The Spreadsheet Space

Massimo Maresca

Abstract

The Spreadsheet Space is a virtual space that supports spreadsheet-based collaboration and analytics. In the Spreadsheet Space, spreadsheet cells reference cells in other spreadsheets, possibly owned by other users and located in other administrative domains, as well as data exposed by software platforms and databases. Cross-spreadsheet references support Spreadsheet Synchronization, i.e., the automatic alignment of spreadsheets among them and with external software platforms, and Spreadsheet Overlays, i.e., coordinated graphs of synchronized spreadsheets and software platforms. The paper introduces the Spreadsheet Space, outlines the software architecture that supports it, reports the feedback obtained in field experiments and compares the Spreadsheet Space approach with existing spreadsheet interconnection mechanisms.

1 Introduction

Spreadsheets are a widespread form of programming, heavily used in a number of application domains ranging from accounting to engineering, marketing, health, and business in general. They are intuitive, they mix development and run time, they support stepwise development, they provide a natural environment for model development, they match the culture of organizations, and, most of all, they do not require code development skills. Domain experts can use spreadsheets at the level of complexity they need and rapidly obtain useful results [1] .

Spreadsheets belong to the category of End User Computing, in which people focus on problems rather than on software technology [2]. Although End User Computing evokes an amateur approach, spreadsheet development is a form of programming [3], and, as such, it must be considered a professional activity subject to engineering rules, in particular for what concerns design, documentation, debugging, testing, maintenance, quality, etc. [4]. Speaking in quantitative terms, the number of End User programmers was recently estimated to be more than an order of magnitude larger than that of mainstream programmers [5]. Today, the number of installations of Microsoft Excel, the most widespread spreadsheet tool included in the Microsoft Office suite, is over one billion[1]. Although most users only exploit a small fraction of the suite functionalities, the number is impressive and demonstrates that the spreadsheet phenomenon deserves careful attention and that research on spreadsheet has a strong impact.

Recently spreadsheets have evolved from personal office tools aimed at improving people productivity to enterprise level tools aimed at supporting collaboration and analytics. With regard to collaboration, spreadsheets were shown to provide a simple and flexible framework supporting collaborative model development [6]. More than that, however, spreadsheets have become the “de facto” standard for tabular data exchange over email and more recently, with the advent of Cloud Computing, the reference tool for tabular data sharing.

With regard to analytics, people traditionally use spreadsheets for personalized post processing and presentation of data exported by Software Platforms, as demonstrated by the ubiquitous “Export to Excel” command. The evolution toward Big Data has created an even stronger need for a smart combination of Cloud computing and End User computing as well as for effective techniques to move data back and forth between the Cloud and the local machines where the Data Scientists work [7]. To improve integration with Software Platforms, spreadsheets

[1] http://money.cnn.com/2013/11/13/technology/enterprise/microsoft-office-google-docs/

have recently incorporated new features to support SQL queries[2] [8], and were proposed for Real Time data access and visualization [9] and data integration and mashup [10].
None of the proposed spreadsheet evolutions, however, leverages the nature of spreadsheets, i.e., the presence of cell references or links. The Spreadsheet Space, on the contrary, takes advantage of the typical permanent asymmetric cell links supported by spreadsheets: the novelty is that these links cross the spreadsheet boundaries over the Internet with no limitations, i.e., they connect spreadsheets stored in different systems and belonging to different users. The Spreadsheet Space leverages spreadsheet interconnection and composition technology [11] to provide a virtual space for tabular data exchange, thus opening a set of opportunities for the development of new spreadsheet utilization models.
The paper proceeds as follows. Section 2 introduces the Spreadsheet Space and presents its operation model, Sections 3 describes the Spreadsheet Space software architecture, Section 4 reports the feedback obtained in field experiments, Section 5 compares the Spreadsheet Space approach against existing solutions for spreadsheet interconnection, and Section 6 provides a concluding remark.

2 The Spreadsheet Space

2.1 Spreadsheet Synchronization

The idea of connecting spreadsheets over the Internet derives from the observation that spreadsheets are based on cell references and that cell references lend themselves to being extended to multiple files, possibly owned by different users and located in different administrative domains. In the same way as internal spreadsheet references denote dependencies among spreadsheet cells, cross-spreadsheet references denote dependencies among cells of different spreadsheets. Thanks to cross-spreadsheet dependency, a cell update in a spreadsheet may trigger cell updates in other spreadsheets, both directly, in those spreadsheets that reference the updated cell directly, and indirectly, in those spreadsheets that reference the updated cell through other intermediate spreadsheets. We call such a behavior inter-spreadsheet synchronization, or more simply Spreadsheet Synchronization.

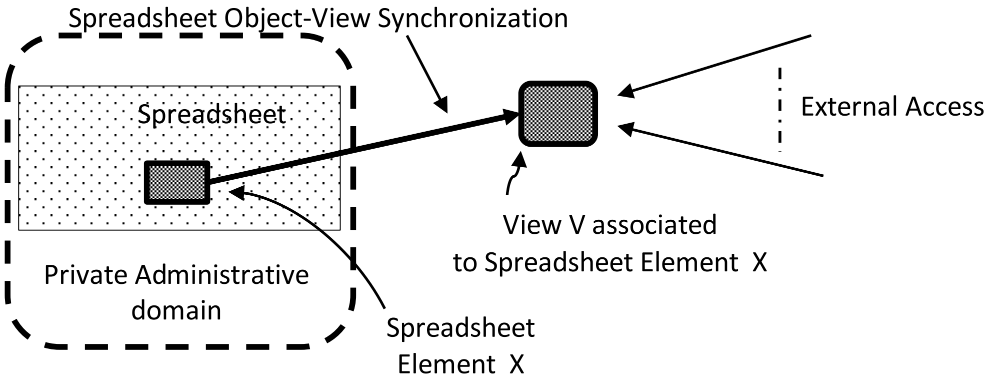
2.2 Spreadsheet Space Operation Model

At a first glance, cross-spreadsheet references and Spreadsheet Synchronization would seem to require just a simple extension to the syntax of reference, based for example on the inclusion of a file name or of a Universal Resource Identifier (URI), to point to other spreadsheets. Unfortunately, such an extension is not sufficient to support Spreadsheet Synchronization over the Internet, as the problem is more complex. First, cross-spreadsheet references may point to spreadsheets belonging to different users and hosted in administrative domains not accessible to the referencing user. Second, cross-spreadsheet references may point to inactive spreadsheets, as spreadsheets may reside in PCs and Laptops that are not permanently active and connected to a network. Third, spreadsheet cell update is controlled by an event management system that only works inside single spreadsheets and not among spreadsheets.
Because of these issues, a platform supporting Spreadsheet Synchronization must include the following three functionalities:

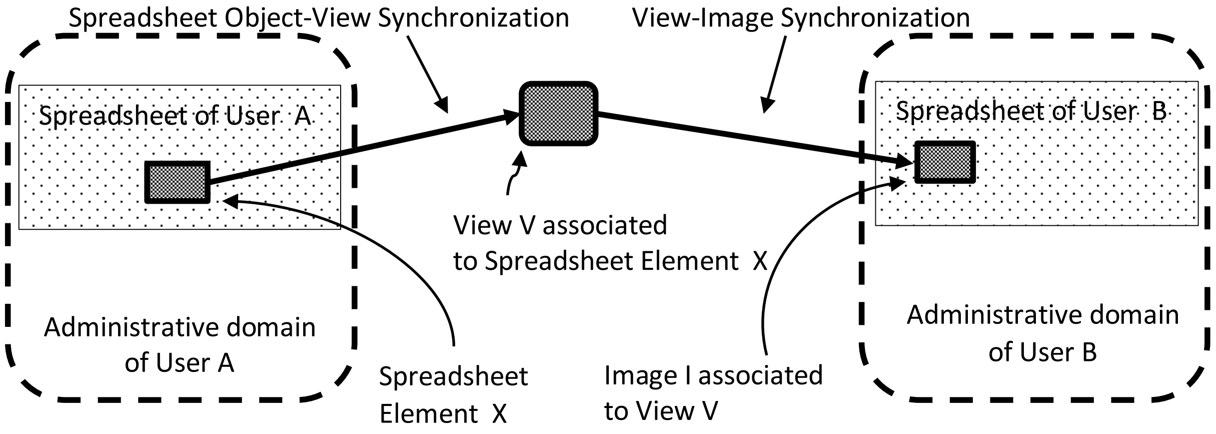
- a functionality that allows spreadsheet users to grant read access rights on their spreadsheets at the cell granularity to specific users, identified on a personal basis;
- a functionality that allows spreadsheet users to create and maintain persistent copies of the cells made available for external reference;
- a functionality that triggers the update of a spreadsheet cell upon the update in a cell of another spreadsheet.

[2] For example Microsoft Power Query (https://support.office.com/), and Google Docs Query (https://support.google.com/docs).

We introduce two abstractions, respectively called View and Image, to support these three functionalities. A View is defined as a window on a Spreadsheet Element, i.e., on a Cell Range or on a Table[3], created by a spreadsheet user, and is implemented as a persistent copy of the Spreadsheet Element, constantly synchronized with it, and made available to a set of target users for external reference (Fig. 1a). The View abstraction directly supports the first two aforementioned functionalities, as it is under the control of the spreadsheet owner, is concerned with a Spreadsheet Element (at the least, a single cell) rather than with an entire spreadsheet, and is made accessible to a set of users identified on a personal basis through a persistent copy [4].



a. View



b. Spreadsheet Synchronization through Views and Images

Fig. 1: Views and Images

An Image is a read-only local copy of an external View in a spreadsheet. An Image maintains the alignment with the corresponding View and, through it, with the Spreadsheet Element associated to the View. Users create Images of remote Views in spreadsheets and then reference cells belonging to other spreadsheets through internal references to Image cells. Thus, spreadsheet synchronization is the result of the combined action of Spreadsheet Element-View synchronization, i.e., the alignment of the Views with the associated Spreadsheet Elements, and View-Image synchronization, i.e., the alignment of the Images with the corresponding Views (Fig.

[3] According to spreadsheet terminology, a Cell Range is a fixed-size two-dimensional array of cells whereas a Table is an extensible record sequence associated to formulas that automatically adapt to extension/contraction.

[4] It is worth pointing out that in other contexts, the term "View" has different meanings. In particular, in database theory, a View is the result of a predefined SQL query, and, as such, it is a snapshot taken at the time the SQL query is executed. On the contrary, in the Spreadsheet Space, a View is a window on a data set that may evolve over time.

1b). An external control system orchestrates the combined action, thus implementing the third aforementioned functionality.
An important generalization of the View abstraction derives from the fact that a View does not necessarily need to be associated to spreadsheet data whereas it can be associated to any tabular data. This allows any Software Platform, and not only spreadsheets, to create Views. Thanks to such an extension, spreadsheets can include Images of Views created by Software Platforms and maintain the alignment with the Software Platform data, thus becoming Software Platform clients. In addition, spreadsheets of different vendors can interoperate with each other.

2.3 Spreadsheet Space Primitives

View-Image operation is based on a Control Plane, in charge of View-Image creation, and on a Data Plane, in charge of View-Image synchronization. The two planes include the following primitives:
Control Plane Primitives

- Expose: Through "Expose", a user creates a View of a Spreadsheet Element: the View can be accessible to everybody or directed to a set of target users.
- Join: Through "Join", a user creates a local Image of a View.

Data Plane Primitives

- Update View: Through "Update View", a spreadsheet uploads the current content of a Spreadsheet Element exposed through a View to the corresponding View.
- Update Image: Through "Update Image", a spreadsheet downloads the current content of a View to the corresponding Image.

While exposing and joining Views always requires explicit user intervention, synchronization may be configured to work either in Manual mode or in Automatic mode. In Manual mode, the emission/acquisition of View/Image updates is explicitly controlled by users, whereas in Automatic mode, it is immediate. While the Manual mode supports data version control, the Automatic mode supports real time data distribution.

2.4 Spreadsheet Overlay

The aforementioned primitives allow users to activate the Spreadsheet Space functionalities directly. However, in workgroup collaboration, it often happens that spreadsheet users do not take the initiative to interact with other users and/or to retrieve data from Software Platforms, while they are requested to do so by a third party who coordinates collaboration. The Spreadsheet Space supports such a use case through a new abstraction called Form, and through an entity called Spreadsheet Overlay (SO).
A Form is a formatted cell range that a spreadsheet user is requested to fill out and expose to other users as a View[5]. A SO, visualized in Fig. *2*, is a complete specification of a spreadsheet-based collaboration scheme that a Coordinator deploys in the Spreadsheet Space. It is a directed graph in which the nodes are associated to the SO participants (i.e., user spreadsheets and/or Software Platforms) while the edges are associated to Forms.
The SO lifecycle includes three stages, namely Creation, Deployment and Operation. To create a SO, the Coordinator identifies the participants, prepares the Forms that they are supposed to fill out and exchange, and enters such a description, through a graphical SO Creation Environment, in the form of a graph like the one shown in Fig. *2*. To deploy a SO, the software component in charge of deployment sends an invitation to the prospective participants. Attached to such an invitation, are the Forms that each participant is requested to fill out/receive. Once all participants have confirmed their participation and have configured their spreadsheets to participate, the SO enters the Operation stage, in which synchronization actually takes place. The Coordinator monitors deployment and operation through a software component that logs all the data exchanges and present the SO status in a graphical form.

[5] The distinctive characteristic of Spreadsheet Space Forms is that they must be prepared using spreadsheets, provided to other spreadsheets through the Spreadsheet Space, filled-out using spreadsheets, and submitted to other spreadsheets through the Spreadsheet Space.

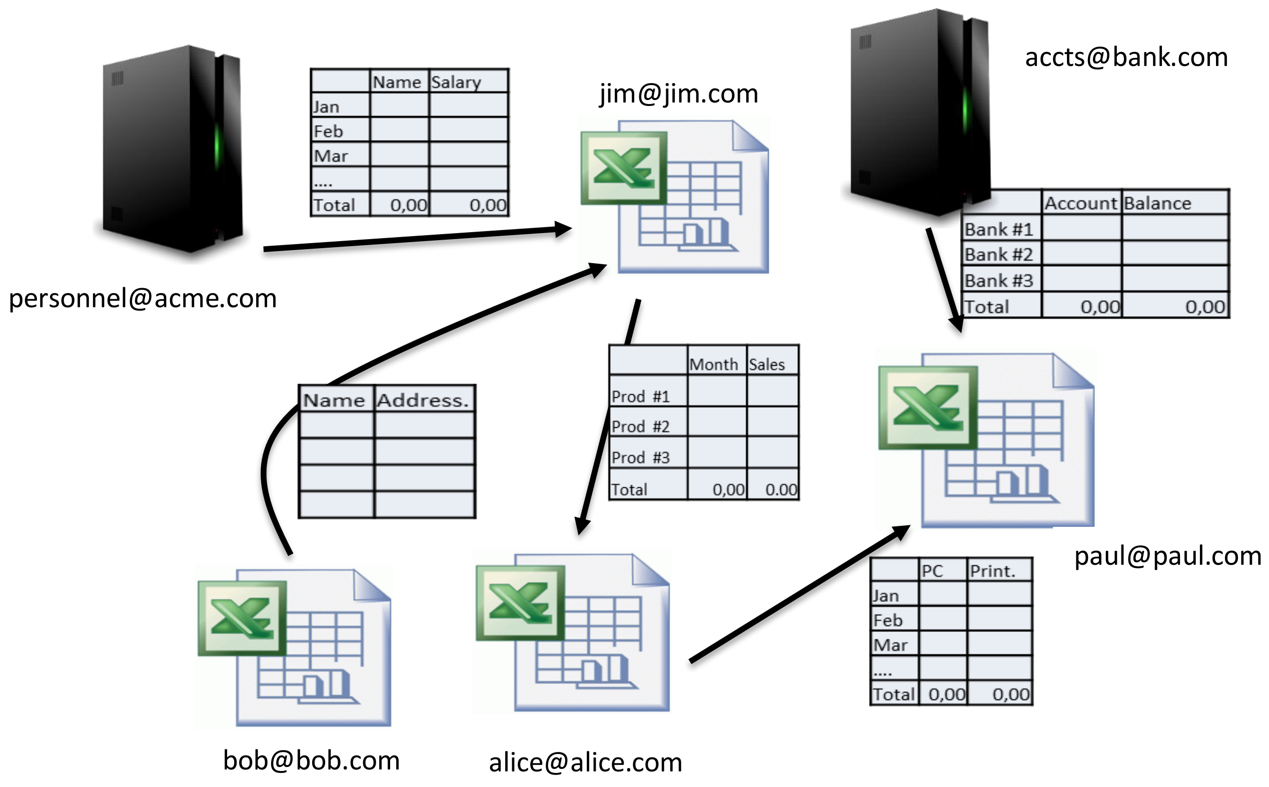


Fig. 2: Spreadsheet Overlay

3 Spreadsheet Space Platform

The Spreadsheet Space relies on a distributed software system. Such a system, shown in Fig. 3a, includes two software components, namely a Server and a Client. While the Server is unique and takes care of persistency and synchronization, the Clients are associated to spreadsheets and Software Platforms to allow them to participate in the Spreadsheet Space.

The Control/Data Plane components in the Clients and in the Server take care of View/Image creation, configuration, removal, and update, while an Event Manager orchestrates operation by processing and issuing the appropriate events (Fig. 3b). All updates are activated by the Clients.

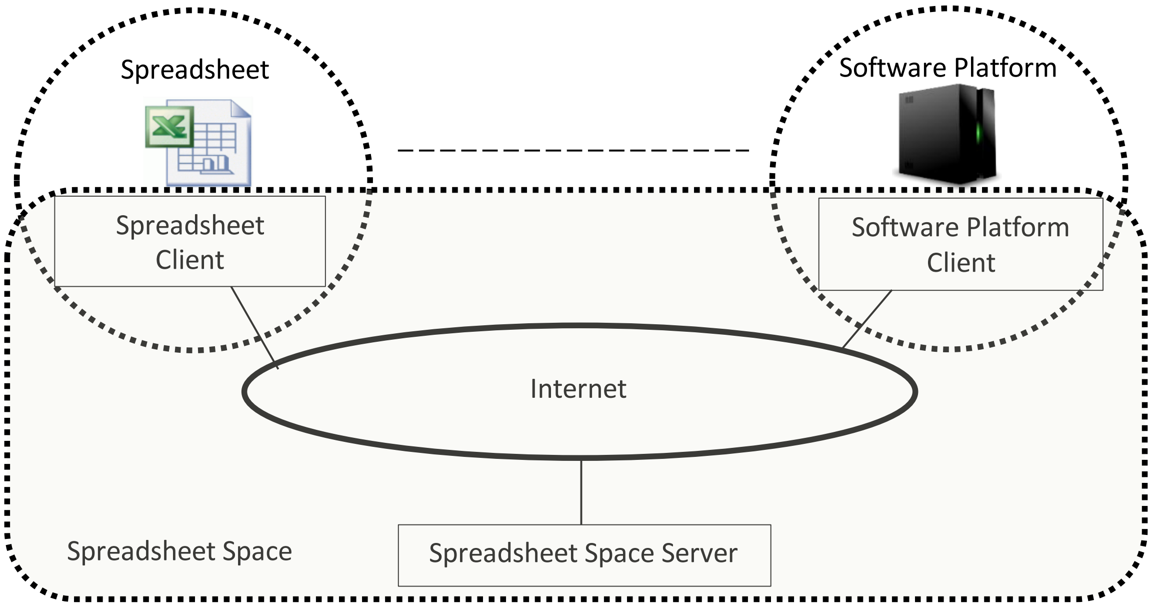
An example of a Control Plane interaction is that in which a source user exposes a View. The source Client

- uploads the View on the View Repository, and
- issues a "New View" event to the View target Clients, through the Event Manager, to invite them to join the View [6].

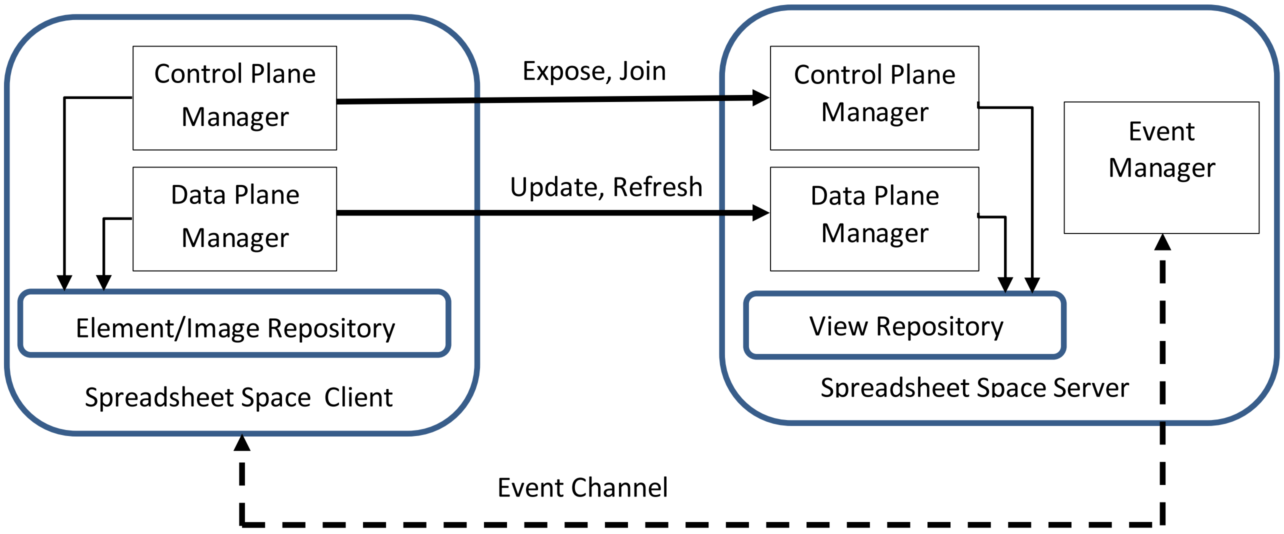
The target Clients

- receive the event notification,
- download the View from the View Repository, and
- issue "View Joined" events to the source Client through the Event Manager.

[6] A special case is that in which a View exposed to the public and not to specific users. In such a case update notifications require explicit user subscription.



a. Architecture



b. Architecture

Fig. 3: The Spreadsheet Space Platform

An example of a Data Plane interaction is that in which a source Client updates an existing View. The source Client

- uploads the update on the View Repository, and
- issues a "View Updated" event to the View target Clients, through the Event Manager, to invite them to refresh their Images.

The target Clients:

- receive the event notification,
- download the update, and
- issue an "Image Updated" event to the source Client, through the Event Manager, to signal update acquisition.

4 Field Experiments

We developed the Spreadsheet Space as a free Internet service (www.spreadsheetspace.net) and performed several experiments. We selected a set of application domains, identified appropriate partner organizations, teamed with partner organization experts to design use cases of interest, and performed the experiments. Table 1 lists the use case considered.

DOMAIN	PARTNER PROFILE	USE CASE
Personal	Stock Exchange	Portfolio Management: Integration of financial information provided by multiple sources (i.e., bank, stock exchange, etc.) in the desktop.
Business	University	Budget Consolidation: Consolidation of research group budgets in global department budget.
Marketing	Enterprise	Product List Management: Synchronization of regional product lists with company central product list.
Public Adm.	Intermodal Hub	Open Data: Publication and processing of freight traffic information.
Accounting	Accountant	Balance Reconciliation: Reconciliation of balances of subsidiary companies belonging to the same holding.
Big Data	Bank	Hadoop-Spreadsheet Integration: Interactive analysis of Hadoop query results.

Table 1 – Examples of Use Cases

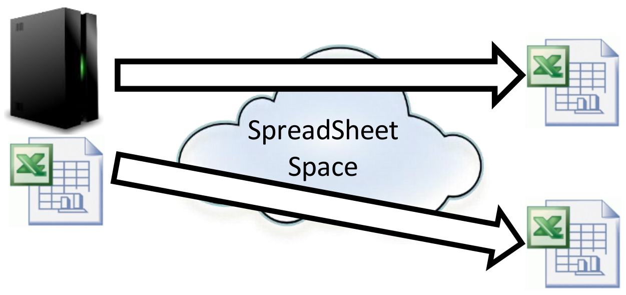
The experiments involved about a hundred Microsoft Excel users and focused on user reaction. The main goal was to detect acceptance problems, hypothesizing that the introduction of a new interaction paradigm might clash with people's emotional and mental attitude [11].

The first feedback from the experiments confirms our hypothesis. The experiments, however, showed that while users are reluctant to accept unstructured spreadsheet interconnection, they are inclined to accept simple and structured interaction patterns that match known user relationships, such as those indicated in Fig. 4. It is worth noticing that although the patterns match known user relationships, they are original, as they refer not to data exchange but to spreadsheet synchronization.

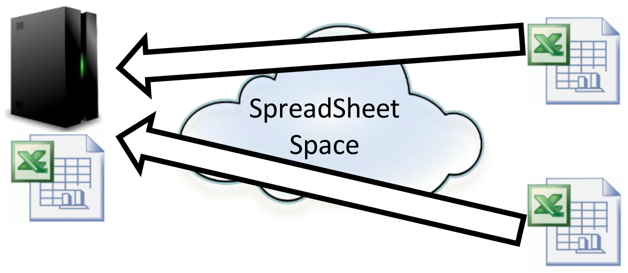
The second feedback is an indication of the relevance of usability, and more specifically of the User Interface and of integration with enterprise services. The User Interface must be simple and friendly and most of all must make users feel that they are permanently in control. Integration with enterprise services extends the user trust in their enterprise systems to the Spreadsheet Space.

The third feedback is an indication of the relevance of security. Although the Spreadsheet Space offers end-to-end encryption, most partner organizations required the deployment of the Spreadsheet Space server in their private Cloud, to enforce data protection, rather than as a service in the public Cloud. In a sense, the feedback is to make the Spreadsheet Space evolve from a single space to a federation of private Spreadsheet Spaces.

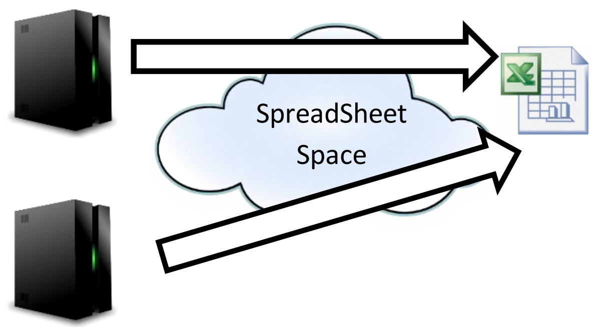
In summary, the experiments led to the identification of the key factors to successful user acceptance, namely the adoption of known interaction patterns, the trust deriving from integration with enterprise services and the sense of protection deriving from deployment in private Cloud.



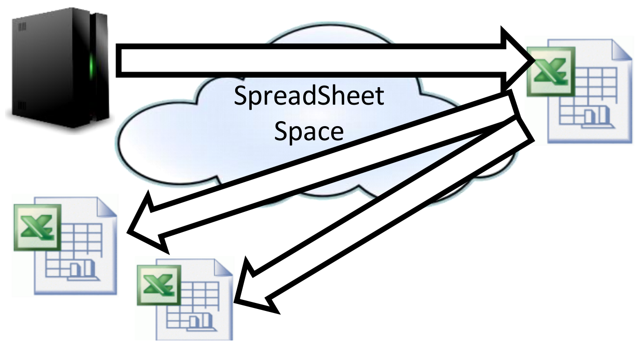
a. *Data Distribution*: A set of spreadsheets maintain synchronization with a data source that exposes data through Views



b. *Data Collection*: A data collector maintains synchronization with a set of spreadsheets that expose data through Views



c. *Enterprise Application Integration*: Spreadsheets synchronize with different Software Platforms and merge data in the desktop.



d. *Data Distribution Chain*: Information propagates from a Software Platform to user spreadsheets through intermediate spreadsheets.

Fig. 4: Regular Interaction Patterns Spreadsheet Space

5 Comparison between the Spreadsheet Space and the Existing Spreadsheet Interconnection Mechanisms

We consider the case of interconnection among spreadsheets and the case of interconnection between spreadsheets and Software Platforms separately.

Interconnection among Spreadsheets

Both Desktop Spreadsheets[7] and Cloud Spreadsheets[8] support interconnection among spreadsheets through cross-spreadsheet references. Desktop Spreadsheets reference external spreadsheets by path-name (e.g., in Microsoft Excel, ='[Spreadsheet_Path-Name]Sheet_Name'!Sheet_Element), while Cloud Spreadsheets do it by URL (e.g., in Google Docs, =importrange("Spreadsheet_URL", "Sheet_Name! Sheet_Element"). A closer look at cross-spreadsheet references shows a set of limitations that significantly restrict their use. The Spreadsheet Space overcomes such limitations, as discussed in the rest of this section using the criteria listed in Table 2.

[7] Microsoft Excel is the de facto standard for Desktop Spreadsheets.

[8] Google Docs and Microsoft Excel Online are the reference systems for Cloud Spreadsheets.

Comparison Criteria:

SCOPE :	Boundaries of cross-spreadsheet references;
PROPAGATION:	Boundaries of cross-spreadsheet data propagation;
AVAILABILITY:	Accessibility of spreadsheets over time;
PRIVACY:	Guarantee that unencrypted data never leave end user systems;
GRANULARITY:	Possibility to grant read access rights to Spreadsheet Elements;
NAMING:	Support of an external namespace.

Comparison

	DESKTOP SPREADSHEETS	CLOUD SPREADSHEETS	SPREADSHEET SPACE
SCOPE	Limited to File System	Unlimited	Unlimited
PROPAGATION	Limited to application instance	Limited to Cloud Provider	Unlimited
AVAILABILITY	Limited to Desktop Connected	Permanent	Permanent
PRIVACY	Full Privacy Supported	Under Cloud provider control	Full Privacy Supported
GRANULARITY	Coarse: Spreadsheet Level	Coarse: Spreadsheet Level	Fine: up to Cell Level
NAMING	No External Namespace Support	No External Namespace Support	External Namespace support

Table 2: Comparison between the Spreadsheet Space and the Native Spreadsheet Interconnection Mechanisms

Scope: While in Desktop Spreadsheets the local path-name nature limits the scope of cross-spreadsheet references to the File System, on the contrary, in Cloud Spreadsheets the global URL nature allows unlimited scope. In the Spreadsheet Space, the nature of View, which is a global reference to a private Spreadsheet Element, allows unlimited scope.

Propagation: We first reiterate that in spreadsheets, data propagation relies on a Publish/Subscribe mechanism through which cells subscribe to other cell updates. As this mechanism is provided by the spreadsheet application, in Desktop Spreadsheets cross-spreadsheet data propagation only works within the same application instance (e.g., Microsoft Excel) while in Cloud Spreadsheets it only works within the Cloud provider perimeter (e.g., Google). In the Spreadsheet Space, cross-spreadsheet data propagation has no limitations, as the Spreadsheet Space platform provides a central Publish/Subscribe mechanism.

Availability: While Desktop Spreadsheets are unavailable when the PCs/Laptops in which they reside are disconnected, Cloud Spreadsheets are permanently available. The Spreadsheet Space provides permanent availability of desktop data by maintaining the Views, implemented as persistent copies of Spreadsheet Elements, in a permanently available server.

Privacy: While Desktop Spreadsheets reside in End User systems, Cloud Spreadsheet reside in the Cloud provider servers in the clear to be processed on the server side. Although Cloud providers comply with strict information protection policies, information delocalization is inherently a threat. The Spreadsheet Space provides full protection by implementing View-Image synchronization over encrypted end-to-end channels. All processing takes place in end systems.

Granularity: Both Desktop Spreadsheets and Cloud Spreadsheets manage read access rights at the spreadsheet level. The Spreadsheet Space manages read access rights at a finer granularity and allows spreadsheets to expose Spreadsheet Elements (i.e., cells, ranges, tables) for external reference while maintaining the rest of the spreadsheet protected.

Naming: Both in Desktop Spreadsheets and in Cloud Spreadsheets, cross-spreadsheet references include either coordinates or names defined in the source spreadsheet, meaning that to configure cross-spreadsheet references, the target users must refer to the source spreadsheet internal structure. In the Spreadsheet Space, on the other hand, View creation includes the generation of an external name, to which the target users refer to configure cross-spreadsheet references.

Interconnection between spreadsheets and Software Platforms

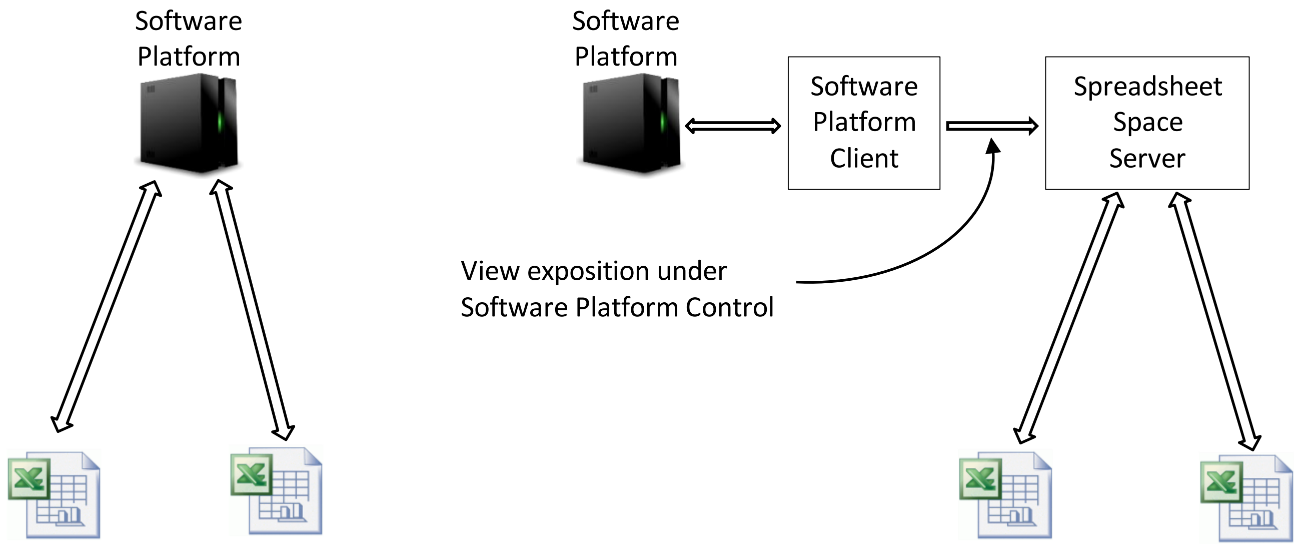
Interconnection between spreadsheets and Software Platforms is currently supported both by native spreadsheet mechanisms (e.g., the Microsoft Excel Connections and the Google Docs Query function) and by a number of spreadsheet extensions based on additional software components (Add-Ins in Microsoft terminology and Add-ons in Google terminology). We focus on native mechanisms, as the aforementioned additional software components do not address spreadsheet interconnection in general while they typically address specific application requirements[9].

The difference between native spreadsheet mechanisms and the Spreadsheet Space is that while native mechanisms implicitly assume that spreadsheets act as clients of Software Platforms (Fig. 5.a), the Spreadsheet Space requires that Software Platforms expose data as Views and that spreadsheets access Software Platform data through such Views (Fig. 5.b). The decoupling between spreadsheets and Software Platforms leads to the following benefits.

Scalability: It improves Scalability by eliminating direct spreadsheet access to Software Platforms. Because in the Spreadsheet Space spreadsheets do not directly access Software Platforms, the load on Software Platforms is not a function of the number of spreadsheet clients; it is a function of the number of exposed Views. The load moves from the Software Platform, where it is critical, to the Spreadsheet Space, where it is not.

Security: It improves Security by decoupling Spreadsheet Space authentication from Software Platform authentication. As spreadsheets never access Software Platforms directly, they are not required to have Software Platform credentials; they only need Spreadsheet Space credentials to authenticate themselves. The improvement is significant because Software Platform access is typically critical and consequently the widespread distribution of Software Platform credentials is insecure.

Control: It improves Control by giving Software Platform operators and Data Scientists the power to configure the Views, to associate them to client spreadsheets, and to monitor and revoke access.



a. Direct Spreadsheet Access to Software Platforms

b. Indirect Spreadsheet Access to Software Platforms through the Spreadsheet Space

Fig. 5: Decoupling between spreadsheets and Software Platforms

[9] Examples of these additional components include Microsoft Power BI and Microsoft Team Foundation Server.

6 Conclusion

We have introduced the Spreadsheet Space, i.e., a virtual space for spreadsheet based collaboration and analytics. The idea behind the Spreadsheet Space is to extend internal spreadsheet references to cross-spreadsheet references over the Internet. Starting from such an idea, the paper introduces Spreadsheet Synchronization, i.e., the possibility to synchronize spreadsheets and Software Platforms over encrypted channels independently form where they are located. The distinctive feature of the Spreadsheet Space is that it provides global reachability and information confidentiality at the same time.

The Spreadsheet Space is nothing but an infrastructure. Its adoption in organizations has implications on data localization, data ownership, data versioning, and auditability, as well as on people interaction in general. The challenge is to develop appropriate models and tools able to accompany the deployment of the Spreadsheet Space in organizations and to take advantage of the opportunities that it offers. If successful, these models and tools will be the real innovation, while the Spreadsheet Space will be just an enabler.

7 Bibliography

[1] Grossman T. 2002. Spreadsheet Engineering: A Research Framework, Proc. European Spreadsheet Risks Interest Group Symposium, Cardiff, Wales, July.

[2] Nardi B. A. 1993. A Small Matter of Programmig, Perspectives on End User Computing, MIT Press.

[3] Ko A.J., Anraham R., Beckwith L., Blackwell A., Burnett N, M., Erwig M., Scaffidi C., Lawrance J., Lieberman H., Myers B., Rosson M. B., Rothermel G., Shaw M., and Wiedenbeck S. 2011. The State of the Art in End User Software Engineering, ACM Computing Surveys, Vol. 43, pp. 21-44.

[4] Erwig S. 2009. Software Engineering for Spreadsheets, IEEE Software, Vol. 26, N. 5, pp. 25-30.

[5] Scaffidi C., Shaw M., and Myers B. 2005. Estimating the number of End Users and End User Programmers, Proc. IEEE Symp. Visual Languages and Human Centric Computing (VLHCC '05), pp. 207-214.

[6] Waqar H., Clear T. 2014. Spreadsheets as Collaborative Technologies in Global Requirements Change Management, Proc. IEEE 9th International Conference on Global Software Engineering (ICGSE '14).

[7] Fisher D., DeLine R., Czerwinski M, Drucker S. 2012. Interactions with Big Data Analytics, ACM Interactions, Vol. XIX (3), May-June, pp. 50-59.

[8] Cunha J., Fernandes J.P., Mendes J, Pereira R. and Saraiva J. 2014. Embedding Model-Driven Spreadsheet Queries in Spreadsheet Systems, Proc. IEEE Symposium on Visual Languages and Uman Centric Computing (VL/HCC), pp. 151-154.

[9] Chang K. S. P. and Myers B. 2015. A Spreadsheet Model for Handling Streaming Data, Proc. 33rd ACM Conference on Human Factors in Computing Systems, pp. 3399-3402, Seoul, Korea.

[10] Obrenovic Z. and Gasevic D. 2008. End-user service computing: Spreadsheets as a service composition tool, IEEE Transactions on Services Computing 1(4), pp. 229–242.

[11] Mangiante S., Maresca M. and Roncarolo, L. 2012. SpreadComp platform: A new paradigm for distributed spreadsheet collaboration and composition, Proc. 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 297 – 305.

[12] Currie G. and Kerrin M. 2004. The Limit of a Technological Fix to Knowledge Management, Epistemological, Political and Cultural Issues in the Case of Intranet Implementation, Management Learning 35(9), pp. 9-29.