УДК 519.766:004.415.5

А.Б. ГОДЛЕВСКИЙ, Н.М. МИЩЕНКО, М.К. МОРОХОВЕЦ, О.Д. ФЕЛИЖАНКО, Н.Н. ЩЁГОЛЕВА

МЕТОДИКА РАЗРАБОТКИ И СРЕДСТВА ФОРМАЛИЗАЦИИ ФУНКЦИОНАЛЬНЫХ СПЕЦИФИКАЦИЙ СИСТЕМ И УСТРОЙСТВ[1]

Аннотация. Предложен подход к построению функциональных спецификаций и к автоматизации перехода от спецификаций на естественных языках к формальным моделям в формате, удобном для последующих верификации этих моделей и валидации построенных по ним программных кодов. Преобразование текстов в модели осуществляется с помощью интерактивной системы OEC. Описанный подход иллюстрируется развернутым примером.

Ключевые слова: функциональная спецификация, поведенческая модель, формальная спецификация систем и устройств, языковой процессор.

ВВЕДЕНИЕ

Одним из важнейших этапов разработки систем (устройств) является моделирование как их отдельных компонентов, так и систем в целом. При этом проверяется соответствие между поведением модели и поведением, которое являлось желаемым при формировании задания на разработку системы. На формальных моделях можно проводить *верификацию* систем, доказывая, например, что определенные критические состояния в модели недостижимы из ее стартовых состояний. Модели можно также использовать для построения тестовых наборов, отвечающих определенным критериям и позволяющих затем проводить *валидацию* изготовляемого устройства.

В процессе проектирования применяются различные виды моделей, которые отображают разные уровни понимания предназначения разрабатываемого устройства: от того, *что* оно должно делать, до того, *как* оно это делает, из каких частей состоит и как они взаимодействуют. Вначале задается верхний уровень (High Level Design), где отражаются общие пожелания заказчика к задачам, которые должно решать проектируемое устройство. Далее на уровне архитектурных решений определяется структура компонентов, из которых будет состоять устройство, и интерфейс, регламентирующий их взаимодействие. Наконец, функционирование отдельных компонентов и их взаимодействие определяется поведенческими моделями. Последние могут быть описанными на специальных языках, например, на графическом языке UCM [1], где в виде процессов (путей в специальных графах) представлены различные сценарии функционирования проектируемой системы. Классический способ описания поведенческой модели устройства — это представление ее в виде транзиционной системы [2], которая задается множеством состояний, включая выделенные начальные и критические состояния, и отношением переходов на множестве этих состояний.

В инженерной практике поведение проектируемой системы может быть задано в виде *функциональных спецификаций* (ФС) — текстовых описаний, регламентирующих переходы в системе. Содержательно ФС состоит из двух частей, одна из которых описывает некоторое состояние системы, а другая — ее возможную реакцию в этом состоянии. По сути, ФС можно рассматривать как правило перехода транзиционной системы, которое представлено на естественном (не-

[1]Работа выполнена в рамках проекта 5228х при поддержке Национальной академии наук Украины и Украинского научно-технологического центра (УНТЦ).

формализованном) языке. Одна из типичных форм ФС имеет вид

> Если «некоторые параметры системы находятся в таких-то диапазонах» [и «произошло некое событие»], то «система должна таким-то образом изменить значение части своих параметров».

Отметим, что выражение «произошло некое событие» означает, что в систему (или ее компонент) проступило сообщение либо извне (от другого компонента), либо от одного из таймеров или изменилось значение одного из параметров, разделяемых с другими компонентами. Такой способ задания спецификаций характерен для систем, управляемых событиями (event driven model), где явно представлен процесс активации компонентов. Однако часть условия о некотором произошедшем событии может не указываться, и тогда система функционирует как обычные последовательные программы, где очередная команда изменяет состояние памяти, а следующая выполняет действия, уже адаптируясь (например, выбирая альтернативу) к этому новому состоянию информационной среды.

Данная работа выполнялась в рамках проекта 5228х «Когнитивная архитектура для понимания программных систем», где ставилась общая задача верификации систем, основанная на определенном классе формальных моделей, которые, в свою очередь, были построены по спецификациям, представленным на *естественных языках* (ЕЯ). Из этого комплекса задач — формализации текстов и верификации построенных по ним моделей, в настоящей статье рассматривается первая. При этом внимание уделяется не построению конкретной формализации для определенной специфицируемой системы, а тому, *как* эту формализацию следует проводить и *как* ее можно автоматизировать. Разработанная методика автоматизации демонстрируется далее на примере устройства кофе-автомата (КА) и состоит в описании последовательных трансформаций от начальных текстовых спецификаций к представлению переходов в виде, воспринимаемом компонентом, осуществляющим верификацию системы.

В описании методики построения функциональных спецификаций сделан акцент на формализации правил перехода в системе (в виде, близком к так называемым базовым протоколам — важному понятию в инсерционном моделировании [3]) и не рассматриваются такие этапы, как выделение множеств агентов и атрибутов системы, а также определения их типов.

Существует ряд работ, посвященных проблеме осознания смысла ЕЯ-текстов с помощью компьютеров, которые основаны на использовании понятия онтологии [4]. В этих работах разрабатываются концепты многих базовых понятий, например, связанных с пространством, временем, которые затем интерпретируются словами и фразами на тех или иных ЕЯ. Онтологический подход к анализу ЕЯ-текстов является фундаментальным, однако полномасштабно применяется и реализуется в больших проектах.

В то же время имеются работы по решению частных задач формализации текстовых спецификаций в целях дальнейшей их обработки формальными методами. Анализируя эти работы, можно выделить такие подходы к данной проблеме:

— применение статистических методов для анализа совокупностей спецификаций большого объема в целях выявления их смысловой близости, при этом спецификации записаны на ЕЯ без ограничений [5];

— использование так называемого ограниченного ЕЯ для записи спецификаций [6], в этом случае предлагается ограничиться простыми языковыми конструкциями без сложноподчиненных предложений;

— конструирование простых формальных языков, близких к ЕЯ, ориентированных на определенную область применения, выражения которых легко преобразуются в формулы логики [7].

Использование естественного языка для представления функциональных спецификаций и работы с ними имеет наряду с очевидными преимуществами определенные трудности, при этом наиболее существенными являются следующие:

— текст на ЕЯ по различным причинам может неоднозначно толковаться;

— для обработки текста на ЕЯ с помощью формальных методов его вначале необходимо преобразовать в текст на формальном языке.

Представление ФС на формальном языке позволяет применять к ним формальные методы анализа. Однако имеются трудности при работе с текстом на формальном языке; требуется высокий уровень знаний и подготовки для:

— представления свойств и функциональных спецификаций;

— понимания результатов работы средств анализа.

