Requirements Metamodeling, Modeling, Verification and Validation

Ing. Ferenčík František
Supervisor: Zdenek Havlice, Ph.D., Assoc.Prof.
ferencik.f@gmail.com
zdenek.havlice@tuke.sk

ABSTRACT

This paper points out the possibility of facilitating and streamlining processes during the development of a software system, starting with the design, through its implementation to maintenance, including verification and validation of requirements. The article focuses on a specific category of software, which is the area of web applications based on the MVC architecture. In the first stage, a method was developed together with a methodology for modeling critical parts of such systems. This method uses the principles of DSM domain-specific modeling. In the second stage, the CASE tool MVCStudio was developed for this method and was built on the WebGME generic modeling platform. For this purpose, a DSML language of domain-specific modeling was created and implemented in the WebGME environment using metamodeling. In the third stage, several extensions of the CASE tool were implemented to support automation and more efficient development processes.

WEB APPLICATION MODELING METHOD

Critical parts

In order to create a method for modeling a specific category of systems, it is necessary to identify its critical parts, and these parts must be supported by models. Based on the principles of the MVC design pattern as well as experience with the development of such web applications, we have identified, that critical parts are data layer, GUI graphical user interface, control of data flow and application state management.

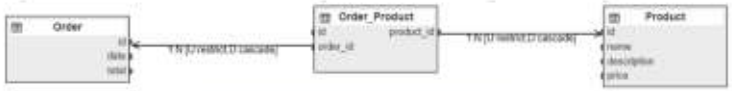
CREATING A METHOD

DATA LAYER MODELING

It is necessary to model data entities persisted in the database and the relationships between them. The inspiration for the model diagram is the physical ERD entityrelational diagram.

The model diagram contains the following elements:
- Entity - database table (user, product, order, etc.);
- Entity attribute - table column; characterizes the type of information;
- Index - a structure that improves the speed of operations
- Primary key - uniquely identifies the record in the table;
- Foreign key - primary key reference; identifies relationships between entities;
- Relation - defines the connection of entities;
- Cardinality - number of entity occurrences in this relationship;
- Relational handling restrictions - enlargement compared to the ERD; RI protection.

The classical ERD diagram has been extended by defining the rules for ensuring RI reference integrity. These rules can be defined for U-update and D-delete operations. RI security methods are restrictive, null setting and cascading. In figure a ModelDiagram containing 3 data entities with defined attributes and relationships is displayed. It is a simple scheme for linking orders and products, which ensures that one order can contain multiple products and at the same time one product can be assigned to multiple orders.



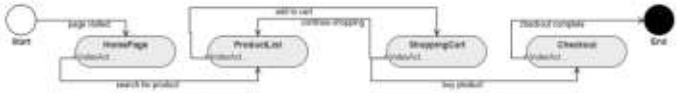
Sample ModelDiagram database layer

STATE CONTROL MODELING

The user switches between different screens on which he performs certain actions. We perceive the set of these screens and transitions as states and transitions between them. The inspiration for the ControllerDiagram is the STD diagram of state transitions.

The ControllerDiagram contains the following components:
- Action – state analogy in STD. Processing the URL of the request triggers a specific action, and thus specific state is entered within the application when loading the URL.
- Start and end terminators – initial or final state of the application.
- Controller – combines several actions, such as displaying the product in the e-shop.
- Transition – oriented state connection. It is possible to add a transition invocation condition, for example, after authentication redirection to the list of orders. If the transition does not lead to a specific action but only to the controller, then the action marked as an index is automatically considered to be the destination of the transition.

In figure is a state diagram of a simple e-commerce. After coming to the website, the user is on the home page, and after using the search, he gets to the list of products, where he can add the product to the cart. He can return to the product list or continue to process the order. After equipping it, he finishes working in the application.



