УДК 004.912

Ю.Н. Возовиков (канд. техн. наук), А.Б. Кунгурцев (канд. техн. наук, проф.), Н.А. Новикова
Одесский национальный политехнический университет г. Одесса
кафедра системного программного обеспечения
E-mail: vozovykov@gmail.com, abkun@te.net.ua, nan_21@mail.ru

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ АВТОМАТИЗИРОВАННОГО СОСТАВЛЕНИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Целью исследования является сокращение времени на составление прецедентов. Для этого на основании анализа реализуемости прецедента определены общие требования к его описанию. Прецеденты (use case, варианты поведения) повсеместно используются при создании информационных систем. От качества написания прецедента во многом зависит время и качество выполнения проекта. Вместе с тем, разрозненные и часто противоречивые правила составления прецедентов приводит к тому, что при описании прецедентов часто до-пускаются ошибки. Формализованное представление прецедентов на языке Unified Modeling Language никак не определяет содержание прецедентов. Таким образом, в области автоматизации описания прецедентов отсутствуют исследования и программные решения. Исследование действий, выполняемых прецедентами, позволило создать классификацию пунктов сценариев. Предложены модели для различных пунктов сценария, что позволило создать множество фрагментов текста и правил их объединения в предложения. Сформулированы условия разворачивания альтернативных сценариев. Определены правила образования структур данных, поддерживающих выполнение сценариев. Предложенная техно-логия за счет автоматизации выполнения многих операций позволяет в среднем на 80 процентов сократить время составления прецедента.
Ключевые слова: *прецеденты, процессы, ограничения, сценарии, система.*

Общая постановка проблемы. Прецеденты (use case, варианты поведения), как наиболее точный способ формулировки функциональных требований к программному продукту, повсеместно используются при создании информационных систем (ИС) [1, 2]. От качества написания прецедента во многом зависит время и качество выполнения проекта.

Многолетнее использование прецедентов позволило выработать ряд рекомендаций по их составлению [3, 4]. Введена классификация прецедентов [4]. Выработаны правила оформления преамбулы [5], описания основных и альтернативных сценариев [5, 6]. Имеются требования к формулировкам некоторых действий [4, 10]. Вместе с тем, разрозненные и часто противоречивые правила составления прецедентов, их разбросанность по различным источникам приводит к тому, что при описании прецедентов часто допускаются ошибки. Существует формализованное представление прецедентов на языке UML [7, 8, 9], однако оно позволяет записать диаграмму прецедентов, установить ассоциации между прецедентами и актерами, но никак не определяет содержание прецедентов. Таким образом, в области автоматизации описания прецедентов отсутствуют исследования и программные решения.

Другая проблема определяется тем, что разработка прецедента – это описание взаимодействия пользователя с проектируемой системой.

Системный аналитик обязательно представляет себе, какими программными средствами будет реализовываться каждый пункт сценария, то есть мысленно создаёт программные модули, способные реализовывать выполнение каждого пункта сценария. Такая предварительная проработка архитектуры программного продукта до настоящего времени не находи-

© Возовиков Ю.Н., Кунгурцев А.Б., Новикова Н.А., 2017

ла отражения в каких-либо документах, что приводило к повторному анализу прецедента на стадии создания диаграмм концептуальных классов.

Постановка задач исследования. Целью исследования является сокращение времени и уменьшение ошибок при описании прецедентов уровня пользователя, а также предварительное формирование модели концептуальных классов.

Для достижения поставленной цели сформулированы следующие задачи:

– определение основных требований к описанию прецедента;

– разработка классификации множества пунктов сценариев;

– составление модели для каждого типа пункта сценария;

– определение условий разворачивания альтернативных сценариев;

– определение создаваемых структур данных;

- разработка технологии автоматизированного составления прецедента.

Требования к описанию прецедента подразделяются на требования со стороны заказчика и разработчика.

1. Требования со стороны заказчика информационной системы ИС:

– должен описывать некоторую последовательность действий, переводящих проектируемую систему в новое состояние (ощутимый результат);

– должен использовать язык заказчика;

– должен быть похожим на действия заказчика в его предметной области.