Проектировщику систем и устройств, а также разработчику функциональных спецификаций наиболее удобно и просто было бы пользоваться привычным ЕЯ и опираться на формальные методы, не погружаясь в их технические подробности. Проблема создания удобных средств, сочетающих преимущества использования ЕЯ для представления ФС и формальных средств их анализа, все еще актуальна. Чтобы избежать некоторых трудностей при применении ЕЯ, прибегают, как отмечалось ранее, к сужению языковых средств (к использованию ограниченного ЕЯ).

Приведенный далее подход к разработке формальных моделей ФС состоит в том, чтобы, не ограничивая разработчика ФС жесткими рамками формального языка, предоставить ему:

— методику (процедуру) формирования ФС на ЕЯ;

— средства программной поддержки для преобразования ФС на ЕЯ в ФС на формальном языке.

Материал статьи организован следующим образом. В разд. 1 представлена методика построения ФС на ЕЯ. Разд. 2 посвящен описанию интерактивной языковой системы обработки ЕЯ-спецификаций (ОЕС) для построения ФС на формальном языке по ФС на ЕЯ. Платформой для создания ОЕС послужила Расширяющаяся Система Программирования ТЕРЕМ (РСП ТЕРЕМ) [8] — инструмент для разработки языковых процессоров. В разд. 3 применение предложенной методики и использование языковой системы проиллюстрированы на примере разработки формальной спецификации кофе-автомата.

1. МЕТОДИКА ПОСТРОЕНИЯ ФУНКЦИОНАЛЬНЫХ СПЕЦИФИКАЦИЙ НА ЕЯ

Создание формальной (формализованной) спецификации предлагается осуществлять в несколько этапов:

— построение моделей разрабатываемого устройства или системы (далее *объекта*) на естественном языке (ЕЯ-моделей);

— формирование словаря рабочей лексики;

— построение модели поведения *объекта* на ЕЯ, ограниченном использованием рабочей лексики;

— преобразование полученной модели поведения в модель на формальном языке.

1.1. Построение ЕЯ-моделей *объекта*. Предполагается, что разработчик ФС начинает работу, зная, какой *объект* (класс объектов) разрабатывается и в каких целях, а также что́ известно о характеристиках данного *объекта* (класса объектов). Задача разработчика — создать формальную спецификацию *объекта*, т.е. такое его формальное описание, которое можно подвергнуть проверке формальными методами, чтобы выяснить, будет ли иметь *объект*, соответствующий этому описанию, некоторые требуемые свойства, а также проверить, будет ли иметь этот *объект* нежелательные свойства (из некоторого списка).

Разработка спецификации начинается с построения ряда моделей *объекта*: функциональной, структурной, информационной. Эти модели могут быть построены пошаговой детализацией.

Функциональная модель детализирует описание функции *объекта*. Для выполнения этой функции могут понадобиться вспомогательные средства. Их состав и функциональные возможности определяются при построении функциональной модели. Последняя также задает способ связи *объекта* со средой его функционирования, кроме того, она может содержать функциональные ограничения, т.е. перечень действий, которые *объект* не выполняет.

Структурная модель определяет декомпозицию *объекта* на взаимодействующие компоненты (называемые также агентами). В общем случае такая декомпозиция является иерархической, поскольку некоторые компоненты верхнего уровня могут быть представлены как композиция компонентов более низкого уровня. (Вопрос о степени подробности описания *объекта* выходит за рамки данной статьи и далее не рассматривается.)

Информационная модель описывает информационные потоки как между *объектом* и средой его функционирования, между отдельными его компонентами, так и внутри этих компонентов. Передаваемая информация может представляться в виде сообщений тех или иных форматов, и в этом случае основной цикл ее обработки состоит в том, что каждый агент воспринимает сообщение, обрабатывает его и, возможно, формирует новое сообщение как реакцию на принятое. На этапе разработки этой модели определяются структуры данных (так называемые атрибуты *объекта*), с помощью которых осуществляются такие информационные связи. Построение информационной модели *объекта* — важный шаг на пути создания формальной спецификации *объекта*.

1.2. Формирование словаря рабочей лексики. Введение атрибутов (при построении информационной модели *объекта*) сопровождается пояснением их смысла, которое учитывается при создании модели поведения *объекта*. Пояснение смысла атрибута имеет следующий вид:

<имя атрибута>=<значение атрибута> **означает, что** <фраза-пояснение>

или

<значение атрибута перечислимого типа> **означает, что** <фраза-пояснение >

или

<имя атрибута> — <фраза-пояснение>.

Фразы-пояснения далее будем называть *клише*, а их совокупность рассматривать как рабочий лексикон.

Пусть, например, при разработке информационной модели некоторого проектируемого устройства вводится атрибут булевого или перечислимого типа с именем F, который характеризует готовность устройства к работе. Назначение и смысл этого атрибута поясняется следующим образом:

$F = 1$ означает, что устройство включено, исправно и готово к работе;

$F = 0$ означает, что устройство вышло из строя.

В результате создаются такие клише: «устройство включено, исправно и готово к работе» и «устройство вышло из строя». В разд. 3 данной работы образование клише и их роль в построении формальных ФС проиллюстрированы более подробно.

1.3. Построение модели поведения *объекта* на ЕЯ, ограниченном использованием рабочей лексики. На основе построенных ЕЯ-моделей *объекта* записываем требования к его функционированию. При этом различаем требова-

ния глобального и локального характера. Под требованием глобального характера понимаем свойство, которое не должно нарушаться на протяжении всего функционирования. Под требованием локального характера понимаем так называемое поведенческое свойство (правило перехода из состояния в состояние). Такие требования описываются в виде ФС, построенных с помощью фраз-клише из рабочего лексикона. Совокупность ФС составляет *модель поведения объекта*.

1.4. Формализация полученной модели поведения. Этот шаг осуществляется с помощью интерактивной языковой системы ОЕС.

2. ЯЗЫКОВАЯ СИСТЕМА ОЕС

Основная функция языковой системы ОЕС — преобразование ФС на ЕЯ в ФС на выходном (полуформальном) языке.

Входной текст для ОЕС представляет собой совокупность ФС на ЕЯ (модель поведения *объекта*) и является совокупностью требований (к *объекту*), оформленных в виде отдельных абзацев. Выходной текст — последовательность преобразованных требований, каждое из которых состоит из одного или нескольких предложений, соответствующих следующему простому синтаксису:

```
Phrase ::= C ----> A.
C ::= Cond | Cond & C
A ::= EA | EA; A
```

Содержательно такое предложение описывает возможную реакцию A специфицируемой системы в состоянии, удовлетворяющем условию C. Здесь Cond — элементарная формула (логики первого порядка), причем, если предикатным символом является «=» (равенство), используется инфиксная форма записи; EA — присваивание, или элементарная формула, или ЕЯ-фраза (причины появления ЕЯ-фраз в выходном тексте будут описаны ниже).

Языковая система ОЕС представляет собой:

— языковой процессор, построенный на основе РСП ТЕРЕМ;

— совокупность грамматик;

— совокупность словарей ключевых слов.

