УДК 681.3.07

Платформонезависимый подход моделирования и разработки распределённых систем

Л. С. Глоба, А. В. Ермольчев, В. Н. Оленюк
Национальный технический университет Украины «КПИ»

Статья посвящена описанию платформонезависимого подхода разработки распределенных программных систем, управляемому моделью, а также построению языка абстрактного описания бизнес-процессов, который используется в данном подходе.

Ключевые слова: *распределенная программная система, моделирование, управление, бизнес-процесс, ошибки, формальный язык, UML, XML, GUI.*

Стаття присвячена опису платформонезалежного підходу розробки розподілених програмних систем, який управляється моделлю, а також побудові мови абстрактного опису бізнес-процесів, яка використовується в даному підході.

Ключові слова: *розподіленна програмніха система, моделювання, управління, бізнес-процес, помилки, формальна мова, UML, XML, GUI.*

The article is intended for description of platform independent approach of model driven distributed systems modeling and development and language which is used in this approach.

Keywords: *distributed software system, software development, modeling, management, business process, errors, formal languages, UML, XML, GUI.*

Вступление

Построение современных распределенных систем, работающих в среде Интернет с различными типами терминальных устройств (персональные компьютеры, мобильные телефоны, смартфоны, КПК и т.д.) требует специальной методологии их создания и развертывания. Одной из широко используемых сегодня методологий является методология разработки программного обеспечения, управляемая моделью (Model-Driven Software Development - MDSD) [1,2]. Ее суть заключается в описании разрабатываемой программной системы с помощью независимого от платформы средства (напр. UML) – создании модели с последующим генерированием зависимой от платформы модели и её реализации. Эта модель представляет собой описание бизнес-процессов как последовательности вызовов различных сервисов, каждый из которых решает строго определенную задачу.

Однако данная методология не предусматривает формализованного подхода к проблеме исправления ошибок на самых ранних этапах жизненного цикла разработки программного обеспечения – на этапе бизнес-моделирования предметной области.

На рынке разработки программного обеспечения существует несколько широко известных технологий, нацеленных на решение задач бизнес-моделирования предметной области. Наиболее известными среди них являются [1,2]:

- SADT;
- ARIS;
- BPMN;

- диаграммы деятельности UML.

Все эти технологии служат только для описания бизнес-процесса с помощью той или иной нотации. В результате моделирования с помощью вышеперечисленных технологий на выходе получается диаграмма соответствующего вида, но она не предоставляет средств обработки и последующего анализа проектируемой распределенной системы в удобном виде.

1. Подход к решению задачи проектирования

Предлагаемый подход [3,4] заключается в предоставлении средства, которое позволяет не только формировать диаграммы, но и выполнять их с учётом характеристик конечного терминального устройства пользователя и опираться на дополнительные данные, внесенные с помощью расширенных нотаций, сохраняемых в базе данных проекта. Под выполнением понимается моделирование процесса взаимодействия участников (актеров) данного бизнес-процесса между собой: программа-человек, человек-программа, программа-программа, человек-человек (например, разговор по мобильному телефону).

В качестве технологии создания базовой диаграммы бизнес-процесса была предложена UML-диаграмма деятельности (activity diagram).

На начальной стадии с помощью известной технологии (например, UML) описывается соответствующий бизнес-процесс. Описание происходит итеративно, т.е. на первой итерации процесс описывается на высоком уровне абстракции; при необходимости, с каждой последующей итерацией описание производится более и более детально.

На следующей стадии все действия этой диаграммы дифференцируются по функциональному признаку, возможности их автоматизации. Можно выделить три ступени автоматизации действий: полностью автоматизируемые (выполнение вычислений, выполнение запросов к базе данных и т.д.), частично автоматизируемые (требующие взаимодействия с оператором) и неавтоматизируемые (взаимодействие человек-человек).

