Always-on Distributed Spreadsheet Mashups

Pierpaolo Baglietto (1), Fabrizio Cosso (2), Martino Fornasa (1), Simone Mangiante (1), Massimo Maresca (1), Andrea Parodi (2), Michele Stecca (1) *

(1) Computer Platform Research Center (CIPI) - University of Padova, University of Genova, Italy
(2) M3S S.r.l. - Genova, Italy

* p.baglietto@cipi.unige.it, fabrizio.cosso@gmail.com, m.fornasa@cipi.unige.it, simone.mangiante@cipi.unige.it, m.maresca@cipi.unige.it, a.parodi@m3s.it, m.stecca@cipi.unige.it

ABSTRACT

We present a platform that supports the creation of distributed data Mashups, implemented through the composition of multiple spreadsheets. The basic idea is to link cell ranges belonging to different spreadsheets in such a way to come up with a distributed spreadsheet. While such a behavior is already supported in private file spaces, our platform extends the operating principle and the functionality to the Internet in a secure way, using a SOA infrastructure as a communication bus. The platform consists of a client component, installed as a Plug-in in spreadsheet applications, and of a server component, accessible as a Web Service, that orchestrates the data exchanges among the client spreadsheets. One of the most original functionalities provided by the platform and described in the paper is that of guaranteeing the propagation of values along chains of linked spreadsheets even when some of the component spreadsheets are off-line.

In addition to the description of the platform the paper also includes the description of a Spreadsheet Mashup framework specifically suited to hierarchical organizations such as those of enterprises. The architecture of the framework is presented in the light of a real-life example.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Management, Design, Experimentation

Keywords

Spreadsheet Mashups, End user programming

1. INTRODUCTION

Mashups are composite applications that combine existing functionalities and data available over the Internet and/or in enterprise domains. Such functionalities and data are typically accessed through SOAP Web Services [2], REST Web Services [8], RSS Feed, Atom Syndication Format, JSON JavaScript Object Notation), and specific APIs (see, e.g., [19]).

Spreadsheets represent one of the most relevant classes of basic components that provide functionalities and data suitable to be integrated in Mashups. The spreadsheet relevance mainly derives from the fact that the spreadsheet user interface is universally known and widely used by non skilled programmers (e.g., business users) to maintain relationships among numerical parameters, to prepare statistics, to draw graphics, to make reports, etc. The usability and the diffusion of spreadsheets also led to the creation of tools that allow developing composite services by means of the well known paradigm based on grids and cells (see [11], [14], [15], [17], and [20]).

Available spreadsheet applications (e.g., Microsoft Excel, OpenOffice Calc) also allow to create Spreadsheet Mashups by configuring connections among cells belonging to different spreadsheets. In a Spreadsheet Mashup of two spreadsheets a "target spreadsheet" uses values associated to cells of a "source spreadsheet".

Spreadsheet Mashups exhibit the interesting "auto-updating" property: as soon as the value of a cell of a spreadsheet changes, such a modification triggers the automatic update of the cells of the target spreadsheet connected to such a cell. As a consequence in a chain of connected spreadsheets the update of a cell propagates to all the target spreadsheets in the chain.

Unfortunately Spreadsheet Mashups have the following two limitations:

- If a spreadsheet belonging to a chain of connected spreadsheets is not currently open, the update of the target spreadsheets located downstream in the chain takes place only when the spreadsheet is opened again.
- The spreadsheets participating in the Mashup must belong to the same domain, as communication among spreadsheets takes place in private environments.

In order to overcome these two limitations, we have developed an "Always-on Distributed Spreadsheet Mashups" architecture. The *always-on* attribute denotes the fact that we overcome the first limitation by supporting the propagation of data over the whole spreadsheet chain independently on the status of the spreadsheets in the chain and the *distributed* attribute denotes the fact that we

