

**Front-end Development on Web 3.0 to access Web Service,  
Content and Future Internet things**Dhumal Monika<sup>1</sup>, Gaikwad Priya<sup>2</sup>, Jagtap Nivedita<sup>3</sup>, Jadhav Prachi<sup>4</sup>, S. S. Nimbalkar<sup>1-4</sup>Student, Department of computer, SVPM's COE, Malegaon (Bk), Baramati, Pune<sup>5</sup>Asst. Professor, Department of computer, SVPM's COE, Malegaon (Bk), Baramati, Pune

---

**Abstract** — The world is coming closer through use of Web. Many technologies are getting invented for the same. So, we represent a front-end that combines traditional services and resources that can be designed to fulfill the requirements of end users converting them into creators and consumers of service based application. The evolution of Web 3.0-based portals and user friendly interfaces has led to a major advance in service usability. However, existing web based service front-ends fall short of end-user expectations. Applications and information portals are still based on inflexible and unfriendly user interfaces. End users do not actually benefit from the advantages of modularity, flexibility and composition promoted by service orientation. Service frontends are not built using formal engineering methods they are constructed ad hoc without tools that could speed up time.

---

**Keywords** - Gadget Authoring Tools, Workspace Edition Tool, User Centric IDE, SOAP, Resource Catalogue, SOA, Web Services, Information Communication Technology, Really Simple Syndication

**I. INTRODUCTION**

The evolution of Web 3.0-based portals and user-friendly interfaces has led to a major advance in service usability. However, existing web-based service front-ends fall short of end user expectations [1]. Applications and information portals are still based on inflexible and unfriendly user interfaces. End users do not actually benefit from the advantages of modularity, flexibility and composition promoted by service orientation [2]. Service front-ends are not built using formal engineering methods they are constructed ad hoc without tools that could speed up time. Service-Oriented Architectures has gained a great deal of interest in the last few years [3]. Very less attention has been given to service front-ends, which we consider to be a fundamental part of SOAs [4]. SOAs increase property reuse, lower the integration charges and increase the strength with which industrials respond to latest demands. Users in many Information and Communication Technology (ICT) areas have started using languages and tools (like spreadsheets) that do not require programming skills to develop their own software solutions. This approach is known as End User Development (EUD) till up-to-day, however, mainstream development and research into SOA has concentrated mainly on scalability and middleware, automating service proposition and service engineering using Business Process Management technologies.

**II. EXISTING SYSTEM**

This is the world of technology many application are upcoming now-a-days. The previously available applications fall short in satisfying end users needs[1]. The architecture empowers users to develop applications based on off-the-shelf and published components through a gradual visual composition process without having to use programming languages.

The architecture is able to wrap any resource or web service for inclusion within the building blocks available to end users. The applications adapt, at design and run time, to the users' navigation context. The architecture enables dataflow based on visual recommendations, supported by the semantics of the data handled by each component. Applications including screen transitions can be created. Screen front-ends are considered as compassable and interlinkable elements with dataflows that govern the transitions. Application events can be handled as if they were data sources, abstracting and making event-oriented programming understandable for end users[5].

For example, mainly focus is on providing visual elements in catalogue for generating Web APIs. They do not allow users to attach an operational back-end to such elements[7,8].

So, for satisfying maximum possible needs of users and user friendly front-end we propose the new system.

**III. PROPOSED SYSTEM**

The components which are necessary for implementing the architecture for the authoring phase are as given below :

**A. Gadget IDE platform:**

Gadget IDE platform is also known as authorware, it is a program that enables user to write multimedia applications

or hypertext. Authoring tools generally enable user to develop a final application entirely linking objects together, such as an illustration, a song, or a paragraph of text. By sequencing objects in proper order, and by mentioning the relationships of objects with each other, the users of authoring tool can develop useful and attractive graphics applications. Authoring tools needs fewer technical knowledge to use them and they are used exclusively for applications that exhibit a combination of audio, graphical, and textual data.

**B. Workspace Edition Platform:**

Workspace Editing Tool intended to design user tailored workspaces, like a mashup editor. Users can use this tool to visually design, reuse and share their workspaces by selecting, connecting and composing the most suitable gadgets for solving a domain problem. The ultimate goal is to create new, modular and anticipated service front-ends (instant applications) by combining gadgets.

**C. Declarative authoring language and gadget template:**

We have created a gadget template to assure the interoperability of the gadgets implemented and used with other catalogue gadgets. To do this, a template includes information on a gadget author, business logic and input and output interface. This mechanism sets up the interoperability of gadgets created by different manufacturers and tools.

**D. Catalogue:**

Catalogue or resource catalogue contains all the metadata about the different building blocks in the architecture (gadgets, screens, flows, workspaces, content delivery resources, application data resources, resource compositions, etc.). We are developing a resource catalogue to enable the workspace editing tool and gadget authoring tool to cooperate. The catalogue is also useful for exploiting external resources, services and components.

**IV. ALGORITHMIC STRATEGY**

**KNN classifier**

**1) Introduction:**

K nearest neighbors is a simple algorithm that accumulates all obtainable cases and identifies new cases on the basis of a resemblance measure (e.g., distance functions). KNN has been used in pattern recognition and statistical estimation previously in the beginning of 1970s as a non-parametric technique.

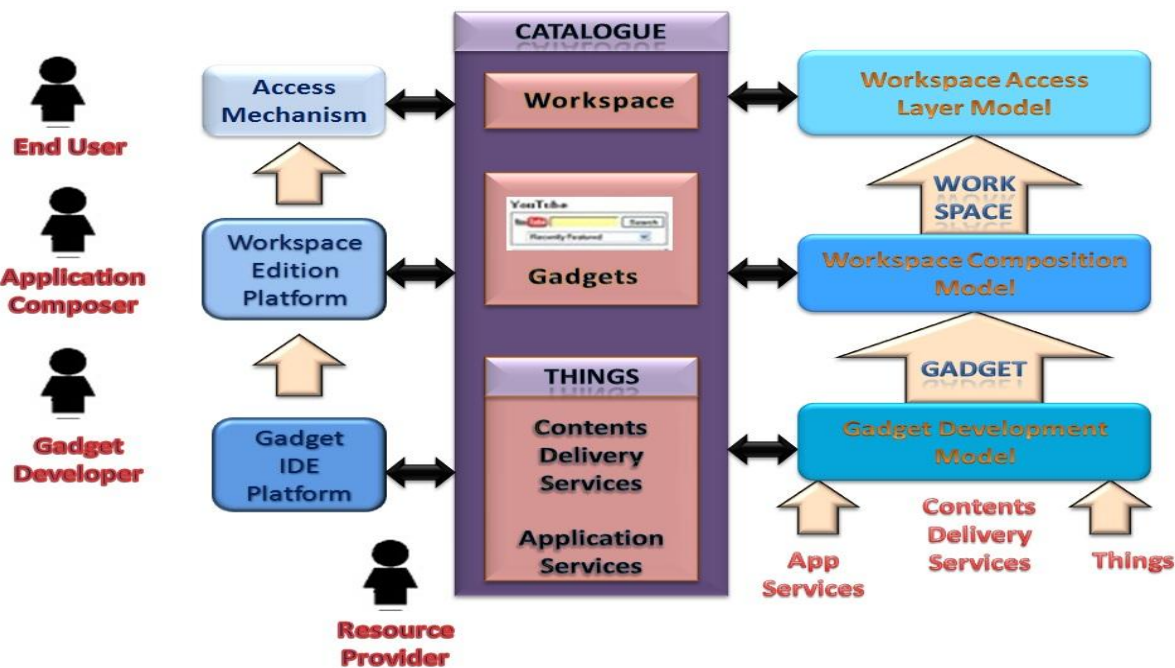


Fig. Proposed architecture

**2) Algorithm:**

A case is classified on the basis of a majority of vote of its neighbors, with the case being designated to the class most common amongst its K nearest neighbors determined by a distance function. If K = 1, then the case is simply designated to the class of its nearest neighbor.

**Steps:**

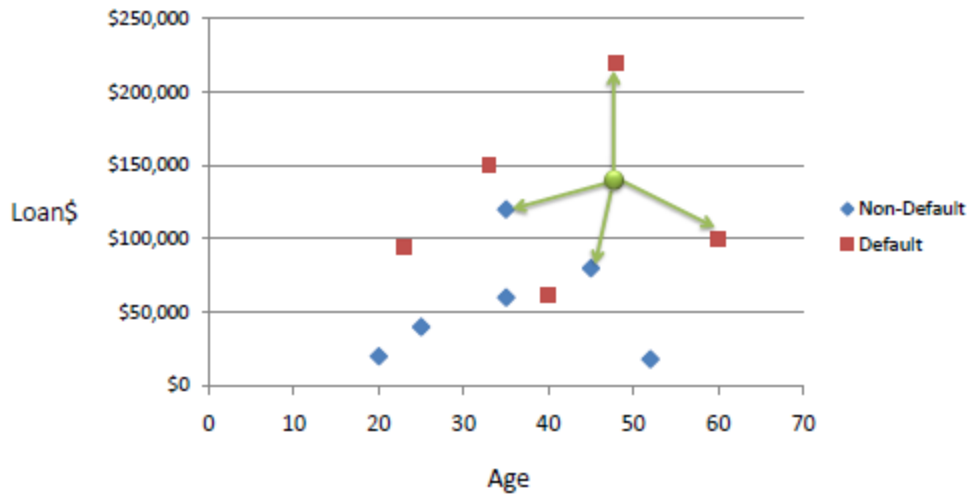
- Determine value of K.
- Calculate Euclidean distance between query instance and all the training tuples.
- Sort the distance determine k<sup>th</sup> minimum distance.
- Use simple majority of nearest neighbor for predicting values.
- 

