Consumer-Oriented Programming Application for Statistical Processing

Zvonimir Pavlic* , Tomislav Lugaric** and Sinisa Srbljic *
* University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia
Consumer Computing Laboratory, Zagreb, Croatia

**Laboratory for Underwater Systems and Technologies

Abstract - Consumer-oriented programming allows consumers that are not professional developers to create personalized software artifacts. Consumer-oriented applications are exposed to consumers through gadgets, small applications displayed in a web browser. Consumers choose a preferred set of gadgets and build personalized workflows on top of the gadgets by interacting them through a graphical user interface. Consumer's application is being built using Geppeto (*Gadget Parallel Programming Tool*), consumer-oriented framework for programming workflows of consumer's applications. Consumer-oriented applications have to satisfy two main requirements: ease of use and adequate set of functionalities. Furthermore, all functionalities of the given application have to be exposed as graphical user-interface elements of the gadgets, to provide simple way of gadget composition for consumers. Demands on consumer-oriented applications are explained through consumer-oriented application for statistical processing. Each gadget implements one statistical function. Consumers can generate their own gadgets as compositions of existing gadgets using Geppeto. The system implements functionalities of data fetching, statistical data processing, displaying the results of data processing as well as defining events and actions triggered by those events.

I. INTRODUCTION

Consumer-oriented programming researches technologies, methods and developing tools suitable for consumer's autonomous creation of personalized applied software [1]. Computer consumers are the largest group of computer users, which have no formal education in computer programming. The most common form of software artifacts suitable for consumer's usage is applied World Wide Web application. At present, consumer network applications are composed as complex sets of interconnected software components [2].

Gadgets, often called widgets, are small standalone applications displayed in a Web browser. The concept of gadgets was first introduced by the Commodore Company in 1985. Inside their environment called Intuition, gadgets were elements of the interface responsible for handling events generated by users and forwarding them to the processes running in background. Today the term gadget represents standalone platform independent application running inside gadget container, such as iGoogle. Gadgets are equipped with graphical user interface for the interaction with the background process, which is usually a Web based source.

Consumers select a set of gadgets relevant to their field of interest and build personalized data flows between gadgets by interconnecting their graphical user interfaces. In order to develop large personalized applications, consumers have to manually interact with multiple gadgets, which is impractical. Consumers can use Geppeto (*Gadget Parallel Programming Tool*), for automating gadget interconnection [2,3,4]. Geppeto is a consumer-oriented framework for programming workflows of consumer's applications, developed at the University of Zagreb, Faculty of Electrical Engineering and Computing (FER). Application is being built by using gadgets as building blocks. Geppeto is implemented as an extension of the Apache Shindig project, open-source gadget container and rendering server [5].

II. CONSUMER-ORIENTED PROGRAMMING APPLICATIONS

Using Geppeto as a programming tool allows consumers to build personalized applications out of gadgets. Consumer-oriented programming applications are exposed to consumers as sets of specified gadgets. A model of a consumer-oriented programming application is shown in Figure. 1 [2].

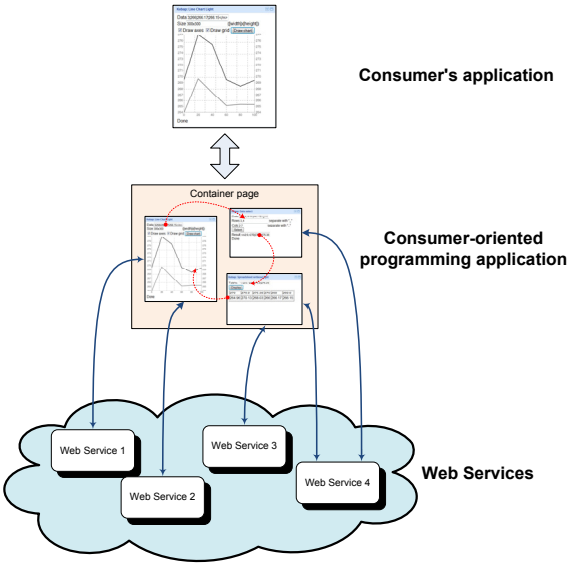

Fig. 1. A model of consumer-programming oriented application

Consumers build their applications by interconnecting gadgets into personalized workflows through gadget's graphical user interface. Gadgets are used as interfaces to lower level applications. Applications are developed using service-oriented architecture (SOA), and run on the hardware platforms [6].