2. Требования со стороны разработчика ИС:

– описание должно иметь два раздела: основной и дополнительный сценарий (расширения);

– в каждом пункте система должна выполнять некоторые действия;

– для каждого действия должно быть указано действующее лицо;

– если вводимая информация требует проверки, то в основном сценарии система дает положительный результат (подтверждает), а в дополнительном – каждый случай отрицательного результата представлен отдельно своим сценарием;

– максимальное количество вводимых и проверяемых данных – 2;

– если система в результате поиска или расчета выводит некоторые данные, которые должен оценить инициатор прецедента, то в основном сценарии он дает положительный ответ (например, «клиент согласен»), а в дополнительном сценарии рассматриваются действия, связанные с его отрицательным ответом;

– если система выполняет некоторый расчет или проверку вводимых данных, то желательно сообщить, какие средства будут при этом задействованы;

Действующими и заинтересованными лицами в прецеденте могут быть люди и системы:

– пользователь системы, который будет с ней непосредственно общаться для выполнения своих профессиональных обязанностей;

– клиент – заказчик некоторой услуги (может оказаться в роли пользователя, например, в режиме онлайн-общения с системой);

– другая система;

– разрабатываемая система.

Классификация пунктов сценария. На основании требований, предъявляемых к описанию прецедента, а также анализа множества сценариев в различных предметных областях с точки зрения характера взаимодействия пользователя и программного продукта предложено следующее множество типов пунктов сценариев.

– **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.

Действие пользователя, которое не укладывается в предложенную классификацию. Очевидно, возможна ситуация, когда пользователь не обнаружил нужной ему операции среди предложенных действий с системой. В этом случае ему предложено создать свободно конструируемый пункт.

Модели пунктов сценария. Модель каждого пункта будет представлена последовательностью элементов. Элементами могут быть заранее заготовленные фрагменты текста tpi, фрагменты текста, формируемые в процессе составления прецедента tuj, либо некоторые поименованные сущности $Client$, $Actor$, $Condition$ и др., которые играют роль синтаксических переменных и также должны получить значения в процессе составления прецедента. Некоторые элементы или группы элементов могут быть объединены метасимволами «[]», что указывает на необязательное присутствие этих элементов в описании пункта прецедента. Метасимвол «+» обозначает конкатенацию строк. Метасимвол «/» обозначает использование одного из двух элементов, разделенных этим символом.

Модель пункта «Создать». Представим пункт в виде кортежа:

$$Create =< N, [Client, tp1, Actor, tp2, tu1], Actor, tp3, Object, [tp4, Conditions,][tp5, ParamList,][tp6] >,$$

где N – номер пункта прецедента;

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

$tp1 – .tp6$ – заранее заготовленные фрагменты текста:

$tp1$ = ” обращается к”;

$tp2$ =” по поводу”;

$tp3$ =” создает в системе”;

$tp4$ =” при условии”;

$tp5$ =” с параметрами”;

$tp6$ =” система подтверждает”;

$tu1$ – текст, формируемый пользователем, например, «заказа 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;

– $Condition =< textCondition, addrCheck >$ – необязательный элемент, где $textCondition$ – формулировка условия, при котором может быть создан объект, а $addrCheck$ – объект, выполняющий указанную проверку;

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

каждый атрибут представляет собой кортеж $< name, Analyeer >$, содержащий название параметра и название объекта, который может проверить корректность значения параметра; Если атрибут не требует проверки, то *Analyeer* представляет собой пустую строку; все атрибуты являются фиксированными записями.

Алгоритм реализации пункта «Создать».

1. Номер пункта формируется автоматически.
2. Можно вводить или не вводить название *Client* .
3. Ввести название *Actor* .
4. Если *Client* !="", то создается фрагмент текста: *Client* +" обращается к" + *Actor* +" по поводу". Вести текст *tu*1, поясняющий причину обращения клиента.
5. Ввести название *Object* . Создается фрагмент текста *Actor* +" создает в системе" + *Object* .
6. Можно включать или не включать в текст *Condition*.
7. Если включено *Condition*, то заполняется таблица:

textCondition	*addrCheck*

и формируется фрагмент текста "при условии" + *Condition*.

8. Можно вводить или не вводить список параметров *ParamList* .
9. Если требуется включить параметры, то заполняется таблица параметров:

Parametr	*Analyseer*
…	…

Если некоторый параметр требует анализа, то для него указывается объект, выполняющий этот анализ. Формируется фрагмент текста " с параметрами" + *Parametr* +… *Parametr* .

10. Если в пункт включено *Condition* и в колонке *Analyseer* появились записи, то добавляется фрагмент текста" Система подтверждает.".

Соединение сформированных фрагментов текста представляет собой пункт прецедента.