Под языковым процессором понимается программа преобразования текста на одном языке (входном), в текст на том же или другом языке (выходном). Настройка языкового процессора на входной язык осуществляется заданием входных параметров, настройка его на выходной язык осуществляется с помощью задания так называемых семантических действий, которые связываются с конструкциями входного языка. С помощью семантических действий также задается форма представления выходного текста.

2.1. Краткие сведения о РСП ТЕРЕМ. Данная система предназначена для разработки гибких языковых процессоров. Она успешно применялась для обработки языков программирования, формализованных подмножеств ЕЯ делового общения [9, 10]. Система реализована в языке Си.

В основе применения РСП ТЕРЕМ лежит метаязык LD для описания спецификаций языковых процессоров, основанных на определении синтаксиса входных языков в виде контекстно-свободных грамматик, описываемых языком БНФ (Бэкусово-Науровских Форм), а также связей синтаксических элементов с семантическими действиями (программами) по обработке входных текстов.

В РСП ТЕРЕМ используется алгоритмический подход к определению семантики на основе аппарата наследующих и синтезирующих действий, реализующих алгоритм перевода фраз, выводимых из нетерминалов грамматики. Кроме семантических действий, множество семантических объектов включает также шаблоны выходного кода.

Связь семантических действий с соответствующими синтаксическими элементами осуществляется с помощью специальных обозначений семантических объектов (действий и их параметров), располагаемых непосредственно после соответствующего элемента в БНФ-тексте входной грамматики. С одним и тем же вхождением синтаксического элемента можно связывать несколько обозначений семантических действий.

Семантические действия делятся на два класса: наследующие действия по вычислению наследующих атрибутов и синтезирующие действия по вычислению синтезируемых атрибутов.

Наследующие действия отличаются от синтезирующих временем их выполнения: первые осуществляются при движении по дереву сверху вниз, а вторые — при движении снизу вверх. В листьях дерева выполняются все действия в том порядке, в котором они нагружены.

Каждое семантическое действие вырабатывает результаты, используя данные, которыми могут быть:

— конструкции, обеспечиваемые синтаксической подсистемой системы ТЕРЕМ (дерево анализа, входная строка, синтаксические таблицы);

— результаты ранее выполненных действий;

— тексты из шаблонов.

Принцип функционирования системы ТЕРЕМ основан на взаимодействии синтаксической и семантической подсистем через промежуточную форму входной программы, а именно дерево анализа, связывающее фразы входного текста с обрабатывающими их семантическими процедурами, обозначениями которых декорированы синтаксические элементы, соответствующие входным фразам.

Система ТЕРЕМ функционирует в виде двух автономно исполняемых модулей: Конструктора и Анализатора. Она включает открытый набор наиболее часто используемых универсальных семантических модулей для формирования семантических подсистем.

Обработка входного текста осуществляется по лексемам, которые считываются из входного текста специальной программой Сканер.

Итак, для реализации языкового процессора с помощью системы ТЕРЕМ необходимо выполнение следующих действий:

— описание синтаксиса входного языка в языке LD модифицированных Бэкусово-Науровских форм, при этом грамматика реализуемого языка должна согласовываться с определением семантики, основной принцип которого состоит в распределении функции перевода по синтаксическим элементам грамматики;

— расширение программы Сканер чтением из входной строки типов данных, которые не содержатся в РСП ТЕРЕМ;

— определение семантики входного языка в виде системы семантических программ и шаблонов;

— описание спецификации входного языка в виде БНФ-грамматики, нагруженной именами семантических действий и шаблонами;

— применение Конструктора к спецификации входного языка для получения Синтаксических таблиц (предполагается, что Конструктор, работающий автономно, не подлежит модификации в связи с реализацией нового языка вследствие того, что его входной язык LD фиксирован);

— сборка языкового процессора;

— задание параметров процессора.

Все перечисленные операции выполняются циклически по мере наращивания возможностей реализуемого процессора.

2.2. Языковой процессор системы ОЕС. Входными данными языкового процессора системы ОЕС являются: текст на входном языке, специальная таблица для синтаксического анализа входного текста и словарь ключевых слов. Таблица строится по грамматике входного языка средствами РСП ТЕРЕМ. Задавая словарь ключевых слов, можно настроить языковой процессор на обработку входного текста, написанного на том или ином языке (естественном или формальном). В экспериментах с ОЕС использовались тексты требований на русском, украинском и английском языках.

2.3. Словари ключевых слов. Словарь представляет собой множество слов и фраз входного языка, выбранных в качестве ключевых, разбитое на разделы. Совокупность слов и фраз одного раздела образует отдельное понятие (например, «глагол», «тип объекта», «служебное слово» и др.). Словарь открыт для пополнения как новыми словами, так и разделами.

2.4. Грамматики и уровни обработки входного текста в системе ОЕС. Обработка входного текста в ОЕС многоуровневая. Первый уровень — предварительная обработка текста, второй уровень — получение текста на выходном языке. Промежуточные результаты доступны для просмотра и редактирования.

Первый уровень обработки текста обеспечивает определение типа каждого предложения входного текста, а также выделение и, возможно, обработку составных частей предложений. Предварительная обработка входного текста может включать один или несколько этапов. На первом этапе (структурирования) осуществляется как определение типа каждого предложеня текста, так и преобразование предложения в зависимости от его типа. Те предложения, для которых это возможно, преобразуются в формат предложений выходного языка системы ОЕС. Последующие этапы выполняются итеративно языковым процессором, настроенным с помощью грамматики выравнивания. На этапе выравнивания языковой процессор просматривает текст, полученный на предыдущем этапе, и в зависимости от типа очередного предложения либо подвергает его структурированию, либо оставляет без изменения (т.е. в том виде, который данное предложение получило на предыдущем этапе обработки).

Рассмотрим подробнее первый этап обработки входного текста. Выполнение этого этапа обеспечивается настройкой языкового процессора с помощью специальной грамматики структурирования. Предложения входного языка делятся на структурируемые и неструктурируемые. Грамматикой определены такие типы предложений: присваивания, замечания, условные предложения (полные и неполные), отсылки (предложения каждого из этих типов относятся к структурируемым), а также предписания (относятся к неструктурируемым предложениям).

Для структурируемого предложения, кроме типа, определяется также его структура (результаты структурирования приводятся в выходном файле). Так, в неполных условных предложениях выделяются условие и так называемое «действие» (часть предложения, описывающая, что происходит или должно произойти, если условие выполнено), а в полных условных предложениях, кроме того, выделяется часть предложения, описывающая, что происходит, если условие не выполнено.

Например, предложение (требование)

If the in Cause the received Direction message is “Signal failed”, then the CMS shall start timer T1, otherwise the CMS shall start timer T3.

классифицируется, структурируется и преобразуется следующим образом:

type: ifotherwise-sentence
condition

the Cause in the received Direction message is “Signal failed”.
action
the CMS shall start timer T1.

otherwise
action
the CMS shall start timer T3.

Выделение структурных частей предложения осуществляется с помощью ключевых слов.

