Widget Transactions Dynamic Processing Architecture Development

Viktoriia Merlak, Daryna Hrebeniuk, Galina Cherneva

***Abstract* — The article is devoted to the development of a widget's dynamic transaction processing architecture. An object is a hierarchical widget in a user interface organization. The subject is the process of the dynamic processing of widget transactions. The purpose of the article is to develop a dynamic transaction processing architecture) for a hierarchical widget. The article discusses the construction of a complex structured user interface in applications that are built on the basis of event-oriented programming. The article discusses the use of hierarchical widgets in companies such as Microsoft and Google, as well as existing developments related to hierarchical widgets. The main components of the dynamic processing of widget transactions, the main objects of the dynamic model are determined. A hierarchy of objects of the dynamic model of transaction processing of the widget is proposed. To interact with the dynamic model and its graphical representation, a special graphical notation is proposed.**

Dynamic processing of widget transactions is a new approach that has its advantages and disadvantages, the class of tasks being solved. The proposed architecture combines both work with unstructured data and the use of a hierarchical data model.

***Index Terms* — dynamic processing, hierarchical widget, transaction.**

I. Introduction

Modern web applications are designed for a large number of users of the distributed data processing system on the Internet. They include the back end (application server and database server) and the front end (many users' web browsers).

Most often, the interface has a hierarchical organization of the image, when other fragments are placed inside one fragment and this hierarchy depends on the current state. The hierarchy is preserved in the resource allocation of the widget [1].

The widget technology is quite new, but its essence is quite simple and is of great importance and popularity in the development of modern startups and projects. The concept of hierarchical widgets is that the states of the dynamic model provide widget elements that correspond to fragments of the image on the user's screen. They set the method and parameters for the formation of the resulting HTML code, as well as links to parent widgets that combine widget elements into a hierarchy [2-3]. Web applications can have a complex user interface that reflects various information from the back-end database, and can accept control commands and data to be entered into the database. Therefore, designing a user interface is a laborious step in creating a web application. For each widget of the interface, you need to solve a difficult task of interaction with the user: generate HTML code (which sets the structure and content of the image on the screen, is sent to the user's browser along with the CSS code, sets the parameters for the appearance of the image) and JavaScript code (sets activity on client side), accept the data entered by the user, check their correctness, process, save in the server database, change the current state to move to a new cycle of interaction with the user [4-5].

The purpose of the article is to develop a dynamic transaction processing architecture for a hierarchical widget.

II. Statement Of The Problem

In the source [6], the author talks about how Flutter (an open-source SDK for creating mobile applications from Google) works using a hierarchical structure of widgets. To be able to generate the pixels that make up the image displayed on the device, Flutter needs to know in detail all the small parts that make up the screen, and in order to identify all the parts it needs to know the structure and content of all widgets.

The article [7] from Microsoft talks about the Common Data Service, which also uses hierarchical widgets. It is possible to select the necessary controls in accordance with the user's needs and explore the hierarchy by expanding/collapsing the widget tree.

The source [8] deals with the results of using hierarchical widgets when building a web application on the example of one of the states of a dynamic model, discusses in detail and compares technologies for organizing testing of user input, as well as generating messages to the user without using and using hierarchical widgets. The article shown that the use of widgets simplifies the organization of XSL transformations and reduces the cost of additional programming of user data validation functions. The analysis shown a high need for widgets in applications that provide for the introduction and verification of user data.

In this article [2] within the framework of the concept of hierarchical widgets, algorithmic support for new elements of the dynamic HSM model has been developed: a controller element for controlling user-entered receiver elements for placing the entered data into the DOM buffer and fixing the detected errors; transition elements, the activity of which depends on the presence / absence of detected errors.

V. Merlak is with the National Aerospace University «Kharkiv Aviation Institute», 17, Chkalova str., Kharkiv, Ukraine
(e-mail: v.merlak@csn.khai.edu).

D. Hrebeniuk is with the National Technical University «Kharkiv Polytechnic Institute», 2, Kyrpychova str., Kharkiv, Ukraine
(e-mail: darina.gg1@gmail.com).

G. Cherneva is with the "Todor Kableshkov" University of Transport, 158 Geo Milev Str. 1574 Sofia, Bulgaria (e-mail: cherneva@vtu.bg).

https://doi.org/10.47978/TUS.2021.71.01.002

The developed algorithms are implemented as part of the HSMI dynamic model interpreter, which operates on the PHP platform, and are ready for practical use when creating web applications.

III. Dynamic Processing Architecture Development

Components of dynamic processing of widget transactions:

– SBSL (System Behavioural Scripting Library) – a library of dynamic models, which contains a set of different scenarios of system behaviour SBS (System Behaviour Scenarios)

– CS (Current state) - current state memory, which stores information about the current state of dynamic models;

– DMD (Dynamic Model Documents) - a set of XML documents corresponding to the states of the dynamic model;

– DMF (Dynamic Model Functions) – a library of data processing functions corresponding to the states of the dynamic model;

– ACF (Archive of Completed Functions) – an archive of executed functions for processing transactions of the widget;

– SBSI (System Behavioral Scripting Interpreter) - a dynamic model interpreter, which, in response to a request R, generates response A by processing a specific dynamic SBS model from the SBSL library based on the current state stored in CS, processing documents with a DMD, and performing DMF data processing functions.

If the dynamic transaction processing is used in a web application, then a set of parameters is used as an input request R, it is acquired together with a URL (for example, parameters of a display form in POST mode), and as a result of A, an HTML code is sent to the client in response. The request parameters indicate the dynamic model being processed and the need to change its current states. Fig. 1 shows how the dynamic transaction processing of the widget works.

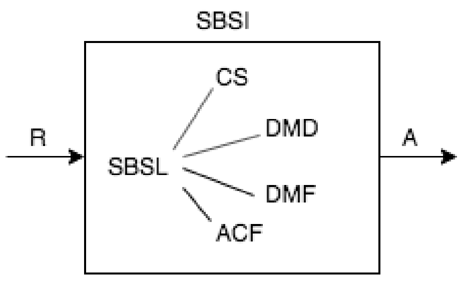

Fig. 1. Workflow of dynamic transaction processing widget

The dynamic SBS model is a finite state transition graph hierarchy. The states are loaded with elements that define the connection with certain data with the DMD, functions over the DMF. With graph arcs - state transitions - associated predicates that determine their activity provide a change in the current state. The dynamic model has a graphical representation (SBS diagram) and an equivalent textual representation with XML-like syntax.

The main objects of the dynamic model are the following components: Model, Sub-model, State, Jump. The root of the tree must be the Model element. The Sub-model element is a child of the Model element; it can contain one or more children of the State type. An element of the State type contains one or more elements of the Jump type that characterize either a transition from this state to another

TABLE I
NOTATIONS OF DYNAMIC MODEL ELEMENTS

Graphic image	Element name	Appointment
sub	Sub-model	A container that contains a set of states of one level of the hierarchy, of which only one is current
sta	State	A container of elements (sub models, transitions, dives, etc.) that are processed sequentially if this state is current
js	JumpS type transition	Specifies the transition from one state to another within the sub model
jsm	JumpSM type transition	Specifies the transition to the internal sub model
act	Action	Specifies the DMF function that is executed in the current state
var	Variable	Sets a variable, saves the state, the value of which is and can change as long as the state is current
con	Container	Container for sources and sinks - specifies the preparation and processing of XML state data in a DOM object
s	Source	Sets a DOM element to an XML document in a DMD or DOM object to load into a real DOM object
r	Receiver	Sets a DOM element to output data to a DMD or output stream by transforming the content of the DOM object
doc	Document	Gives access to process the file
b	Button	Initiates some event
arc	Archive	Saves executed transaction processing functions of the widget

(transition of the JumpS type), or a transition between sub models (transition of the JumpSM type). The Jump element also contains some overhead information and an Aim attribute that sets the direction of the transition. Fig. 2 graphically depicts the hierarchy of objects in the dynamic model, which was described above.