Модель пункта «Ввести данные»:

$$InputData =< N,[Client, tp1, tu1,]Actor, tp2, DataList[, tp3, tp4] >,$$

где *tp*1 =" предоставляет данные о";

*tp*2 =" вводит в систему";

*tp*3 =" Система проверяет данные";

*tp*4 = "Система подтверждает корректность данных";

*tu*1 – 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 (является фиксируемой записью);

addrCheck – ориентировочный адрес объекта, осуществляющего проверку данного (служебная информация). Если адрес отсутствует, то значение данного не проверяется, Если в списке будет указано более двух проверяемых данных, то аналитику выводится рекомен-

дация о разбиении данного пункта сценария на два пункта. Если в списке нет проверяемых данных, то аналитику выводится рекомендация об объединении данного пункта с соседним пунктом;

addrKeep – ориентировочный адрес объекта, сохраняющего значение данного (служебная информация).

Реализация пункта «Ввести данные». В результате выполнения пункта заполняется таблица:

dataName	*addrCheck*	*addrKeep*

Если некоторое данное требует анализа, то для него указывается объект, выполняющий этот анализ (*addrCheck*). В пункт добавляется фрагмент текста *dataName*, *dataName*,…., *dataName*, где приведены все названия данных из таблицы. Если существует хотя бы одно данное, которое требует проверки, то формируется фрагмент текста "Система проверяет данные"+ *dataName*,…, *dataName* . "Система подтверждает корректность данных".

Модель пункта «Запросить значение»:

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

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

*tp*1 ="желает получить", фраза, формируемая пользователем, например, «стоимость услуги»;

*tp*2 ="вводит в систему запрос на получение";

Data =< *dataName*, *addrKeep* > – запрашиваемое у системы данное;

addrKeep – название объекта содержащего данное;

*tp*3 ="при условии";

Condition =< *textCond*, *addrKeep* > – условие получения данного, например, *textCond* может быть, например, «за прошлый месяц»;

addrKeep – название объекта, способного проверить и выполнить условие;

*tp*4 ="Система подтверждает корректность условия"- необязательный элемент;

*tp*5 ="Система выводит + *nameData* + *Client* / *Actor* + согласен";

Реализация пункта «Запросить значение». В результате выполнения пункта заполняется таблица:

dataName	*addrKeep*

В пункт добавляется фрагмент текста *dataName* . Если *dataName* должно быть получено при выполнении некоторого условия, то заполняется таблица:

textCond	*addrKeep*

В пункт добавляется текст "при условии + *textCond* +Система подтверждает корректность условия." Если *Client* не включался в текст пункта, то добавляется фрагмент текста "Система выводит + *nameData* + *Actor* + согласен.". В противном случае – " Система выводит + *nameData* + *Client* + согласен.".

Модель пункта «Запросить список»:

$$ListInquiry =< N,[Client,tp1,tu1,]Actor,tp2,List,[tp3,Condition,tp4,]tp5 >$$

где $tp1$=" желает получить список", фраза, формируемая пользователем, например, «предоставляемых услуг»;

$tp2$=" вводит в систему запрос на получение";

$List =< listName, addrKeep >$ – запрашиваемый у системы список;

$listName$ – название списка,

$addrKeep$ – место хранения (формирования) списка;

$tp3$=" 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;

$tp4$=" Система подтверждает корректность условия";

$tp5$=" Система находит и выводит $listName$";

Реализация пункта «Запросить список». В результате выполнения пункта заполняется таблица:

$listName$	$addrKeep$

В пункт добавляется фрагмент текста $listName$. Если формирование списка должно производиться с соблюдением некоторого условия, то заполняется таблица

$textCond$	$addrKeep$

В пункт добавляется текст " при условии + $textCond$ +Система подтверждает корректность условия." . Добавляется фрагмент текста "Система находит и выводит $listName$".

Модель пункта «Выбрать из списка»:

$$SelectElem =< N,[Client,tp1,]Actor,tp2,Elem,tp3,tp4 >,$$

где $tp1$=" указывает на нужный элемент из списка";

$tp2$=" выбирает из списка";

$Elem =< elemName, elemType >$; элементы могут быть двух типов (всегда одинаковые для всего списка): $elemType = Data / Control$; если тип элемента $Data$, то система должна предоставить $Client / Actor$ запрашиваемые данные; если тип элемента $Control$, то ему соответствует множество пар $\{< elemName_i, N_i >\}$, $i = \overline{1,k}$; N_i – номер пункта сценария, которому будет передано управление, k – количество элементов в списке);

$tp3$=" и вводит его в систему. ";

$tp4$=" Система предоставляет значение + $Data$ + $Client / Actor$ +согласен. ";