Некоторые предложения выходного текста, полученного на первом этапе обработки, необходимо обрабатывать повторно (выделять составные части).

Рассмотрим, например, следующее предложение:

Если принято сообщение 'вброшен посторонний предмет Т' и бокс возврата ВМ не пуст, то Т попадает в бокс возврата ВМ, бокс возврата ВМ не пуст или бокс возврата ВМ переполнен.

На первом этапе обработки данное предложение будет классифицировано, структурировано и преобразовано в два предложения (соответствующие альтернативным действиям исходного требования), которые затем итеративно обработаются:

type: ifor-sentence
Если принято сообщение 'вброшен посторонний предмет Т' и бокс возврата ВМ не пуст, то Т попадает в бокс возврата ВМ, бокс возврата ВМ не пуст.
Если принято сообщение 'вброшен посторонний предмет Т' и бокс возврата ВМ не пуст, то Т попадает в бокс возврата ВМ, бокс возврата ВМ переполнен.

Каждое из предложений данного выходного текста является условным, однако их составляющие (условия, «действия») не выделены.

Чтобы в результате предварительной обработки входного текста получить текст, не содержащий неструктурированных предложений, хотя средствами ОЕС их структурирование возможно, нужна дополнительная обработка.

Такая дополнительная обработка (выравнивание) выполняется на втором этапе. Для этого языковой процессор настраивается с помощью специальной грамматики выравнивания.

Второй уровень обработки текста обеспечивается языковым процессором, настроенным с помощью грамматики «условий-действий», который преобразует предложения, полученные в результате предварительной обработки, в предложения выходного языка системы ОЕС. Особенностью этого уровня обработки является использование вспомогательной грамматики, дополняющей грамматику «условий-действий». Средства РСП ТЕРЕМ дают возможность строить языковой процессор, использующий две грамматики: основную и вспомогательную. Некоторые нетерминальные символы основной грамматики определяются во вспомогательной. При таком подходе одну и ту же основную грамматику можно использовать с различными вспомогательными. В данном случае вспомогательную грамматику применяем для определения понятия «клише» (фразы из словаря рабочей лексики, связанной с некоторым атрибутом и поясняющей его смысл). Используя эту связь, можно задавать преобразования фраз ЕЯ в выражения формального языка.

Приведем пример использования клише. Пусть в информационной модели некоторого объекта введен атрибут *OK* перечислимого типа, среди значений которого есть значение *cr*, имеющее смысл «кофе готов, стаканчик с кофе на полочке, окно выдачи открыто».

Если фраза-клише «кофе готов, стаканчик с кофе на полочке, окно выдачи открыто», связанная со значением *cr* атрибута *OK*, встречается в требрвании вида

«Если кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, принято сообщение 'клиент забрал кофе', то окно выдачи закрывается, таймер wind_timer отключается.»,

то это означает, что выполняется условие $OK = cr$, и тогда данную фразу-клише из входного текста можно преобразовать к (формальному) виду $OK = cr$.

Если эта фраза встречается в предложении-требовании вида

«Если клавиша СТАНДАРТ подсвечена и принято сообщение 'нажата клавиша СТАНДАРТ', то значение НМ уменьшается на 70, на табло Tab высвечивается новое значение НМ, кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, включается таймер wind_timer, значение Sigma увеличивается на 70, значение Sigma_St увеличивается на 1, клавиша СТАНДАРТ не подсвечивается, клавиша ДВОЙНОЙ не подсвечивается.»,

то это означает, что атрибут перечислимого типа OK получает значение cr, и тогда рассматриваемую фразу-клише можно преобразовать к (формальному) виду $OK := cr$.

Таким образом, с помощью словаря рабочей лексики можно определить во вспомогательной грамматике понятия «клише-условие» и «клише-действие» и связать с каждой фразой-клише семантическое действие, которое задает преобразование этой фразы в формальное выражение. В результате получается гибкая языковая система, настраиваемая на предметную область.

Отметим, что при построении информационной модели *объекта* некоторые шаги его функционирования (действия) могут рассматриваться как элементарные (недетализируемые). С ними не связываются никакие атрибуты (это так называемые неатрибутируемые действия). Фразы входного текста, описывающие такие действия, не преобразуются к формальному виду (в результате в выходном тексте появляются ЕЯ-фразы).

3. ИСПОЛЬЗОВАНИЯ МЕТОДИКИ И ЯЗЫКОВОЙ СИСТЕМЫ ОЕС НА ПРИМЕРЕ КОФЕ-АВТОМАТА

Рассмотрим задачу: специфицировать автомат, который продает кофе (обычную порцию либо двойную), причем предполагается, что объемы порций и их стоимость известны. В данном случае подлежащий специфицированию объект — класс устройств (автоматов), а функция, которую должен выполнять каждый из них, — порционная продажа. Исходя из этой постановки, строим функциональную, структурную и информационную модели КА, а затем — модель поведения (совокупность поведенческих правил его функционирования).

3.1. Функциональная модель КА:

3.1.1. Основная функция КА и уточнение параметров.

Кофе-автомат продает клиентам порции кофе двух видов: обычную («стандарт») за 70 с (центов) и двойную («двойной») за 125 с.

3.1.2. Способ связи КА с клиентом (со средой).

Автомат принимает от клиента монеты номиналом 1 с, 5 с, 10 с, 25 с, 50 с. Готовый заказ (порция кофе) появляется в окошке выдачи и клиенту предоставляется определенное время, чтобы забрать его, в противном случае заказ утилизируется без возможности возврата денег.

3.1.3. Вспомогательные функции.

Если сумма денег, которые вбросил клиент, достаточна для покупки обычной (двойной) порции кофе, на панели заказа подсвечивается соответствующая кнопка, после нажатия на которую в окошке выдачи появляется стаканчик с заказанным кофе.

Для удобства клиента сумма вброшенных им в автомат денег, а также информация о готовности автомата к работе или предупреждения о заминках высвечиваются на специальном табло.

Клиент также может вернуть свои деньги, не сделав заказа.

Чтобы автомат мог выполнять свою основную функцию многократно, после каждого действия клиента включается таймер, по которому через определенное время (на данном этапе не уточняется) автомат приводится в начальное состояние.

3.1.4. Функциональные ограничения КА.

Для простоты рассматриваем случай, когда автомат не дает сдачу, т.е. сумма вброшенных денег должна быть кратной стоимости допустимой порции кофе или превышать ее.

Проанализировав приведенную функциональную модель КА, выделяем составляющие, необходимые для выполнения функций, перечисленных в функциональной модели, и создаем структурную модель КА.

3.2. Структурная модель КА. Кофе-автомат имеет две составляющие: внешнюю («видимую») и внутреннюю («невидимую»).

Внешняя часть состоит из:

— табло;

— панели заказа;

— щели (слота) для вбрасывания монет;

— окошка выдачи заказа с полочкой, на которую помещается стаканчик с кофе;

— кнопки возврата монет;

— бокса возврата монет.

