INTEGRATION OF FUNCTIONAL AND INTERFACE REQUIREMENTS OF AN WEB BASED SOFTWARE: A VDM BASED FORMAL APPROACH

Souvik Sengupta¹, Ranjan Dasgupta²
¹Bengal Institute of Technology, Kolkata, West Bengal, India
²National Institute of Technical Teachers’ Training and Research, Kolkata, West Bengal, India
¹mesouvik@hotmail.com, ²ranjandasgupta@hotmail.com

ABSTRACT
Analysing user requirements properly is the key to any software development process. In contemporary web based software developments the user interface requirements are predominant among the non functional requirements. The end-users’ involvement in interface requirements has made it challenging for the designers to map these interface requirements with functional requirements. As a consequence we often use different specification languages for different types of requirements but then interface requirements, if handled separately May lead to traceability problem for the developers, especially in web based application development.

In order to make requirements analysis both comprehensible and rigorous, formal requirements specification methods are very important. In this paper we have tried to describe how interface requirements can be specified in a formal way in connection with the functional requirements for a better understanding in designing. The Vienna Development Method Specification Language (VDM-SL) being a well established formalizing tool for user requirements specification, we propose some add-ons to this tool to help in formalizing these types of requirements. We have also used this technique in a case study to strengthen our proposal.

KEY WORDS

1. Introduction
Any Software product inevitably fails if there is a lack of understanding in any of the requirements by the designing team because then a correct implementation of an incorrect design does not make any sense. As testing is usually done at the later stage of the Software Development Life Cycle (SDLC), often on the coding, it is already very late to realize the mistake. A formal specification of the requirements not only ensures correct understanding but also helps in testing the design against the requirement very early in the development process. VDM-SL is one of the well established tools for formalizing the user requirements based on mathematical algebraic expressions. The popularity of internet has made the users so convenient with the web based software that the stakeholders these days are not confined within their functional need, instead they also advise (even demand) on the presentation part of the software, that is how the user interface should look like. The problem is that the end users, generally being unaware of technical constraints, sometimes demand for some interface requirements which are very hard to map with the functional requirements. These types of requirements are therefore required to be described in a more formal way for easier understanding and mapping with the corresponding functional requirement(s).

In this paper we have explored the VDM-SL approach of formal modelling of the user requirements and it is observed that the complex event driven structure of the web based applications are very hard to formally describe using the existing features of the language. As the interface requirements can not be specified independent of the functional requirements, issues like hyperlink navigation and sequencing, client-server architecture, stateless nature of HTTP, message passing etc. come as the main confronts against the formal modelling of such compound requirements. One solution to this problem can be specifying these two types of requirements in two different ways but then there is always a problem in mapping these two types and even harder to formally describe.

In this paper we have introduced a new modelling technique which incorporates the existing features of VDM-SL and additionally introduce some new data types, methods and rule based logic to handle these types of challenges. It also gives liberty to the Requirement Analysts to specify the compound user requirements that integrate functional as well as interface requirements.
2. Related works

One of the biggest challenges in the software industry is to ensure that a software product meets all user specifications. Hence Requirement Specification of software systems has gained immense importance in the present competitive world of developing software more quickly, more efficiently and more accurately [1]. So in order to reduce the development time, to enhance the maintenance, and to involve all stakeholders, the Model-driven Software Development (MDSD) paradigm has been used [3]. On the other hand work has been done on establishing and enriching traditional approaches of formal specification like Z and VDM [8, 9, 10]. In [6] a framework is developed for requirement tracing using UML for change tracking and influence analysis. Pohl et al [5] have described an approach based on scenarios and meta-models to bridge requirements and architectures. In [4] an approach is taken in developing the semantics and tool support for VDM, and applying in industry to identify achievements and challenges in providing lightweight but effective formal methods. New techniques for specifying and testing software requirements and design which emerge from integrating the three specification techniques namely informal specification technique (natural language), semi-formal specification technique (graphical models, e.g. entity relationship diagram, data flow diagram, and data structure diagram), and formal specification technique (e.g. Z, VDM, OBJ) are proposed in [7]. Asmaa Alsumait et al [2] proposed a framework called SECURE (Scenario and Use-Case based Requirement Engineering) which is based on an enriched version of UCM (Use-Case Maps) model. The user interface requirements are classified into three categories namely presentation dimension, task dimension and dialog dimension. It is observed that the UCM’s core notations are not sufficient to describe the inherent complexity of the interface requirements. In a similar work [11] Egbert Schlungbaum classified model based UI into three categories namely: a) domain model b) task model c) presentation model. The Board for Software Standardization and Control (BSSC), of the European Space Agency who produce advisory material for software developers conforming to ESA’s Software Engineering Standards, [12] has discussed on some limitations of traditional VDM-SL, which are the main confronts in specifying interface requirements. It is seen that VDM-SL is one of the most popular formal languages used in software development, its mathematical approach allows unambiguous specification and development of high integrity software [1]. In this paper some extended notation and feature set are introduced to model requirements of a web based software application.