$tp5$=" Происходит переход к пункту + N_i + сценария. ";

Реализация пункта «Выбрать из списка». Действующему лицу предоставляется таблица:

$elemName_j$	$Data_j$	
$elemName_{j+1}$	$Control_{j+1}$	N_i

Если полю *elemName* записи из списка соответствует значение *Data* в поле $Data/Control$, то формируется строка "Система предоставляет значение $+Data+Client/Actor+$согласен". Если в поле $Data/Control$ записано *Control*, то создается фрагмент текста "Происходит переход к пункту + N_i + сценария. ";

Модель пункта «Ввести услугу (документ)»:

$$InputService =< N,[Client,tp1/tp2,tu1,]Actor,tp3,serviceName,tp4>,$$

где *Service* – наименование услуги (документа), определяющего дальнейшие действия; $Service =< serviceName, Nm >$, где *serviceName* – название сервиса (документа), *Nm* – номер пункта сценария, который соответствует указанному сервису;

$tp1 =$" желает работать с документом";

$tp2 =$" желает использовать";

$tu1$ – текст определяющий сервис (например, «осмотр ходовой части», «регулировка развала/схождения») или документ (например, «заявление на предоставление льготного тарифа»);

$tp3 =$" вводит в систему";

$tp4 =$" Система проверяет возможность выполнения услуги (документа) и исполнителя. Система подтверждает.".

Реализация пункта «Ввести услугу (документ)». Пользователю предоставляется таблица

$serviceName_1$	Nm_1	$addrKeep_1$
$serviceName_2$	Nm_2	$addrKeep_2$
…	…	…

Если в таблице имеется нужное название сервиса (документа), то его вводят в систему. Если нужное название в таблице отсутствует, то *Actor* заполняет новую строку таблицы. Создается фрагмент текста $serviceName +$" Система проверяет возможность выполнения услуги (документа) и исполнителя. Система подтверждает.".

Модель пункта «Повторение действий»:

$$CyclAction =< N, Client / Actor, tp1, Np >,$$

где $tp1 =$" желает повторить действия начиная с пункта";

Np – номер пункта основного сценария, расположенный выше пункта *N*.

Реализация пункта «Повторение действий»:

1. Ввести/не вводить название *Client*.

2. Если *Client* !="", то создается фрагмент текста: $Client +$ " желает повторить действия начиная с пункта "+ *Np*.

3. Если *Actor* !="", то создается фрагмент текста $Actor +$ желает повторить действия начиная с пункта" + *Np*.

Модель свободно конструируемого пункта:

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

где *tu1, tu2* – фразы, формируемые пользователем.

Алгоритм реализации свободно конструируемого пункта:

1. По желанию пользователя вводится фраза *Client* + *tu*1.

2. Пользователь вводит фразу *Actor* +" *tu*2.

3. Пользователю предлагается ответить на вопрос «Вводятся в систему данные?». При положительном ответе предлагается заполнить таблицу

dataName	*addrCheck*	*addrKeep*

4. Пользователю предлагается ответить на вопрос «Выводит система данные?». При положительном ответе предлагается заполнить таблицу

dataName	*addrKeep*

Если *dataName* должно быть получено при выполнении некоторого условия, то заполняется таблица:

textCond	*addrKeep*

Формирование альтернативных сценариев. Все пункты основного сценария могут предусматривать альтернативные сценарии. Рассмотрим алгоритм построения альтернативного сценария на примере пункта «Создать». Для других пунктов только укажем условия построения альтернативных сценариев.

Алгоритм построения альтернативного сценария для пункта «Создать»:

1. Если в пункте нет условия *Condition* , то выход.

2. Формируется условие альтернативного сценария: *N.l* . Условие *textCondition* не выполняется. Невозможно создать объект *Object* .

3. Если *ParamList* пустой, то выход.

4. Анализируется $ParamList =< data_1,...,data_i,...,data_n >$. Для каждого атрибута $data_i$, для которого в предусловии к прецеденту не найдена фраза «установлено значение $data_i$», либо он не фигурировал в $DataList = \{d_1, d_2,...,d_n\}$, расположенного ранее пункта «Ввести данные», создается альтернативный сценарий следующего формата:

N.l. Значение параметра $data_i$ не определено, невозможно создать объект *Object* .