III. REQUIREMETS ON CONSUMER-ORIENTED PROGRAMMING APPLICATIONS

Consumer-oriented programming applications are developed for a wide range of computer consumers. Lacking sufficient education, average consumers, unlike the professional programmers, do not understand the main concepts of programming crucial for developing new applications, such as objects, procedure calls and program loops. Consumers understand the level of abstraction of graphical user interfaces and data flows, which they constantly use in everyday interaction with the computer. Therefore, all functionalities of the given application have to be exposed as graphical user-interface elements of the gadgets, to provide simple way of gadget composition for consumers.

In order to develop personalized applications by interconnecting gadgets, a consumer has to learn how to use each gadget. Application components are required to have simple and concise graphical user interfaces that are easy to learn, understand and use.

Applications are built by combining multiple gadgets. Consumers must be given the opportunity to use only functionalities they consider necessary to build their applications. Gadgets should be functionally decomposed, so that each gadget implements single functionality. Granularity of the system must be well-balanced to match the consumer's level of abstraction.

In the consumer application, interconnected gadgets exchange data among themselves. Gadgets can be developed by professional programmers, as well as other consumers. All gadgets in a consumer-oriented programming application have to implement standardized data format for exchanging data between gadgets. Additionally, data format used in gadgets must be understandable to consumers, in order to support data-flow programming and usage of gadgets developed by other consumers as building blocks.

Data-flow programming in consumer-oriented programming applications is achieved by providing consumers with three groups of gadgets that support data flow: gadgets for providing data, gadgets for personalized data processing and gadgets for presentation of given results.

Consumers expect their applications to have functionalities similar to applications developed by professional programmers. Consumer-oriented programming applications should provide similar expressive capabilities as modern programming languages used to build professional applications. Moreover, consumers which are experts in their field of interest require domain-specific sets of gadgets in order to develop domain-specific applications.

Furthermore, consumer-oriented programming applications should have high extending possibilities and components reusability. Programmer of application can easily add new gadgets implementing new functionalities demanded by the consumers of that application.

In addition, consumers can add their personalized gadgets, developed as composition of gadgets, to consumer-oriented programming application. Other users can use these gadgets as building blocks for their own personalized gadgets. This feature reveals a great application development potential with consumers as authors of new applications [2].

IV. CONSUMER-ORIENTED PROGRAMMING APPLICATION FOR STATISTICAL PROCESSING

A consumer-oriented programming application for statistical processing is developed at University of Zagreb, Faculty of Electrical Engineering and Computing. This application is the result of collaboration with NOAA (National Oceanic and Atmospheric Administration, US Department of Commerce). Application provides statistical processing of oceanographic, climate and atmospheric data. The application is intended for consumers who are experts in fields of statistical analysis and meteorology, but have no programming skills.

Functionality of the consumer-oriented programming application for statistical processing is similar to functionality of programming language R [7]. R is a programming language and software environment for statistical computing and graphics. The R language has become a *de facto* standard among statisticians for the development of statistical software, and is widely used for statistical processing and data analysis. Consumers using this application can develop their own personalized applications with similar functionalities as the ones developed by professional programmers using the R programming language.

Each gadget within the application implements one statistical function. Graphical user interface of the gadget contains elements for data input, processed data output and buttons for initiating consumer's action. Graphical user interface of the gadget is shown in Figure. 2.

Figure 3 shows the classification of gadgets within the application by their functionality. Gadgets are divided into three basic groups: gadgets for fetching data, gadgets for processing data and gadgets for displaying results.