На третьей стадии каждый из видов деятельности может быть описан с помощью языка, который позволяет выполнить данный бизнес-процесс в специализированной среде промежуточного слоя программного обеспечения. В качестве такого языка предлагается к рассмотрению XML-подобный язык FaceXML.

2. FaceXML

2.1. Обзор FaceXML

Рассматриваемый язык предназначен для описания полностью и частично автоматизируемых действий UML-диаграммы «activity diagram» или ей подобной. Под полностью автоматизируемым действием понимается некоторая последовательность операций, выполняемая в строго определенном порядке без вмешательства оператора. Под частично автоматизируемым действием – последовательность операций, требующая на одном или нескольких этапах своего выполнения вмешательства оператора. Очевидно, что взаимодействие системы с оператором подразумевает использование соответствующего интерфейса. Учитывая тенденции развития программных систем,

предполагается, что интерфейс является графическим. Таким образом, язык его описания должен включать следующие возможности:

- предоставлять способ описания графического интерфейса пользователя;
- описывать частично-автоматизируеме действия;
- иметь возможность выполнения последовательности операций для полностью автоматизируемых действий.

К тому же, описание графического интерфейса должно содержать данные о внешнем виде интерфейса, а также информацию о поведении системы в целом в контексте поведения элементов данного графического интерфейса при воздействии на них со стороны пользователя.

Предлагаемое решение в виде XML-подобного языка описания действий состоит из двух частей: непосредственно FaceXML для описания внешнего вида графического интерфейса пользователя (Graphical User Interface – GUI) и FaceXML Processing Language (FPL) для описания полностью автоматизируемых действий. Кроме этого, FPL используется для решения задачи описания поведения графического интерфейса при воздействии на него пользователя.

Таким образом, FaceXML решает две задачи:

- описание внешнего вида GUI;
- описание поведения GUI при взаимодействии с пользователем.

Согласно этому общая структура описания действия изображена на рис.1.

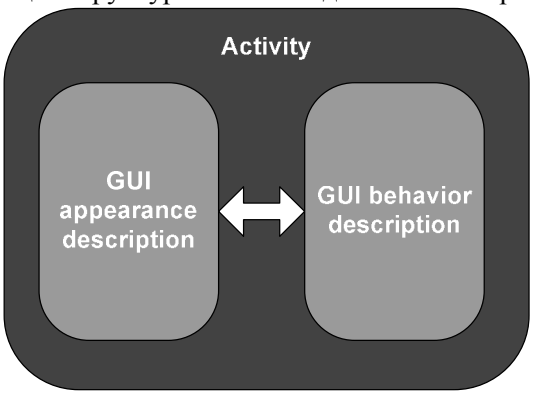

Рис.1. Общая структура описания действия с помощью FaceXML.

В целях упрощения внесения изменений и расширения созданных описаний было предложено отделить отображаемые данные от способа их отображения. Для этого описание внешнего вида GUI разделено на описание данных (model) и на описание способа отображения этих данных (view). Описание поведения графического интерфейса заключается в описании обработчиков событий, генерируемых элементами GUI. С учетом вышесказанного структура описания принимает вид, изображенный на рис.2.

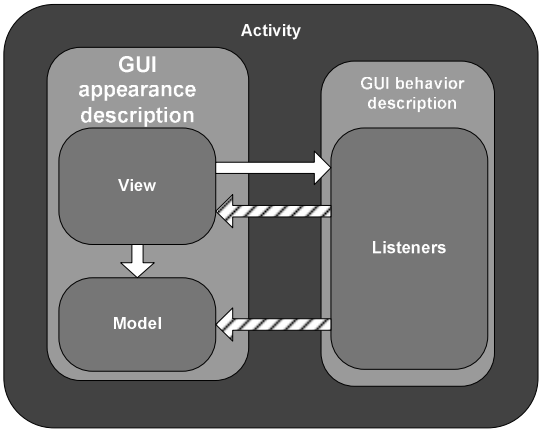

Рис.2. Структура описания действия с помощью FaceXML.