На табло для клиента высвечиваются сообщения: приглашение угоститься, сумма денег, вброшенных клиентом, предупреждение о переполнении/неисправности бокса возврата монет.

На панели заказа имеются две кнопки (клавиши): «стандарт» и «двойной», предназначенные для заказа соотвествующих порций кофе. Кнопка срабатывает при ее нажатии, если она подсвечена, и она становится подсвеченной, если клиент вбросил достаточно денег для выполнения заказа.

Щель (слот) для вбрасывания монет может быть открыта или закрыта. Как отмечалось ранее, в открытую щель следует бросать монеты номиналом 1 с, 5 с, 10 с, 25 с, 50 с.

Окошко выдачи заказа либо открыто, если заказ готов и стаканчик с кофе стоит на полочке, либо закрыто в остальных случаях. Предполагается, что имеется сенсор, реагирующий на то, что стаканчик с кофе забирают с полочки. Окошко также управляется таймером.

Кнопка возврата монет может быть нажата или нет, она используется клиентом для возврата денег при отказе от заказа.

Бокс возврата монет представляет собой лоток, в который помещаются монеты, возвращаемые клиенту при отказе от заказа. Бокс имеет ограниченную емкость. Предполагается, что имеется сенсор, реагирующий на переполнение бокса возврата монет.

Внутренняя часть кофе-автомата состоит из:

— накопителя монет;

— хранилища монет.

В накопителе находятся монеты, поступившие от клиента для заказа кофе. Накопитель монет связан со слотом для вбрасывания монет, боксом возврата монет и с хранилищем монет. Емкость накопителя ограничена. Предполагается, что имеется сенсор, реагирующий на его переполнение.

В хранилище монет находятся деньги, уплаченные за кофе, и деньги, забытые в накопителе (невостребованные клиентом в течение некоторого времени, контролируемого таймером). Хранилище монет связано с накопителем монет.

При выполнении заказа определенной порции кофе соответствующая сумма из накопителя монет поступает в хранилище монет.

С учетом построенной структурной модели КА внесем уточнения в его функциональную модель, а именно в описание вспомогательных функций.

Уточнения к разделу 3.1.3. функциональной модели КА. Если сумма денег, вброшенных клиентом, которая высвечивается на табло, достаточна для покупки обычной (двойной) порции кофе, то подсвечивается соответствующая кнопка на панели заказа. Клиент может нажать на такую кнопку, и тогда в окошке выдачи появится стаканчик с заказанным кофе. Если клиент не сразу отреагировал на выставленный ему стаканчик, то (посредством таймера) через минуту ему подается напоминание, а если и после этого он еще через минуту не забирает заказанное, то заказ аннулируется без возврата потраченной на него суммы.

Клиент также может нажатием на кнопку возврата монет вернуть все вброшенные им деньги, не сделав заказа, или часть денег, оставшуюся после выполнения заказа (например, в случае, если вброшены деньги для двух порций кофе, а заказана лишь одна).

Внесенные уточнения не приводят к увеличению составляющих КА, поэтому не нужно пересматривать структурную модель. Однако эти уточнения необходимо принять во внимание при разработке информационной модели.

3.3. Информационная модель КА. Рассматриваемый КА функционирует в среде, ответственной за его исправность и наполнение ресурсами (электричество, вода, продукты, посуда и т.д.), которая сигнализирует о его исправности, но от самого этого механизма в данном случае абстрагируемся. Для описания этой характеристики состояния автомата введен атрибут F булевого типа. Таким образом:

$F = 1$ означает, что автомат включен, исправен и готов к работе;

$F = 0$ означает, что автомат вышел из строя.

Действия пользователя и реакции изнутри неспецифицированной части автомата моделируются с помощью сообщений, которые воспринимает автомат. Для этого вводится атрибут перечислимого типа MES со значениями:

1 c — в слот вброшена монета 1 цент;

5 c — в слот вброшена монета 5 центов;

10 c — в слот вброшена монета 10 центов;

25 c — в слот вброшена монета 25 центов;

50 c — в слот вброшена монета 50 центов;

anything — в слот вброшена нестандартная монета или посторонний предмет;

standard — клиент заказал обычную порцию кофе;

double — клиент заказал двойную порцию кофе;

return — клиент потребовал возвратить монеты;

SL_over — сообщение от сенсора в КА о переполнении накопителя монет;

box_BM_over — сообщение от сенсора в КА о переполнении бокса возврата монет;

coffee_taken — сообщение от сенсора о том, что клиент взял стаканчик с кофе с полочки;

invalid — сообщение непосредственно от автомата, что он не готов к работе;

empty — входных сообщений не имеется.

С табло связан атрибут Tab символьного типа.

С кнопками заказа «стандарт» и «двойной» связаны атрибуты OPS и OPD булевого типа, описывающие состояния кнопок: «подсвечена»–«не подсвечена» (можно делать заказ или нельзя). Таким образом:

OPS = 1 означает, что кнопка (клавиша) «стандарт» подсвечена;

OPS = 0 означает, что кнопка (клавиша) «стандарт» не подсвечена;

OPD = 1 означает, что кнопка (клавиша) «двойной» подсвечена;

OPD = 0 означает, что кнопка (клавиша) «двойной» не подсвечена.

Для описания состояния слота для вбрасывания монет (свободен–блокирован) используется атрибут SL булевого типа:

SL = 1 означает, что слот блокируется (закрыт);

SL = 0 означает, что слот не блокируется (открыт).

Окошко выдачи заказа может быть открытым или закрытым, а полочка — пустой или непустой. Для описания состояний окошка с полочкой введен атрибут OK перечислимого типа со следующими значениями:

cr — кофе готов, стаканчик с кофе на полочке, окно выдачи открыто;

ca — окно выдачи закрывается;

cl — окно выдачи закрывается и стаканчик с кофе убирается с полочки.

Для описания состояний бокса возврата монет введен атрибут перечислимого типа BM со значениями:

em — бокс возврата пуст;

ov — бокс возврата переполнен;

ne — бокс возврата не переполнен и не пуст.

С накопителем монет связаны целочисленные атрибуты:

HM — сумма денег в накопителе;

Sigma_leaved — текущая сумма забытых в накопителе денег.

С хранилищем монет связаны такие целочисленные атрибуты:

Sigma — текущая сумма заработанных денег;

Sigma_St — текущее количество проданных обычных порций;

Sigma_double — текущее количество проданных двойных порций.

Контроль времени осуществляется двумя таймерами:

wind_timer — контролирует интервалы времени между открыванием окошка со стаканчиком кофе, взятием стаканчика клиентом и закрыванием окошка;

gen_timer — контролирует интервал времени после последнего воздействия клиента на автомат, после чего сбрасывает значения всех атрибутов (кроме используемых для учета) в начальное состояние.