**3) Complexity of kNN:**

Basic kNN algorithm stores all examples. Suppose we have n examples each of dimension d O(d) to compute distance to one example O(nd) to nd one nearest neighbor O(knd) to nd k closest examples. Thus complexity is O(knd)

**4) Example:**

Consider the following data regarding credit default. Loan and Age are two numerical variables and default is the target[6].



Now we can use the training set to categorize an anonymous case (Age=48 and Loan=\$142,000) using Euclidean distance.

Age	Loan	Default	Distance
25	\$40,000	N	102000
35	\$60,000	N	82000
45	\$80,000	N	62000
20	\$20,000	N	122000
35	\$120,000	N	22000
52	\$18,000	N	124000
23	\$95,000	Y	47000
40	\$62,000	Y	80000
60	\$100,000	Y	42000
48	\$220,000	Y	78000
33	\$150,000	Y	8000
48	\$142,000	?	

Euclidean Distance  $D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$

If K=1 then the nearest neighbor is the final case in the training set with default=Y.

$$D = \text{Sqrt}[(48-33)^2 + (142000-150000)^2] = 8000.01 \gg \text{Default}=\text{Y}$$


With K=3, there are two default=Y and one default=N out of three closest neighbors. The prediction for the anonymous case is again default=Y.

**Standardized Distance**

One significant disadvantage in calculating distance measures straightly from the training set is in the case where variables have

diverse measurement scales or there is a combination of categorical and numerical variables. For example, if one variable on the basis of yearly income in dollars and the other is on the basis of age in years then income will have a large influence on the distance computed. One solution to standardize the training set is given below.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
<b>0.7</b>	<b>0.61</b>	<b>?</b>	



$$X_s = \frac{X - Min}{Max - Min}$$

By using the standardized distance on the identical training set, the anonymous case returned a diverse neighbor which is not a good indication of robustness.

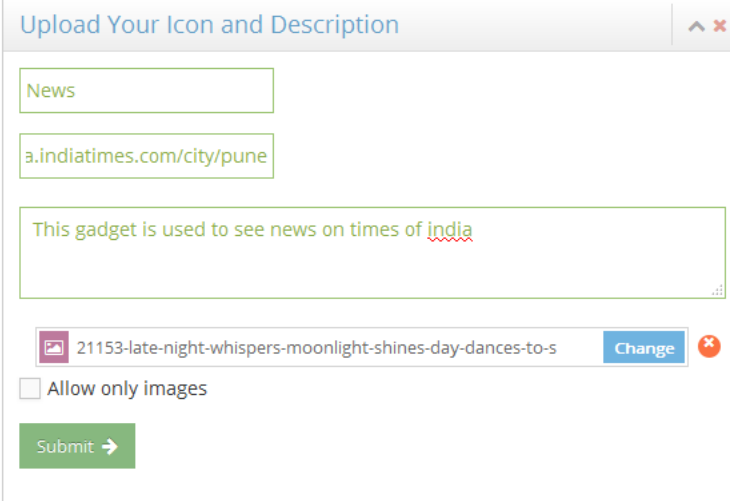
### V. GOALS

1. Reduce technical overhead
2. Tailor services to their individual needs
3. User without programming skill can build web application
4. Architecture enables user to exploit their unique expertise to build application
5. Application support users in their routine work in open innovative creation process
6. Wrap any resource or web service for inclusion within the building blocks

