Programming tool and estimation of energy consumption for an Autonomous Device of Experimentation in Aquaculture

Rómel A. Cabo, Carlos J.Sosa and Juan A. Montiel
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria (ULPGC)
Las Palmas de Gran Canaria, Spain

Abstract—**This paper presents the design and implementation of a multiplatform graphical user interface (GUI) oriented to estimate the energy consumption and program an Autonomous Experimentation Device (DAE) in Aquiculture. The DAE executes research experiments based on experimental primitives. The main key of the GUI is the use of graphical features like create, eliminate, drag and drop those experimental primitives. This feature provides a Human Interface for non-engineers like biologists, geneticists or dieticians in aquaculture. Based on the experiment specified by the user and the primitive database, the GUI tool estimates the energy consumption and evaluate the usage of the batteries resources (time horizon) of the Autonomous Experimentation Device (DAE). To materialize the objectives were used elements of the methodology of Rational Unified Process (RUP) and the UML (Unified Modeling Language), the Eclipse integrated development environment, the Java programming language and the ZK framework. Finally, this work is framed in the Work Package 8 of the European project AquaExcel2020.**

Keywords: DAE, RUP, UML, Eclipse, Java, ZK, MVVM, GUI.

I. Introduction

In the current context within the field of aquaculture, technology is being invested with the implementation of electronic systems to improve the production of various species as an industrial interest [1]. The DAE allows a user to schedule the timed execution of a set of measurement primitives and basic mathematical operations on these measurements to form fish research experiments [2]. Our mission has been to develop software that allows managing and planning such experiments implemented by users in aquaculture with the DAE [3].

It was determined to select the main guidelines of the RUP (Unified Rational Process) methodology to meet the needs in the software development process. This is a process guided by the architecture, use cases and it worked in conjunction with the end user, who will use the application [4]- [6]. The methodology will be complemented with UML (Unified Model Language) for the description of the different modeling diagrams, formalizing the process with its documentation, since it provides the artifacts and models for it [6]- [8].

For the selection of the language and framework to be used in the development of the GUI, it was considered that the developer had basic knowledge about the Java programming language [9]- [13]. The ZK framework was chosen because it fulfilled with the functionalities, tools and resources required to obtain the desired final product. In addition, with ZK it was not necessary to have knowledge about the JavaScript web language, nor about Ajax, which were not mastered and in this way the development time could be optimized [14].

II. Software design Pattern

The application development process has been governed by a modern architectural design pattern called MVVM (Model-View-ViewModel) as part of software engineering (Fig 1). This is provided and documented by the ZK framework, enabling a better reuse and maintenance of the code [14].

Model: Contains the classes with the data and information of the application, as well as the business rules [14].

View: It is the user interface layer, defined in the .zul file with the ZK components. The interaction of a user with components triggers events that are sent to the ViewModel [14].

ViewModel: It is responsible of exposing the data of the Model to the View and for providing the actions and the business logic necessary for the interaction of the users from the View. It is an abstraction of the View containing its state and behavior, but it must not have references to the components of the graphical user interface; the framework itself is responsible for the control of communication and synchronization between View and ViewModel [14].

The "Binder" is the element of the MVVM design pattern responsible for the synchronization of data between View and ViewModel, ensuring that any changes made to the components of the GUI of a View are automatically transferred to the ViewModel and vice versa. Application developers only must define the data link relationship between the UI component attribute and the target object, usually a ViewModel, by data link annotation expression [14].

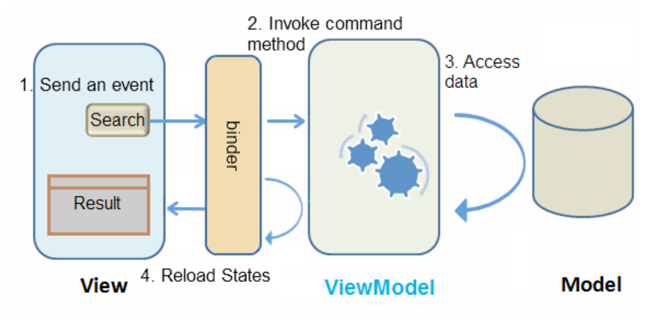


Fig 1. Design pattern flow MVVM [14]

