A Practical Strategy for ERP to Cloud Integration

Sorin POPA[1, 2]*, Mircea-Florin VAIDA[1]

[1] Technical University of Cluj-Napoca,
26-28 George Baritiu Street, Cluj-Napoca, 400027, Romania,
Mircea.Vaida@com.utcluj.ro

[2] Coresystems AG,
Dorfstrasse 69, Windisch, 5210, Switzerland,
Sorin.Popa@coresystems.ch. (**corresponding author*)

Abstract: The Enterprise Resource Planner is a half-centenary giant that gained popularity in the industry-software market segment throughout the years. Influenced by the latest technology trends, such as mobility and cloud, it keeps software integration as one of its greatest challenges. The present article introduces a cloud-based Field Service Management solution, meant to empower mobility, manage workflows and enhance process automation for ERP enabled companies. Motivated by ERP vendor dependence and a limited adoption rate, the solution's objective is to design and implement optimized cloud generic integration capabilities. Using a commercial, world-wide-distributed environment as a continuous feedback and validation method, the results of our implementations show good application performance and feature offering, but rather low usability metrics and sub-optimal commercial adoption. Based on a post-implementation critical analysis of the existing integration applications and the legacy solution infrastructure, the paper proposes an improved integration strategy, zooming in from business requirements and API constraints, to application architecture and data flows that would overcome technical limitations, reduce prototyping and integration time and decrease maintenance efforts. The implementations and improvement model represent specific practical contributions to a productive, enterprise-oriented solution, proven both successful and highly competitive in respect to its market competitors.

Keywords: ERP, cloud, integration, framework, strategy, connectivity.

1. Introduction

The Enterprise Resource Planner (ERP) is a software system that integrates internal and external management information across an entire organization, automating and facilitating the flow of data between critical back-office functions. Having its name first coined by Gartner in 1990, its evolution witnessed various phases, from the birth development of its ancestors in the 60s and 70s, to its expansion and consolidation stages in the 90s and 2000s [2,8,9,17]. Both redesigns and perspective adjustments were required due to the numerous technical challenges that emerged throughout its lifecycle, but also because of the standardization efforts to align industry processes and business workflows [7,14,18].

Among the numerous trends we have seen consolidating around this colossus, mobility is by far the most important one, because of the empowerment it provides for employees and executives alike, accelerating the exchange of critical information via innovative applications and enabling real-time collaboration between business customers, partners and staff [15].

The cloud is also an important ERP trend, especially today. The industry has shown itself sceptical about adopting this delivery model at first, refusing to place sensitive data outside the company firewall [1]. However, as the advantages of this delivery model became obvious, hesitations have been evaporating either naturally or by migrating through intermediate solutions, as a private cloud deployed inside the company environment [11,15,21].

A large number of integration solutions have been emerging lately for most of enterprise software types [4,10]. Small to big, proprietary to open source, on premise to on demand, these solutions try to integrate aspects of the modern technology, usually being delivered hand in hand with buzzwords like: process automation, machine learning or big data analytics.

Despite the interest shown in this area and the number of solution already productive, there is not yet a concept capable of generically scaling and adapting from one ERP requirements set to the other, shaping once more application integration as one of the most complex enterprise aspects in the distributed cloud world.

2. Coresuite.com

Considering the financial potential of the enterprise segment and evaluating the cloud and mobility trends as appealing and adoptable,

Coresystems AG [6] seized the opportunity to introduce to the ERP integration market a cloud based FSM (Field Service Management) Solution, Coresuite.com, offering integrated tools and workflows meant so serve the field workforce, the service centre and management alike in field service oriented companies.

Coresuite started a little over half a decade ago, as a pure mobility solution, meant to provide fast access to critical data for companies working with SAP B1 (SAP Business One) in their back office. Commercial opportunity arose from the licensing model, but also from fortunately random design decisions like *offline capabilities* for the mobile apps, useful to service technicians working in screened environments.

It continued to grow by adding process automation, workflow management and self-servicing client applications to its portfolio, but also by providing new ERP integration options, like the "Microsoft CRM Connector" or "Excel Importer", visible in Figure 1 [13].

3. Motivation and Objective

The solution started offering mobility features against SAP B1 and was thus limited to only a slice of the entire ERP market segment. In addition, SAP was trying already to provide built-in mobile application for B1.

Motivated by ERP vendor independence and envisioning an accelerated commercial adoption by including partners and customers themselves in developing new integration projects, the solution's objective was to design and implement optimized cloud generic integration capabilities for the cloud solution.

4. Existing Solutions and Strategy

There are already wide ranges of integration solutions available, on premise or cloud based, proprietary or open source, targeting large or small enterprises. Table 1 puts them side by side against some of the most important solution KPIs (Key Performance Indicators).

Table 1. Comparison of existing solution types

KPI	Integration Solution Types			
	On Premise ESB	**Cloud Based ESB**	**Cloud Based ERP**	**Built-in ERP Mobility**
ERP Vendor Independence	✔	✔	✖	✖
Integration Know-How Independence	✖	✖	✔	✔
Additional Infrastructure Free	✖	?	?	✔
Fast and Simple Prototyping	✖	✖	?	✔
Meta-Service Support	✔	✔	✖	✖

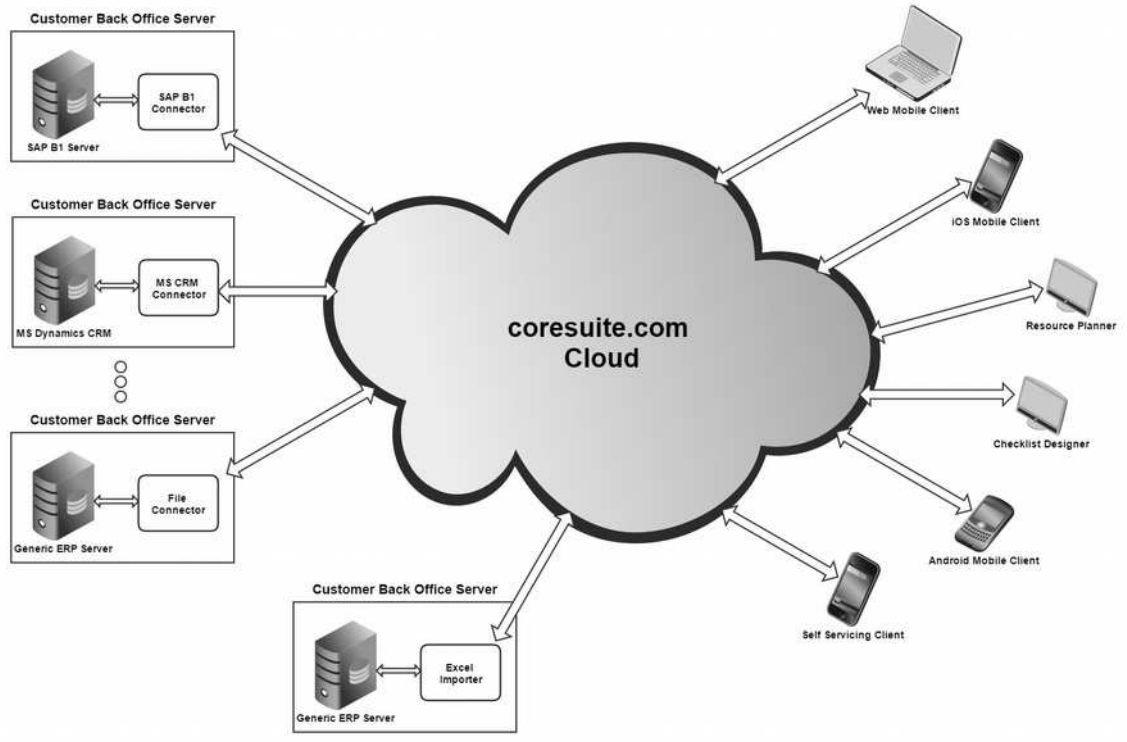

Figure 1. Coresuite Solution Overview

The first and most important is *ERP vendor independence*. Coresuite aims for global commercial adoption and thus it needs to easily scale across customer environments, regardless of their backend ERP type and technology. In respect to this, cloud-based ERPs and built-in mobility can already be excluded from the equation, as they relate to a specific ERP technology.

Integration know-how independence describes technological and business knowledge (not) required by internal or third party developers that integrate against the solution. This is usually a significant drawback for ESBs, as their overall complexity typically exceeds the required functionality of a single integration scenario by orders of magnitude.

Traditionally, on premise solutions require *additional infrastructure* and costs compared to cloud based offerings, from IT employees and specific technical skill, to deployment and maintenance overhead. Nevertheless, usually caused by privacy concerns towards having their data system outside the company firewall [1], a lot of modern organizations choose the private cloud as a stepping stone, compromising responsibility externalization in favour of increased control and security metrics. Cloud-based integration options do not always imply not needing additional infrastructure, but rather provide the option.

Adopting an integration solution for mobility or added functionality, at both the business and technical level implicitly, often boils down to proof-of-concept and *prototyping time and complexity*. While this is provided implicitly in the built-in ERP mobility case, it can or cannot be well supported in an ERP, but it is definitely not a walk in the park at the ESB level.

Finally, transporting data from one place to another is the core of an integration project, but only a fraction of the overall functionality. This typically implies *meta-services* managing aspects like: configuration, messaging, security, commands or logging. While ESBs most often provide support for these requirements, ERPs do not usually put out more than data synchronization plugin endpoints.

Since none of the existing solution types are able to accomplish an integration framework that is ERP vendor independent, requires no or little additional know-how and infrastructure overhead, provides fast and simple prototyping and supports meta-services, the solution was required to design its own strategy.

In parallel with developing in-house ERP-specialized connectors, for: Microsoft CRM (Customer Relationship Management) and Navision, SAP B1 (Business One) and ECC, (Enterprise Central Component), salesfore.com, QuickBooks, etc., Coresystems designed and implemented two generic integration tools: the "Transporter" library and the "File Connector", to support customer and partner self-integration efforts, detailed as follows.

