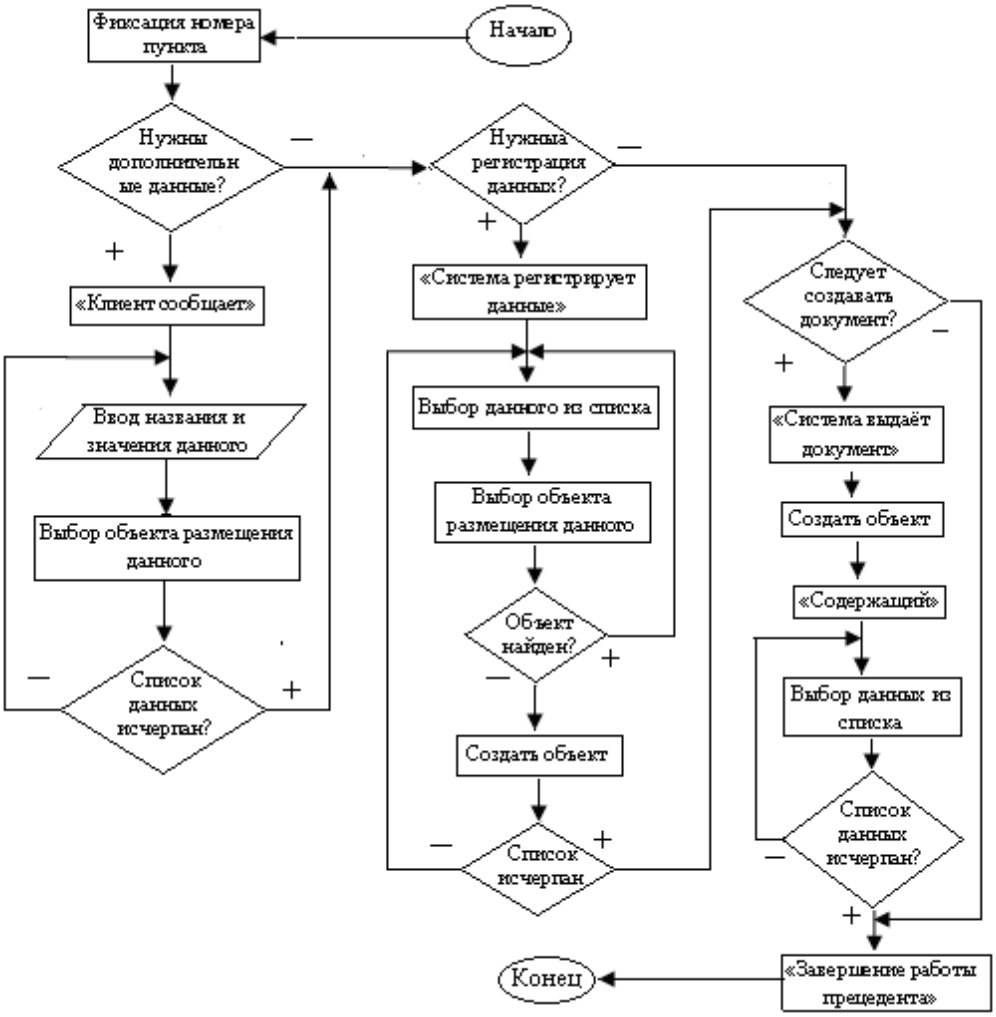

Рис. 2. Алгоритм формирования пункта сценария «Завершить прецедент»

Также была создана новая версия программного продукта UseCaseEditor1. На рис. 3 представлен пример формирования пункта прецедента.

Испытания показали сокращение времени на описание ВИ порядка 10 % за счет введения нового пункта «Завершить прецедент» вместо использовавшегося ранее «Свободно конструируемый пункт». Ситуации, когда бы потребовался «Свободно конструируемый пункт», не возникло, что косвенно указывает на полноту предложенной классификации пунктов сценариев.