N.l.1. Пользователь вводит новое значение данного $data_i$. Переход к пункту N, где N – номер (целое число без знака) пункта «Создать» в основном успешном сценарии (например, N =1);

N.l. – номер пункта с заключением системы, образуется процедурой конкатенации $concat(N, letter_j)\ j = 1,26$; (например, $N.l = 1.a$);

N.l.1. – номер первого пункта альтернативного сценария

Для пункта **«Ввести данные»** создается альтернативный сценарий, если значение некоторого данного $d_i =< name, addrCheck, addrKeep >$ из *DataList* , требующего проверки (указан *daddrCheck*), не соответствует требованиям системы.

Для пунктов **«Запросить»** и **«Запросить список»** формируются альтернативные сценарий, если в пунктах имеется условие *Condition* .

Для пункта **«Выбрать из списка»** создается альтернативный сценарий, если полю *elemName* записи из списка соответствует значение *Data* в поле *Data* / *Control* .

Для пункта **«Ввести услугу (документ)»** создается альтернативный сценарий, если *Service* отличается от реализованного в основном успешном сценарии, но предусмотрен в

системе. В противном случае происходит завершение работы прецедента.

Для пункта **«Повторение действий»** создается альтернативный сценарий, если *Client* / *Actor* не желает повторения действий.

Представление элементов разрабатываемой системы. При реализации каждого пункта сценария предполагалось использование ряда элементов разрабатываемой системы, которые названы объектами. В большинстве случаев, указанные элементы в дальнейшем, действительно, будут представлять собой объекты некоторых классов. Однако процесс составления прецедента не предусматривает детальную проработку архитектуры проектируемого программного продукта. На этом этапе нужно только зафиксировать, что должна делать система, что и где сохранять. Для этого предлагается следующая структура объекта, которая приведена в табл. 1.

Таблица 1 – Структура объекта

Название объекта		
Хранение		
Данные	Услуги	Документы
…	…	…
Проверка		
Данные	Услуги	Документы
…	…	…

Предложенная структура объектов позволяет учитывать все действия, выполняемые системой, исключить дублирование данных и функций, а также является основой для построения диаграмм концептуальных классов. Кроме этого, названия объектов, данные, услуги и документы подлежат включению в словарь предметной области [11].

В соответствии с предложенными моделями и правилами их реализации был создан программный продукт UseCaseEditor. На рис. 1 представлен пример формирования пункта прецедента типа «Создать».

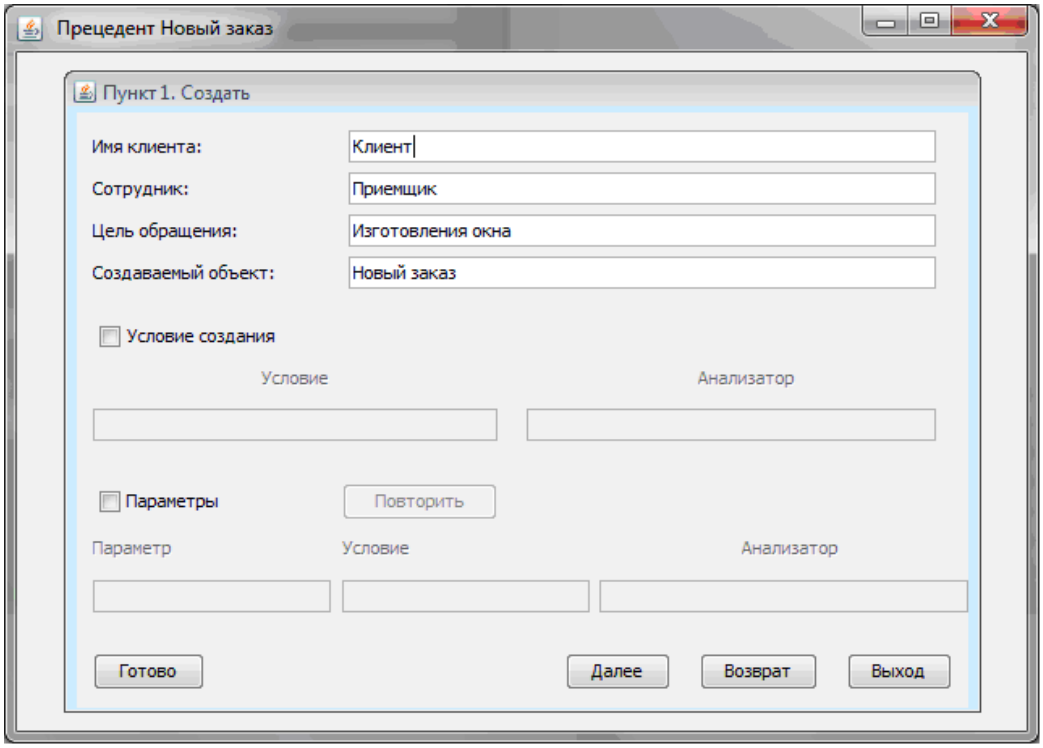

Рисунок 1 – Набор пункта «Создать»