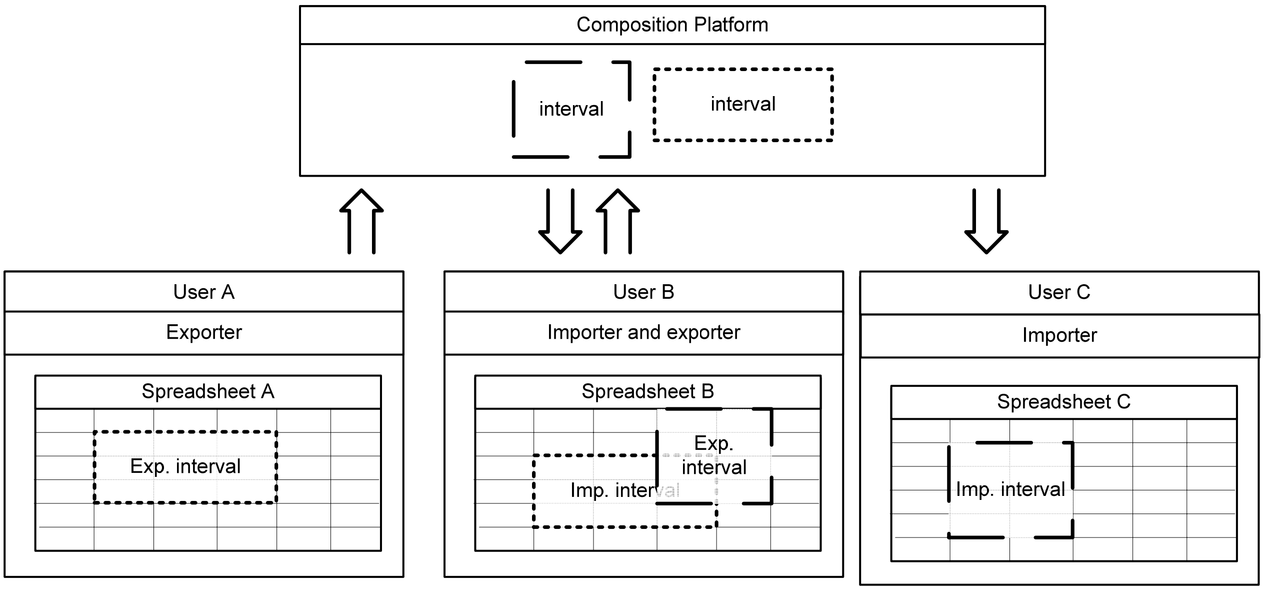


Figure 1. Spreadsheet Mashup operation.

overcome the second limitation by supporting the composition of spreadsheets over the Internet.

The paper is organized as follows: in Section 2 we describe the requirements, the architecture, and the operating principles of the platform, in Section 3 we present a Spreadsheet Mashup framework for hierarchical organizations, in Section 4 we describe the prototype implementation, in Section 5 we focus on the distinctive features of our proposal. In Section 6 we compare the proposed solution with some related work and in Section 7 we provide some concluding remarks.

2. REQUIREMENTS, PLATFORM ARCHITECTURE, AND OPERATING PRINCIPLES

2.1 Platform requirements

The Spreadsheet Mashup framework presented in this paper was designed to fulfil the following requirements:

Req. 1. Support of lightweight creation of Situational Applications: being widely used and well known by end users spreadsheet-based tools can be used as enablers for the creation of Situational Applications (i.e., a "rapidly created or contextually customized application that addresses immediate need of an individual or a community", see [5]) that do not require programming skills;

Req. 2. Support of reuse of already existing data: enterprise users often use spreadsheets to store information and/or data extracted from other company's software systems (e.g., ERP system). Thus it is important to provide a tool that allows to combine data belonging to different spreadsheets easily and friendly;

Req. 3. Compliance with the distributed and hierarchical structure of enterprises: the platform is supposed to manage the case in which the spreadsheets belonging to a composition are stored in different machines (e.g., different units of the same company may be geographically distributed). Moreover, the Spreadsheet Mashup framework must be aware of the different roles of the employees inside the company in order to apply the correct access rights according to the hierarchical structure of the company;

Req. 4. Support of automatic updates and Always-on distributed Composite Spreadsheets: the maintenance of complex relationships over distributed spreadsheet chains is necessary to preserve data consistency. In particular the system is supposed to synchronize the linked spreadsheets automatically even when one or more components of the distributed spreadsheet are offline.

2.2 Platform architecture and operating principles

The system is based on a client-server architecture: one or more clients interact with a composition platform across a network.

- The **client** part is a Plug-in module integrated in the spreadsheet application. The module interacts with the composition platform by means of Web Services and allows the user to configure the exportation and the importation of spreadsheet cells by means of a graphical interface.
- The **composition platform** exposes interfaces that support the execution of spreadsheet data exportation and importation, manages user accounts, synchronizes data across dependent spreadsheet and does version control. The platform hosts a spreadsheet engine that runs one or more instances of the spreadsheet application.

Figure 1 illustrates the system operation. Spreadsheet A exports a cell range (dotted area) and Spreadsheet B imports the same range. In the same way, Spreadsheet B can export a range (dashed

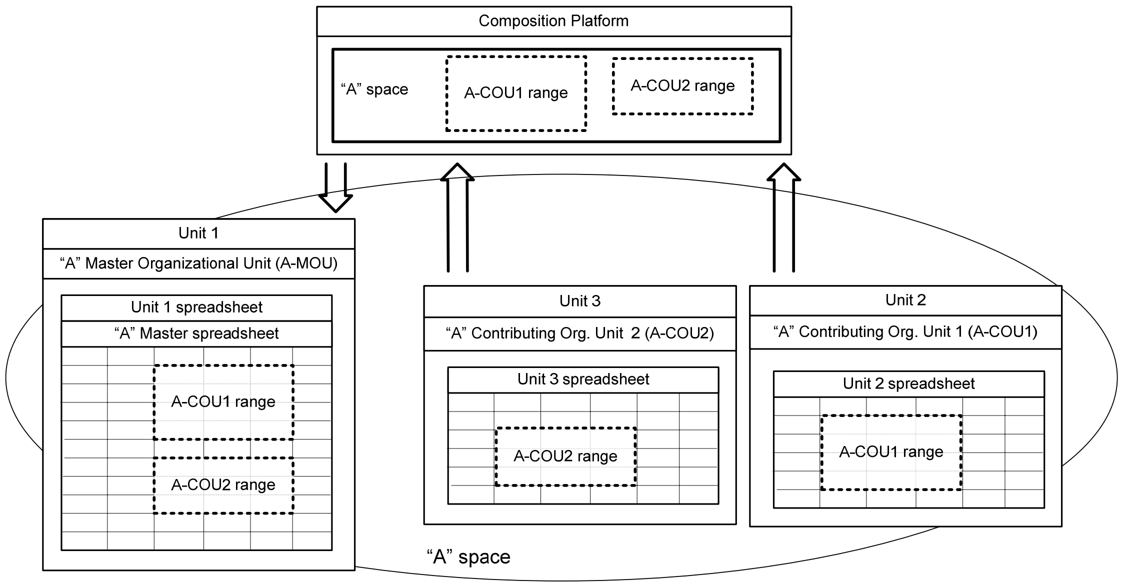


Figure 2. Hierarchical Enterprise Spreadsheet Mashup, single space.