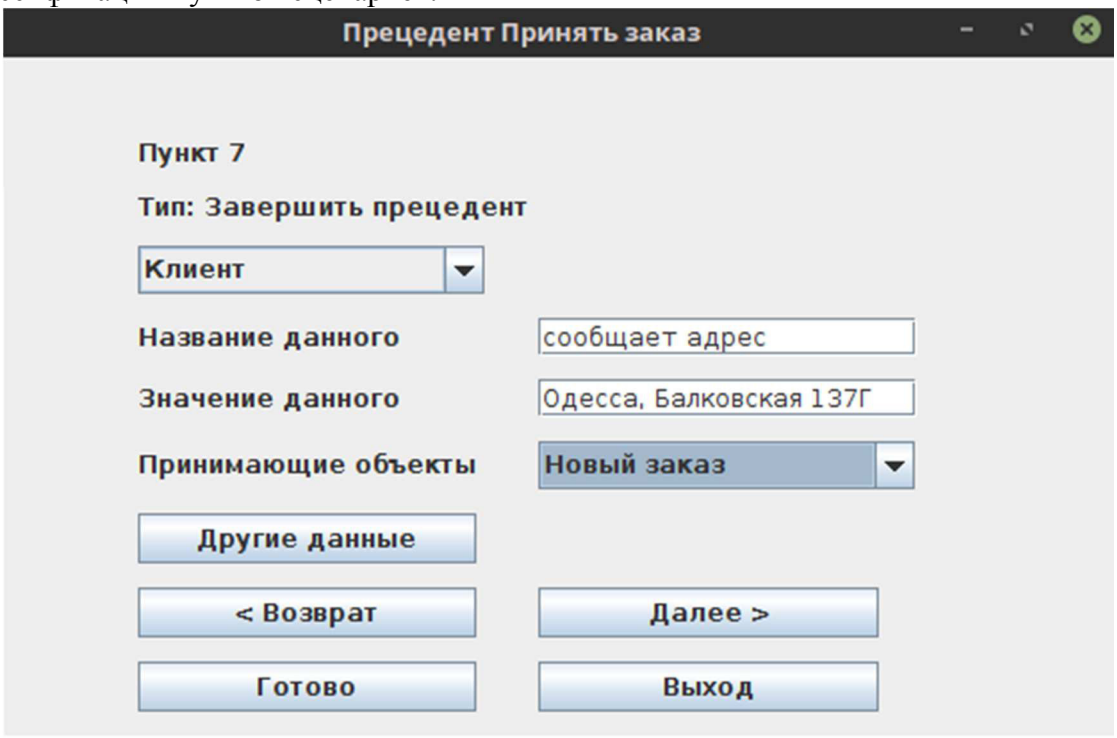

Рис. 3. Пример формирования пункта прецедента «Завершить прецедент»

Выводы в соответствии со статьей. Уточнена классификация возможных пунктов сценариев вариантов использования. Внесены изменения в трактовку пунктов «Создать», «Запросить значение», «Свободно конструируемый пункт». Предложен новый пункт «Завершить прецедент». Разработаны соответствующие математические модели. Принятые решения реализованы в новой версии программного продукта UseCaseEditor1. Проведенные эксперименты показали эффективность предложенной технологии в плане сокращения времени на описание прецедента и уменьшения количества ошибок.

Список использованных источников

1. *Прецеденты* в спецификации программ [Электронный ресурс]. – Режим доступа : http://club.shelek.ru/viewart.php?id=232.

2. *Создание* проекта. Анализ прецедентов. Реализация прецедентов. Уточненное описание прецедента [Электронный ресурс]. – Режим доступа : http://vunivere.ru/work72704.

3. *Леффингуэлл, Д. Уидриг*. Принципы работы с требованиями. Унифицированный подход / Леффингуэлл, Д. Уидриг. – М. : Вильямс, 2002. – 450 с.

4. *Алистер Коберн*. Современные методы описания функциональных требований к системам / Алистер Коберн. – М. : Лори, 2002. – 266 с.

5. *Леонов И. В.* Фирма «ЭСКЕЙП». Прецеденты (use cases) и их связи. Исполнители (actors) и их связи [Электронный ресурс] / И. В. Леонов. – Режим доступа : http://www.interface.ru/fset.asp?Url=/rational/vmr2.htm.

6. *Крэг Ларман*. Применение UML 2.0 и шаблонов проектирования / Крэг Ларман. – М. : Вильямс, 2008. – 736 с.

7. *Лененков А. В.* Объектно-ориентированный анализ и проектирование с использованием UML IBM и Rational Rose / А. В. Лененков. – М. : Бином ; ИНТУТЕ, 2006. – 320 с.

8. *Буч Г.* Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. – М. : ДМК-Пресс, 2007. – 496 с.

9. *Рамбо Дж.* UML 2.О. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб. : Питер, 2007. – 544 с.

10. *Возовиков Ю. Н.* Информационная технология автоматизированного составления вариантов использования/ Ю. Н. Возовиков, А. Б. Кунгурцев, Н. А. Новикова// Наукові праці Донецького національного технічного університету. – Покровськ, 2017. – № 1 (30). – С. 46–59.

11. *Кунгурцев А. Б.* Метод автоматизированного построения толкового словаря предметной области / А. Б. Кунгурцев, Я. В. Поточняк, Д. Ф. Силяев // Технологический аудит и резервы производства. – 2015. – № 2/2(22). – С. 58–63.