3. Requirements in web based software

Web Application Development these days has requirements engineering, a phase that comes before design and programming, to play a more important role that determines the success of the software. These requirements are not limited to functional requirements but also non functional requirements like quality, technical and interface requirements play major roles. A functional requirement can be specified by identifying the state at which the data is to be input to the system, its input data domain, the output data domain, and the type of processing to be carried on the input data to obtain the output data. Where as the interface requirements fit into the input and output part of the functionalities where the user intervention is required to start and user interpretation of result is required to complete the functionalities. Like any other software applications, Web-based applications too have certain characteristics which act as complex factors in these applications. A Web-based application allows the information processing functions to be initiated remotely from a client (browser) and executed partly on a Web server, application server and/or database server. Issues like navigational requirements, presentational requirements, adaptive requirements, transactional requirements, are the salient features in web requirement engineering [13]. Moreover Web applications also have features that are not present in stand alone, client-server and distributed software systems. These include session control, cookies, the stateless aspect of HTTP, and security issues. In addition to that effective web based software design requires attention to usability issues like Scalability, Visual Design, and Interactivity etc.

Considering all these issues, in this paper we have tried to propose a technique which helps towards formalization of such challenging requirements using VDM-SL. We have introduced some new components in the specification language which can partially cover the above said challenges.

4. VDM based formalization

VDM-SL stands for "The Vienna Development Method-Specification Language" is a technique for the formal specification of Software systems. VDM-SL consists of a mathematical model built from simple data types like sets, lists and mappings, along with operations which change the state of the model. In this section we provide an overview of VDM-SL and selection V deals with the proposed add-ons.

Building blocks of VDM-SL

- Data Types
- Functions and Operations
- Set
- Composite Objects and Invariants
- Map
- Sequence

4.1 Data Types

The type of data that a variable represents. The primitive data types are as followed

N: natural number
N1: natural number \( \geq 1 \)
R: Real,
Z: Integer,
B: Boolean,
Token: structure less tokens,
Char: character

User Defined types:

Light::<RED>|<GREEN>|<YELLOW>

Composite Data Types: Composite object type associates *more than one* type with an object in VDM-SL

\[
\begin{align*}
\text{Time} & : \text{::} \\
\text{hour} & : \text{N} \\
\text{Minute} & : \text{N} \\
\text{second} & : \text{N}
\end{align*}
\]

4.2 Specifying the state
State refers to the permanent data that must be stored by the system, and which can be accessed by means of operations. The state is specified by declaring variables, known as state variables.

Example:

state anysystem of

var1: Z

end

4.3 Specifying the operations
Each operation specified in VDM-SL as follows:
i) operation header ii) external clause iii) precondition iv) post condition

Example:

increment () -------------- (i)

ext rw var1:Z --------------- (ii)

pre var1<10 ------------- (iii)

post var1=var1+1 ----------- (iv)

4.4 Specifying function
A function is a set of assignments from one set to another. The main difference with operation is that it cannot directly access the state variables. Functions are used inside the operations.

Example :

\[
\begin{align*}
\text{add} (x : \text{R}, y : \text{R}) z : \text{N} \\
\text{pre} & \text{ true} \\
\text{post} & z = x + y
\end{align*}
\]

4.5 Declaring sets
A set is an unordered collection of objects in which repetition is not significant

variableName: ElementType

Example : Day = <MON> | <TUE> | <WED> | <THU> | <FRI> | <SAT> | <SUN>

importantDays : Day-set

aNumber: N –set

4.6 Declaring sequences
A sequence is an ordered collection of objects; In a sequence, repetitions are significant.

\[
s = [a, d, f, a, d, c] \\
[a, d, f] \neq [a, f, d]
\]

4.7 Specifying a state invariant
we can incorporate a restriction of the state into the specification by creating a **global** constraint, known as a **state invariant**. Invariant must be satisfied under any state of the system.

\[
\text{inv mk-IncubatorMonitor(t) } \Delta \text{MIN } \_ \_ \text{MAX}
\]

4.8 Maps
Computing systems often involve relating two types of value together. The first set is referred to as the **domain**, the second is referred to as the **range**.

users : user-name \( \rightarrow \) Password

The **domain** operator \((\text{dom})\) returns a set that contains domain of all the maps.