III. System Arquitecture

The ZK´s Server + client Fusion architecture provides the synchronization of the states of the components between the browser and the server automatically and transparently to the application. The ZK client engine and the ZK update engine simplify the implementation working together with safety, efficiency and robustness. The ZK application runs on the server accessing back-end resources, assembling the user interface with components, shown in Fig 2. In addition, listen to the user's interactions to later manipulate the components and update the user interface. Optional client-side functionality can be added for greater interactivity, such as event handling, visual effects personalization, or even composition of the user interface without server-side encoding. ZK allows uninterrupted mergers that range from exclusively server-centric to customer-focused in a productive and flexible manner. The communication between layers is established with the HTTP protocol (Hypertext Transfer Protocol) through Ajax requests and responses (Asynchronous JavaScript And XML) [14]-[16].

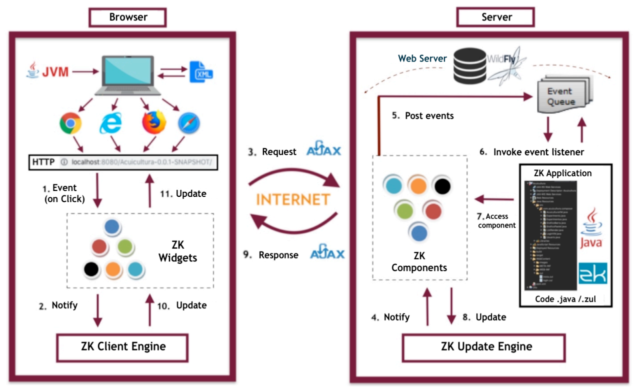


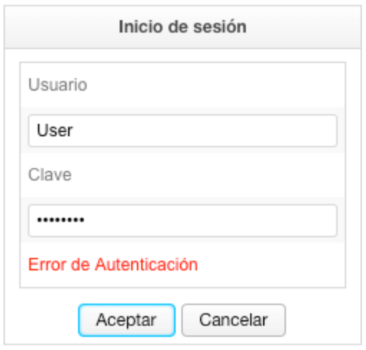
Fig 2. System architecture flow

The GUI objects are built with a widget and a component with a one-to-one relationship, for each widget a component is intended. Widgets are JavaScript objects with visual aspect that handle events that occur in the browser. These represent the UI objects with which the user interacts. A component is a Java object that has all the behavior of a UI object, it does not have a visual part, it is executed on the server and it is manipulated by a Java application [14].

IV. Results and Validation

The tool is developed for programming and estimating energy consumption for the use of the DAE at the Aquaculture environment. The user stories already implemented and functionalities of each modules are demonstrated. The secure access of the application and the correct functioning of all graphical components are checked at the beginning of the session. In addition, the acceptance and validation of the tool by end users is achieved.

The initial interface of the system is the login (Fig 3) where a user accesses the application depending on the type of user and its corresponding key. The software tool is accessible only by two types of user modules, Aquaculture User and Aquaculture Administrator. The user passwords are provided by the developer of the application, and only he as responsible can modify it with the due authorization and agreement with the client.



Inicio de sesión

Usuario: User

Clave: ••••••••

Error de Autenticación

Aceptar | Cancelar

Fig 3. Graphical login interface

Once logged in, the user will be redirected to (Fig 4) the Aquaculture Administrator or Aquaculture User module, where they can use the corresponding interfaces and functionalities for each module, related to the programming and management of aquaculture experiments based on the energy consumption of the autonomous experimentation device.

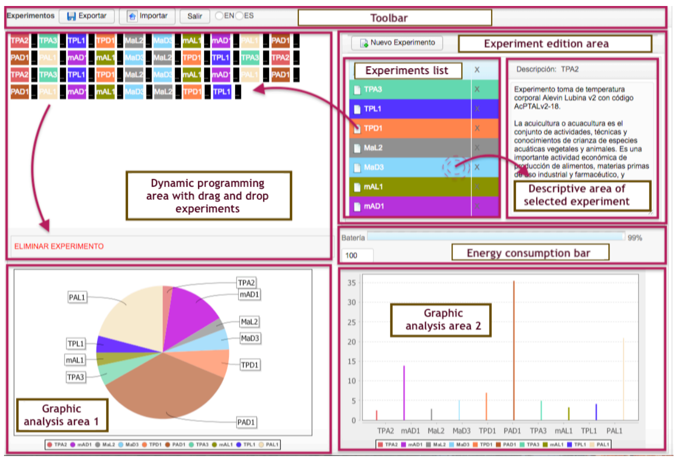
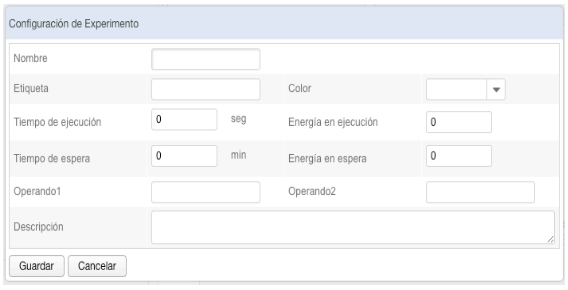