Сплошные стрелки показывают направление зависимостей между составляющими описания, заштрихованные – направление внесения изменений во время выполнения бизнес процесса.

2.2. Формальная грамматика FaceXML

Порождающая грамматика G языка L(G) может быть формально представлена как четверка:

$G = (V_t, V_n, P, S)$, где

- V_t — алфавит нетерминальных символов;
- V_n — алфавит терминальных символов;
- $P = \{V_t \; x \; V_n\}$ – продукции или все возможные синтаксические цепочки, построенные путем комбинации V_t и V_n по правилам данного языка;
- S – стартовый символ.

В контексте FaceXML множеству V_t соответствуют:

- теги *<E>id</E>*;
- их атрибуты *<E attr="id" />*;
- их значения *id*.

Наиболее важными терминальными символами V_t для FaceXML являются:

<activity/>,<view/>, <model/>, <interaction/>, <activator/>, <text_editor/>, <text_outputer/>, <ranger/>, <list/>, <selector/>, <text_model/>, <list_model/>, <command/>, <name/>, <event_listener_map/>, <processor/>, <map_item/>, <action_event/>, <model_ref/>, id, enable, disable, occurrence и т.д.

Продукциями являются экземпляры FaceXML-объектов, или теги *<E1/>*, *<E2/>*,.. с конкретными значениями атрибутов и их комбинации. *S* является стартовым символом:

S = HA

H= <?xml version="1.0" encoding="UTF-8"?>…

A = <activity…> Activity </activity>

при этом, в FaceXML-тексте имеются следующие секции: *A-Activity, V-View, M-Model, I-Interaction*.

Таким образом, действие, описанное с помощью FaceXML имеет следующий вид:

```
<activity>
    <view>
        /*описание представления*/
    </view>
    <model>
        /*описание данных*/
    </model>
    <interaction>
        /*описание поведения*/
    </interaction>
</activity>
```

Описание *View* состоит из описаний элементов графического интерфейса пользователя.

```
<view>
<<ui_element> id="..." <parameters>>
    <event_listener_map>
        <map_item>
            <<event_type>>
<<event_subtype> occurrence = "concrete_happening"/>
            </<event_type>>
            <listener_ref>ID_312</ listener_ref>
        </map_item>
        ...
    </ event_listener_map >
    <model_ref>ID_211</model_ref>
 </<ui_element> >
...
</view>
```

<ui_element> является отображением одного элемента графического интерфейса пользователя.

Иерархия элементов графического интерфейса пользователя, изображена на рис.3.

В описание элемента входят также ссылка на модель данных – <model_ref/>, - которые отображаются данным элементом, и карта, сопоставляющая события, генерируемые данным элементом и ссылки на обработчики этих событий – <listener_ref/>. При выполнении бизнес-процесса среда выполнения связывает элементы с соответствующими моделями данных, а также перенаправляет возникшие события нужным обработчикам.

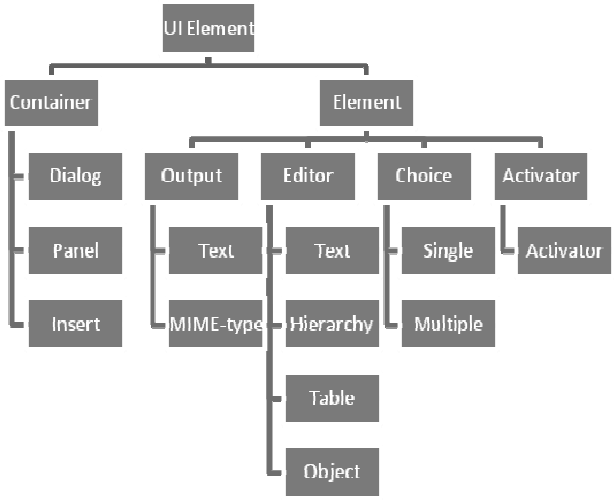

Рис.3. Иерархия элементов GUI.