If \( m1 = \{a \mapsto 1, b \mapsto 2, c \mapsto 2, d \mapsto 3, e \mapsto 4\} \) then \( \text{dom} m1=\{a, b, c, d, e\} \)

4.9 A standard template for VDM-SL specifications

**types**

\( \text{SomeType} = \ldots \)

**values**

\( \text{constantName} : \text{ConstantType} = \text{someValue} \)

**state**\( \text{SystemName of attribute1 : Type} \)

\( : \)

\( \text{attributen : Type} \)

\( \text{inv mk-SystemName(i1; Type, ..., i_n; Type) } \Delta \text{Expression(i1, ..., i_n)} \)

\( \text{init mk-SystemName(i1; Type, ..., i_n; Type) } \Delta \text{Expression(i1, ..., i_n)} \)

**end**

**functions**

specification of functions .....  

**operations**

specification of operations .....  

5. Proposed add-ons to VDM

5.1 New data types:
We have observed some new data types are required in order to meet the modelling criteria of the interface requirements in web based software. The following features influenced our model :
The webpage is a container for all the web interface controls
- The hyperlink is the navigational path between two web pages
- User can input data by textbox, radio button, dropdown box, checkbox
- Event associated with these controls are traceable at client side
- Submit button is used to send clients information to the server.

<table>
<thead>
<tr>
<th>Data Types</th>
<th>Symbol</th>
<th>Declaration</th>
<th>Initialization</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web-Page</td>
<td>WP</td>
<td>p:WP</td>
<td>Cannot be initialized using ‘=’ operator</td>
<td>It uses + operator to include other controls see table 5.1</td>
</tr>
<tr>
<td>Hyperlink</td>
<td>HL</td>
<td>l: HL</td>
<td>l = “Kolkata”</td>
<td>Text of the link is visible as “kolkata”</td>
</tr>
<tr>
<td>Text Box</td>
<td>TB</td>
<td>b: TB</td>
<td>b= “Hello Kolkata”</td>
<td>Textbox now contain “Hello Kolkata”</td>
</tr>
<tr>
<td>Check Box</td>
<td>CB</td>
<td>c: CB</td>
<td>c= “checked”</td>
<td>Check box is ticked, the other value allowed is “unchecked”</td>
</tr>
<tr>
<td>Radio Button</td>
<td>RB</td>
<td>r: RB</td>
<td>r= “on”</td>
<td>Radio button is selected, the other value allowed is “off”</td>
</tr>
<tr>
<td>Drop down Box</td>
<td>DB</td>
<td>d: DB</td>
<td>No write operation</td>
<td></td>
</tr>
<tr>
<td>Push Button</td>
<td>PB</td>
<td>b: PB</td>
<td>b = “OK”</td>
<td>The label on the button is “OK”</td>
</tr>
<tr>
<td>Submit Button</td>
<td>SB</td>
<td>s: SB</td>
<td>s= “Submit Data”</td>
<td>The label on the submit button is “Submit Data”</td>
</tr>
</tbody>
</table>

Table 5.1 shows details of these data types and their use.

We can declare variables of these data types in usual VDM-SL way like:

L1: HL means L1 is hyperlink type variable

OR T1: TB means T1 is textbox type variable

5.2 Other than these we will introduce two more special data types

| ActiveControl   | AC which is enumeration of < WP | HL | TB | CB | RB | DB | PB | SB > |
|-----------------|---------------------------------|
| ActiveEvent     | AE which is enumeration of <load| submit| redirect| click| select| check| change> |

The speciality is that we can not instantiate (declaration and initialization of variable) these two types directly instead they are instantiated at runtime by the system and passed to intervention function [sec 5.3] as parameter. However we can always declare variable of these types.