References

1. *Pretsedenty v spetsifikatsii programm [Precedents in the program specification*]. (n.d.). Retrieved from http://club.shelek.ru/viewart.php?id=232 [in Ukrainian].

2. Sozdanie proekta. Analiz pretcedentov. *Realizatciia pretcedentov. Utochnennoe opisanie pretcedenta [Create a project. Analysis of precedents. Implementation of precedents. Refined description of the use case]*. (n.d.). Retrieved from http://vunivere.ru/work72704 [in Russian].

3. Leffinguell, D. U. (2002). *Printcipy raboty s trebovaniiami. Unifitcirovannyi podkhod. [Principles of work with requirements. A unified approach]*. Moscow: Viliams [in Russian].

4. Kobern, A. (2002). *Sovremennye metody opisaniia funktcionalnykh trebovanii k sistemam.[Modern methods of describing functional requirements for systems*]. Moscow: Lori [in Russian].

5. Leonov, I. V. (n.d.). *Pretsedenty (use cases) i ikh svyazi. Ispolniteli (actors) i ikh svyazi [Use cases and their connections. Performers (actors) and their connections]*. Retrieved from http://www.interface.ru/fset.asp?Url=/rational/vmr2.htm [in Russian].

6**.** Larman, K. (2008). *Primenenie UML 2.0 i shablonov proektirovaniia. [Application of UML 2.0 and design patterns]*. Moscow: Viliams [in Russian].

7. Lenonenkov, A. V. (2006). *Obektno-orientirovannyi analiz i proektirovanie s ispolzovaniem UML IBM i Rational Rose [Object-oriented analysis and design using UML IBM and Rational Rose*]. Moscow: Binom; INTUTE [in Russian].

8. Buch, G, Rambo, Dz. & Dzhekobson, I. (2007). *Iazyk UML. Rukovodstvo polzovatelia [The UML language. User guide]*. Moscow: DMK-Press [in Russian].

9. Rambo, D. & Blakha, M. (2007). *UML 2.O. Obektno-orientirovannoe modelirovanie i razrabotka [Object-oriented modeling and development]*. SPb: Piter [in Russian].

10. Vozovikov, Iu. N., Kungurtsev, O. B. & Novikova, N. O. (2017). Informatsionnaia tekhnologiia avtomatizirovannoho sostavleniia variantov ispolzovaniia [Information technology of automated drawing up of use cases]. *Naukovi pratci Donetckogo natcionalnopo tekhnichnopo universitetu – Scientific works of Donetsk National Technical University,* 1, 46–59 [in Russian].

11. Kungurtsev, A. B., Potochniak, Ya. V. & Siliaev, D. A. (2015). Metod avtomatizirovannogo postroeniia tolkovogo slovaria predmetnoi oblasti [The method of automated construction of an explanatory dictionary of the subject domain.]. *Tekhnologicheskii audit i rezervy proizvodstva – Technological audit and production reserves,* 2, 58–63 [in Russian].

UDC 004.912

Oleksii Kungurtsev, Nataliia Novikova, Maria Reshetnyak, Yana Cherepinina

UPDATED CLASSIFICATIONS AND MODELS OF SCENARIOS OF USE CASES

Urgency of the research. *Use cases are one of the important ways of presenting functional requirements to the projected information systems. Their description is quite a complicated and time-consuming process. Therefore, studies aimed at automating this process are relevant.*

Target setting. *To automate the description of use cases, it is necessary to develop classification and modeling of scenario items. In this paper, the problem of adjusting the existing classification of items of use cases.*

Actual scientific researches and issues analysis. *The latest publications on the application of use cases were considered. Based on the classification proposed in the work "Information technology automated generation of use cases".*

Uninvestigated parts of general matters defining. *Creating a model for the "Completing of the precedent" item.*

The research objective. *Correction of classifications of items of use cases, creation and modification of model points of scenarios.*

The statement of basic materials. *An improved model for describing the use cases, reflecting both the functional requirements and the conceptual classes of the projected software product is proposed. Classification of points of the scenario of use cases is supplemented with a new item "Complete use case". Changes have been made to previously proposed models and new models have been created for the items in the "Create", "Request value", "User actions that do not fit into the proposed*

classification". The corresponding algorithms have been created and the software product has been modernized, supporting the technology of automated generation of use cases.

Conclusions. *The implementation of the proposed models and algorithms allowed to reduce the time for describing the use cases to 10 %, and also to create models of conceptual classes for the projected software product.*

Keywords: *use cases; scenarios; models; precedents*; *conceptual classes.*

Fig.: 3. References: 11.