area) that is imported by Spreadsheet C and so forth. In the above example, Spreadsheet A is an **exporter**, Spreadsheet C is an **importer,** and Spreadsheet B is both an importer and an exporter.

Data propagation is implemented through the following mechanisms:

- The **exporter contribution update** is periodically performed by each exporter Plug-in towards the platform. The Plug-in checks the spreadsheet for exported range modification and refresh the range image on the platform. In this way, the platform always contains the latest update of the exported ranges. If the user computer is offline, the Plug-in caches the modifications, and exports them to the platform as soon as possible.
- The **importer spreadsheet update** is periodically performed by the importer Plug-in. The Plug-in periodically polls the platform for new contributions or updates. As soon as a new contribution or update is detected the Plug-in imports the contribution and inserts it in the spreadsheet. In this way, the spreadsheet always contains the latest update and updates are propagated in real time.

If a user spreadsheet is both importer and exporter, and at least an exported cell is a function of an imported cell (via direct inclusion or formulas) we call it an **intermediate spreadsheet** (in our Figure 1 example, Spreadsheet B is an intermediate spreadsheet via direct inclusion of the dashed range in the dotted range). Intermediate spreadsheets allow building spreadsheet chains.

As we assume that a user personal computer can be switched-off in every moment, a scenario that includes one or more intermediate spreadsheets presents a crucial issue: if an intermediate spreadsheet computer is offline (e.g., switched off or not connected to the network), the data updates cannot propagate through the chain. In order to overcome this issue, the system should support a feature, namely the **data propagation function,** which works as follows:

- When a client Plug-in realizes that the spreadsheet is intermediate it uploads the entire spreadsheet on the platform. Every time an intermediate spreadsheet is updated the Plug-in performs a new upload.
- When an intermediate spreadsheet is offline, the platform runs a local spreadsheet engine (i.e. an instance of the spreadsheet software) in order to recalculate the exported ranges based on fresh import ranges.

In Figure 1 example, Spreadsheet B is an intermediate spreadsheet, and it is uploaded on the platform in order to assure data updates from A to C.

3. HIERARCHICAL ENTERPRISE SPREADSHEET MASHUP (HESM)

The hierarchical nature of enterprise organizations leads to the Spreadsheet Mashup use case described in this section. We face the case in which an enterprise organizational unit, which we call Master Organizational Unit (MOU), establishes and maintains a global view of a business through a spreadsheet based representation. A number of enterprise organizational units, called Contributing Organizational Units (COU), hierarchically inferior with respect to the MOU, contribute their data to the MOU view as to allow the MOU to establish and maintain the business global view as required.