Описание данных помещается в секции <model/> и состоит из описания отдельных элементов модели данных.

```
<model>
 <<data_model_type> id="...">
     <<data_1>>some_data</<data_1>>
     ...
 </<data_model_type>>
 ...
</model>
```

<data_model_type> – это тип модели данных.

Возможные типы модели данных:

- текстовые данные;
- иерархические данные;
- табличные данные;
- списочные данные;
- дата;
- MIME тип.

Тип модели данных может состоять с других типов данных, которые описываются с помощью тэгов <data_1>...<data_n>. Например, иерархический тип имеет следующую структуру:

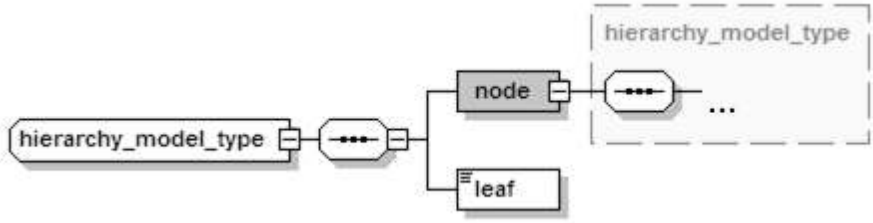

Рис. 4 Пример структуры иерархического типа данных.

Каждый элемент графического интерфейса может отображать несколько типов модели данных, в свою очередь одна модель данных может быть отображена несколькими видами элементов графического интерфейса. Например, списочная модель данных, может быть отображена как елементом Single Choice, так и Multiple Choice. В это же время, Text Editor, может служить редактором, как текстовых данных так и даты.

Описание обработчиков событий находится в секции <interaction/>, который состоит из множества описаний отдельных обработчиков.

```
<interaction>
 <listener id="..." type="ClientCommand|FlowCommand">
    <parameters>
       <parameter>
          <name>par_name</name>
          <element>element_ref</element>
          <property>property_name</property>
       </parameter>
       ...
    </ parameters>
    <processor>
       <operator_1>
       ...
    </processor>
 </listener>
...
</interaction>
```

Каждый обработчик содержит список параметров, необходимых при выполнении обработки (<parameters/>). Под параметрами понимаются значения свойств элементов графического интерфейса или модели данных. Также обработчик содержит непосредственно список инструкций, согласно которым производится обработка (<processor/>). Инструкции описываются с помощью FaceXML Processing Language.

3. FaceXML Processing Language

FaceXML Processing Language представляет собой процедурный язык программирования, предназначенный для описания полностью автоматизируемых действий и обработчиков событий для FaceXML. Все инструкции данного языка записываются в виде XML-тэгов.

Инструкции языка FPL бывают двух видов: операторы и выражения. Операторы - это самостоятельные составляющие программы, написанной с помощью FPL. Программа представляет собой множество операторов, записанных в определенной последовательности. Оператор может включать в себя другие операторы и выражения. Выражение – это инструкция FPL, предназначенная для вычисления какого-либо значения или извлечения значения свойства с контекста бизнес-процесса.