5.3 Intervention function
It is invoked whenever there is an intervention in the system. This can be client side events like clicking button, submitting, selecting from the dropdown list etc. or server side events like redirecting, session expired etc. The keyword intv is used to define an intervention function. It always takes two parameters, the first one is a variable of ActiveControl type and the second parameter is of the ActiveEvent type. The values are set by the environment based on the intervention type. For instance if submit button is clicked then the values are SB and Click respectively.

intv <system>{ ActiveControl, ActiveEvent } {} |

5.4 Event tracer functions
The event tracer function can work only within the intervention function. It also takes two parameters, the first one is a variable of ActiveControl type and the second parameter is of the ActiveEvent type. Then it returns the name of the control that has fired that event. We use the symbol É for event method. For instance if the submit button named s1 is clicked and the values passed are SB and Click respectively, then É(SB,click) returns the value “s1” as a String.

Example:

B1 : PB
intv SomeSystem(X: ActiveControl, Y : ActiveEvent)
{
  if É(X,Y) = “S1” then
   // do something
  end if
}
### 5.5 Proposed Operators and their use:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Use</th>
<th>Return type</th>
<th>Meaning</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navigate Page</td>
<td><img src="image1" alt="Symbol" /></td>
<td>1.HL p:WP 1</td>
<td>HL</td>
<td>Link 1 leads to page p</td>
<td>Link should be included in a WP to work</td>
</tr>
<tr>
<td>Navigate Place</td>
<td><img src="image2" alt="Symbol" /></td>
<td>1.HL p:WP 1</td>
<td>HL</td>
<td>Link 1 leads to a particular part of page p</td>
<td></td>
</tr>
<tr>
<td>Navigate New window</td>
<td><img src="image3" alt="Symbol" /></td>
<td>1.HL p:WP 1</td>
<td>HL</td>
<td>Link 1 opens a page p in a new window</td>
<td></td>
</tr>
<tr>
<td>Navigate Frame</td>
<td><img src="image4" alt="Symbol" /></td>
<td>1.HL p:WP 1</td>
<td>HL</td>
<td>Link 1 opens a page p in a different frame</td>
<td></td>
</tr>
<tr>
<td>Download</td>
<td><img src="image5" alt="Symbol" /></td>
<td>1.HL f:TOKEN 1</td>
<td>TOKEN</td>
<td>Link 1 leads to downloading the content f</td>
<td></td>
</tr>
<tr>
<td>Include</td>
<td><img src="image6" alt="Symbol" /></td>
<td>P:WP x:TB p:WP 1</td>
<td>WP</td>
<td>Textbox x is included inside the webpage p</td>
<td>The RHS can be any component</td>
</tr>
<tr>
<td>Pass Parameter</td>
<td><img src="image7" alt="Symbol" /></td>
<td>x:WP y:WP 1</td>
<td>WP</td>
<td>Webpage x pass parameter(s) to the webpage y</td>
<td>among HL, TB, CB, DB, PB, SB</td>
</tr>
<tr>
<td>Retrieve Value</td>
<td><img src="image8" alt="Symbol" /></td>
<td>p:WP r:RE p:RE 1</td>
<td>String</td>
<td>Webpage p retrieves the value of the component r which is passed to it</td>
<td>The RHS can be any component</td>
</tr>
</tbody>
</table>

Table 5.3: Symbol and meaning of the operators.

### 6. A case study

Consider the following requirement specification:

*In a web-based system, the first page will ask the user for userid and password. After authentication, if it is a valid entry then the home page will be displayed with a welcome message otherwise the login page will display error message and will ask to re-enter the login details.*

#### 6.1 Listing requirements

- **R1.** The login page will be displayed with a message to enter userid and password against two textboxes.

- **R2.** The given values should be authenticated once submit button is pressed

- **R3.** The home page will be displayed with a welcome message for a valid authentication

- **R4.** The login page will be displayed with an error message and asking to re-enter userid and password, in case of invalid authentication

Next we will try to formally specify the above requirements using the conventional VDM-SL then we have analysed the shortcoming of the existing model and how these are taken care in our proposed model.

#### 6.2 Formal specification using conventional VDM types

*UserName = String*

*Password = String*

*Signal = <Welcome>|< Re Enter>*

*state SecuritySys of*

*authorized : Name → Password*

*inv mk-SecuritySys(a,i) Δ i ⊆ dom a*

*Login (nameIn : UserName, passwordIn : Password ) msg : Signal*

*Ext rd authorized : Name → Password*

*Pre TRUE*
Post \((\text{authorized}(\text{nameIn}) = \text{passwordIn} \land \text{msg} = \text{<Welcome>})\)

\[ \lor (\text{nameIn} \notin \text{dom authorised} \lor \text{authorized}(\text{nameIn}) \neq \text{passwordIn}) \land \text{msg} = \text{<Re Enter>}) \]

### 6.3 Limitations of the conventional specification

The limitation in the above specification is that it cannot address issues related to the interface requirements. For instance:

I) In the above requirement it is specified that the user should enter user-id and password in the text boxes which means in the web page there should be two textboxes, one for each entry. As it is already specified in the requirement, we cannot leave it on designer’s choice. Interestingly a design from the above VDM can allow userid and/or password to be picked up from a dropdown list which is against the requirements

II) There are also presentation requirements like there should be two separate webpages for login and home page. But as it is not specified in the VDM-SL, one can club the login interface with in the home page by placing a login panel at any corner of the page, which is against the requirement.

III) The error message will be visible on the login page only after any fail in authentication. Screening the error message to the user can alternatively be done by means of showing the message in the second page and then redirected back to the first page.

So these are the issues that can lead to ambiguity for a designer as we cannot specify them formally using the conventional VDM-SL, which we will try to do with our proposed model in the next section.

### 6.4 Specification in the proposed model

**types**

\(\text{SomeType} = \ldots\)

**values**

\(\text{constantName} : \text{ConstantType} = \text{someValue}\)

**state** \text{UserAuthentication} of

\(\text{authorized : Name} \xrightarrow{m} \text{Password}\)

login: WP

Home : WP

Uid : TB

Pwd : TB

Sub : SB

**init mk-** \text{UserAuthentication}

\((\text{lg:WP,Hm:WP,u:TB,p:TB,s:SB}) \Delta\)

\[\text{lgn} = \text{lg} \mid + \text{disp(“Enter your id :”)} \text{ [R1]}\]

\[\text{lgn} = \text{lg} \mid + \text{disp(“Enter your password :”) [R1]}\]

\[\text{lgn} = \text{lg} \mid + \text{p [R3]}\]

\[\text{lgn} = \text{lg} \mid + \text{s [R4]}\]

end operations

**intv** \text{UserAuthentication} \((X: \text{ActiveControl}, Y: \text{ActiveEvent})\)

**ext** \text{rw login,Home}

**pre** \(Y=\text{clicked} \land \tilde{E}(X,Y) = “S1” \text{ [R2]}\)

**post**

\((\text{Home =login} \to \text{Home} \land \text{authorized (Home login) = Home login} \land \text{Home = Home login} \land \text{disp(“welcome”)}) \lor \text{authorized (Home login) } \notin \text{dom authorised} \lor \text{authorized (Home login) } \neq \text{Home login}) \land \text{login = Home login} \land \text{login = login login} \land \text{disp(“Wrong! Try Again”)}) \text{ [R4]}\)

### 7. Advantage

The formal specification using our model is advantageous than one described in sec. 6.1 in the sense that now it is less abstract to be interpreted differently by different designers. As the specification is made with help of interfacing components like webpage, textbox, submit button, link etc. the interface requirements associated with functional requirements can be addressed in a better way.

In the above specification all the functional requirements can be easily mapped with the interface requirement. Moreover as this formal specification is based on VDM model so it is possible to verify the correctness, completeness and consistency checking of the specification. We can test these requirements against the design of the software which encourages greater rigor in the early stages of software development life cycle.

### 8. Conclusion

The primary objective of this paper is to address the challenge of formal modelling of interface requirements in a way which can be easily traced against functional requirements. In this paper we have introduced some add-ons to the VDM-SL which is particularly suited for specifying web-based software requirements. We have
observed that the specification proposed using our model have good readability, well designed structuring mechanism, and precise semantics.

The formal basis of our method facilitates the better analysis of specifications so that the issues of interface requirements along with the functional requirements can be dealt more efficiently. Another aspect of this work is the traceability of the design improves significantly as the model is less abstract in specifying the functional requirements and also takes care of the essential features of web based applications while dealing with the interface requirements.

However in this presentation, we have not been able to explore some important issues that arise to consider challenges in technical requirements (like stateless nature of HTTP, session, and cookies) we require more add-on components for the specification language, which is the future scope of this work.

References


[12] ESA Board for Software Standardisation and Control, “Guide to the software requirements definition Phase” ESA PSS-05-03 Issue 1