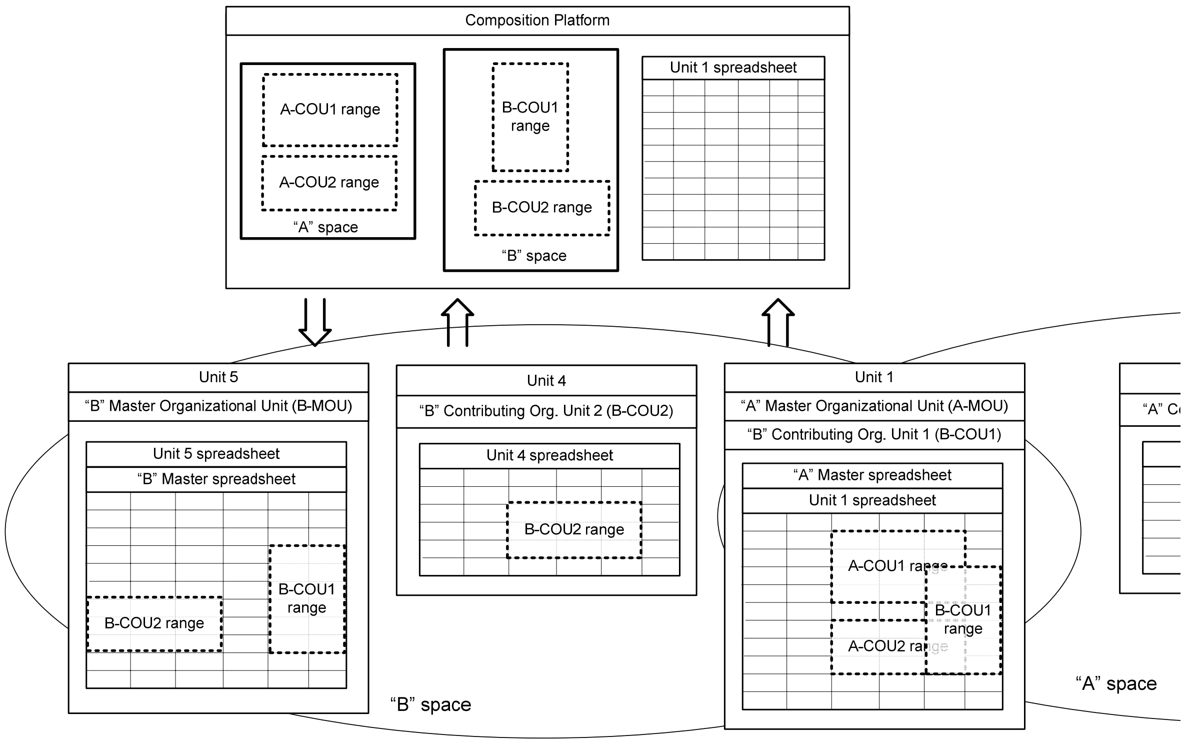


Figure 3. Hierarchical Enterprise Spreadsheet Composition, chain of spaces.

In order to link the reference scenario to reality we describe the following case. A certain number of Car Dealers (CDs) are organized in a hierarchical way: all the CDs located in the same area are managed by an Area Sales Manager (ASM) whereas all the ASMs of a region are managed by a Region Sales Manager (RSM) who needs to have and maintain regional car sales figures. In such a scenario, we suppose that all the actors keep track of the cars sold on their local spreadsheets (note: the CD Spreadsheets are compliant with a template provided by the ASM and the ASM Spreadsheets are compliant with a template provided by the RSM). All the spreadsheets belong to the same "Spreadsheet Mashup" in which an update done by an actor in the lowest level of the hierarchy (i.e., a CD) triggers the update of the spreadsheets located in the higher levels of the hierarchy (e.g., both the ASM and the RSM). In this way all the sales data are constantly up to date.

Hierarchical Enterprise Spreadsheet Mashup (HESM) is a Spreadsheet Mashup framework based on an entity called **Space**. A **Space** is associated to a space owner, i.e., the MOU, and to a number of contributors, i.e. the COUs. The MOU creates and maintains the so called **Master Spreadsheet**, the cells of which include references to data exported by the COUs spreadsheets.

A **Space** is created by the MOU, which assigns membership privileges to the COUs. Each COU is not aware of the presence of the other COUs and is only allowed to see and control its own data. The HESM configuration process consists of the following phases:

- The MOU creates the **Master Spreadsheet**;
- The MOU creates the so called **Spreadsheet Contribution Templates (SCT);**
- The MOU sends the SCTs to the COUs;
- The COUs:
  - import the SCTs;
  - create the appropriate links between the SCTs and the internal data structures so as to have the SCTs dynamically updated, and
  - perform **SCT acceptance registration**.

Figure 2 shows a space called "A" with its MOU and two COUs.

In the car dealers scenario described at the beginning of this section, each ASM creates a space to collect sales figures. So, each ASM is the MOU of a space in which each CD belonging to the corresponding area is a COU.

The **Space** concept is based on a unidirectional relation from COUs to MOU. In order to allow more complex and hierarchical interactions, we exploit here the *spreadsheet chain* concept described in the previous section. In a *chain of spaces*, a user can be at the same time the MOU (data importer) of one space and one of the COUs (data exporters) of another space. In this way, contributions can propagate through chains of spaces.

In the car dealer example, each ASM is at the same time the MOU of a space and one of the COUs of a space owned by the RSM. In this way the sales figures introduced by CDs propagate along the spreadsheet chain and reach the RSM spreadsheet. Such an example is shown in Figure 3.

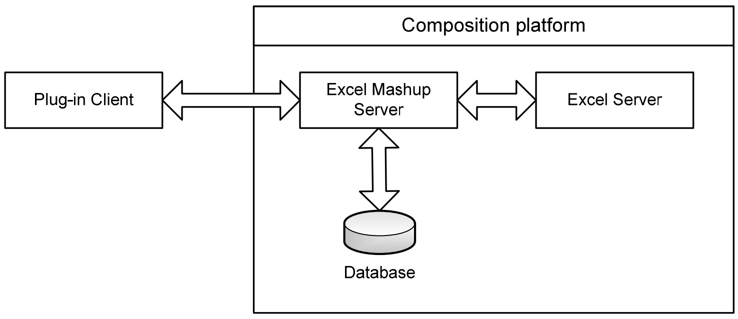


Figure 4. Prototype architecture.

Of course, a chain of spaces relies on the data propagation function described in previous section. With reference to Figures 2 and 3, if "Unit 1" is offline and the platform receives fresh data from "Unit 2", the platform runs the spreadsheet engine in order to recalculate B-COU1 range using fresh A-COU1 and A-COU2 ranges, and propagates the result to "Unit 5".

4. PLATFORM IMPLEMENTATION

Microsoft Excel is by far the most widely used spreadsheet application. Hence we chose to build the platform prototype on top of this Microsoft technology. The prototype architecture is depicted in Figure 4: the two main platform components are the Plug-in Client and the Composition Platform. We adopted an architecture based on a central Composition Platform in order to fulfill Req. 4 by providing a central repository that performs automatic synchronization even when one or more contributor are offline.