5. The Transporter Library

The "Transporter" is a .NET library that simplifies the communication protocol with the Cloud, abstracting a REST/HTTP based cloud API into communication object data queues. It connects to the implementing application using a couple of interfaces that allow bi-directional control and provides a handful of micro services enabling synchronization and meta-capabilities (configuration, security, commands and logging).

5.1 Download service

An example of Transporter service can be seen in the download flow diagram in Figure 2. In order to optimize performance, the handshake requires a *confirmation* step, as transferring and processing data happens asynchronously.

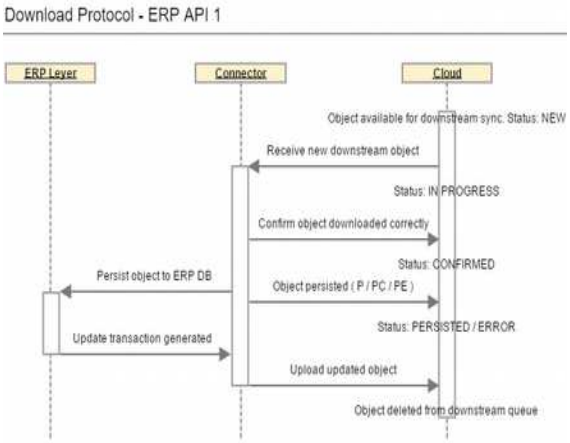

Figure 2. Transporter Download Protocol

Due to the master-to-master replication pattern, a *persist* step is also required to associate the ERP id to cloud id. Finally, since the business rules remain on the ERP side, an *upload* step is required to close the loop, ensuring the cloud possesses the latest version of data.

5.2 Upload and failed imports service

Another example, this time at the component level, is visible in Figure 3, showing the Upload and associated Failed Imports workflows. Transporter queue processing time is non-deterministic, so "ERP Connector" will have to continuously poll for data processing results. A *transaction identifier* will be used throughout the process to uniquely identify any transaction.

Based on an ERP notice (1), the Connector will place the data in a queue (2), it will be transferred to the cloud (3) and confirmed (4). The connector will read the confirmation (4) and delete the original transaction.

Nevertheless, the confirmation only means a successful upload. In case the data processing fails (a), the data object will be queue (b) and downloaded by the Transporter (c), read (d) and persisted (e) by the Connector for human analysis and confirmed back to the Transporter (f) and the cloud (g), completing the cycle.

6. The File Connector

The "File Connector" (FC) is a full connector implementation, developed on top of the Transporter library, with an open-ended XML file interface. It implies the existence of an "XML Generator" (XG) on the ERP side, which can be implemented at any level of complexity, from an Extract Transform Load (ETL) tool to an Object Relational Mapper (ORM) system[12].

The application provides complex services, from cloud database management, security, metadata and remote logging to full duplex, priority-managed synchronization, some of which will be presented further in this article. The protocol abstraction is therefore moved from code-level queues to Operating System (OS) level folders, to/from where the xml files will be produced/consumed.

The application also provides configuration and monitoring capabilities, both at the UI (User Interface) level, visible in Figure 4, and the file (advanced user) level.

Figure 4. File Connector User Interface

6.1 Download service

A representative example of service for the FC is the *download service* as well, being

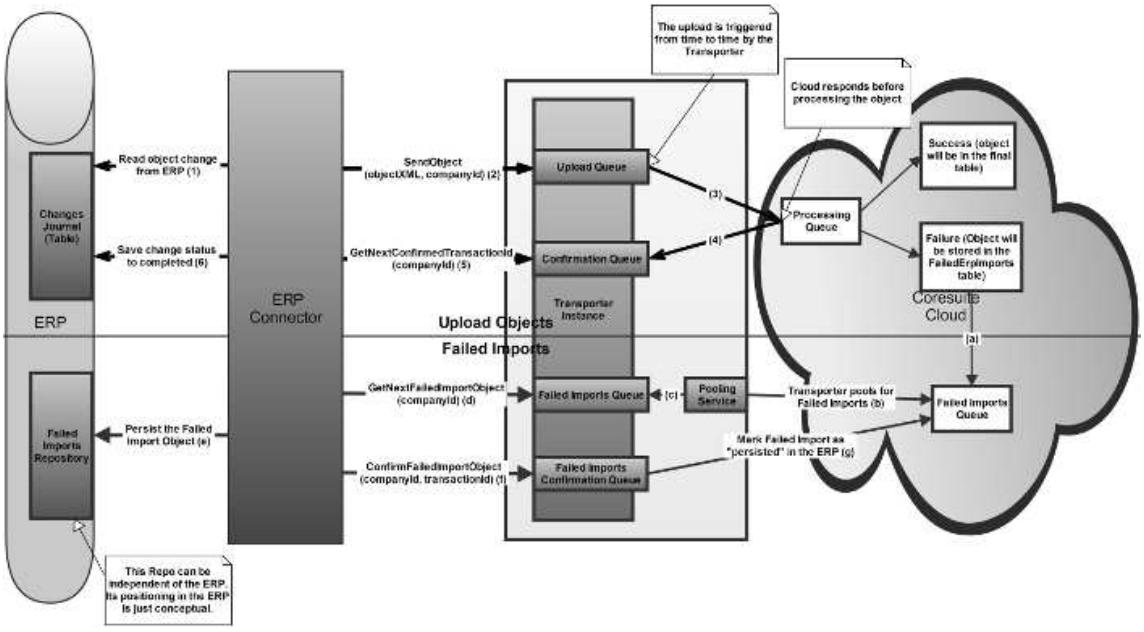

Figure 3. Transporter Data Download and Failed Imports Service Workflows

responsible for generating the *persist* (ERP id to cloud id mapping) mechanism. Its diagram can be analysed in Figure 5.

When a new object is produced by a client application, it gets downloaded by the FC (0) and stored into the *incoming folder* (1), from where the XG will parse it (2) and persist it into its ERP Table (3), saving the ERP identity into a temporary "Identifier Table" location (4).

Based on an ERP notification (5), the XG will assemble the data (6), leveraging the identity information it saved in the "Identifier Table" (7) and generate an xml file into the outgoing directory (8, 8b), that will be read by the FC (9) and uploaded to the cloud (10).

6.2 Architecture

In tone with the API and Transporter, the FC provides a "compact database" component (Figure 6), that enables persisting intermediate states of all processed transactions, allowing it to recover from complex failure scenarios (application crash, network fault, etc.).

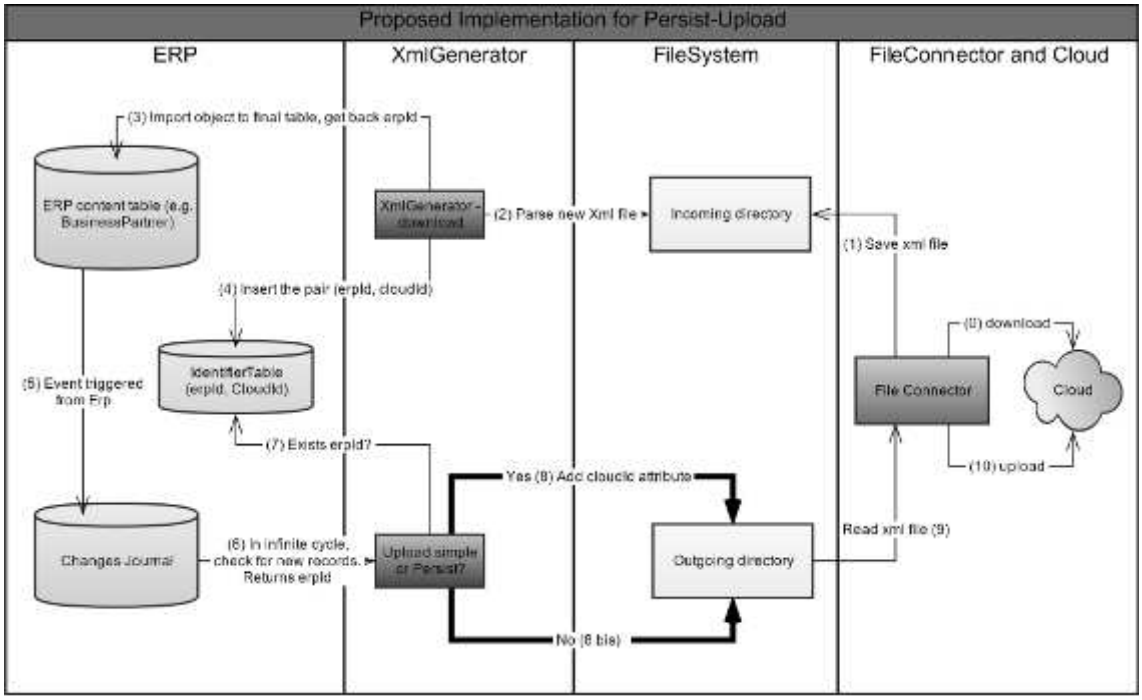

Figure 5. File Connector Download Protocol

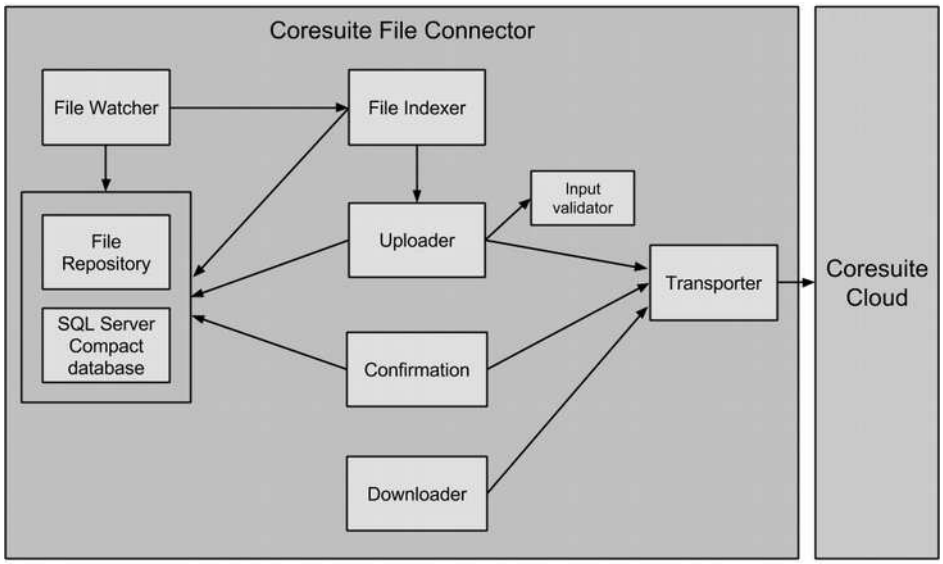

Figure 6. FC Component Architecture

7. Implementation Results

In order to provide a relevant quantification of the implemented options, we benchmarked them against some of the other in-house connector implementations to outline both the technical and business impacts.

7.1 Performance

Based on an optimistic approach, the FC brings improvements at the Cloud protocol level, by processing data asynchronous from the upload step. Choosing a random, constant-size business object (BO) and a random object batch number to synchronize, the FC outruns the SAP B1 Connector by a factor of 9, as visible in Table 2.

Table 2. Upload performance benchmarking

Integration Application	SAP B1 Connector	File Connector
Object Count	1580	1580
Sync Time [sec]	94.74	10.53
Throughput [sec]	16.68	150.05
Throughput [min]	1000.63	9002.85

In terms of *application load time*, the total time is composed of the initialization and start of the Transporter library, connector business layer and ERP interface. Since the File Connector does not implement an ERP data interface and it is based on an improved Transporter version – but even if we exclude the Transporter library differences, its load time is significantly smaller than that of the SAP B1 Connector, as shown in Table 3.

Table 3. Load time comparison

App.	SAP B1 Connector			File Connector	
Comp.	Transp. v1	ERP Interface	Business Layer	Transp. v2	Business Layer
Load Time [sec]	3.688	21.509	12.087	0.655	5.495
	37.284			6.15	

7.2 Code metrics

Regarding *cyclomatic complexity*, both FC and Transporter scored similarly with the other applications in terms of "per-method average", with a value of 5.76, as visible in Table 4.

Being below the literature threshold of 10 [19], this appears as a good indicator for this metric. The "per-project" value, in return, is more relevant in respect to the project size.

Table 4. Cyclomatic complexity comparison

Application	Cyclomatic Complexity			Project Avg. Lines of Code
	Per-Method Average	Per-Method Max	Per-Project	
Transporter v1	6.5	15	1069	2175
Transporter v2	5.65	34	2040	4201
SAP B1 Connector	5.39	90	10808	1864.18
File Connector	5.76	34	4140	1777.60
MS CRM Connector	5.62	49	6808	730.68
Excel Importer	4.13	15	754	661.50

Depth of inheritance, as a metric, conflicts with itself as a higher number infers behaviour unpredictability and greater design complexity, but at the same time a higher potential of code reuse. [16]. Although there is no established metric in the literature, we see the FC at the top of the chart in Figure 7.

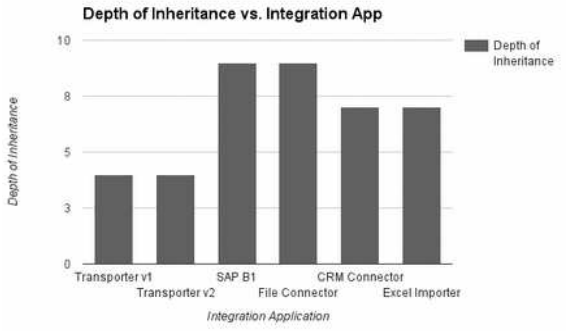

Figure 7. Depth of Inheritance Comparison

Good software design dictates that types and methods should have high cohesion and low coupling. *Class coupling*, proposes a value of 9 for a single member as a good reference [3], but the aggregated per-project value, with respect to the average project size, is again a more pertinent metric.

Table 5. Class coupling comparison

Application	Per-Project Class Coupling		Average Lines of Code Per-Project
	Average	Max	
Transporter v1	182	182	2175
Transporter v2	286	286	4201
SAP B1 Connector	201.73	834	1864.18
File Connector	201.2	292	1777.6
MS CRM Connector	107.14	512	730.68
Excel Importer	130	180	661.5

Finally, computing the *Maintainability Index* of the implemented applications, reveals again a striking similarity not only between the FC and the SAP B1 Connector, but all the other applications as well, as visible in Table 6.

Table 6. Maintainability index comparison

Application	Maintainability Index	# of Projects Per Solution	Lines of Code
Transporter v1	83	1	2175
Transporter v2	84	1	4201
SAP B1 Connector	83.94	11	20506
File Connector	82.95	5	8888
MS CRM Connector	82.36	22	16075
Excel Importer	84.92	2	1323

7.3 Features and functionality

The features that discriminate between the integration options belong to the enhancements done in the second version of the Cloud ERP API, and propagate to the Transporter libraries and implementing connectors. However, the critical functionality of synchronization is homogeneously provided for all of them, in a manner more or less user friendly, specific to every application. A functionality offering comparison is visible in Table 7.

Table 7. Feature comparison of existing solution types

Functionality	Tran. v1	SAP B1 Conn.	Tran. v2	File Conn.	MS CRM Conn
Company Management	✔	✔	✔	✔	✔
Security Service	✔	✔	✔	✔	✔
Metadata Service	✔	✔	✔	✔	✔
Log Service	✔	✔	✔	✔	✔
Download Service	✔	✔	✔	✔	✔
Upload Service	✔	✔	✔	✔	✔
Backend Request Service	✖	✖	✔	✔	✔
Notification Service	✖	✖	✔	✔	✔
Database Profile Service	✖	✖	✔	✔	✔
UI Configuration	✖	✔	✖	✔	✔

7.4 Integration usability

Although not always a quantifiable metric, the FC provides an improvement in terms of prototyping and developing speed compared to the existing options (Table 8).

Table 8. Table styles

Integration Option	Cloud ERP API	Transporter Library	File Connector
Implementation Time	Months (Assumed)	Weeks-Months	Days-Weeks

Between 2012 and 2016 there were a total of ten ERP integration attempts by solution partners, more than Transporter based attempts in the same period, demonstrated by Table 9.

Table 9. Table styles

Integration Option	Cloud ERP API	Transporter Library	File Connector
Integration Attempts	0	7	10

The FC does not collect data on the ERP type, thus tracing a running connector back to an ERP type is rather inaccurate. However, at the time of writing this paper there were eighteen accounts using the FC, which is actually less than against the Transporter library (Table 10).

Table 10. Table styles

Integration Option	Target ERP	Active Accounts
Transporter v1	SAP ECC	25
Transporter v1	MS Dynamics Navision	10
File Connector	N/A	18

7.5 Business adoption

If we perform an ERP connector type count, aggregated across Coresuite's productive cloud environment, the distribution overwhelmingly inclines towards the SAP B1 Connector, making the FC account for only 1.5% of the active accounts, as presented in Figure 8.

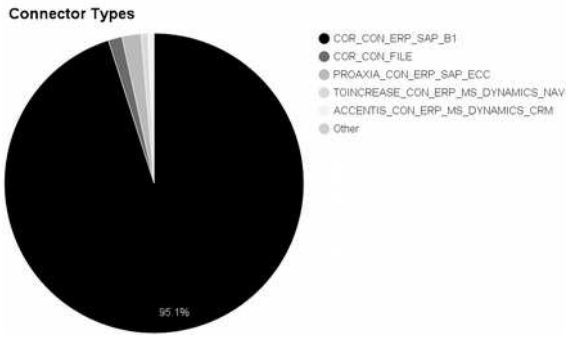

Figure 8. Connector Type Distribution

Nevertheless, between 2009 and 2016 the number of accounts (and cloud users) has been continuously growing, at an almost exponential rate, as visible in Figure 9.

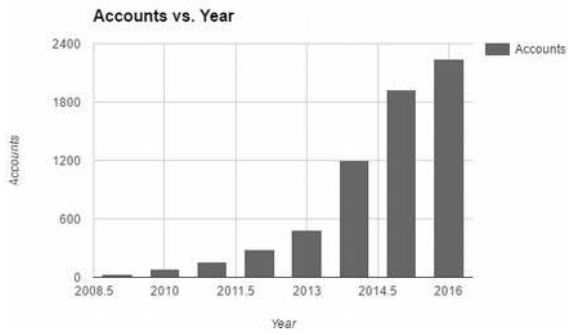

Figure 9. Coresuite commercial adoption

8. Retro-Analysis

Based on the usability and adoption of the externally exposed integration options, but also on the existing applications and infrastructure, we were able to outline a few of our solution's biggest, integration related limitations:

- The protocol *lacks atomicity and isolation in favour of consistency and durability*;
- Integration *does not easily scale to new ERPs*, as the infrastructure was designed with a lot of baked-in ERP specifics;
- The applications *lack flexibility*, contrary to the ERP world that is built on customization;
- We lack application control, internally, as most connectors are deployed behind the customer firewall.

8.1 Protocol

Zooming in on the infrastructure, the protocol provides the following *positive* aspects:

- *Performant batch upload* capabilities, very useful because of the ERP's typically large objects counts;
- *Object dependency support*, implemented via a weak referencing system;
- *Object data versioning*, that enables clean encapsulation of data structure and associated business logic.

From a *constructive* perspective, however, the protocol contains the following drawbacks:

- *ERP specificity*, making it hard to scale to different backend technologies;
- *Poor authentication design*;
- Other *low-level design issues*, like mixing of concerns, god services, communication data duplication or handshake complexity.