Множество значений атрибута MES определяет алфавит входных воздействий, выходные реакции автомата задаются значениями атрибутов из набора (Tab, OPS, OPD, SL, OK), которые клиент имеет возможность наблюдать визуально (надписи на табло, подсветка кнопок панели заказа, состояние слота, состояние окошка выдачи). Состояние непосредственно кофе-автомата определяется значениями всех перечисленных атрибутов, кроме MES и Tab.

Следующий шаг построения ФС в соответствии с предлагаемой методикой — формирование словаря рабочей лексики.

Рассмотрим информационную модель КА. При описании атрибутов сформировались фразы-клише, описывающие различные ситуации, возникающие при функционировании КА, например, «кнопка (клавиша) «стандарт» подсвечена», «слот блокируется», «кофе готов, стаканчик с кофе на полочке, окно выдачи открыто», «бокс возврата переполнен» и др. С каждой такой фразой связан определенный атрибут и его значение. Совокупность этих фраз-клише составляет рабочую лексику, которая используется при построении модели поведения КА (функциональных спецификаций КА).

3.4. Функциональные спецификации КА. Построим модель поведения КА на ЕЯ, ограниченном использованием рабочей лексики. Эта модель представлена в виде совокупности правил функционирования КА, сформулированных на основе ЕЯ-моделей с использованием клише. Правила оформляем отдельными абза-

цами, которые для удобства пронумерованы.

1. Стоимость «стандарта» равна 70 с, стоимость «двойного» равна 125 с.

2. Если автомат включен и исправен, то на табло высвечивается либо приглашение угоститься, либо отображается сумма НМ денег, уже находящихся в приемнике (превышающая 0 центов); слот SL открыт.

3. Если на табло высвечивается приглашение угоститься, то предполагается, что сумма НМ монет в приемнике равна нулю.

4. Если КА выключен или не исправен, то слот SL закрыт и табло отключено.

5. Если принято сообщение 'вброшена монета одного из номиналов (1 с, 5 с, 10 с, 25 с, 50 с)', то значение НМ увеличивается на соответствующую величину, на табло Tab высвечивается новое значение НМ.

6. Если принято сообщение 'вброшен посторонний предмет T' и бокс возврата ВМ пуст, то T попадает в бокс возврата ВМ и бокс возврата ВМ не пуст. Если принято сообщение 'вброшен посторонний предмет T' и бокс возврата ВМ не пуст, то T попадает в бокс возврата ВМ, бокс возврата ВМ не пуст или бокс возврата ВМ переполнен.

7. Если принято сообщение 'переполнен бокс возврата ВМ', то на табло Tab высвечивается “Return box overloading”, бокс возврата ВМ переполнен, слот SL блокируется.

8. Если сумма денег НМ не менее 70 и клавиша СТАНДАРТ не подсвечена, то клавиша СТАНДАРТ подсвечивается.

9. Если сумма денег НМ не менее 125 и клавиша ДВОЙНОЙ не подсвечена, то клавиша СТАНДАРТ подсвечивается и клавиша ДВОЙНОЙ подсвечивается.

10. Если клавиша СТАНДАРТ подсвечена и принято сообщение 'нажата клавиша СТАНДАРТ', то значение НМ уменьшается на 70, на табло Tab высвечивается новое значение НМ, кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, включается таймер wind_timer, значение Sigma увеличивается на 70, значение Sigma_St увеличивается на 1, клавиша СТАНДАРТ не подсвечивается, клавиша ДВОЙНОЙ не подсвечивается.

11. Если клавиша ДВОЙНОЙ подсвечена и принято сообщение 'нажата клавиша ДВОЙНОЙ', то значение НМ уменьшается на 125, на табло Tab высвечивается новое значение НМ, кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, включается таймер wind_timer, значение Sigma увеличивается на 125, значение Sigma_double увеличивается на 1, клавиша СТАНДАРТ не подсвечивается, клавиша ДВОЙНОЙ не подсвечивается.

12. Если кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, принято сообщение 'клиент забрал кофе', то окно выдачи закрывается, таймер wind_timer отключается.

13. Если кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, таймер wind_timer включен в течение 1 мин, то на табло Tab высвечивается “Please, take your coffee”.

14. Если кофе готов, стаканчик с кофе на полочке, окно выдачи открыто, таймер wind_timer включен в течение 2 мин, то окно выдачи закрывается и стаканчик с кофе убирается с полочки, таймер wind_timer отключается.

15. Если принято сообщение 'приемник монет НМ переполнен', то слот SL блокируется и на табло Tab высвечивается “Return box is overflown. You may order coffee or take your money back”.

16. Если таймер gen_timer срабатывает, то таймер gen_timer отключается, счет забытых денег Sigma_leaved увеличивается на НМ, НМ обнуляется.

17. Если принято сообщение 'нажата кнопка возврата монет' и бокс возврата ВМ не переполнен, то монеты из накопителя монет пересылаются в бокс возврата, НМ обнуляется, бокс возврата ВМ не пуст или бокс возврата ВМ переполнен.

18. Если принято сообщение 'автомат вышел из строя', то таймер wind_timer отключается и таймер gen_timer отключается, слот SL блокируется.

Первые четыре правила глобальные, а остальные являются локальными.

3.5. Формализация функциональных спецификаций. Следующий шаг — формализация полученной модели поведения. Рассмотрим модель поведения КА, ограниченную локальными правилами (т.е. совокупность ранее приведенных правил, начиная с пятого). Приведем результат преобразования этой модели, выполненного языковым процессором системы ОЕС. Составляющие одно правило (требование) предложения (ФС) сгруппированы. Вместо нумерации требований используется разделитель ---------- между ними.

```
MES = 1 ----> HM := HM  + 1 ; Tab := val (HM + 1).
MES = 5 ----> HM := HM  + 5 ; Tab := val (HM + 5).
MES = 10 ----> HM := HM  + 10 ; Tab := val (HM +10).
MES = 25 ----> HM := HM  + 25 ; Tab := val (HM +25).
MES = 50 ----> HM := HM  + 50 ; Tab := val (HM +50).
----------

(MES = anything) & (BM = em) ----> T попадает в бокс возврата BM ; BM := ne.
(MES = anything) & (BM = ne) ----> T попадает в бокс возврата BM; BM := ne.
(MES = anything) & (BM = ne) ----> T попадает в бокс возврата BM; BM := ov.
----------

(MES = box_BM_over) ----> Tab := “Return box overloading”; BM := ov; SL := 0.
----------

(HM >= 70) & (OPS = 0) ----> OPS := 1.
----------

(HM >= 125) & (OPD = 0) ----> OPS := 1; OPD := 1.

----------

(OPS = 1) & (MES = standard) ----> HM := HM – 70; Tab := val (HM – 70);
OK := cr; start(wind_timer); Sigma := Sigma + 70; Sigma_St := Sigma_St +1;
OPS := 0; OPD := 0.
----------

(OPD = 1) & (MES = double) ----> HM := HM – 125; Tab := val (HM – 125);
OK := cr; start(wind_timer); Sigma := Sigma +125; Sigma_double :=
Sigma_double +1; OPS := 0; OPD := 0.
----------

(OK = cr) & (MES = coffee_taken) ----> OK := ca; stop(wind_timer).
----------

(OK = cr) & (wind_timer = 1') ----> Tab := “Please, take your coffee”.
----------

(OK = cr) & (wind_timer = 2') ----> OK := cl; stop(wind_timer).
----------

(MES = SL_over) —— SL:= 0; Tab := “Return box is overflown. You may order coffee or
take your money back”.
----------

expired(gen_timer) ----> stop(gen_timer); Sigma_leaved := Sigma_leaved +HM;
HM := 0.
----------

(MES = return) & (BM = ne) ----> монеты из накопителя монет пересылаются в бокс
возврата; HM := 0; BM := ne.
(MES = return) & (BM = ne) ----> монеты из накопителя монет пересылаются в бокс
возврата; HM := 0; BM := ov.
----------

(MES = invalid) ----> stop(wind_timer); stop(gen_timer); SL := 0.
---------- .
```