4.1 Plug-in Client

The Plug-in Client, written in C# using Visual Studio Tool for Office[1], provides a graphical interface that allows users to control the import/export operations; the interface maintains the same look and feel as the other Microsoft Excel functionalities. The Plug-in interacts with the Composition Platform through Web Services technology, as Web Services are widely developed and supported in enterprise environment. In particular we adopt standard SOAP messaging..

Users perform **exportation** by selecting a cell range and launching the *export* command on the Plug-in graphical interface (see Figure 5). Then, the user is prompted to insert an identifier, a brief description, a list of the users enabled to view and import the range, and to select the exportation mode (automatic or manual): in the automatic mode, every cell range modification is automatically exported to the composition platform whereas in the manual mode the user has to manually trigger the exportation.

Cell range to be exported is converted to an XML document and wrapped in a SOAP request that is sent to the Excel Mashup Server (see Figure 7). The Excel Mashup Server module stores the exported cells in the database. If the automatic mode is selected, each successive exported range modification is intercepted by the Plug-in and sent to the server.

[1] http://msdn.microsoft.com/it-it/vsto/default(en-us).aspx

Additionally, the Plug-in stores metadata on the spreadsheet, including exportations and importations information, user identification information and server address.

A user performs **importation** by browsing a list of possible ranges that he is allowed to import (see Figure 6). When the user selects a range, he is prompted to select the importation mode (automatic or manual). Then, the Plug-in sends a SOAP request for the requested interval to the Excel Mashup Server. The Excel Mashup server replies with the latest version of the requested interval and the Plug-in prepares the spreadsheet with the necessary space to fit the imported cell range. Finally, the range is inserted in the local spreadsheet.

If the automatic mode is selected, the Plug-in refreshes the imported range by periodically polling the Excel Mashup Server. If the manual mode is selected, the user has to manually trigger such a check. The imported range has to be constantly monitored by the Plug-in because the user might move or change it, thus invalidating the synchronization.

In order to perform the data propagation function (see Section 2.2), if the Plug-in realizes that the spreadsheet contains an automatic exportation dependent on an automatic importation, it uploads the entire spreadsheet on the platform. Otherwise, only cell ranges are exchanged.

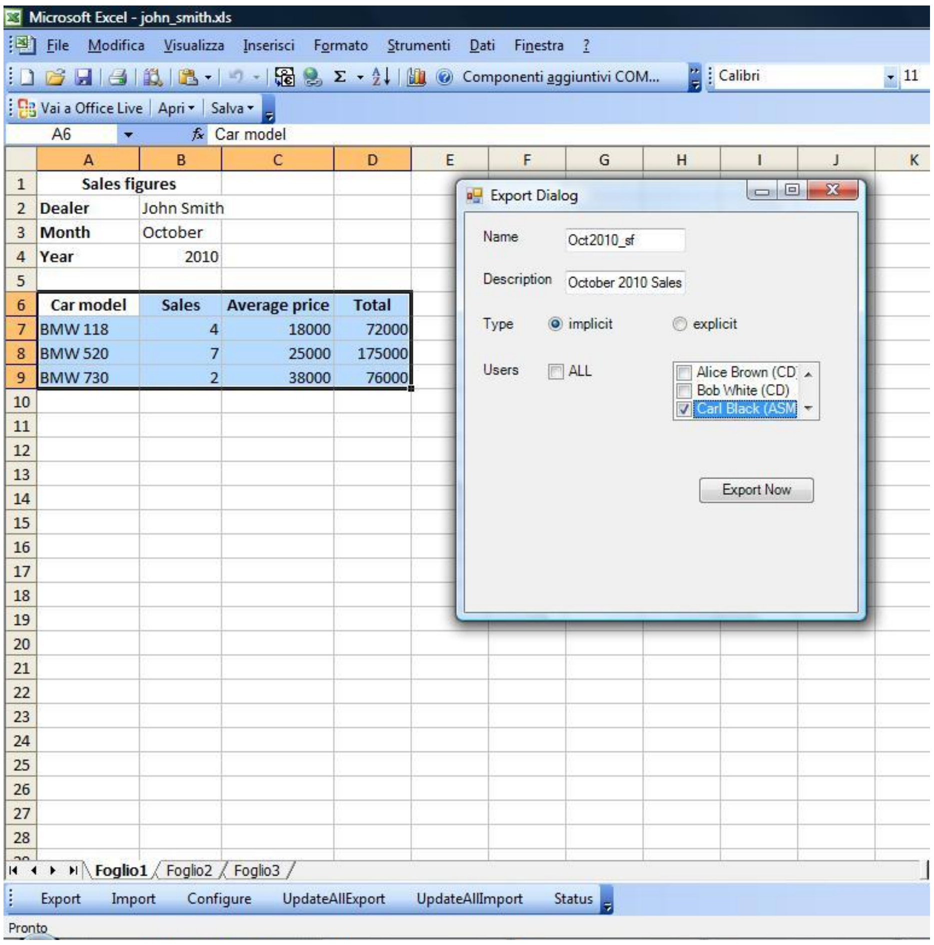
4.2 Composition Platform

The Composition Platform includes the following three modules:

- Excel Mashup Server
- Excel Server
- Database

The Excel Server is a .NET application which controls a Microsoft Excel instance equipped with the Plug-in Client. When the Excel Mashup Server realizes that an intermediate spreadsheet should be re-evaluated, it invokes the Excel Server module, which opens the file, performs the calculations over the new values and updates the exportations. The Excel Mashup Server module and the Excel Server module interact through a dedicated protocol.

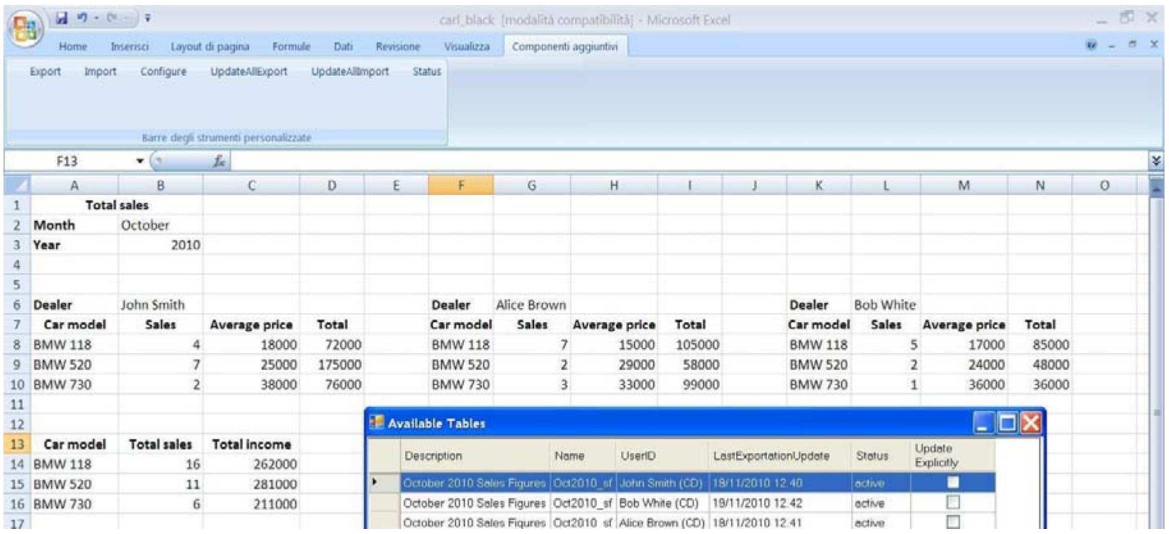
The Database module is a Microsoft SQL Server instance; it stores user's credentials, exported cell ranges and dependencies among data along spreadsheet chains. Such dependencies are tracked in order to determine if the Excel Server module should re-evaluate an offline Excel file uploaded on the platform after a range update.


Microsoft Excel - john_smith.xls
File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?
Componenti aggiuntivi COM...
Calibri 11
Vai a Office Live Apri Salva
A6 Car model
Foglio1 Foglio2 Foglio3
Export Import Configure UpdateAllExport UpdateAllImport Status
Pronto
Export Dialog
Export Now

	A	B	C	D
1	Sales figures			
2	Dealer	John Smith		
3	Month	October		
4	Year	2010		
5				
6	Car model	Sales	Average price	Total
7	BMW 118	4	18000	72000
8	BMW 520	7	25000	175000
9	BMW 730	2	38000	76000

Name: Oct2010_sf
Description: October 2010 Sales
Type: [✓ implicit] explicit
Users: ☐ ALL — ☐ Alice Brown (CD) ☐ Bob White (CD) ☑ Carl Black (ASM

Figure 5. Cell range exportation. With reference to the car dealer example presented in Section 3, car dealer *John Smith* is exporting Oct. 2010 sales figures to ASM *Carl Black*.


carl_black [modalità compatibilità] - Microsoft Excel
Home Inserisci Layout di pagina Formule Dati Revisione Visualizza Componenti aggiuntivi
Export Import Configure UpdateAllExport UpdateAllImport Status
Barre degli strumenti personalizzate
F13
Available Tables

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Total sales													
2	Month	October												
3	Year	2010												
4														
5														
6	Dealer	John Smith				Dealer	Alice Brown				Dealer	Bob White		
7	Car model	Sales	Average price	Total		Car model	Sales	Average price	Total		Car model	Sales	Average price	Total
8	BMW 118	4	18000	72000		BMW 118	7	15000	105000		BMW 118	5	17000	85000
9	BMW 520	7	25000	175000		BMW 520	2	29000	58000		BMW 520	2	24000	48000
10	BMW 730	2	38000	76000		BMW 730	3	33000	99000		BMW 730	1	36000	36000
11														
12														
13	Car model	Total sales	Total income											
14	BMW 118	16	262000											
15	BMW 520	11	281000											
16	BMW 730	6	211000											

Description	Name	UserID	LastExportationUpdate	Status	Update Explicitly
October 2010 Sales Figures	Oct2010_sf	John Smith (CD)	18/11/2010 12.40	active	☐
October 2010 Sales Figures	Oct2010_sf	Bob White (CD)	18/11/2010 12.42	active	☐
October 2010 Sales Figures	Oct2010_sf	Alice Brown (CD)	18/11/2010 12.41	active	☐

Figure 6. Cell range importation. With reference to the car dealer example presented in Section 3, ASM *Carl Black* is importing sales figures from three car dealers.