### VI . IMPLIMENTAION

#### User-centric IDE or gadget authoring tool:

We have created gadget authoring tool for creating new gadgets. Gadgets developed using web development tool can be used on any mashup platform, like, Yahoo! Dapper, EzWeb, JackBe or OpenKapow, and it acts both as a workspace editing tool and as access layer. Gadgets are referred to as screenflows in the gadget authoring tool and are assembled by composing one or more resource representations known as screens. Each screen is composition of a form and several back-end resources and/or authoring resources, for example, Amazon, IGoogle, Yahoo!, eBay, Microsoft back-end resources, etc., and they are offered by means of REST (REpresentational State Transfer)-like homogeneous interfaces. The gadget authoring tool uses building blocks available in catalogues. These building blocks have differing degrees of abstraction, including screenflows, screens, forms or back-end resources. Each element is an aggregation of the elements lower down in the hierarchy. The back-end resources can be operators or wrapped services, and are composed of an API and invocation data. All building blocks are composed of a description, an interface, for example, an image, a HyperText Markup Language (HTML) or XML form, an interface described in XUL (XML based User interface Language), etc.) that users can tailor using WYSIWYG (What You See Is What You Get) mechanisms as in other HTML editors, a precondition, a post condition and the option of including a wrapped resource. A building blocks precondition and postcondition are composed of facts, and a fact is just a data item with a meaning associated.



The screenshot shows a web-based form titled "Upload Your Icon and Description". It contains several input fields: a text box with "News", a URL box with "a.indiatimes.com/city/pune", and a larger text area with "This gadget is used to see news on times of india". Below these is a preview area showing a small image and the text "21153-late-night-whispers-moonlight-shines-day-dances-to-s" with a "Change" button. There is also a checkbox labeled "Allow only images" and a green "Submit" button with a right-pointing arrow.

Fig: Gadget authoring tool

The preconditions and post conditions are items used internally by the gadget authoring tool. They are not displayed to users because they are not user friendly mechanisms. The gadget authoring tool is able to create gadgets, which are a user-friendly component useful for taking advantages of web services. The services must first wrap a template that indicates how to invoke the service, which data it accepts and the data which it outputs. The gadget authoring tool uses this information to return the service for users as a black box with inputs and outputs and suggest which elements to interlink to the service to produce or consume such outputs. The gadget authoring tool is responsible for wrapping the service call at run time. The gadget authoring tool checks the syntax and data types in order to suggest the interlinks and ensure that they are valid. Other types of web resources should also be tailored as building blocks considering their inputs, outputs and invocation protocol.

### **Workspace editing tool and run time access layer:**

We have created a workspace editing tool for users without programming skills to create solutions to improve their routine work and also satisfy their personal requirements. The tool also acts as a run time composite application access layer. The workspace editing tool is offering workspace design and editing support based on catalogued gadgets and run time support as a workspace access layer. The platform acting as a development environment, can be used to interlink gadgets. Therefore, execution of one gadget can generate useful data for the other components which are part of the mashup, leading to a dataflow among gadgets. At the run time, users may use the created workspace to manage user-defined dataflows, support the implied invocations of web services and access any remote resources that are gadget data input sources, solving any resulting cross-platform problems. A Run time solution made up of a workspace composed of several gadgets to search a keyword in Amazon services, Web news, Flickr, Wikipedia, YouTube, etc. Creating the flows, the design environment uses the data semantics in the knowledge framework to visually suggest which outputs can be rerouted to the inputs of other components, which will lead to a valid dataflow. The workspace editing tool uses a knowledge framework to built into the environment. The framework helps users to share the workspace that they have built with other users for their advantage, subject to its parameterization to another problem. It monitors and captures user behavior at run time and design time. For this purpose, it creates behavioral rules that can be used through the inference engine on the other platforms and are able to suggest new gadgets of interest for the workspace created by users or new components and background resources for enhancing existing gadgets or creating new gadgets with the gadget authoring tool.

### **Search Web services:**

1. For Accessing web services user has to registered by filling up registration form.
2. For searching type name of web service into the search field provided. Ex. Google, Facebook.