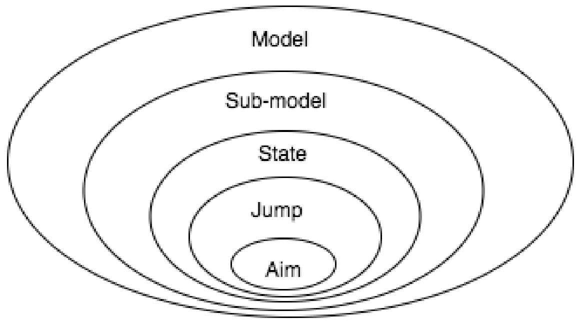

Fig. 2. The hierarchy of objects of the dynamic model of transaction processing of the widget

To interact with the dynamic model and its graphical representation, a special graphical notation is proposed with the following rules of use:

– dynamic model tree nodes are rectangular signs (sometimes with an invisible border) with the name of the corresponding element inside;

– connecting line that defines a parent-child pair (it joins the parent sign to the right or bottom, and to the child sign to the left or top);

– the order of child nodes is set hierarchically (top to bottom, right to left);

– the order of the nodes that are attached to the parent at one point is determined by the height of the branch of the connecting lines relative to the point of union: branches from above preceded branches from below.

The elements listed above in Table 1 have many built-in attributes that allow fine tuning, external transformation files such as CSS, Javascript, and others.

Fig. 3 below provides an example of a simple web application model based on dynamic transaction processing of a widget.

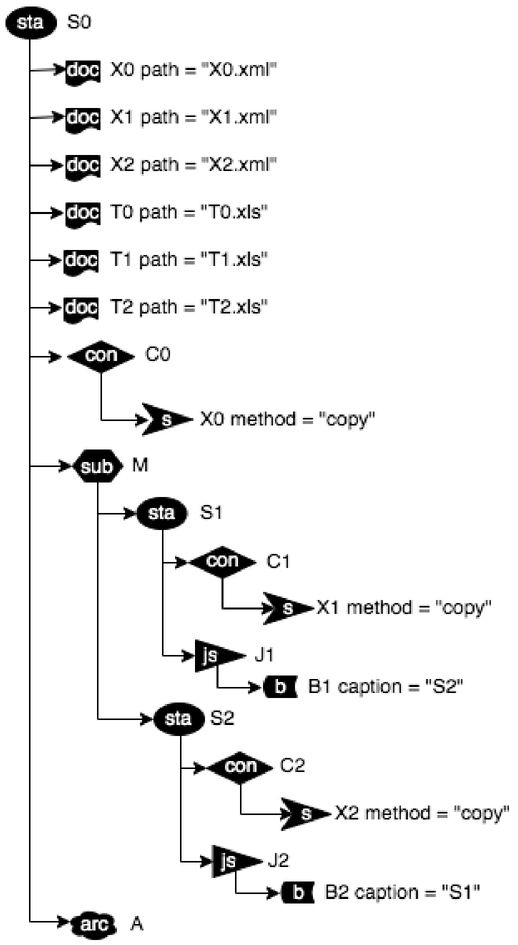

Fig. 3. An example of a simple model of a web application based on dynamic transaction processing of a widget.

The root state S0 is the starting point of the dive and contains the following elements as child states: three XML document definitions with DMD (X1, X2, and X3), three XSL transformation styles (T1, T2, and T3), one SBSL sub model (M) and an archive A that stores the executed transaction processing functions of the widget. XML documents are an attribute set of data that can be modified in HTML with the appropriate XSL styles.

In the root state S0 in the dynamic DOM object C0, the data of the XML file X0 is processed using the XSL transformation table T0 and the data is output to the data area of this state. Sub model M defines two substates for the parent state S0: S1 and S2, containing the data X1 and X2, respectively. They are processed in the same way as the root state. J2 and J1 are transitions from one state to another within the sub model, which provide a change in the current state upon actuation of buttons B2 and B1, respectively.

The figure below shows a schematic representation of the page structure built by the dynamic transaction processing interpreter of the widget when processing the dynamic model using the example above (Fig. 4). First, the user is in state S1, which is a substate of the root state S0 (Fig. 4a). After pressing the button B1 ("Go to Text 2"), the transition J1 is triggered and the application switches to the state S2 (Fig. 4b). It should be noted that only the text field changes, the title does not change.