The Excel Mashup Server is an ASP.NET application which contains a set of functions that allow clients to authenticate themselves and perform CRUD operations on exported and imported data. It serves client SOAP requests by invoking the other modules when necessary.

As a future work the Composition Platform will be equipped with a versioning mechanism, in order to maintain data exportation changes history. Users will be allowed to browse different versions during the importation process.

5. DISCUSSION

To the best of our knowledge, the existing Spreadsheet Mashup platforms do not support neither the "distributed" feature, which is the possibility to compose spreadsheets over the Internet nor the "always-on" feature, which is the possibility to support data propagation through spreadsheet even when some of the spreadsheets are off-line. Such features are of high relevance for enterprises.

We notice that most of the existing Spreadsheet-based Mashup systems focus on graphical service creation platform and support of heterogeneous data sources. Our platform does not include a graphical service creation environment and only processes spreadsheet data sources. The absence of a graphical service creation environment is due to the fact that we consider spreadsheet composition as a result of a joint action of multiple spreadsheet owners. However it is also possible to use an existing graphical service creation environment to maintain a Mashup-oriented view of spreadsheet composition, i.e., a view in which the component spreadsheets appear as blocks while the links among them appear as edges.

The platform described in this paper can be seen as complementary with respect to the platforms that import different data sources and convert them to spreadsheet cells. In this sense the contribution of our approach is the adoption of the spreadsheet cell as an interaction paradigm, with no regard to the fact that the cells may derive from different data sources. For example, some spreadsheet applications (e.g., Microsoft Excel 2007) support data importation from external data sources in which the information updates are provided as RSS feeds. Thanks to such a feature, spreadsheet applications can be seen as gateway between such external data sources and the internal spreadsheet world. For instance, Microsoft Excel 2007 provides the "Import External Data" function that allows importing an XML file (e.g. an RSS Feed) and keeping it aligned with the external source. Leveraging both this native feature and the architecture described in Section 2.2, it is possible to create data Mashups that merge spreadsheet data with RSS Feed items in an Always-on/Distributed way.

The Always-on feature is enabled by a remote server that keeps all the links among the spreadsheet participating in a composition constantly updated even when one or more of these are offline. This approach naturally fits with the emerging Cloud Computing paradigm [16] where servers can be deployed and executed "in the cloud" to reduce costs thanks to the "pay as you go" schema [4]. For instance, an "Always-on Distributed Spreadsheet Mashup" service provider might implement the system as an Amazon Machine Image – AMI deployed on the Amazon EC2 Infrastructure as a Service platform [3]. From the end user point of view, i.e. who actually composes spreadsheets, this service is delivered according to the Software as a Service paradigm [16] because the Composition feature is simply used without any knowledge about function interval implementation (e.g., Car Dealers and Managers mentioned in Section 3 will simply use their spreadsheets without caring about where and how the systems works).