Фразы «T попадает в бокс возврата BM», «монеты из накопителя монет пересылаются в бокс возврата», встречающиеся в приведенной модели поведения, являются примерами описания неатрибутируемых действий. Уровень подроб-

ности описания для изменения состояний бокса возврата в построенной информационной модели КА ограничивается выделением трех его состояний: бокс возврата пуст, переполнен, не переполнен и не пуст. Содержимое бокса возврата (а также другие его возможные характеристики) в модели не уточняются. Поэтому авторы настоящей статьи считают возможным ограничиться простой констатацией действия, касающегося бокса возврата, и фиксацией изменения его состояния.

Учитывая общие соглашения, рассматриваемые как контекст по отношению к функциональным спецификациям КА, дополняем полученные ФС с помощью их редактирования вручную. Приведем эти соглашения:

— КА выполняет свои функции, если он исправен, т.е. $F = 1$;

— после каждого приема сообщения область MES для их хранения освобождается, т.е. MES := empty;

— таймер gen_timer, реагирующий на действия клиента, включается (переустанавливается) всякий раз, когда клиент совершает какое-либо действие, т.е. выполняется команда reset(gen_timer);

— если автомат неисправен или не имеется какого-либо ресурса, необходимого для его работы, то это означает, что автомат уже нефункциональный, т.е. $F := 0$.

В результате получаем модель поведения КА, дополненную согласно общим соглашениям (добавления выделены жирным курсивом).

(F = 1) & MES = 1 ----> HM := HM + 1; Tab := val (HM + 1)***; MES := empty; reset(gen_timer)***.
(F = 1) & MES = 5 ----> HM := HM + 5; Tab := val (HM + 5)***; MES := empty; reset(gen_timer)***.
(F = 1) & MES = 10 ----> HM := HM + 10; Tab := val (HM + 10)***; MES := empty; reset(gen_timer)***.
(F = 1) & MES = 25 ----> HM := HM + 25; Tab := val (HM + 25)***; MES := empty; reset(gen_timer)***.
(F = 1) & MES = 50 ----> HM := HM + 50; Tab := val (HM + 50)***; MES := empty; reset(gen_timer)***.

(F = 1) & (MES = anything) & (BM = em) ----> T попадает в бокс возврата BM; BM := ne; ***MES := empty; reset(gen_timer)***.
(F = 1) & (MES = anything) & (BM = ne) ----> T попадает в бокс возврата BM; BM := ne; ***MES := empty; reset(gen_timer)***.
(F = 1) & (MES = anything) & (BM = ne) ----> T попадает в бокс возврата BM; BM := ov; ***MES := empty; reset(gen_timer)***.

(F = 1) & (MES = box_BM_over) ----> Tab := “Return box overloading”; BM := ov; SL := 0; ***MES := empty***.

(F = 1) & (HM >= 70) & (OPS = 0) ----> OPS := 1.

(F = 1) & (HM >= 125) & (OPD = 0) ----> OPS := 1; OPD := 1.

(F = 1) & (OPS = 1) & (MES = standard) ----> HM := HM – 70; Tab := val (HM – 70); OK := cr; start (wind_timer); Sigma := Sigma +70; Sigma_St := Sigma_St +1; OPS := 0; OPD := 0 ***; MES := empty; reset(gen_timer)***.

(F = 1) & (OPD = 1) & (MES = double) ----> HM := HM – 125; Tab := val (HM – 125); OK := cr; start (wind_timer); Sigma := Sigma + 125; Sigma_double := Sigma_double +1; OPS := 0; OPD := 0 ***; MES := empty; reset(gen_timer)***.

(F = 1) & (OK = cr) & (MES = coffee_taken) ----> OK := ca; stop (wind_timer) ***; MES := empty; reset(gen_timer)***.

(F = 1) & (OK = cr) & (wind_timer = 1') ----> Tab := “Please, take your coffee”.

(F = 1) & (OK = cr) & (wind_timer = 2') ----> OK := cl; stop (wind_timer).

(F=1) & (MES = SL_over) ----> SL := 0; Tab := “Return box is overflown. You may order coffee or take your money back”***; MES := empty***.

(F* = 1*) & expired (gen_timer) ----> stop (gen_timer); Sigma_leaved := Sigma_leaved + HM; HM := 0.

(F* = 1*) & (MES = return) & (BM = ne) ----> монеты из накопителя монет пересылаются в бокс возврата; HM := 0; BM := ne ***; MES := empty***.

(F* = 1*) & (MES = return) & (BM = ne) ----> монеты из накопителя монет пересылаются в бокс возврата; HM := 0; BM := ov ***; MES := empty***.

(MES = invalid) ----> ***F := 0;*** SL := 0; stop (wind_timer); stop (gen_timer) ***; MES := empty***.

ЗАКЛЮЧЕНИЕ

В данной работе описаны средства разработки формальных спецификаций систем и устройств, которые дают возможность построить функциональную спецификацию объекта (системы или устройства) на естественном языке, а затем осуществить ее формализацию. В результате предложены:

— методика разработки функциональных спецификаций систем и устройств на ЕЯ;

— языковая система ОЕС, позволяющая преобразовать ЕЯ-спецификацию в формальный текст.

Назначение методики — предоставить план построения ФС объекта. Применение предложенной методики дает возможность:

— разрабатывая ФС, оставаться в рамках ЕЯ;

— строить ФС на ЕЯ таким образом, чтобы облегчить ее перевод на формальный язык.

Описанная в настоящей работе языковая система ОЕС является интерактивной и эксплуатировалась в экспериментальном режиме. Она позволяет пошагово преобразовывать входной текст к формальному виду, при этом промежуточные результаты преобразований доступны пользователю. Входной текст может быть написан на украинском, русском или английском языках. Эксперименты, проведенные с системой, дали возможность наметить такие направления ее усовершенствования и развития:

— разработка и реализация дружественного интерфейса с системой;

— автоматизация формирования словаря рабочей лексики;

— автоматизация построения вспомогательной грамматики по словарю рабочей лексики;

— создание средств интерфейса для работы со словарями ключевых слов.