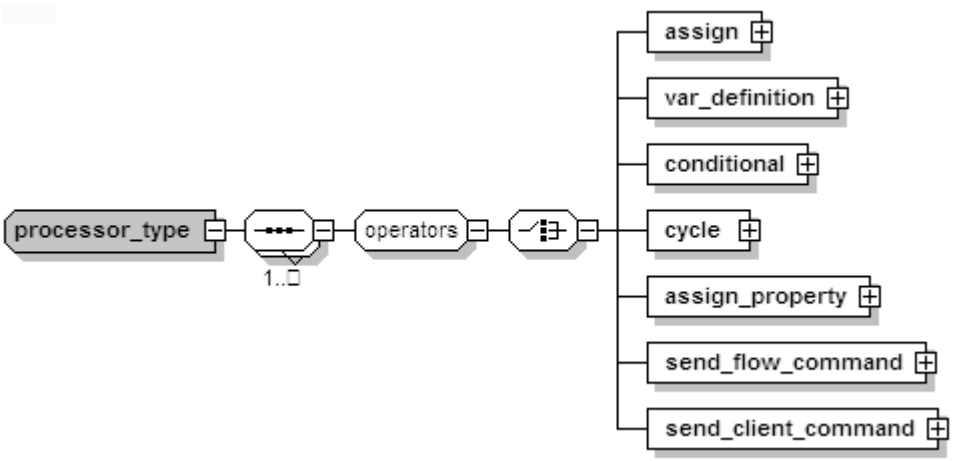

Рис. 5 Список операторов языка FPL.

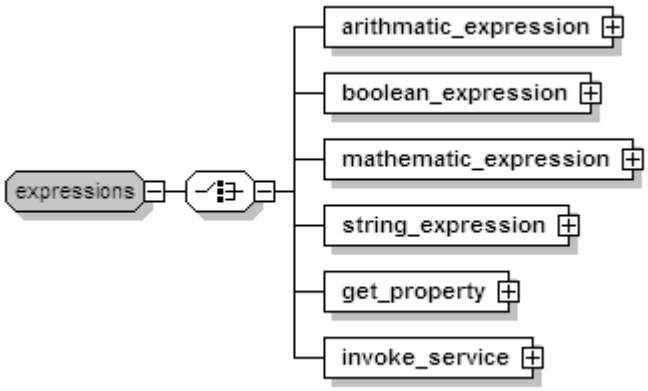

Рис. 6 Список выражений языка FPL.

Пример оператора, записанного с помощью FPL.

```
<assign>
    <variable>
        current_ns_state
    </variable>
    <boolean_expression>
        <operand_1>current_ns_state</operand_1>
        <operation>NOT</operation>
    </boolean_expression>
</assign>
```

Приведенный оператор присваивания, присваивает переменной *current_ns_state* значение логического выражения *not,* примененного к этой же переменой.

Таким образом, наличие расширенного описания бизнес-процесса, которое включает модели бизнес-процесса, внешнего вида графического интерфейса, поведения интерфейса и stub-services (заглушки сервисов), позволяет

моделировать поведение распределенной системы на самых ранних этапах ее разработки.

Выводы

В целях уменьшения количества ошибок на этапе бизнес -моделирования жизненного цикла разработки программного обеспечения распределенных систем предложен инструментарий, позволяющий исполнять бизнес - модели распределенных систем и по результатам моделирования их поведения вносить изменения и корректировки.

Предложенный подход обладает еще одним преимуществом. Поскольку, описание производится на независимом от платформы исполнения XML-подобном языке, элементы языка не являются привязанными к какой-либо платформе, можно говорить о независимости данного описания от платформы исполнения. Это позволяет при наличии адаптера исполнять бизнес-модель на любой целевой платформе.

ЛИТЕРАТУРА

1. Э. Таненбаум, М. Ван Стен. Распределенные системы. Принципы и парадигмы. – изд. «Питер», 2003 г. 880 стр. ISBN: 5-272-00053-6.
2. Хандхаузен Р. Знакомство с Microsoft Visual Studio 2005 Team System – СПб: Питер, 2006.
3. L.S. Globa, Prof., Dr.Sci.Tech CASE Tools for Distributed IT-System Accounting Multithreading, Polish J. of Environ. Stud ", Vol. 16, No. 5B (2007), 135-140
4. L.S. Globa, Prof., Dr.Sci.Tech “The integrated environment for design, programming, testing and reengineering of the distributed information resources”, CriMiCo 2006.

Надійшла 01.04.2009.