Fig 4. Graphical user interface for the aquaculture project

We can export a project that has been made in edition area of experiments by the Aquaculture Administrator user, obtaining a file with extension .xml in a certain location of the computer. Likewise, with a previously exported file, it could be imported into a new work project to reuse or edit it having access to each of its characteristics and properties in the experiment area of editing.

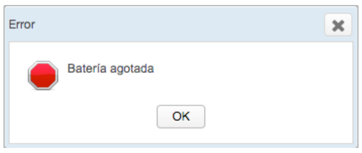
If we click on the "New experiment" button, we access to (Fig 5) the configuration interface or creation of new experiments. In this interface, we configure each experiment with its properties and characteristics.


Configuración de Experimento

Nombre			
Etiqueta		Color	
Tiempo de ejecución	0 seg	Energía en ejecución	0
Tiempo de espera	0 min	Energía en espera	0
Operando1		Operando2	
Descripción			

Fig 5. Experiment configuration interface

This tool allows us to add quantities of experiments with their consumption parameters, showing us the progress of consumption and autonomy limit of the device battery. Once the battery is exhausted, the software system notifies us with a message as shown below.


Error

Batería agotada

Fig 6. Total battery consumption alert

A set of tests have been developed to find errors and validate the software tool. This is a contribution to increase the study and research of sustainable aquaculture. It is a GUI that can be used by any kind of user.

V. CONCLUTIONS

The software application has been developed to allows users the graphically describe the experiments, according to the configuration of their parameters and timing.

The RUP is complemented with the UML as a methodology to achieve the necessary requirements. This served as a guide of, organization and model during the phases of the software development process.

An XML interface has been built allowing to abstract the implementation details of the primitives of measurement, computation and timing.

The GUI has been implemented to allows users to program the DAE parameters intuitively and with a minimum knowledge of algorithmic programming.

The algorithm has been constructed making it possible to estimate the energy consumption of the DAE and the execution time horizon of the experiments.

ACKNOWLEDGMENT

This work has been supported by the Institute for Applied Microelectronics (IUMA) and the University of Las Palmas de Gran Canaria (ULPGC).

REFERENCES

[1] “European research infrastructures (including e-Infrastructures)”, HORIZON 2020, Work Programe 2014 – 2015.

[2] “Selection and Breeding Programs in Aquaculture”, ISBN: 978-1-4020-3341-4, Springer, 2015.

[3] http://www.aquaexcel2020.eu/about/overview, last acces: 2019/01/07.

[4] Ahmad K. Shuja and Jochen Krebs, “IBM Rational Unified Process Reference and Certification Guide: Solutions Designer”, 2008.

[5] https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational, last acces: 2018/11/04.

[6] https://en.wikipedia.org/wiki/Rational_Software, last acces: 2018/12/15.

[7] http://www.uml.org/, last acces: 2018/12/14.

[8] https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado, last acces: 2018/12/14.

[9] Vartan Piroumian “Java Gui Development” August 1999.

[10] https://es.wikipedia.org/wiki/Interfaz_gráfica_de_usuario, last acces: 2018/08/04.

[11] https://java.com/es/, last acces: 2019/01/07.

[12] http://www.eclipse.org/, last acces: 2019/01/07.

[13] https://docs.oracle.com/javase/7/docs/api/, last acces: 2018/12/14.

[14] https://www.zkoss.org, last acces: 2019/01/08.

[15] Xavier Vilajosana Guillén, Leandro Navarro Moldes “Arquitectura de aplicaciones web”, Universidad Oberta de Catalunya, 2007.

[16] https://es.wikipedia.org/wiki/Cliente-servidor, last acces: 2018/12/14.