Предполагаемое участие человека в системе, т.е. неполная автоматизация, связано как с неоднозначностью, присущей текстам на ЕЯ, которую сложно формализовать, так и с высокой ценой ошибки, что может быть внесена в формальную модель и затем быть выявляемой на этапе верификации модели. Вместе с тем авторы продолжают работу над тем, как повысить степень автоматической обработки с использованием понятия онтологий, ориентированных на предметные области.

Задача формализации спецификаций, представленных на ЕЯ, может рассматриваться и в такой постановке: предварительно переформулировать входной ЕЯ-текст так, чтобы, используя ограниченный ЕЯ, минимизировать количество употребляемых в нем слов и упростить тем самым последующий анализ текста. Интересный опыт в этом направлении был продемонстрирован в популярной книге [11], автор которой, используя лишь 1000 слов английского языка, описал несколько нетривиальных явлений. Эту книгу, как одну из нескольких вышедших в 2015 г. и шести примечательных для себя, отметил Билл Гейтс. В своем отзыве [12] на нее он написал, что «if you can’t explain something simply, you don’t really understand it»[2]. В то же время стремление к ограниченности ЕЯ мож-

[2]Тот, кто не может объяснить нечто простыми словами, тот в дейтвительности не понимает этого.

но сочетать с использованием некоего словаря терминов, характерных для предметной области специфицируемого объекта.

СПИСОК ЛИТЕРАТУРЫ

1. International Telecommunications Union. Recommendation Z.151. — User Recommendation Notation (URN), 2008. — 190 p. — http://www.itu.int/rec/T-REC-Z.151-2008II-I/en
2. Handbook of process algebra / J.A.Bergstra, A.Ponce, S.A. Smolra (Eds.). — North-Holland, 2001. — 1342 p.
3. Летичевский А.А. Инсерционное моделирование // УСиМ. — 2012. — № 6. — С. 3–15.
4. Sowa J.F, Knowledge representation: Logical, philosophical, and computational foundations. — Pacific Grove (CA): Brooks Cole Publishing Co., 1999. — 594+xiv p.
5. Natt och Dag J., Gervasi V., Brinkkemper S., Regnell B. Speeding up requirements management in a product software company: linking customer wishes to product requirements through linguistic // Proc. of the 12th IEEE International Conference on Requirements Engineering (RE 2004), 6–10 Sept. 2004, Kyoto, Japan. IEEE Computer Society 2004. — P. 283–294. — http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.477.3979&rep=rep1 &type=pdf.
6. Zhoe J., Lu Y., Lundqvist K., Lonn H., Karlsson D., Liwang B. Towards feature-oriented requirements validation for automotive systems // Proc. of the 22nd IEEE International Requirements Engineering Conference (RE'14), 25-29 Aug., Karlskrona, Sweden. — P. 428–436. — http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6912296&url=http%3A%2F% 2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D6912296.
7. Filipovikj P., Nyberg Rodriguez-Navas M.G. Reassessing the pattern-based approach for formalizing requirements in the automotive domain // Proc. of the 22nd IEEE International Requirements Engineering Conference (RE'14), 25–29 Aug., Karlskrona, Sweden. — P. 444–450. — http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6903646&filter%3DAND%28p_IS_Number%3A6912234%29&pageNumber=3.
8. Мищенко Н.М., Щёголева Н.Н. О проектировании языковых процессоров на ПЭВМ // Кибернетика и системный анализ. — 1993. — № 2. — С. 110–117.
9. Мищенко Н.М. Применение РСП Терем для реализации языка параллельного программирования Маяк // Проблемы создания Супер-ЭВМ, Супер-систем и эффективность их применения: Тез. докл. 1-й Всесоюз. конф. (15-17 сентября 1987г., Минск). — Минск: Ин-т математики АН БССР, 1987, ч. 1. — С. 129–131.
10. Берестовая С.Н., Годлевский А.Б., Гороховский С.С., Капитонова Ю.В., ЛетичевскийА.А., Мищенко Н.М. О реализации языков семейства МАЯК для многопроцессорных вычислительных комплексов с макроконвейерной организацией вычислений // Кибернетика. — 1989. — № 3. — С. 29–34.
11. Munroe R. Thing explainer: complicated stuff in simple words // Houghton Mifflin Harcourt, 24 Nov. 2015. — P. 61. — http://www.amazon.co.uk/Thing-Explainer-Complicated-Stuff-Simple/dp/1473620910?tag=duckduckgo-ffsb-uk-21.
12. Gates B. A basic guide for curious minds. — https://www.gatesnotes.com/books/thing- explainer.

Надійшла до редакції 09.03.2016

О.Б. Годлевський, Н.М. Міщенко, М.К. Мороховець, О.Д. Феліжанко, Н.М. Щоголева

МЕТОДИКА РОЗРОБКИ ТА ЗАСОБИ ФОРМАЛІЗАЦІЇ ФУНКЦІОНАЛЬНИХ СПЕЦИФІКАЦІЙ СИСТЕМ І ПРИСТРОЇВ

Анотація. Запропоновано підхід до побудови функціональних специфікацій та до автоматизації переходу від специфікацій, поданих природними мовами, до формальних моделей у форматі, що є зручним для подальшої верифікації цих моделей та валідації побудованих за ними програмних кодів. Перетворення текстів у моделі здійснюється за допомогою описаної інтерактивної системи ОЕС. Описаний підхід ілюструється розгорнутим прикладом.

Ключові слова: функціональна специфікація, поведінкова модель, формальна специфікація систем та пристроїв, мовний процесор.

A.B. Godlevsky, N.M. Mishchenko, M.K. Morokhovets, O.D. Felizhanko, N.N. Shchogoleva

A DEVELOPMENT TECHNIQUE AND FORMALIZATION MEANS FOR FUNCTIONAL SPECIFICATIONS OF SYSTEMS AND DEVICES

Abstract. In this paper, we propose an approach to the construction of functional specifications and to automation of the transition from specifications in natural language to formal models in a format suitable for the subsequent verification of these models and validation of software code built on them. Converting texts to models is carried out by means of the interactive system OEC described in the paper. Our approach is illustrated by a detailed example.

Keywords: functional specification, behaviour model, the formal specification of systems and devices, language processor.

Годлевский Александр Богуславович,
кандидат физ.-мат. наук, старший научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: a.godl49@gmail.com.

Мищенко Надежда Михайловна,
кандидат физ.-мат. наук, старший научный сотрудник, Киев, e-mail: nadmykh@ukr.net.

Мороховец Марина Константиновна,
кандидат физ.-мат. наук, старший научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, доцент Национального технического унверситета Украины «Киевский политехнический институт», Киев, e-mail: marina.morokhovets@gmail.com.

Фелижанко Ольга Дмитриевна,
научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: felizhanko@yandex.ua.

Щеголева Наталья Николаевна,
научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: nat@incyb.kiev.ua.