УДК 004.912

Олексій Кунгурцев, Наталія Новікова, Марія Решетняк, Яна Черепініна

УТОЧНЕННЯ КЛАСИФІКАЦІЇ І МОДЕЛЕЙ ПУНКТІВ СЦЕНАРІЇВ ВАРІАНТІВ ВИКОРИСТАННЯ

Актуальність теми дослідження. *Варіанти використання є одним з важливих способів подання функціональних вимог до інформаційних систем, що проектуються. Їх опис – досить складний і трудомісткий процес. Тому дослідження, спрямовані на автоматизацію цього процесу, є актуальними.*

Постановка проблеми. *Для автоматизації опису варіантів використання необхідна розробка класифікації та моделювання пунктів сценаріїв. У цій роботі вирішується проблема коригування наявної класифікації пунктів варіантів використання.*

Аналіз останніх досліджень і публікацій. *Були розглянуті останні публікації з теми застосування варіантів використання. За основу прийнята класифікація, запропонована в роботі «Інформаційна технологія автоматизованого складання варіантів використання».*

Виділення недосліджених частин загальної проблеми. *Створення моделі для пункту «Завершення прецеденту».*

Постановка завдання. *Коригування класифікацій пунктів варіантів використання, створення і внесення змін у моделі пунктів сценаріїв.*

Виклад основного матеріалу. *Запропоновано удосконалену модель опису варіантів використання, що відображає як функціональні вимоги, так і концептуальні класи проектованого програмного продукту. Класифікація пунктів сценарію варіантів використання доповнено новим пунктом «Завершити варіант використання». Внесено зміни в раніше запропоновані моделі й створені нові моделі для пунктів сценарію «Створити», «Запросити значення», «Вирішення проблеми, які не вкладаються в запропоновану класифікацію». Створено відповідні алгоритми і виконана модернізація програмного продукту, що підтримує технологію автоматизованого складання варіантів використання.*

Висновки відповідно до статті*. Реалізація запропонованих моделей і алгоритмів дозволила скоротити час на опис варіантів використання до 10 %, а також створити моделі концептуальних класів для проектованого програмного продукту.*

Ключові слова*: варіанти використання; сценарії; моделі; прецеденти; концептуальні класи.*

Рис.: 3. Библ.: 11.

Кунгурцев Алексей Борисович – кандидат технических наук, профессор, Одесский национальный политехнический университет (просп. Шевченко, 1, г. Одесса, 65044, Украина).
Кунгурцев Олексій Борисович – кандидат технічних наук, професор, Одеський національний політехнічний університет (просп. Шевченка, 1, м. Одеса, 65044, Україна).
Kungurtsev Oleksii – PhD in Technical Sciences, Professor, Odessa National Polytechnic University (1 Shevchenko Av., 65044 Odessa, Ukraine).
E-mail: abkun@te.net.ua
ORCID: http://orcid.org/0000-0002-3207-7315

Новикова Наталия Алексеевна – старший преподаватель, Одесский национальный морской университет (ул. Мечникова, 34, г. Одесса, 65029, Украина).
Новікова Наталія Олексіївна – старший викладач, Одеський національний морський університет (вул. Мечнікова, 34, м. Одеса, 65029, Україна).
Novikova Nataliia – Senior Lecturer, Odessa national maritime University (34 Mechnikova Str., 65029 Odessa, Ukraine).
E-mail: nataliya.novikova.31@gmail.com
ORCID: http://orcid.org/0000-0002-6257-9703

Решетняк Мария Юрьевна – бакалавр, Одесский национальный политехнический университет (просп. Шевченко, 1, г. Одесса, 65044, Украина).
Решетняк Марія Юріївна – бакалавр, Одеський національний політехнічний університет (просп. Шевченка, 1, м. Одеса, 65044, Україна).
Reshetnyak Maria – bachelor, Odessa National Polytechnic University (1 Shevchenko Av., 65044 Odessa, Ukraine).
E-mail: 0957011250m@gmail.com

Черепинина Яна Витальевна – бакалавр, Одесский национальный политехнический университет (просп. Шевченко, 1, г. Одесса, 65044, Украина).
Черепініна Яна Віталіївна – бакалавр, Одеський національний політехнічний університет (просп. Шевченка, 1, м. Одеса, 65044, Україна).
Cherepinina Yana – bachelor, Odessa National Polytechnic University (1 Shevchenko Av., 65044 Odessa, Ukraine).
E-mail: cherepinina98@gmail.com

Уточнение классификации и моделей пунктов сценариев вариантов использования / А. Кунгурцев, Н. Новикова, М. Решетняк, Я. Черепинина // Технічні науки та технології. – 2018. – № 1 (11). – С. 79-88.