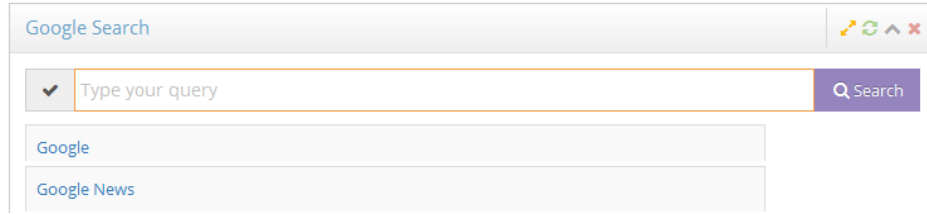


Fig: Search Web services

**Calendar Events:**

1. User can add his/her own events into calendar.
2. User can add events daily, weekly and monthly.

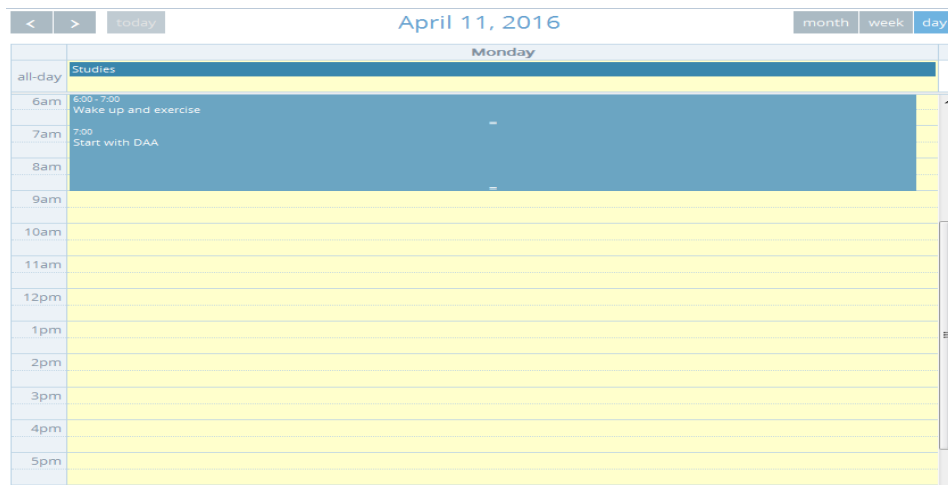


Fig: Calendar Events

**Results:**

Tool	Mean of time	Std. dev.	Std. Error	95% lower b.	95% upper b.	Min.	Max.
Yahoo!Pipes and Dapper	46.40	6.69	0.1579	41.23	51.16	30.50	67.50
iGoogle	63.62	7.89	0.2530	56.19	71.88	35.50	72.50
Apple Dashboards	68.21	15.28	0.0699	44.79	69.90	40.50	87.50
Presented architecture	42.89	6.15	0.1952	38.15	50.95	27.50	61.30

**VII . APPLICATIONS**

1. Food and Drug Administrator.
2. In organizations to reduce technical overhead.
3. E-Business.
4. E-Learning.

5. Networking among society.
6. Private individuals will benefit from intuitive, unsophisticated ways to discover, remix and use the web services that they consider interesting and useful.

### **VIII . CONCLUSION**

The proposed system empowers users to build applications based on off-the-shelf and published components using a visual composition process without having to use programming languages or skills. The system is able to wrap any resource and web service for combining them within the building blocks available to end users. The system enables dataflow based on visual recommendations, supported by the semantics of the data handled by each component. The architecture enables users to discover and exploit their unique intelligence to build applications that support their routine work in an open innovative creation process. The architecture helps users without programming skills to develop composite web applications based on front-end components.

### **XI . REFERENCES**

- [1] J.J. Hierro, T. Janner, D. Lizcano, M. Reyes, C. Schroth, J. Soriano, Enhancing userservice interaction through a global user-centric approach to service oriented architectures, in: Proceedings of the Fourth International Conference on Networking and Services, ICNS 2008, IEEE Computer Society Press, 2008, pp. 194203 (authors in alph. order).
- [2] A.P. McAfee, Enterprise 2.0: The dawn of emergent collaboration, MIT Sloan Management Review 47 (2006) 21–28.
- [3] G. Alonso, F. Casati, H. Cuno, V. Machiraju, Web Services Concepts, Architectures and Applications. Data-Centric Systems and Applications, Springer, 2004.
- [4] C. Schroth, O. Christ, Brave new web: Emerging design principles and technologies as enablers of a global soa, in: Proceedings of the IEEE International Conference on Services Computing, 2007, SCC 2007, pp. 597604, 2007.
- [5] Juan Alfonso Lara, David Lizcano, María Aurora Martínez, Juan Pazos, Developing front-end Web 2.0 technologies to access services, content and things in the future Internet, in: Universidad a Distancia de Madrid, UDIMA, 28400, Collado Villalba, Madrid, Spain
- [6] [http://www.saedsayad.com/k\\_nearest\\_neighbors.htm](http://www.saedsayad.com/k_nearest_neighbors.htm)
- [7] D. Bianchini, V.D. Antonellis, M. Melchiori, A semantics-enabled web application programming interface registry, in: DEXA Workshops'11, 2011, pp. 247–251.
- [8] D. Bianchini, V. De Antonellis, M. Melchiori, Semantics-enabled web application programming interface selection patterns, in: Proceedings of the 15<sup>th</sup> Symposium on International Database Engineering and Applications, ACM, New York, NY, USA, 2011, pp. 204–208.