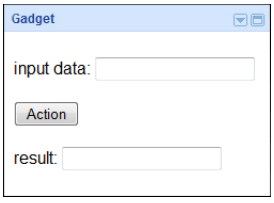

Fig. 2. Gadget's user interface

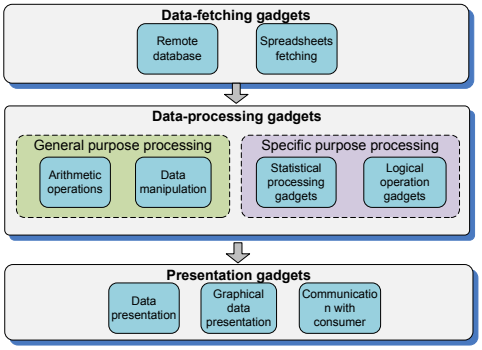

Fig. 3. Classification od gadgets

There are two sets of data fetching gadgets. First set of gadgets provides data from remote database ERDDAP (the Environmental Research Division's Data Access Program) [8]. ERDDAP is a collection of data servers that provides a simple, consistent way to download subsets of scientific datasets in common file formats and generates graphs or maps. There are gadgets for fetching data like sea surface temperature, concentration of chlorophyll in sea water, wind, COADS surface marine observations and sea currents. The second subset of data-fetching gadgets provides input from web-based spreadsheet editors, such as Google Spreadsheet. Data fetching gadgets also translate data into standardized data exchange format, which is implemented in all gadgets contained in this application. Data is displayed to consumers as tables, which are common-used data format in statistical processing. Data format is based on XML, and is suitable for transporting tables between the gadgets, which consumers use in their everyday interaction.

Data processing gadgets are divided to general purpose processing and specific purpose processing gadgets. General purpose processing gadgets are functionally independent from statistic data analysis, and can be used in other consumer-oriented programming applications, such as personal financial application. There are gadgets for basic arithmetic operations on data sets, as well as number of gadgets for data manipulation, like selection of rows and columns from table and table concatenation.

Specific-purpose gadgets contain gadgets for statistical processing, calculating arithmetic mean and arbitrary statistical regression on provided data set. Logical gadget is used for defining logical conditions in the consumer's applications, and therefore enables decision-making. Consumer can enter logical clauses defined by comparison of data values and combine them in logical conjunction or disjunction. Logical gadget can also be used in creating user events, later to be trigger for Geppeto "TriggerMe" gadget.

Presentation gadgets are used to display the results of data processing. Data can be displayed in table and graph form. There are gadgets for plotting different types of graphs: scatterplot, line-chart, scatterplot matrix, 3d-scatterplot, 3d-wireframe, time-series plot and plot of seasonal temperature normals. In addition, a special gadget is developed for communicating with consumer via consumer's e-mail address. Communication gadget allows consumers to design applications that continuously

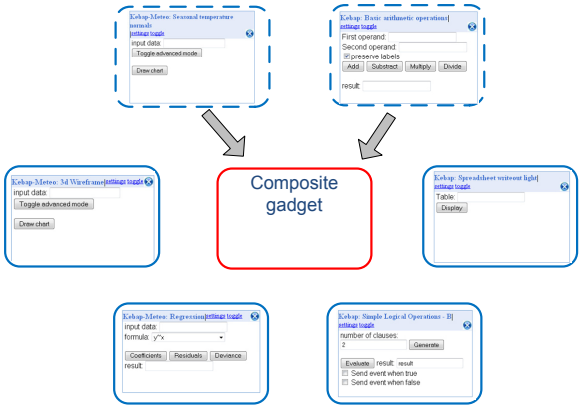

Fig. 4. Programming by composition

monitor the system and alert them if an event of interest happens.

V. PROGRAMMING IN GEPPETO

Consumes create their own personalized gadgets by combining previously developed gadgets into one composite gadget [2]. Composition of gadgets is shown in Figure. 4. Consumer chooses the set of gadgets from consumer-oriented programming application website. Chosen set of gadgets provides required set of functionalities for consumer application. Gadgets are loaded into Geppeto container by entering gadget's URL into container interface [3]. These gadgets are considered source gadgets for the composite gadget. After loading chosen gadgets, consumer adds programmable gadget to container.