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
       xmlns:soap="http://schemas.xmlsoap.org/
soap/envelope/"
       xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<exportData xmlns="http://www.m3s.it/MashUp">
  <request>
    <UserID>John Smith (CD)</UserID>
    <Name>Oct2010_sf</Name>
    <ExportationID>2404c4e3-28ea-458e-ad25-
43f4715a5924</ExportationID>
    <Description>October 2010 Sales
Figures</Description>
    <isExplicit>false</isExplicit>
    <AllowedUsers>
       <string>Carl Black (ASM)</string>
    </AllowedUsers>
    <LastUpdate>2010-11-
22T10:52:29Z</LastUpdate>
    <RequestRangeValues> [truncated] &lt;?xml
version="1.0" encoding="utf-
8"?&gt;&lt;ExportedRange&gt;&lt;sheetName&gt;Fog
lio1&lt;/sheetName&gt;&lt;rowCount&gt;4&lt;/rowC
ount&gt;&lt;columnCount&gt;4&lt;/columnCount&gt;
&lt;startingRowIndex&gt;6&lt;/startingRowIndex
    </RequestRangeValues>
    <Status>active</Status>
  </request>
  <sourceExportations/>
</exportData>
</soap:Body>
</soap:Envelope>
```

Figure 7. Sample exportation SOAP message.

6. RELATED WORK

Mashups are a relatively new technology aimed at the support of lightweight creation of composite services. Because of their young age a clear classification of the different Mashup approaches is still missing in the literature. Although both academia (see, e.g., [6] and [18]) and industry (see, e.g. [10], [12], and [13]) are working on the design and implementation of Mashup platforms, unfortunately these platforms cannot be easily compared because of the lack of a systematic classification of the available Mashup approaches (see [1] and [9] and for the proposal of some classification models). The specific domain of spreadsheet-based Mashup tools, which this paper considers, was faced in the proposals reported below.

IBM - A1 [14] allows including JAVA objects (usually used as adapters to external data sources) in spreadsheets but in this case the developer is required to deal with the Java programming language.

StrikeIron SOA Express for Excel [21] leverages on the spreadsheet technology as Mashup creation environment where the developer can import data provided through the Web Service technology by dragging & dropping the WSDL (Web Service Description Language) file into the spreadsheet. The Plug-in provided by this solution is in charge of automatically update the spreadsheet when external data change.

Extensio Extender for Microsoft Excel [7] uses spreadsheets as an integration point where users can easily import and combine data provided in different formats. It also provides a functionality that allows to update data in back-end sources directly from Excel (e.g., if a spreadsheet cell is importing data from a database, it is possible to propagate the changes of that cell to the corresponding field in the back-end database).

AMICO:CALC [17] introduces an extension of the language already defined for the traditional spreadsheets formulas (i.e., it adds new functions) to enable the combination of SOAP Web Services, XML-RPC, etc. into spreadsheets.

Husky [20] supports the creation of composite services by introducing a semantics related to how data and formulas are spatially placed on the spreadsheets.

Finally SpreadATOR [15] is another project in which the Mashup creation consists dragging and dropping external data in the spreadsheet.

Compared to the system proposed in this paper, the IBM – A1 approach requires programming skills to users while all the other solutions focus on the integration of different data sources in spreadsheets. Our approach, on the contrary, aims at the composition of spreadsheets in a distributed and hierarchical way which reflects the structure of enterprises as explained in the business scenario in Section 3. Moreover the solution discussed in this paper proposes an Always-on spreadsheet composition that keeps working even when one or more components of the Mashup are not available.

7. CONCLUSION

We have presented a platform that supports the creation of distributed Spreadsheet Mashups, implemented through the composition of multiple spreadsheets. The platform extends a functionality already provided by spreadsheet applications in private file spaces by supporting secure spreadsheet composition over the Internet.

The paper presents and discusses the platform, which guarantees data propagation along chains of connected spreadsheets even when the spreadsheets are off-line, and one possible Spreadsheet Composition framework, specifically suited for hierarchical application domains such as those of enterprises.

A prototype based on Microsoft environment was developed to demonstrate the concept. The prototype, described in the paper, is currently being extended to the open source environment.

At the moment the prototype is being used as a tool to establish experimental testbeds in a number of application environments, both in private domains and in public domains. The objective is to refine the platform specifications, to update the software and to test the reaction of users to the adoption of the platform.

8. REFERENCES

[1] Agnes Koschmider, et al., Elucidating the Mashup Hype: Definition, Challenges, Methodical Guide and Tools for Mashups, Proceeding of "2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)", April 2009

[2] Alonso G., et al., Web Services. Springer Verlag, 2003

[3] Amazon Elastic Cloud Computing, http://aws.amazon.com/ec2/

[4] Armbrust, M., et al., “Above the Clouds: A Berkeley View of Cloud Computing,” UCB/EECS-2009-28

[5] Balasubramaniam S., et al., Situated Software: Concepts, Motivation, Technology, and the Future, IEEE Software, vol. 25, no. 6, pp. 50-55, Nov./Dec. 2008, doi:10.1109/MS.2008.159

[6] Braga D., et al., Mashing up search services, Internet Computing, Volume 12, Number 5, September-October 2008, IEEE Press, Pages 16-23

[7] Extensio Extender for Microsoft Excel, www.extensio.com

[8] Fielding, R., T. Representational state transfer (REST). Chapter 5 in Architectural Styles and the Design of Network based Software Architectures. Ph.D. Thesis, University of California, Irvine, CA, 2000

[9] Fischer T., et al., An Evolutionary Algorithm for Automatic Composition of Information-gathering Web Services in Mashups, ecows, pp.39-48, 2009 Seventh IEEE European Conference on Web Services, 2009

[10] Gurram, R. et al., A web based mashup platform for enterprise 2.0, Proc. of the International Workshops on Web Information Systems Engineering, pp.144–151, 2008

[11] Hoang, D. D., et al., An analysis of spreadsheet-based services mashup. In Proceedings of the Twenty-First Australasian Conference on Database Technologies - Volume 104 (Brisbane, Australia, January 01 - 01, 2010). H. T. Shen and A. Bouguettaya, Eds. Conferences in Research and Practice in Information Technology Series. Australian Computer Society, Darlinghurst, Australia, 141-150

[12] IBM Mashup Center, www.ibm.com/software/info/mashup-center/

[13] JackBe Presto Service, http://www.jackbe.com

[14] Kandogan, E., et al., A1: End-user programming for web-based system administration, in `UIST 2005', ACM, New York, NY, USA, pp. 211-220

[15] Kongdenfha, W., et al., Rapid development of spreadsheet-based web mashups. In Proceedings of the 18th international Conference on World Wide Web (Madrid, Spain, April 20 - 24, 2009). WWW '09. ACM, New York, NY, 851-860. DOI= http://doi.acm.org/10.1145/1526709.1526824

[16] Mell P., and Grance T., The NIST Definition of Cloud Computing, NIST Report, July 2009

[17] Obrenović Ž. and Gašević D., "End-User Service Computing: Spreadsheets as a Service Composition Tool," IEEE Trans. Services Computing, vol. 1, no. 4, 2008, pp. 229–242

[18] Pietschmann S., et al., A Thin-Server Runtime Platform for Composite Web Applications, iciw, pp.390-395, 2010 Fifth International Conference on Internet and Web Applications and Services, 2010

[19] Programmable Web Portal, www.programmableweb.com.

[20] Srbljic, S., et al., (2007), `Husky editor'. http://www.husky.fer.hr

[21] StrikeIron SOA Express for Excel, http://www.xignite.com/ExcelAddIn/ExcelAddInMain.aspx