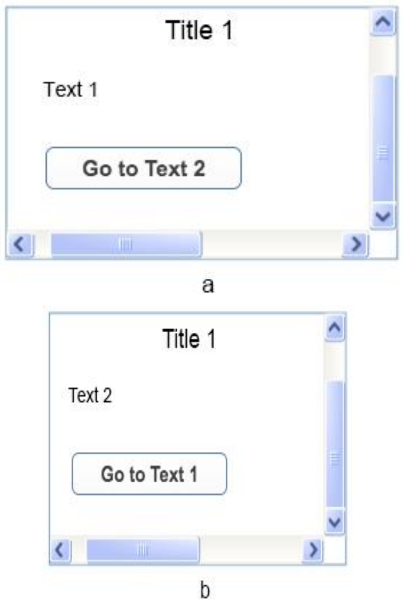

Fig. 4. An example of a simple model of a web application based on dynamic transaction processing of a widget (a - state S1, b - state S2)

Any situation implemented by dynamic transaction processing of the widget can be represented as a generated set of events. From this it follows that dynamic transaction processing is closely related to the ideas of event-driven programming, but, despite the similarities, there are differences in technological terms. Dynamic transaction processing focuses on server-side scripting and the SBSL interpreter is built as a PHP script. This is due to the desire to automate program design, the use of a model-based approach and the desire to make a business process model without unnecessary complexity through the use of asynchronous data processing.

In dynamic transaction processing, it is possible to use client scripts in contexts of different situations. For this, the scenario for each situation is saved in a separate XML file. But flexibility in scripting comes at the price of a lack of design capability in the process of creating a program model.

The use of server-side scripting makes the application more private and secure from the end user, unlike client applications.

Based on the foregoing, we can conclude that the proposed approach to dynamic processing of widget transactions does not call for replacing existing solutions, but to help efficiently use resources. When designing a program, you should choose the right tools that will simplify the implementation of the idea.

Class of tasks for which the dynamic transaction processing widget approach is suitable:

– work with data based on documents and document collections;

– work with complex structured and complexly connected data;

– development of applications that require storing the history of operations (archive of system states);

– automation of business processes that have a situational focus (testing, decision making).

IV. Conclusion

This approach is proposed for building a complex-structured interface in Internet applications, where the interface is formed according to the "top-down" principle, that is, the structure of the model should also be flexibly scalable.

Data storage and processing is performed in XML format, which requires active use of DOM technology, because it does not impose restrictions on the structure of the document. Also, any document can be represented as a tree of nodes connected by parent-child relationships. Dynamic transaction processing of a widget is a new approach that has its own advantages and disadvantages, a class of tasks to be solved. The use of dynamic transaction processing combines both working with unstructured data and using a hierarchical data model.

References

[1] N. Kuchuk, V. Merlak, "The method of redistributing resources of the university e-learning system on a hyperconvergent platform," *Radioelectronic and computer systems*, 2019, *no. 1*, pp. *91-98*, DOI. 10.32620/reks.2019.1.10.

[2] V. Kanashin, V. Mironov, "Hierarchical widgets: user data control algorithms in web applications on the basis of situation-oriented databases," *Bulletin of UGATU*, vol. *18*, *no. 1 (62)*, pp. *204-213*, Oct. 2018.

[3] V. Merlak, I. Zykov, H. Molchanov, "Situatio-oriented approach for designing vines" *Control, Navigation and Communication Systems*, vol. *4*, *no. 50*, pp. *91-98*,2018, https://doi.org/10.26906/SUNZ.2018.4.125

[4] M. Marinov, "Four-Dimensional Encoding of Character Sequences and Evaluation of their Similarities and Differences," *Proceedings of the Technical University of Sofia*, vol. *70*, *no.* 2, pp. *11-20*, 2020, https://doi.org/10.47978/TUS.2020.70.02.008

[5] G. Cherneva, "Fractal Models for Approximation of Random Processes," *Proceedings of the Technical University of Sofia*, vol. *67*, *no.* 2, pp. *171-176*, 2017.

[6] D. Boelens, "How Flutter works," *available at* https://habr.com/en/post/476018/#ierarhicheskaya-struktura-vidzhetov.

[7] Microsoft Documentation, "Visualize hierarchical data with model-driven apps," available at https://docs.microsoft.com/en-us/powerapps/maker/data-platform/visualize-hierarchical-data.

[8] V. Kanashin, V. Mironov, "Hierarchical widgets: experience of use in the web application on the basis of situation-oriented database," *Bulletin of UGATU*, vol. *18*, *no. 2 (63)*, pp. *185-196*, Oct. 2018.