Programming in Geppeto consists of two steps: defining user interface of consumer's composite gadget and defining the composite gadgets logic by building personalized data flow through selected gadgets.

Designing the user interface of a composite gadget is done by adding graphical user elements from one or more other gadgets that consumer has chosen as source gadgets. Adding elements is done via the right click menu which can be brought up when the mouse pointer is over a user interface element. To add an element to the composite gadget, the user right-clicks the element, selects "add" and the element is copied into the composite gadget. The right click menu also has the option of removing a user interface element.

When programming data flow and the logic behind the composite gadget, actions are specified using the right-click menu. Defining a sequence of actions is done by selecting "When clicked" action on an element. The sequence of actions defined by the "When clicked" option will be reproduced each time when the selected user-interface element is clicked. Right-click menu provides options for interconnecting source gadgets. Actions of source gadgets are defined by option "click" done on action buttons of source gadget. On input elements consumer can type in text. Communication between source gadgets is done by copying output of one

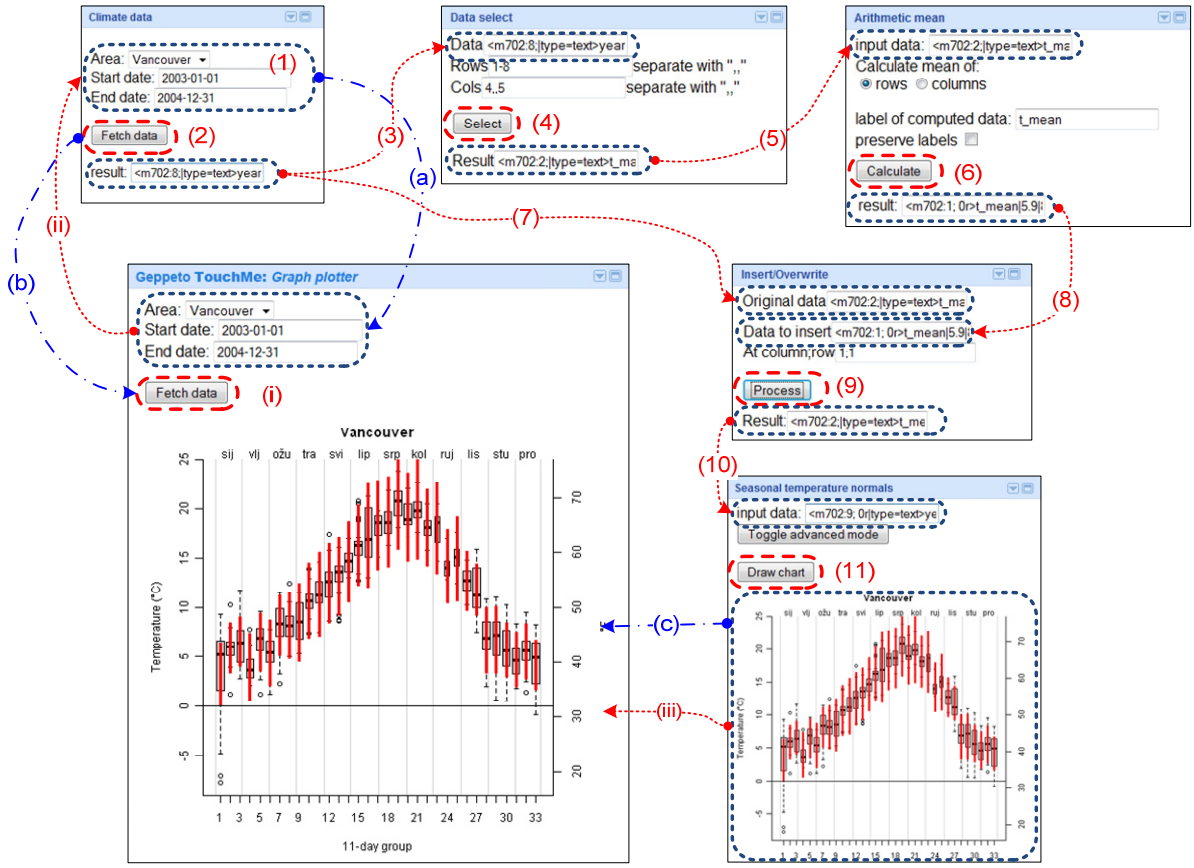

Fig. 5. Consumer application example

gadget and pasting it to input of another gadget. All sequences the user generates can be viewed and reorganized in a table which is stored together with the generated composite gadget. Table of actions can be later edited, and used to define concurrency of actions of the composite gadget data flow. In addition to gadgets programmable by gadget composition (TouchMe), the system also offers time programmable (TickMe) and event programmable (TriggerMe) gadgets.

VI. CONSUMER APPLICATION EXAMPLE

This section demonstrates the simple consumer-programmed application for statistical processing. The goal of this application is to plot a graph of seasonal temperature normals, which indicates the climate of given area. Figure 5 illustrates the workflow of this application.

Consumer wants to plot a graph based on data for selected area through given period of time.

In order to plot a seasonal temperature normals graph, "Seasonal temperature normals" gadget must be provided with data consisting of three sets of values: maximal temperatures, "t_max", minimal temperatures, "t_min", and mean temperatures, "t_mean". "Climate" gadget provides data only for minimal and maximal temperatures. Consumer has to compute the mean temperature for each date.

"Climate" gadget provides various meteorological data for selected area and specified period of time. Consumer selects the area and specifies the starting and the ending date of observation (1). Consumer clicks on the "Fetch data" button of "Climate" gadget in order to fetch data (2). After that, he copies provided data to "Data select" gadget (3).

In "Data Select" gadget, consumer enters parameters for selecting fourth and fifth column, where minimal and maximal temperatures are stored. By clicking "Select" button (4), consumer selects the temperatures columns and copies them into the "Arithmetic mean" gadget (5).

"Arithmetic mean" gadget can calculate arithmetic mean of rows or columns of provided data. In this case, arithmetic mean of rows must be calculated. Consumer also enters the label for calculated column of arithmetic means, "t_mean", in order to achieve data consistency with the original data. By clicking on the "Calculate" button, the arithmetic mean is calculated (6).

When the arithmetic mean of temperatures is calculated, it has to be re-attached to the original data, to provide all three sets of values for plotting. Combining data tables can be done by using the "Insert/Owerwrite" gadget. Consumer copies data from the "Climate" gadget to the "Original data" field of "Insert/Overwrite" gadget (7). Than, he copies data from "Arithmetic mean" to "Data to insert" field "Insert/Overwrite" gadget (8). Consumer enters position of insertion and by clicking the "Process" button combines two tables (9). Data is now ready to be sent to the "Seasonal temperature normals" gadget (10).

Consumer clicks on the "Draw chart" button in the "Seasonal temperature normals" to plot the graph (11).

To draw seasonal temperature normals graph for some other area or different time period, consumer would have to repeat previous actions and connect gadgets manually,

which is very impractical. To automate application workflow, consumer creates programmable composite "TouchMe" gadget called "Graph plotter" and defines gadget's GUI and behavior using Geppeto.

Graphical user interface (GUI) of the composite gadget is defined. Consumer adds input GUI elements from source gadget for data-fetching, called "Climate" using Geppeto drop-down menu. Added GUI elements are for selecting the observation area, starting date of observation and final date of observation (a). He defines the action element of composite gadget by adding "Fetch Data" button from "Climate" gadget (b). The output GUI element is defined by adding plotted graph from "Seasonal temperature normals" gadget to composite gadget (c).

To build workflow logic, consumer selects "Fetch Data" button at the composite gadget and defines set of actions after selecting "When Clicked" option in the Geppeto drop-down menu (i). Data flow of the application is started by copying input parameters of composite gadget to "Climate" gadget (ii). The rest of data-flow actions are defined as follows by copying outputs of one gadget to input of another one, as previously done manually. Control flow is programmed by clicking on the consumer action buttons on source gadgets interfaces in the same order as described before. Complete workflow of the consumer's application is defined by actions (2) – (11). Finally, the output of "Seasonal temperature normals" gadgets is copied to the composite gadget, so that the graph in composite gadget will be refreshed.

With every click on the button "Fetch data" in the composite gadget, programmed sequence of actions will be reproduced. Table of actions for consumer's applications is shown in the Table 1.

TABLE 1. TABLE OF ACTIONS

wait for click FetchData at GraphPlotter
select select at ClimateData
copy Startdate at GraphPlotter to Startdate at ClimateData
copy Enddate at GraphPlotter to Enddate at ClimateData
click FetchData ad ClimateData
copy result at ClimateData to Data at DataSelect
\"1-8\" => Rows@ DataSelect
\"4,,5\" => Cols@ DataSelect
click Select at DataSelect
copy Result at DataSelect to inputdata at ArithmeticMean
typein „t_mean" to label at ArithmeticMean
click Calculate at ArithmeticMean
copy result at ClimateData to Originaldata at Insert/Overwrite
copy result at ArithmeticMean to Datatoinsert at Insert/Overwrite
\"1;1\" => Atcolumnrow@Insert/Overwrite
click Process at Insert/Overwrite
result@Insert/Overwrite =>inputdata@SeasonalTemperatureNormals
click Drawchart@SeasonalTemperatureNormals
graph@ SeasonalTemperatureNormals => graph@GraphPlotter

VII. CONCLUSION

Consumer-oriented programming applications are exposed to consumers through gadgets, small applications displayed in a web browser. Consumers build their own applications using previously developed gadgets as building blocks. Program logic and data flow is defined by interconnecting gadgets graphical user interfaces, using Geppeto, web automation framework based on consumer-programmable gadgets.

Since consumers have no experience or formal education in programming, applications suitable for consumer-oriented programming must satisfy certain criteria. Those criteria are: ease of use, graphical user interface, standardized data format, functional decomposition, well-balanced granularity which suits consumer's level of abstraction, defined data-flow (data fetching, data processing, presentation of results), adequate capabilities of expression, extendibility and reusability.

Requirements on consumer-oriented programming applications are explained through application for statistical data processing, which satisfied those requirements.

VIII. ACKNOWLEGMENT

The authors acknowledge the support of the Ministry of Science, Education, and Sports of the Republic of Croatia through research project „Computing Environments for Ubiquitous Distributed Systems" (036-0362980-1921) as well as Google, Inc. for the Google Research Award project "End-User Tool for Gadget Composition". The authors also whish to thank Roy Mendelssohn from National Oceanic and Atmospheric Administration (NOAA) for his valuable help with consumer-oriented application for statistical processing. Furthermore, many thanks to Goran Narancic from University of Toronto, Miroslav Popovic and Dejan Skvorc from Faculty of Electrical Engineering and Computing at University of Zagreb for their help with preparing this manuscript.

IX. REFFERENCES

[1] Škvorc, D., Programiranje prilagođeno potrošaču, PhD thesis, University of Zagreb, Faculty of Electrical Engineering and Computing

[2] Srbljić S., Škvorc, D. Skrobo, D., Widget-Oriented Consumer Programming, Automatika 50(2009), 3-4, str 252-264

[3] Geppeto home page, http://www.ris.fer.hr/, 27.01.2011.

[4] Popović, M., Consumer Program Synchronisation. PhD thesis, University of Zagreb, Faculty of Electrical Engineering and Computing

[5] Apache Snindig, The Apache Software Foundation, http://shindig.apache.org/index.html, 27.01.2011.

[6] Newcomer, E., Lomow, G., Understandnig SOA with Web services, Addison Wesley Professional, 2004.

[7] The R Project for Statistical Computing, http://www.r-project.org/, 25.01.2011.

[8] ERDDAP home page, http:// coastwatch.pfeg.noaa.gov/erddap/index.html, 23.01.2011.