ControllerDiagram - state diagram of a simple e-commerce

MODELING OF SYSTEM MODULES

ControllerDiagram diagrams can group controllers within larger logical units - modules. In figure we see the connection of application modules. The Front module contains the primary e-commerce business logic services and the Admin module contains order fulfillment services.

Kompozícia modulov pomocou ControllerDiagram

USER INTERFACE MODELING

This ViewDiagram is not just about the GUI. After processing the request, it is necessary to return data that can be in various formats. The most used is the HTML code that the browser interprets into the GUI. There are other data transfer formats that are used to communicate with other programs. It is, for example, the JSON format. The file can also be the

The components of the ViewDiagram are:
- Format – response type definition - HTML, JSON, File and others.
- HTML element – base unit containing information or enabling user input to be recorded. For example, title, form, text input, link, etc.
- JsonObject – used for JSON format. It contains data in a structured text form of the

In figure we can see the product screen. The page contains the name, description, product price, available quantity in stock and a form for adding to the basket containing another model of the form containing field for entering the quantity and a button for adding to the basket.

Fig. 4: ViewDiagram – product page GUI screen

DATA FLOW MODELING

In order for the origin of the processed data to be clear and where the results of operations are stored or displayed, it is necessary to model this process in the DataFlowDiagram.

It is inspired by the DFD data flow diagram and contains the following components:
- Process – any process that changes data and produces output based on inputs, such as the process of creating an order with sending notification emails.
- Input / output – process input and output data, respectively; for example, email for

The DataFlowDiagram in figure contains a process that retrieves from the product EAN database and API access data. Subsequently, with this data, the API calls the vendor, which returns the number of pieces in stock. This number is finally assigned to the corresponding element of the GUI template.

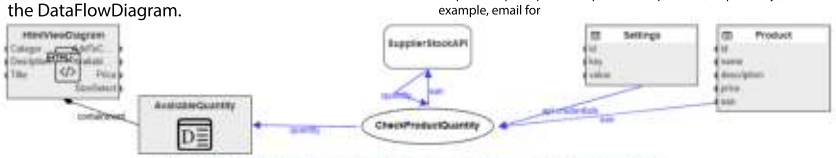


Fig. 5: DataFlowDiagram zistenia množstva produktu na sklade

DEVELOPMENT OF METHODOLOGY

There is no general and universal procedure for designing a system and compiling its individual models. However, we recommend finding inspiration in our approach and adapting the individual steps to one´s needs according to the current situation:

1. Based on the defined system requirements, it is necessary to design and model the data layer. It is not a problem if certain errors and inconsistencies appear in the original models, because these models can be modified at any time later. It will also help to detect design errors in a timely manner. Thus, even if the structure of the data is not yet completely clear at the beginning, it is necessary to model at least what is known and other elements will be continuously supplemented during the design and compilation of other models.

2. The main system functions are also hidden in the system requirements definitions. These functions need to be identified and divided into interrelated groups. It is a good practice to focus these groups on larger functional units and create modules from them (e.g. administration, user profile, front module, etc.).

3. In this step, for each module, we re-examine the functions of the system and again try to divide them into interrelated smaller groups. We perceive these groups as the states of the application in which the user is currently located (e.g. home screen, product list, shopping cart, etc.) Within these states, certain bases can also be defined, which we refer to as actions. They are important in supporting the partial functions provided by the given controller (e.g. checking the condition of the product in stock).

4. In the next step, we need to design and model the GUI and possibly other types of views (HTML template, JSON string or file). Each type of view contains its specific components, from which it is necessary to compile this view according to the defined requirements. For example, when modeling a GUI template, components such as a form, various input fields, links, buttons, and many other elements are available.

5. In this last step, we have modeled and have made available everything necessary to create a data flow diagram. It is advisable to have the previous diagrams already created, because it uses them to a large extent. Therefore, we recommend modeling this type of diagram completely at the end. The data stream can connect the data layer with the view layer through processes in the controllers.

Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic