The Dicode Workbench: A Flexible Framework for the Integration of Information and Web Services

Guillermo de la Calle
and Eduardo Alonso-Martínez
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
28660 Boadilla del Monte, Madrid, Spain
{gcalle,ealonso}@infomed.dia.fi.upm.es

Manolis Tzagarakis
and Nikos Karacapilidis
Computer Technology Institute & Press "Diophantus"
and University of Patras
26504 Rio Patras, Greece
tzagara@upatras.gr, nikos@mech.upatras.gr

ABSTRACT
Aiming to address requirements concerning integration of services in the context of "big data", this paper presents an innovative approach that (i) ensures a flexible, adaptable and scalable information and computation infrastructure, and (ii) exploits the competences of stakeholders and information workers to meaningfully confront information management issues such as information characterization, classification and interpretation, thus incorporating the underlying collective intelligence. Our approach pays much attention to the issues of usability and ease-of-use, not requiring any particular programming expertise from the end users. We report on a series of technical issues concerning the desired flexibility of the proposed integration framework and we provide related recommendations to developers of such solutions. Evaluation results are also discussed.

General Terms
Management, Design, Human Factors.

Keywords
Service integration, e-applications service architectures, data exchange, mashup application, data-intensiveness, collaborative work, collective intelligence.

1. INTRODUCTION
Traditional software systems were usually designed to work as standalone applications. However, during the last years, new requirements and challenges have emerged; systems frequently need to exchange heterogeneous data and interoperate with other applications. The associated integration of data and services is a complex and challenging issue, which depends on many factors such as system architectures, operating systems, type of the components, information to be integrated, coupling and use of the systems, performance requirements, data heterogeneity and semantics, user interfaces, middleware, and availability of resources [1].

At the same time, individuals and organizations are confronted with the rapidly growing problem of information overload [2]. An enormous amount of content already exists in the "digital universe", i.e. information that is created, captured, or replicated in digital form, which is characterized by high rates of new information that is being distributed and demands attention. This enables us to have instant access to more information than we can ever possibly consume [3]. When working together, people have to cope with this diverse and exploding digital universe; they need to efficiently and effectively collaborate and make decisions by appropriately assembling and analyzing enormous volumes of complex multi-faceted data residing in different sources.

In all domains of our societies, e.g. e-business, e-commerce, e-learning, e-science, and e-government, it is nowadays easier to get the data in than out. Big volumes of data can be effortlessly added to a database; the problems start when we want to consider and exploit the accumulated data, which may have been collected over a few weeks or months, and meaningfully analyze them towards making a decision. Admittedly, when things get complex, we need to identify, understand and exploit data patterns; we need to aggregate big volumes of data from multiple sources, and then mine it for insights that would never emerge from manual inspection or analysis of any single data source. In such settings, the meaningful exploitation of collective (human) intelligence is of great importance and certainly sets a big research challenge.

As results from the above, tasks performed (and tools used) nowadays are increasingly information and interaction intensive. Thus, issues related to the guidance of the information worker through the space of available data and the indication of relevant information to facilitate and augment collaboration and decision making activities are of major importance. Generally speaking, information management related tasks need to be streamlined and automated. Recent findings clearly indicate that information management costs too much when it is not well organized and meaningfully automated [4]. They also call for investments in innovative software that reduces or eliminates time wasted, reduces management overheads, streamlines collaborative processes, and automates the overall workflow. Return on such investments can be both tangible (e.g. time or money saved) and intangible (e.g. more valuable information, easier extraction of

hidden information, increase of information workers' satisfaction and creativity, improved collaboration).

Aiming to address the above requirements concerning interoperation and integration of information and services in the context of "big data", this paper presents the approach developed in an FP7 EU project, namely Dicode (http://dicode-project.eu/). The proposed solution (i) ensures a flexible, adaptable and scalable information and computation infrastructure, and (ii) exploits the competences of stakeholders and information workers to meaningfully confront information management issues such as information characterization, classification and interpretation, thus incorporating the underlying collective intelligence. At the same time, it pays much attention to the issues of usability and ease-of-use, not requiring any particular programming expertise from the end users.

The remainder of the paper is structured as follows: The next section briefly reports on existing integration technologies and related integration efforts. Section 3 provides an overview of the Dicode project, focusing on the diversity of the services involved and their underlying technology. Section 4 is devoted to the project's approach towards integration of information and services, namely the Dicode Workbench; issues discussed include the technological solutions adopted to cope with the diversity of services, the design adopted towards facilitating interoperability, the interface and interoperability of services, data integration from both a conceptual and a technological point of view, as well as examples of use of the proposed approach. Finally, evaluation results and concluding remarks are given in Section 5.

2. BACKGROUND

A key challenge for integration systems is the adaptation of existing (legacy) systems. To ensure a successful integration, two factors are crucial: coupling and adaptation to standards. These factors may hamper the tasks of designing and implementing software systems, converting them into products that do not scale and hence will not be used. The coupling of a system is given by the degree of interdependency among modules and programs. It is desirable that this interdependency remains as little as possible, since a loose coupling between components facilitates the modification of any of the modules without affecting the rest of the parties. Adaptation to standards relies on the correct design and documentation of the system. Well-planned and designed systems have interfaces for integration between its modules. By using standards, the need to develop specific software to perform this integration is minimized.

Service-oriented architectures (SOA) [5] aim to address the above requirements; they are based on well-defined standards, which are focused on low coupling between modules. SOA is not a tool, technology or product, but a concept, a set of rules and principles to design software, regardless of the technology used in its development. SOA relies on the creation of some interfaces that abstracts away the underlying complexity. By using such interfaces, clients and providers may establish communication by only knowing the inputs and outputs of the services. SOA is usually implemented using *web services*. There are two main technologies to develop web services: WS-* (SOAP-based web services) and RESTful [6]. RESTful web services appeared as an evolution of WS-* to alleviate its complexity. RESTful services give more importance to information while WS-* are more focused in message exchange.

Following the philosophy of SOA and based on web services technologies, a new concept for the integration of applications and services has appeared during the last years, namely the *web mashup* [7]. Web mashups are web applications combining data, functionalities, services and applications from different sources to define more complex services. The main features of mashups are the combination and aggregation of services and the visualization within a common interface. Two styles of mashup applications can be defined considering its architecture: client-based mashups and server-based mashups. The former are focused in user's web browsers to integrate and manipulate data coming from external sources. On the other hand, server-based mashups manipulate the information on the server and return the final results to the user's web browser. Independently from the style, mashup architectures are usually divided into three layers:

- data layer, to represent the information; technologies like XML, JSON and KML are used to codify the information.
- access layer, to retrieve the data used in the mashup; usually, data is accessed using API services implemented as web services (SOAP, REST).
- visualization layer, to present the aggregated information to users; user interfaces are developed using HTML, CSS, JavaScript or AJAX.

Generally speaking, there are several initiatives for developing mashup applications, such as *Yahoo Pipes* [8], *Google Mashup Editor* (GME), *Microsoft Popfly*, *Intel Mash Maker* [9], and *QedWiki*. All of them enable users to combine different services to create complex applications, but they present a common drawback: creating a new mashup application requires high technical (programming) skills that end-users usually do not have. Thus, most users cannot easily take advantage of this technology.

Projects and initiatives with objectives similar to those of Dicode include GRANATUM (http://granatum.org), Doc@Hand (http://www.doc-at-hand.org), Health-e-Child (http://www.health-e-child.org), Virolab (http://www.virolab.org), SIMBioMS (http://simbioms.org), VPH Network of Excellence (http://www.vph-noe.eu), and Ricordo (http://www.ricordo.eu). These either do not deal with big data issues, or they do not focus on the full (i.e. technical, conceptual and interface) integration of data mining and collaboration support issues which traditionally have been addressed independently [10-11]. Paying much attention to the data intensiveness of contemporary business contexts, Dicode aims to achieve such a full integration.

Other approaches developed to address information and service integration issues are discipline-specific. For instance, several efforts have been conducted in the area of Bioinformatics. A well-known example is *myExperiment*, an online research environment that supports the social sharing of bioinformatics workflows, i.e. procedures consisting of a series of computational tasks, which can then be reused according to their specific requirements [12]. The *Galaxy Project* (http://galaxy.psu.edu/) offers a web-based platform allowing researchers to perform and share analyses. *BioCatalogue* [13], a Web based service registry, allows users to annotate and comment on the available services in order to assist them in identifying suitable services.

In any case, the above approaches demonstrate a set of limitations, mainly concerning flexibility in the integration of services offered, as well as incorporation of the collective intelligence. As made

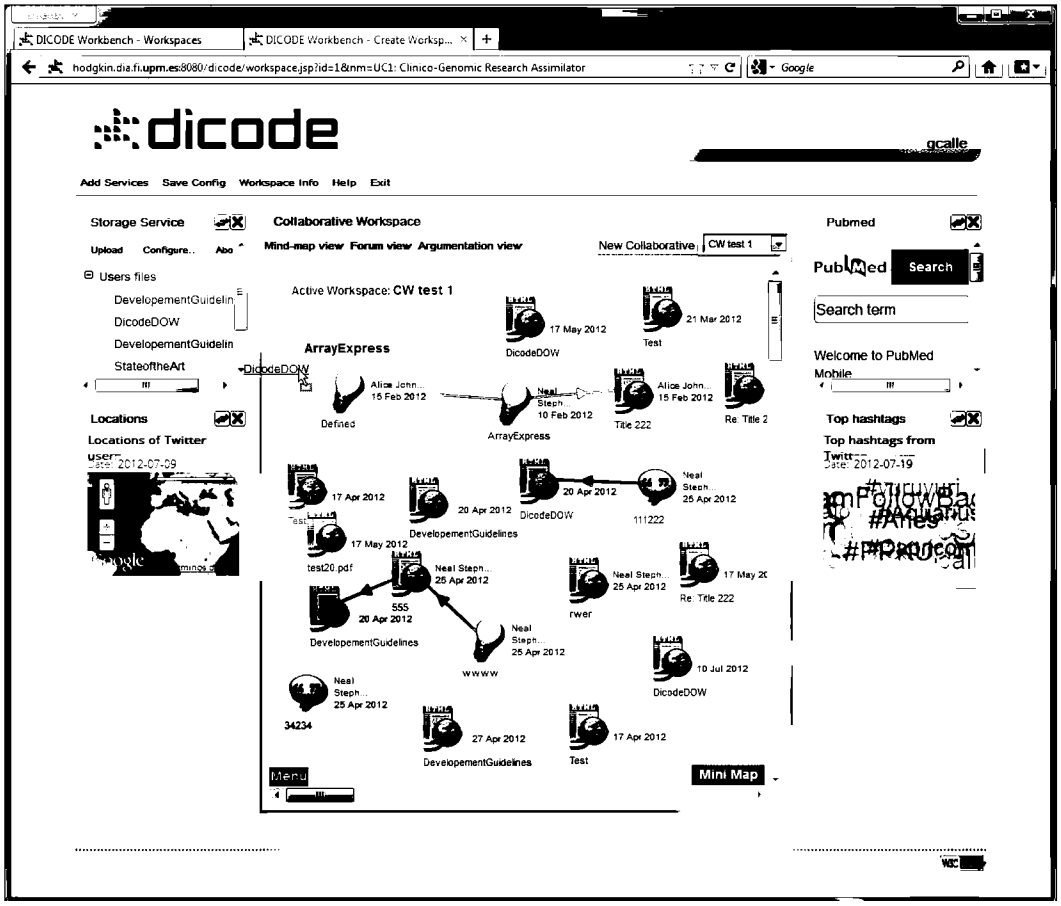

Figure 1. A snapshot of the Dicode workbench

clear in the next sections, in the Dicode project we have developed a highly flexible integration framework that enables users to easily set-up and work with their own collaborative workspaces (exploiting diverse web applications) by just using the mouse, without any need for programming. At the same time, through appropriately integrated collaboration and decision making support services, we provide users with means to foster and augment the human intelligence in order to extract the necessary insights for the solution of the issue under consideration.

3. THE DICODE PROJECT

The goal of the Dicode project is to provide the necessary infrastructure and environment that makes it easy for people to cope with a diverse and exploding digital universe when working together. In particular, Dicode aims at enabling efficient and effective collaboration and decision making by appropriately assembling and analyzing enormous volumes of complex multi-faceted data residing in different sources. The environments under which Dicode is designed to be used are characterized by data-intensiveness and cognitive complexity. Representative examples of such environments, which serve as the test-bed of the Dicode approach, concern:

- clinical researchers and bio-scientists which need to locate and assemble huge amounts of data (including clinico-genomic data, molecular pathways, DNA sequence data etc.) and collaborate in order to draw new, insightful conclusions;
- radiologists, radiographers, clinicians, patients and researchers in the pharmaceutical industry, which engage into a decision making process with respect to clinical decisions and drug testing by examining a huge amount of heterogeneous and annotated datasets (including blood tests, physical examinations, X-rays, MRI scans, and free text journals of patients on their experience from treatments);
- marketing and brand specialists who need to forage web pages, blogs, forums and wikis for high level knowledge, such as public opinions about specific products and services, who monitor public responses to a new marketing launch, or who need to filter, collate and analyze findings to inform their strategy.

At the heart of Dicode's approach is the synergy of human (collective) and machine (artificial) intelligence, which traditionally have been considered separately. In the project's context, such synergy is accomplished by providing, integrating and orchestrating a set of interoperable services that reduce the data-intensiveness and complexity overload at critical decision points to a manageable level, thus permitting stakeholders to be more productive and concentrate on creative and innovative activities.

3.1 Diversity of Services in Dicode

Acknowledging the fact that in the context of big-data, users are required to use and orchestrate a diverse range of services, Dicode aims at providing the necessary technical infrastructure to

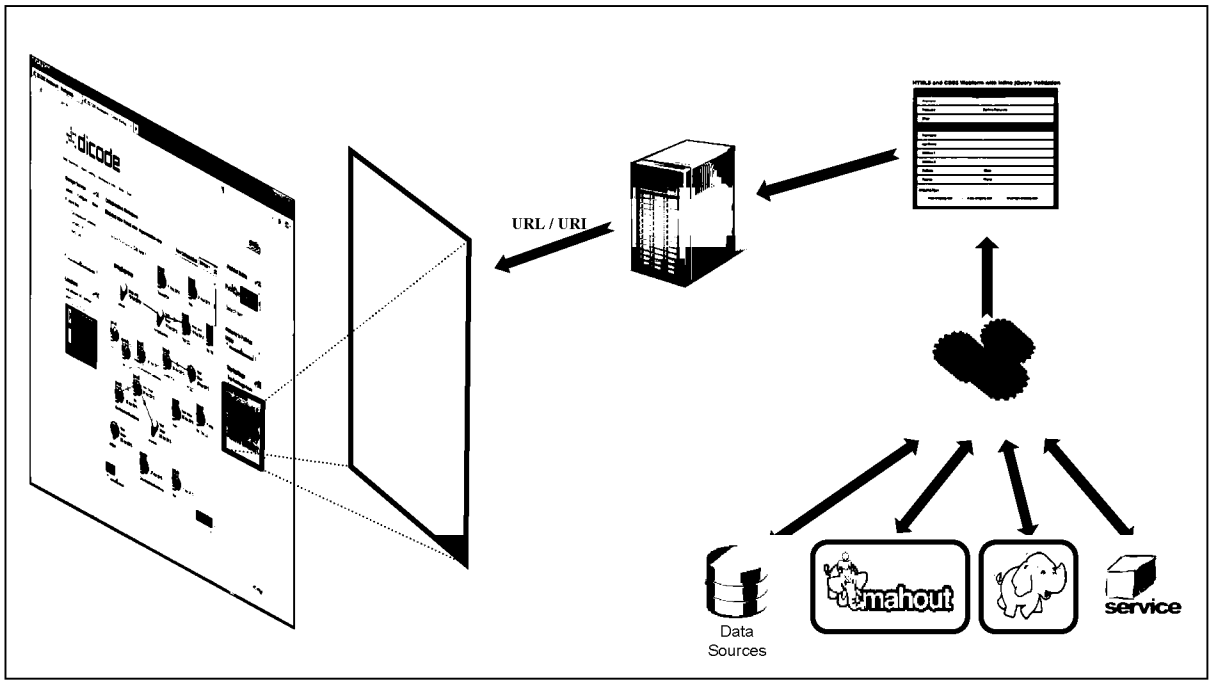

Figure 2. Structure of a service integrated within the Dicode workbench

integrate such required services and make them easily accessible to the end users. The categories of services provided by the Dicode technical infrastructure include:

- *Data acquisition services*, which enable the purposeful capturing of tractable information that exists in diverse data sources and formats, including external repositories. Such services offer the necessary interfaces to retrieve and store locally remote resources. A range of technologies are supported to retrieve data that include web services as well as web crawlers to acquire web-based resources.
- *Data pre-processing services*, which efficiently transform raw data in a way that is suitable for use within the system. Such services enable the transformation of documents into appropriate forms, as well as data cleansing (e.g. removing noise from web pages, discarding useless database records).
- *Data mining services*, which are built on top of a cloud infrastructure and other prominent large data processing technologies to offer functionalities such as high performance full text search, data indexing, classification and clustering, directed data filtering and fusion, and meaningful data aggregation. Advanced text mining techniques, such as named entity recognition, relation extraction and opinion mining, aid the extraction of valuable semantic information from unstructured texts.
- *Collaboration support services*, which facilitate the synchronous and asynchronous collaboration of stakeholders through adaptive workspaces, efficiently handle the representation and visualization of the outcomes of the data mining services (through alternative and dedicated data visualization schemas), and accommodate a workflow engine that enables the orchestration of a series of actions for the appropriate handling of data in each case.
- *Argumentation support services*, which augment individual and group sense-making and decision-making by supporting stakeholders in arguing about relevant information and knowledge, as well as by providing them with appropriate notifications and recommendations (taking into account parameters such as preferences, competences, expertise etc.). Based on pre-defined formalisms, this category of services deploys a set of reasoning mechanisms to aid stakeholders in monitoring the evolution of a collaborative discussion and assessing its outcome.

The above services rely on different frameworks to provide their functionality: they can be REST-based or rely on the SOAP framework (http://www.w3.org/TR/soap12-part1/). The last two categories of services, as well as the interoperation of them with the data mining services, are specifically targeted towards the exploitation and enhancement of the collective intelligence of the stakeholders involved.

4. THE DICODE WORKBENCH

4.1 Description

Central to the Dicode approach is the concept of "workbench". Workbenches are Web-based applications that integrate - at the level of the user interface - various data mining and collaboration support services and make them available to the users (Figure 1). The objective is to provide users with a uniform and easy access to the available services. The type and number of services appearing on the workbench can be configured by end users according to the needs of the particular context and problem under consideration. New services can be easily integrated into the workbench. A widget-based approach [14] has been adopted to implement and integrate services within the Dicode workbench.

Since the Dicode workbench is a web application, the idea of web widgets is appropriately applied.

A web widget can be defined as a small stand-alone software application that can be embedded within a web page and executed (used) by the end user. It is important to note that in widget applications, the host does not control the content of the widgets. This means that the behavior of the widget's contents is at the responsibility of the widget itself. Widget functionality or content cannot be managed or modified by the host. The host can only control whether the widget is shown or not, as well as the placement of the widget on the screen.

There exist different technologies, libraries and frameworks to develop widgets, but the most common ones are HTML, JavaScript and Adobe Flash. For the Dicode workbench, we used HTML5, CSS3 and JavaScript to implement the web interface (front-end) for the services developed by service providers. Other options are also acceptable.

As shown in Figure 1, widgets within the Dicode workbench are distributed in three "logical" columns: two small ones on the sides and one bigger in the middle of it. By default, the *collaborative workspace* (i.e. a particular web service supporting multiple-view argumentative collaboration among stakeholders) appears in the center, while the rest of services are located on the sides. However, the Dicode workbench allows users to maximize any of the widgets located on the sides. When a widget is maximized, it changes its position with the widget that is in the middle at that moment (thus reflecting where the focus of the attention is each time). Users sharing a workspace (e.g. belonging to a particular community of practice) may use the same set of services; in any case, they can customize their own view of the shared workspace by relocating (moving) the widgets. The position of the widgets is stored when a collaboration session terminates.

The Dicode workbench also allows users to easily add new services to the workspace. Users can search and add new services, provided that these have been previously registered by the service providers or developers in the system (the process of registering a new service is discussed in Section 4.2).

Apart from these aspects, which are primarily oriented towards end users, service developers have to consider that each service will be loaded into an *iframe element* [15]. Thus, all services have to be suitable to be displayed in such HTML control. Issues concerning development of services to be integrated in the Dicode workbench are discussed in the next subsection.

4.2 Developing services for the Dicode workbench

As mentioned above, the Dicode workbench uses iframe elements to display the services, but it is not responsible for their behavior. The only technical requirement for a service to be integrated into the workbench is that it can be loaded and displayed into an iframe. Inside an iframe, any web application that usually uses web technologies such as HTML, CSS and JavaScript can be displayed. The recommendation for developers is to use state-of-the-art web technologies such as HTML5, CSS3, JavaScript or jQuery.

Figure 2 depicts the structure of a service integrated within the Dicode workbench. The service performs a concrete task or a set of tasks; for instance, it may retrieve information from a database, analyze datasets or execute complex algorithms. The results of this task are presented through a web interface. This web interface is deployed in a web server and it is accessible via a URL/URI. This URL/URI is used by the Dicode workbench to display the web interface within an iframe element. In this way, a bidirectional communication can be established, i.e. information may flow from the service to the Dicode workbench and from the workbench to the service, for instance, to communicate actions executed by the user in the workbench or data provided by the user to parameterize the service.

To integrate an application or a service in the Dicode workbench, service providers and developers have to follow the following steps:

1. **Develop the service**, i.e. to implement the "logic" of the service. A service might be as simple as displaying a message or perform an addition, and as complicated as running complex algorithms using high performance toolkits. These services can use any technology or library because the Dicode workbench is not aware about it.

   Developers should also take into account that the service will be probably invoked from outside the service. Thus, a public interface to invoke the service should be created. Our recommendation is to create web service interfaces based on RESTful or WS-* (SOAP) services [6].

2. **Develop a web interface of the service,** to enable users interacting with the service. Users should be able to invoke and use the service from this web interface. In fact, this web interface acts as a "wrapper" for the service. Additionally, if the service needs or can be invoked with different parameters, the web interface could also provide facilities to users to establish such parameters.

   Designers and developers have to carefully consider the available space devoted to widgets in the Dicode workbench. Widgets cannot properly display traditional web applications designed for high resolutions. Applications for widgets are more similar to mobile applications regarding user interface. In particular, for the Dicode workbench, widgets can have two possible states as commented for the layout shown in Figure 1: maximized in the middle or minimized on the sides. Ideally, the web interface should suit to the actual resolution by using a "liquid and elastic design" [16].

   Depending on the type of integration required for the service, developers have also to consider some additional requirements listed in Section 4.3.

3. **Deploy the service and the web interface.** Both elements have to be accessible through the Internet via an URL/URI. Thus, they have to be deployed in a web server.

4. **Register or publish the service.** The Dicode workbench can display only those services that have been previously registered in the system. A registry of annotated services is maintained by the Dicode workbench. To register a service, service providers have to provide metadata about their service, such as its name and description, annotations according to the sensemaking operations contained in the DicodeONtology (DON) [17], or the URI where the service is running. The latter is the most important field to integrate the service within the workbench.

4.3 Integrating services

As discussed above, the Dicode workbench can be considered as a mashup web application, allowing users to share resources under a common framework. Mashup applications usually consist of applications showing together different components such as, for instance, *iGoogle* [18]. But traditionally, these components neither share information nor communicate in any other way. In the Dicode project, we are moving one step ahead by enabling users to move data from one widget to another, just by using the mouse. The system architecture is designed to maintain a loose coupling among all integrated resources.

Besides the integration at the level of the user interface (called "light integration" in Dicode), services are also integrated at a deeper, semantic level (called "full integration"). Such integration allows services to exchange data in order to share data between services. This allows for example to support "drag-and-drop" functionality between the Dicode services, in order to support the passing of necessary data from one service to another. The two different modes of integrating services in Dicode are described in more detail below.

1. **Light integration.** It can also be called "visual integration". This consists in the traditional mashup approach, i.e. services/applications are displayed together within the same web interface. No interactions happen between services, thus each service works as a standalone application.
2. **Full integration.** Services are not only displayed within the same application, but data can be exchanged among them. Different mechanisms have been developed to communicate data among services. Web interfaces of these services need to implement a set of functions to properly carry out such communication.

The Dicode workbench provides both integration methods to end users. Service developers can select the level of integration desired for their services.

4.3.1 Light Integration

This is the integration approach followed by the mashup application, where different components are just displayed together within a common interface. To carry out this form of integration in the Dicode workbench, service developers only need to follow the steps described previously, i.e. develop the service, develop the web interface, deploy both elements in a web server and publish the service (URI to the web interface) in the Dicode workbench. As soon as these steps are successfully completed, services can be located and added by users to their shared workspaces.

4.3.2 Full Integration

This integration approach allows interaction and data sharing between components integrated within the same platform. Interactions in the Dicode workbench are envisioned as events triggered when users move (drag with the mouse) items from one widget to another.

We have designed a loosely coupled architecture based on the idea of message passing interfaces (MPI) [19]. In particular, we exploited the *postMessage* mechanism provided by HTML5 [20]. This mechanism allows applications running in different windows or frames to communicate information (plain text) across different origins and domains. Although the content of the message can only be plain text, this is enough to communicate almost everything using, for instance, URIs or REST references.

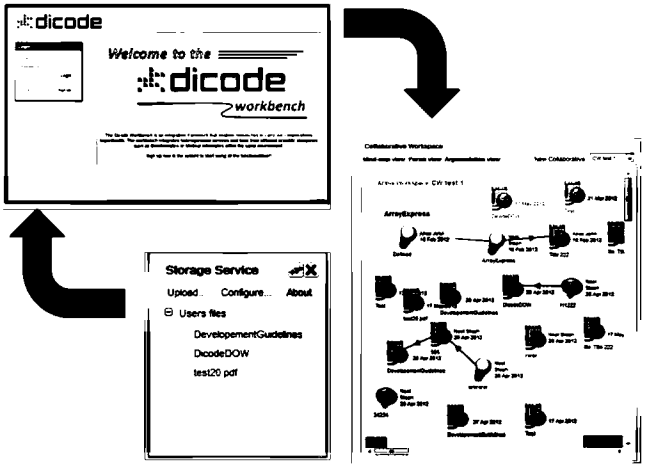

Figure 3. Communication between widgets in the Dicode workbench

As shown in Figure 3, the Dicode workbench acts as a message "mediator" between the different widgets. When the Dicode workbench detects that the user wants to move one element from one widget to another, it takes the element from the origin source and send a message containing the element to the target widget. Then, the target widget receives the message, interprets it and performs the actions associated to that message. Both reception and sending messages are optional for widgets (iframes), while it is at the responsibility of service developers to incorporate them. These functionalities are implemented in JavaScript using the facilities of HTML5. Some examples of the code to be incorporated in the web application interface are given below.

a) Sending an item

There are two requirements needed to send items in the Dicode workbench:

- All HTML elements which can be dragged must be labeled as *draggable;*
- To define the information to be sent when the item is dragged.

```
<!doctype html>
<html>
<head>
<scriptsrc="js/dragDrop.js"></script>
</head>
<body>
  <a href="#" draggable="true" ondragstart="processDragStart(this.href,
     event)"> item1 </a>
  <a href="#" draggable="true" ondragstart="processDragStart(this.href,
     event)"> item2 </a>
  <a href="#" draggable="true" ondragstart="processDragStart(this.href,
     event)"> item3 </a>
</body>
</html>
```

Figure 4. Example code to allow items to be dragged

To label an element as *draggable*, an attribute to the tag of this element has to be added as shown in the example code given in Figure 4. Working like this, browsers can identify those elements as *draggable*.

In Figure 4, a file named *dragDrop.js* and containing JavaScript code is imported. This file contains the functions to handle the drag behavior of an item and receives the associated messages. The full content of *dragDrop.js* file is given in Figure 5.

To establish the information to be sent, it is needed to add an event to the *draggable* items and define a function to process the event. Usually, this function will define the information to be exchanged between widgets. There are two options to add an event to an item:

- To include the event in the HTML code as shown in Figure 4; *ondragstart* is the name of the event that is triggered when users start a drag action; *processDragStart* is the name of the function to process the event.
- To invoke the function *addEvent*, included in the code shown in Figure 5, for each draggable item. Using tools and libraries such as, for instance, cssQuery [21], the DOM of the document can be examined looking for items defined as draggable ones.

Once the "listeners" for the events are established, service developers have to codify the functions to attend the events. As described above, we have adopted a message passing strategy. Thus, that function should be used to construct the message that the service wants to communicate to the other services. In Figure 5, an example of such function is provided. In this case, one message is created following the message structure adopted. After the message is created, it is sent to the parent window, i.e. the Dicode workbench.

b) Receiving an item

To receive messages in the application, two requirements are needed:

- To create a listener to receive messages in the application/service. An example of how to create such a listener is shown at the end of the code given in Figure 5. This listener will be associated to the window/iframe where the application is running.
- To define and codify the treatment of the information received into the message. In the example of Figure 5, the function *OnMessage* has been defined for such purpose. In this case, the function *OnMessage* checks the origin of the message to prevent from unauthorized uses, and then the content of the message is shown in a popup window. Treatment of messages can be as simple as presented, but it

```
/* dragDrop.js */

varaddEvent = function(obj, evType, fn){
if (obj.addEventListener) { //W3C DOM
obj.addEventListener(evType, fn, false);
    } else if (obj.attachEvent) { //IE DOM
obj['e' + evType + fn] = fn;
obj[evType + fn] = function() { obj["e" + evType + fn](self.event); };
obj.attachEvent("on" + evType, obj[evType + fn]);
    }
};

functionprocessDragStart(ref, e){
var message;
message = '{"Item": {\n' + '\t"type": "File",\n' + '\t"name": "",\n' + '\t"uri": "'+ref+'",\n' +'\t"format": "",\n'
                + '\t"description": ""\n' + '\t}\n}';
if (parent.postMessage) {
parent.postMessage(message,"*");
    } else {
alert ("Your browser does not support the postMessage");
    }
};

functionOnMessage (event) {
var message = event.data;
//Check the location of the caller

//Opera earlier than version 10
if ('domain' in event) {
if (event.domain != "hodgkin.dia.fi.upm.es:8080") { return;}
}

// Firefox, Safari, Google Chrome, Internet Explorer from version 8 and Opera from version 10
if ('origin' in event) {
if (event.origin != "http://hodgkin.dia.fi.upm.es:8080") { return; }
}

// TODO Treatment of the received message
alert ("RECEIVED: "+message);
};

onload = function () { addEvent(window, "message", OnMessage); };
```

Figure 5. Complete code of JavaScript file*dragDrop.js*

can be as complex as service developers need.

Methods proposed can be refined by service developers by using jQuery [22, 23] or other libraries. At the moment, interactions between widgets are triggered by users when they move elements from one widget to another. However, this architecture could be extended to allow widgets/services to trigger events for sending data to other widgets/services by following a "publish-subscribe" design pattern.

c) Message formats

The message passing approach requires that both emitter and receiver agree on a common format for exchanging information. In the Dicode project, we have adopted a message format based on JSON [24]. A preliminary set of basic messages has been defined to be used by the applications/services within the Dicode workbench. These are listed in Table 1 (parts appearing in italics have to be completed by the sender with the proper information). This set is not closed; it could be easily expanded with more types of messages.

Table 1. Set of JSON message in the Dicode workbench

Type of item	Message format

File	{Item: {
	type: File,
	name: *name*,
	uri: *uri*,
	format: *format*,
	description: *description*
	}
	}

Image	{Item: {
	type: Image,
	name: *name*,
	uri: *uri*,
	description: *description*
	}
	}

Text	{Item: {
	type: Text,
	content: *content*
	}
	}

Link	{Item: {
	type: Link,
	uri: *uri*
	}
	}

4.4 Usage Examples

The mechanisms facilitating the integration of the diverse range of services provided in Dicode allows users to introduce different work practices, which differ from those with which they engaged during their daily routine. Before the Dicode workbench, users working with high-throughput scientific data (such as those found in clinico-genomic, biomedical and marketing research) had to "cross" them among several applications, which can be web and desktop based. Such work practices introduce great overhead in managing the relevant data and applications, ultimately disrupting the everyday work of users. In the context of the Dicode workbench, such tasks can be streamlined and automated, relieving users from such concerns.

In particular, within the context of the Dicode workbench users are able to:

- Pass parameters and start the execution of data mining services by simply dragging and dropping data sets from one widget onto the desired algorithm residing in a different widget.
- Invoke data pre-processing services by dragging and dropping onto them the data they need to transform.
- Access the outcomes of the invoked data mining services, which are automatically published in the respective widgets, after the services terminate their execution.
- Discuss the outcomes of the data mining services and involve them in decision making processes, by uploading them into the collaboration workspace (by dragging and dropping the relevant files from one widget to the other).

The integration model in Dicode also allows the uploading of data mining algorithms into collaboration workspaces; such algorithms can be treated as any other collaboration item. In addition, such items (representing data mining algorithms) can be executed from within the collaboration workspace. Our approach allows users to monitor their status. After the execution of these algorithms, their outcomes are automatically uploaded into the collaboration workspace, and can be part of an ongoing discourse. For instance, users may comment on them, express arguments in favor or against the algorithm and/or the data source chosen etc. (more about this kind of collaborative analysis of data and the associated incorporation of the collective intelligence offered by the Dicode approach can be found in [25]).

5. DISCUSSION AND CONCLUSIONS

The Dicode project goes through an ongoing evaluation process. The first evaluation round aimed to assess a series of key success indicators concerning the maturity of the technology used, as well as the usability and acceptability of Dicode services in three real-life contexts (clinic-genomic research, medical decision making, and opinion mining from Web 2.0 data). Evaluators were asked to read a service-specific scenario, experiment with the Dicode services (and workbench), and fill in a mixed-type questionnaire (responses expected were in both quantitative and qualitative form). As far as the Dicode workbench is concerned, the sample consisted of 58 evaluators with varying background knowledge in bioscience fields. Answers to the quantitative questions of the questionnaires were given in a 1-5 scale, where 1 stands for *'I strongly disagree'* and 5 for *'I strongly agree'* [26-27].

Figure 6 summarizes the evaluators' responses relative to the overall quality of the Dicode Workbench. As shown, the evaluators agreed (median: 4, mode: 4) that its objectives are met, that it is novel to their knowledge, that are satisfied with its performance and that they are overall satisfied with it. The evaluators were neutral (median: 3, mode: 3) with respect to whether the Workbench addressed the data intensive decision making issues. Related comments were: *'Some kind of 'roadmap' would be appreciated'*; *'Getting started is a bit confusing for a new user'*. As long as its acceptability is concerned (such results are not shown in Figure 6), the evaluators agreed (median: 4, mode: 4) that the Workbench has the full set of functions they expected, that its interface is pleasant and that they will recommend it to their peers/community.

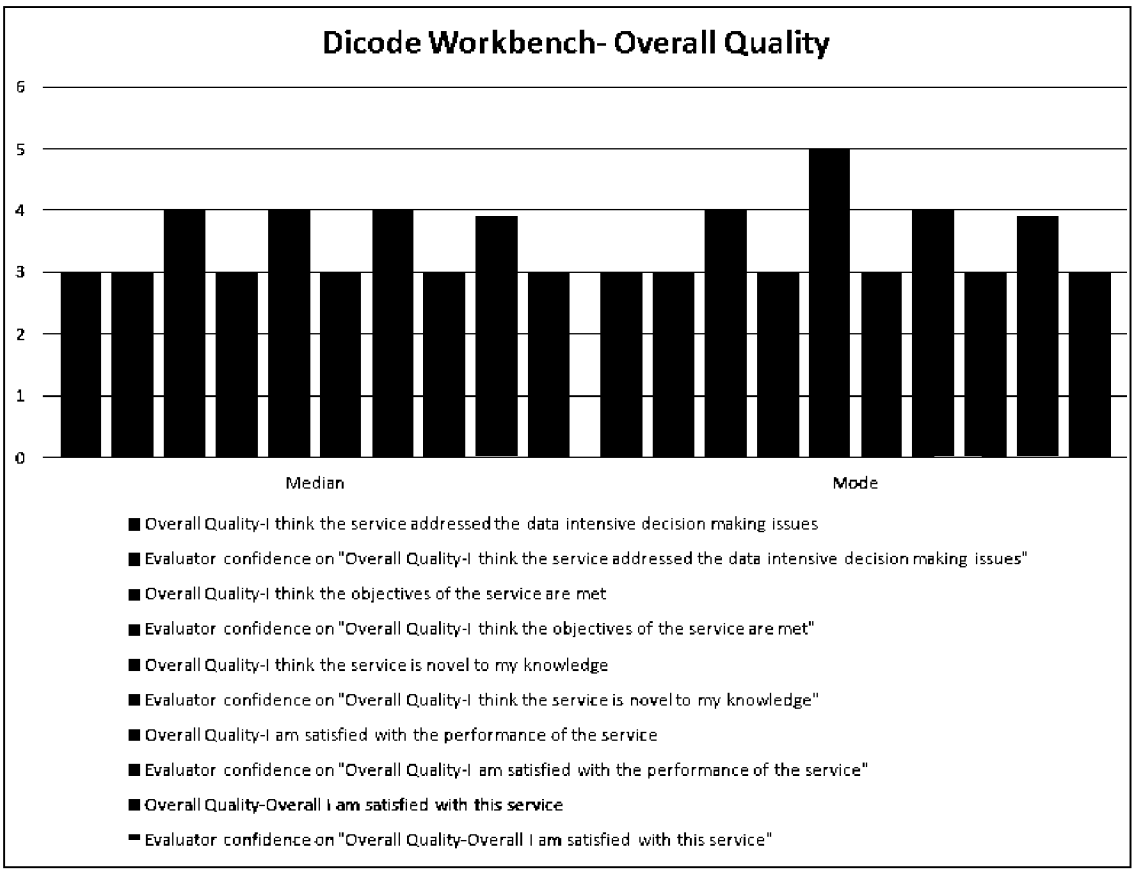

Figure 6. Evaluation results for the Dicode Workbench

Taking into account the feedback received from the first evaluation phase of the Dicode project, it is shown that our overall approach offers an innovative solution that reduces the data-intensiveness and overall complexity of real-life collaboration and decision making settings. Future work directions concern (i) the improvement of Dicode workbench in terms of its documentation, functionalities of user interface and overall performance, and (ii) its testing in various data-intensive contexts towards further assessing its applicability and potential.

We argue that the Dicode workbench provides a flexible, easy-to-use and scalable integration framework, which provides two types of integration: (i) *Light integration*, during which any web application can be used within the workbench as long as it publishes a REST-based interface. Users may register and integrate any web application by providing the application's URL. Such features make the workbench a powerful tool even for novice users. (ii) *Full integration*, where a set of mechanisms for exchanging data within the context of the workbench is defined, thus enabling the integrated services to exchange data. Such way of integrating services goes beyond mashup-based approaches, which in general do not allow sharing of information among applications; support for such features must be explicitly designed and implemented.

The Dicode workbench exploits and augments the underlying collective intelligence. Our approach is able to enhance the way that people think and act during a collaborative and data-intensive task by making it easier for them to mine and interpret data, and accordingly propose, amend or enhance collective actions [28]. Such advancements will ultimately shape innovative work methodologies for dealing with the problems of information overload and cognitive complexity in diverse collaboration and decision making contexts. Both individual and collaborative sense making will be augmented through the meaningful exploitation of prominent data processing and data analysis technologies. The solution offered is user-friendly and built on the synergy of human and machine intelligence. It masks the overall complexity of the underlying issues, thus allowing stakeholders to easily interact with large and complex data, providing them with meaningful recommendations upon which they can base their decisions and actions. Moreover, machine-tractable knowledge concerning the full lifecycle of collaboration and decision making is accumulated and maintained. Consequently, the solution offered by the Dicode project augments the productivity of stakeholders.

7. REFERENCES

[1] Ziegler, P., and Dittrich, K. R. 2004. Three Decades of Data Integration - All Problems Solved? In *18th IFIP World Computer Congress (WCC 2004)*, Volume 12, Building the Information Society,volume 2004, 3-12.

[2] Economist. 2010. Data, data everywhere, [http://www.economist.com/node/15557443].

[3] Kirsh, D. 2000. A Few Thoughts on Cognitive Overload, *Intellectica*, 1(30): 19-51.

[4] Eppler, M.J. and Mengis, J. (2004). The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines. *The Information Society*, 20(5):325–344.

[5] Newcomer, E., and Lomow, G. 2004. *Understanding SOA with Web Services* (1st ed.). Pearson.

[6] Pautasso, C., Zimmermann, O., and Leymann, F. 2008. Restful web services vs. "big" web services: making the right architectural decision. In: *Proceeding of the 17th international conference on World Wide Web. WWW '08*. ACM, New York, NY, USA, 805-814. DOI:http://dx.doi.org/10.1145/1367497.1367606.

[7] Yu, J., Benatallah, B., Casati, F., and Daniel, F. 2008. Understanding mashup development. *IEEE Internet Computing* 12 (5), 44-52. DOI:http://dx.doi.org/10.1109/MIC.2008.114

[8] Loton, T. 2008. *Working with Yahoo! Pipes, No Programming Required*. Lotontech Limited.

[9] Ennals, R., Brewer, E.,Garofalakis, M.,Shadle,M., and Gandhi, P. 2007. Intel mash maker: join the web. *SIGMOD Rec.* 36 (4), 27-33.

[10] Maglogiannis, I., Delakouridis, C., and Kazatzopoulos, L. 2006. Enabling collaborative medical diagnosis over the internet via Peer-to-Peer distribution of electronic health records. *Journal of Medical Systems*, 30(2):107-116. DOI=http://dx.doi.org/10.1007/s10916-005-7984-1

[11] Chronaki, C.E., Katehakis, G., Zabulis, X.C., Tsiknakis, M. and Orphanoudakis, S.C. 1997. WebOnCOLL: Medical collaboration in regional healthcare networks. *IEEE Trans. Inform. Technol. Biomed.*, 1(4):257 -269.

[12] Goble, C.A., Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., et al. 2010. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 38:W677–W682.

[13] Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., et al. 2010. BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic acids research*, 38:W689–W694.

[14] Swick, R.R. and Ackerman, M.S. 1988. The X toolkit: more bricks for building user interfaces, or widgets for hire. In *Usenix Winter 1988 conf.*, 221-228.

[15] W3schools – Iframe. [http://www.w3schools.com/tags/tag_iframe.asp]

[16] Cederholm, D. 2005. *Bulletproof Web Design: Improving flexibility and protecting against worst-case scenarios with XHTML and CSS*. New Riders Press.

[17] Thakker, D., Yang-Turner, F., Lau, L., and Dimitrova, V. 2011. Socio-technical Ontology Development for Modelling Sensemaking in Heterogeneous Domains. *Workshop on "Ontologies Come of Age in the Semantic Web" at the 10th International Semantic Web Conference (ISWC)* (Bonn, Germany, October 2011). OCAS2011. 60-71.

[18] iGoogle. [http://www.google.es/ig].

[19] Snir, M., Otto, S. W., Walker, D. W., Dongarra, J., and Huss-Lederman, S. 1995. *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA.

[20] Hickson, I. 2010. *HTML5 Web Messaging*. Editor's Draft 15, W3C (June 2010). [http://dev.w3.org/html5/postmsg/].

[21] cssQuery.[http://dean.edwards.name/my/cssQuery/].

[22] jQuery.[http://jquery.com/].

[23] Ben Alman – jQuerypostMessage. [http://benalman.com/projects/jquery-postmessage-plugin/].

[24] JSON. [http://www.json.org/].

[25] Karacapilidis, N., Rüping, S., Tsiliki, G. and Tzagarakis, M. 2012. Towards a Meaningful Analysis of Big Data: Enhancing Data Mining Techniques through a Collaborative Decision Making Environment. In: Markus Helfert, Chiara Francalanci and Joaquim Filipe (eds.), *Proceedings of the 1st International Conference on Data Technologies and Applications* (Rome, Italy, July 25-27, 2012). DATA 2012. 141-146.

[26] Nielsen, J. 1991. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing.

[27] Norman, D.A. 1998. *The Design of Everyday Things*. The MIT Press.

[28] Alag, S. 2008. *Collective Intelligence in Action*. Manning Publications Co.