Испытания программного продукта проводились с участием двух групп из 10 студентов Одесского национального политехнического университета в процессе изучения дисциплины «Анализ требований к программному обеспечению». Группа студентов, которая использовала UseCaseEditor допустила 4 ошибки, группа студентов, которая не использовала UseCaseEditor допустила 22 ошибки и потратила на составление прецедента на 80% больше времени (с учетом времени на исправление ошибок).

Технология составления прецедента представлена на рис. 2. Системный аналитик работает с мастерами формирования фрагментов прецедента. Мастера реализуют предложенные в работе модели. Результаты работы представлены в прямоугольниках с толстыми линиями.

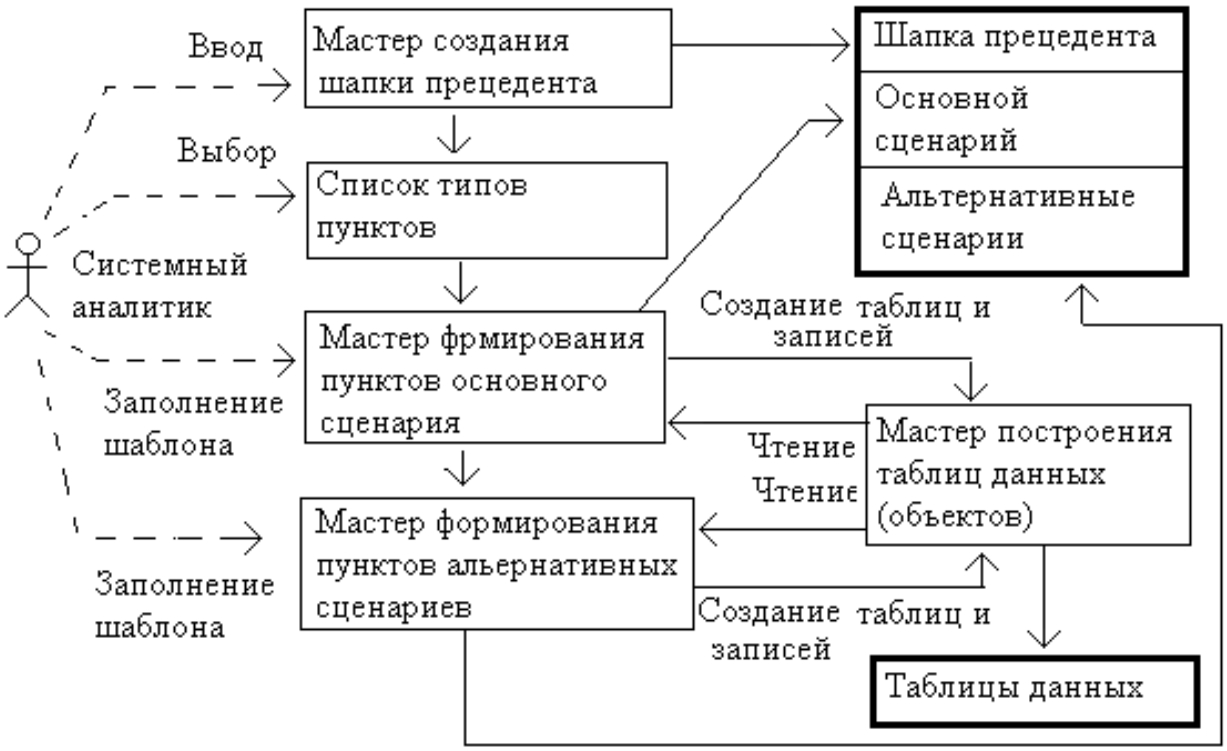

Рисунок 2 – Технология создания прецедента

Выводы.

1. Предложена классификация возможных пунктов сценариев с точки зрения видов и результатов взаимодействия пользователя с будущей системой. Это дало возможность создать модели пунктов и алгоритмы их формирования, позволяющие из заранее заготовленных фрагментов текста в автоматизированном режиме формировать тексты пунктов сценариев.

2. Разработаны формальные правила выявления ситуаций, требующих создания альтернативных сценариев, что позволяет избежать множества ошибок, связанных с неполным анализом пунктов сценариев.