8.2 ERP interface

The ERP communication's strongest points are:

- *Database improved reading speed*;
- Wrapping the interface to *overcome ERP limitations* (i.e. mutable primary keys, etc.);
- *Auto-generated read queries*, as a trade-off between an ORM and statically typed queries.

8.3 Application

Application wise, the most relevant aspects are:

- *External, centralize configuration*;
- Application *(app domain) level isolation* between synchronizing companies or between the connector application and the ERP interface;
- As a drawback, a decent amount of *over-engineering*, as previously seen in Figure 6.

8.4 Data Management

Data handling, nevertheless, brings out the following constructive aspects:

- *String programming*, a common anti-pattern found systematically in most projects;
- *Isolated data mapping*, a requirement that enables customizability and extensibility;

9. Proposed Improvement Model

Considering the application and business level metrics in chapter and the analysis in chapter, we propose an improvement model that would overcome the present limitations, with the following characteristics:

9.1 API requirements

Regarding the *API implementation*, the top two most important requirements are:

- *Synchronous operations*, removing the need for queues and simplifying handshake at the (manageable) cost of a lower upload performance;

– *Atomicity and Consistency*, removing the "Identifier Map" complexity overhead;

9.2 Connector architecture

Figure 10 introduces and overview of the proposed application architecture, promoting a plug-in-able design with the following top advantages:

– *Modularity*, keeping the ERP specifics (i.e. "ERP Layer", "ERP Configuration UI") as "thin" as possible and easily replaceable for scaling to new ERP backend technology
– High *(process level) isolation*, between the application and the ERP layer and between the synchronizing companies themselves.
– *Remote Controlling*, through the external "Connector Controller", enabling performant monitoring and maintenance.

9.3 Synchronization process

Finally, the data flows and the associated infrastructure should be significantly simplified. As opposed to the complexity previously visible in Figure 3, the upload process – as a relevant example – would look like in Figure 11, completing the cycle with only one call to the cloud.

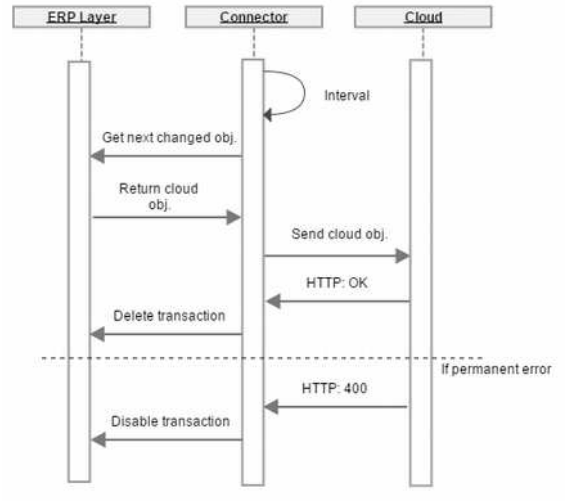

Figure 11. Improved Download Process

10. Conclusions

First, the implemented external integration options show good performance and technical maintainability metrics and proved themselves successful across connectivity projects. Their limited adoption rates, however, can be a matter of usability, but can also be related to the Coresystems' partner model attractiveness.

Second, although the existing infrastructure has some known limitations, its usability is unquestionable, proved by the exponential account and user growth throughout the last half of a decade.

Third, we do not believe in a "one size fits all" solution, thus the proposed improvement model will not be a final consumer application serving any ERP integration requirement, but rather a flexible framework to support scalability and customizability.