3. Предложены правила и методика формирования объектов, содержащих данные, необходимые для выполнения сценариев, что дает возможность сократить время на этапе проектирования программного продукта. Предложена технология автоматизированного создания описаний прецедента.

Разработан программный продукт, реализующий предложенные модели и технологию. Проведенные эксперименты показали эффективность предложенной технологии, прежде всего, с точки зрения сокращения времени на составление прецедента (до 80 %), а также в плане удобства работы пользователя.

Список использованной литературы

1. Гради Буч. Язык UML. Руководство пользователя / Гради Буч, Джеймс Рамбо, Ивар Джекобсон. – М.: Издательство ДМК Пресс, 2007 – 496 с.
2. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования / Крэг Ларман. – М.: Издательство Вильямс, 2008 – 736 с.
3. Леффингуэлл, Д. Принципы работы с требованиями. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг. – М.: Издательский дом «Вильямс», 2002 – 450 с.
4. Алистер Коберн. Современные методы описания функциональных требований к системам. Москва: Издательство Лори, 2002 – 266 с.
5. Материалы академической программы корпорации IBM: Essentials of visual modeling, Fundamentals of Rational Rose [Электронный ресурс] // Rational University. – Режим доступа: http://www.ibm.com/ru/software/info/students.
6. Создание проекта. Анализ прецедентов. Реализация прецедентов. Уточненное описание прецедента [Электронный ресурс] – Режим доступа: http://vunivere.ru/work72704.
7. Леонов, И.В. Прецеденты (use cases) и их связи. Исполнители (actors) и их связи. [Электронный ресурс]. – Режим доступа: http://www.interface.ru/fset.asp?Url=/rational/vmr2.htm
8. Лененков, А.В. Объектно-ориентированный анализ и проектирование с использованием UML IBM и Rational Rose. — М: Издательство Бином, 2006 – 320 с.
9. Дж. Рамбо, Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – Санкт-Петербург: Издательство Питер, 2007 – 544 с.
10. Прецеденты в спецификации программ [Электронный ресурс] – Режим доступа: http://club.shelek.ru/viewart.php?id=232
11. Кунгурцев, А.Б. Метод автоматизированного построения толкового словаря предметной области / А.Б. Кунгурцев, Я.В. Поточняк, Д.Ф. Силяев // Х.: Технологический аудит и резервы производства – № 2/2(22), 2015 – С. 58 – 63.

References

1. Gradi Buch (2007), The Unified Modeling Language: Users Guide, DMK Press, Moscow, Russia.
2. Krjeg Larman (2008), Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Vil'jams, Moscow, Russia.
3. Leffingujell, D. (2002), Managing Software Requirements. A Unified Approach, Vil'jams, Moscow, Russia.
4. Alister Kobern (2002), Writing Effective Use Cases. Lori, Moscow, Russia
5. Materialy akademicheskoj programmy korporacii IBM: Essentials of visual modeling, Fundamentals of Rational Rose, available at: http://www.ibm.com/ru/software/info/students.
6. Sozdanie proekta. Analiz precedentov. Realizacija precedentov. Utochnennoe opisanie precedenta [Creating a project. Analysis of precedents. Implementation of precedents. Refined description of the use case], available at: http://vunivere.ru/work72704.
7. Leonov, I.V. Precedenty (use cases) i ih svjazi. Ispolniteli (actors) i ih svjazi [Use cases and their connections. Performers (actors) and their connections], available at: http://www.interface.ru/fset.asp?Url=/rational/vmr2.htm.
8. Lenonenkov, A.V. (2006), Obektno-orientirovannyj analiz i proektirovanie s ispol'zova-niem UML IBM i Rational Rose [Object-oriented analysis and design using UML IBM and Rational Rose], Binom, Moscow, Russia.
9. Dzh. Rambo and Blaha M. (2007), UML 2.0. Object-Oriented Modeling and Design with UML. Piter, Sankt-Peterburg, Russia.
10. Precedenty v specifikacii programm [Precedents in the program specification], available at: http://club.shelek.ru/viewart.php?id=232.

11. Kungurcev, A.B., Potochnjak, Ja.V., and Siljaev, D.A.(2015), Metod avtomatizirovannogo postroenija tolkovogo slovarja predmetnoj oblasti, Tehnologicheskij audit i rezervy proizvodstva [The method of automated construction of an explanatory dictionary of the subject domain], № 2/2(22), Kharkov, Ukrain.