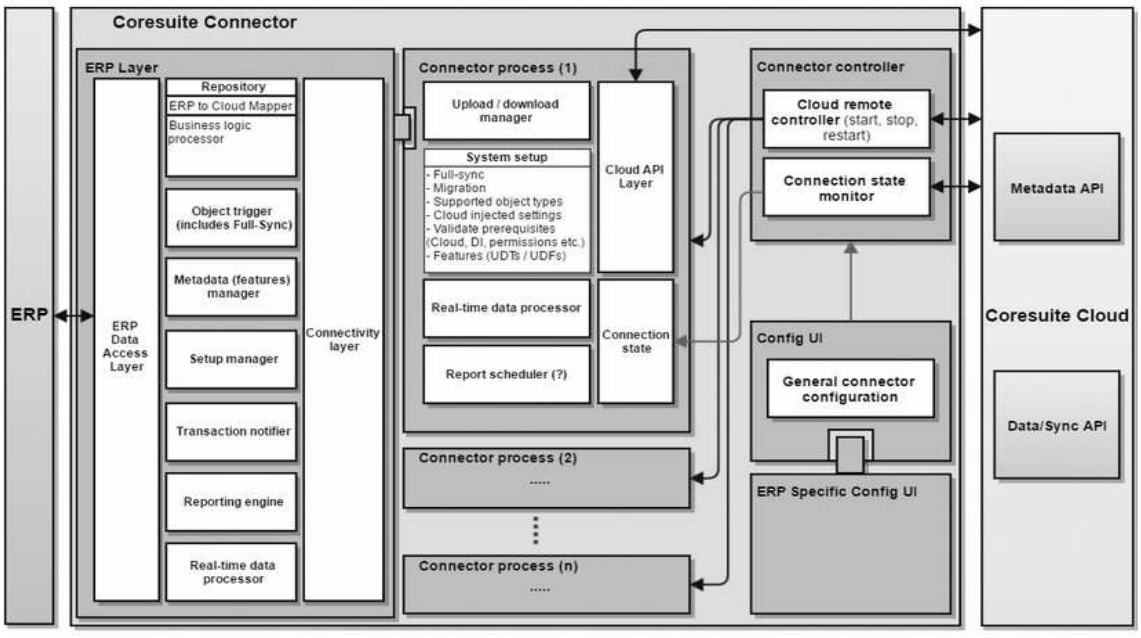

Figure 10. Proposed Integration Model Architecture

Acknowledgements

Coresystems AG [6], a Swiss company offering cloud-based field service management products, supports the work of this paper.

REFERENCES

1. ALBOAIE, L., M.-F. VAIDA, **Trust and Reputation Model for Various Online Communities**, Studies in Informatics and Control, vol. 20(2), 2011, pp. 143-156.
2. BARAY, S., S. HAMEED, A. BADII, **Analysing the Effectiveness of Implementing Enterprise Resource Planning in the Printing Industry**, Euro. and Medit. Conf. on Inf. Syst. (EMCIS), July 2006, Costa Blanca, Alicante, Spain.
3. CHIDAMBER, S. R., KEMERER, C. F., **A Metrics Suite for Object Oriented Design**, IEEE Trans. on Software Engineering, vol. 20(6), 1994, pp. 476-493.
4. COLE, A., **Venturing onto the Private Cloud**, retrieved June 4th, 2010, http://www.itbusinessedge.com/
5. COLTMAN, T. R., T. M. DEVINNEY, D. F. MIDGLEY, **Customer Relationship Management and Firm Performance**, J. Inf. Tech. vol. 26(3), 2011, pp. 205-219.
6. **Coresystems AG Official Website**, http://www.coresystems.ch.
7. GANESHAN, R. E. J., M. J. MAGAZINE, P. STEPHENS, **A Taxonomic Review of Supply Chain Management Research**, Quantitative Models for Supply Chain Management, 1999.
8. HERALD, H., **Extended ERP Reborn in B-to-B**, InfoWorld, Aug. 27 – Sept. 3 2001, vol. 23(35/36), p. 21, Trade Publication.
9. JACOBS, F. R., F. C. 'Ted' WESTON Jr., **Enterprise Resource Planning (ERP)—A Brief History**, J. Operations Management, vol. 25(2), March 2007, pp. 357-363.
10. KIF, M., **Microsoft's Enterprise Service Bus (ESB) Strategy**, http://blogs.msdn.com/, retrieved January 23rd, 2007.
11. MCKENDRICK J., **Enterprise Service Busted,** retrieved July 22nd, 2008, http://www.zdnet.com/article/enterprise-service-busted/.
12. POPA, S., M.-F. VAIDA, **A Practical Abstraction of ERP to Cloud Integration Complexity: The Easy Way**, 15th RoEduNet International Conference – Networking in Education and Research (RoEduNet NER), Bucharest, September 2016 – proceedings.
13. POPA, S., M.-F. VAIDA, **Outspreading Enterprise Capabilities into the Cloud – A Commercial Case Study**, ACTA Technica Napocensis, vol. 57(3), 2016 procs.
14. SACHAN, A., S. DATTA, **Review of Supply Chain Management and Logistics Research**, Intl. J. of Physical Distribution & Logistics Management, vol. 35(9), 2005, pp.664-705.
15. SEROTER, R., **Patterns of Cloud Integration**, http://www.pluralsight.com, retrieved September 4th, 2013.
16. SHATNAWI, R., **A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems**, IEEE Transactions on Software Engineering, vol. 36(2), 2010, pp: 216-225.
17. SOJA, P., **Success Factors in ERP Systems Implementations: Lessons from Practice**, J. Ent. Information Management, vol. 19(6), 2006, pp. 646-661.
18. SUBRAHMANIAN, E., S. RACHURI, S. FOUFOU, R. D. SRIRAM, **Product Lifecycle Management Support: A Challenge in Supporting Product Design and Manufacturing in a Networked Economy**, Int. J. Product Lifecycle Management, vol. 1(1), 2005.
19. WATSON, A. H., T. J. McCABE, T. J., **Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric**, 1996, NIST Special Publication 500-235.
20. WINER, R. S., **Customer Relationship Management: A Framework, Research Directions, and the Future**, University of California, Berkeley, April 2011.
21. ZHANG, Q., L. CHENG, R. BOUTABA, **Cloud Computing: State-of-the-Art and Research Challenges**, 20.04.2010, The Brazilian Computer Society, Internet Serv Appl 1: 7.