Поступила в редакцию:
17.04.2017

Рецензент:
д-р техн. наук, доц. Вовна А.В.

Ю.М. Возовиков, О.Б. Кунгурцев, Н.О. Новікова
ОНПУ «Одеський національний політехнічний університет»

Інформаційна технологія автоматизованого складання варіантів використання.

Метою дослідження є скорочення часу на складання прецедентів. Для цього на підставі аналізу реалізації прецеденту визначені загальні вимоги до його описання. Прецеденти (use case, варіанти поведінки) повсюдно використовуються під час створення інформаційних систем. Від якості написання прецеденту залежить час і якість виконання проекту. Разом з тим, розрізнені та зазвичай суперечливі правила складання прецедентів призводить до того, що під час описання прецедентів зазвичай допускаються помилки. Формалізоване уявлення прецедентів на мові Unified Modeling Language ніяк не визначає зміст прецедентів. Таким чином у області автоматизації описання прецедентів відсутні дослідження та програмні рішення. Дослідження дій, які виконуються прецедентами, дозволило створити класифікацію пунктів сценаріїв. Запропоновано моделі для різних пунктів сценарію, що дозволило створити безліч фрагментів тексту та правил їх об'єднання в пропозиції. Сформульовані умови розгортання альтернативних сценаріїв. Визначено правила утворення структур даних, що підтримують виконання сценаріїв. Запропонована технологія завдяки автоматизації виконання багатьох операцій дозволяє в середньому на 80 % скоротити час складання прецеденту.

***Ключові слова**: прецеденти, процеси, обмеження, сценарії, система.*

Y. Vozovykov, O. Kungurtsev, N. Novikova
Odessa National Polytechnic University

Information technology using automated options.

Precedents (use case, variants of behavior) are widely used in the creation of information systems. The quality and quality of the project depends on the quality of the precedent writing. At the same time, disparate and often contradictory rules of precedent setting lead to the fact that when describing precedents mistakes are often made. The formalized representation of use cases in the language of Unified Modeling Language does not determine the content of precedents. Thus, in the field of the automation of the description of use cases, there are no studies and software solutions. The aim of the study is to reduce the time and error in describing user-level precedents, as well as the preliminary formation of a model of conceptual classes. To achieve the goal, the following tasks are formulated: 1) definition of the requirements for the description of the precedent; 2) development of classification of a set of scenario items; 3) compiling a model for each type of scenario item; 4) determining the conditions for the deployment of alternative scenarios; 5) definition of the created data structures; 6) development of the technology of automated precedent writing. The requirements for describing the precedent on the part of the customer and the developer of the information system are defined. A classification of possible points of scenarios is suggested in terms of the types and results of user interaction with the future system: 1. Create, 2. Enter data, 3. Request value, 4. Request list, 5. Select from list, 6. Enter service (document), 7. Repetition of actions, 8. Arbitrary action. This made it possible to create point models and algorithms for their formation, allowing from pre-prepared fragments of text in an automated mode

to form the texts of script points. The technology of automated precedent creation is developed. In accordance with the proposed models and rules for their implementation, the software product UseCaseEditor was implemented, which implements the proposed models, algorithms and technology. The experiments carried out showed, in practice, a significant reduction in time (up to 80 percent) when applying the proposed technology in comparison with the traditional way of writing precedents.

Keywords: *use case, processes, constraints, scenarios, system.*

Возовиков Юрий Николаевич, Украина, окончил Одесский национальный политехнический университет, к.т.н., старший преподаватель кафедры системного программного обеспечения. Институт компьютерных систем «Одесский национальный политехнический университет» (пр. Шевченко, 1, Одесса, 65044, Украина).
Основное направление научной деятельности – разработка методов, и средств повышения производительности информационных систем.

Кунгурцев Алексей Борисович, Украина, окончил Одесский национальный политехнический институт, к.т.н., профессор, профессор кафедра системного программного обеспечения. Институт компьютерных систем «Одесский национальный политехнический университет» (пр. Шевченко, 1, Одесса, 65044, Украина).
Основное направление научной деятельности – разработка методов, и средств повышения производительности информационных систем, создание средств общения с автоматизированными системами на естественном языке.

Новикова Наталия Алексеевна, Украина, окончила Одесский государственный университет, старший преподаватель кафедры техническая кибернетика им. проф. Р.В. Меркта. Одесский национальный морской университет (ул. Мечникова, 34, Одесса, 65029, Украина).
Основное направление научной деятельности – системы автоматизации обучения.