OUTSPREADING ENTERPISE CAPABILITIES INTO THE CLOUD – A COMMERCIAL CASE STUDY

Sorin POPA[1,2] Mircea-Florin VAIDA[1]
[1]*Communications Department, Technical University of Cluj-Napoca, 26-28 George Baritiu Street, Romania*
[2]*Coresystems AG, Dorfstrasse 69, Windisch, Switzerland*
Sorin.Popa@coresystems.ch Mircea.Vailda@com.utcluj.ro

Abstract: Gravitating around the ERP integration concept, the present paper aims to introduce a case study of a world-wide distributed commercial solution, expanding into the cloud to provide ERP empower enterprises with mobility, process automations or workflow management. Motivated by ERP vendor independence and global commercial adoption, its objective is to provide a generic integration strategy, easily scaling between back office environments. Expanding on history and evolution, the paper presents some of the most important integration options, concluding on the adoption results and the strategy perspectives.

Keywords: *ERP, cloud, integration, process automation, connectivity.*

I. INTRODUCTION

During the last half of a century, we have seen the Enterprise Resource Planning (ERP) concept emerge, shape, branch into various forms and then consolidate, but the last couple of years have fundamentally shifted the entire area [1-5]. Various technical challenges arose from the evolution of its concepts and trends, requiring both redesigns and mentality shifts, but kept a lot of people busy with trying to find a "one size fits all" solution [6-16].

Mobility is one of the greatest technology assets of the current century. Like previously shown, with the "smart-devices retail explosion" it has transformed people's lives, but ultimately it has transformed the way businesses engage with their customers, partners and staff, using innovative applications that enhance and accelerate the exchange of critical information.

With mobile websites systematically losing ground to popular operating systems' native apps (iOS, Android), most of the software vendors already understood the importance of this compelling concept and started offering it in a way or another. Despite its unquestionable value, mobility is typically a valid but not sufficient reason for purchasing a cloud solution on top of your ERP. More often than not, we find companies buying in to such software solutions because of the additional business capabilities and workflows.

On the business side however, it comes hand in hand with process automation, another staggering actor of the light-speed century which implies: cost reduction, streamline working and increased productivity, progress tracking and transparency and why not, standards enforcement and better high volume demand management [17].

On top of this, modern world's buzzwords come to the rescue: "Machine Learning" combined with process automation turns out to be a quite appealing mix. Big Data is again a word full of potential. A long with managing and storing a large volume of data, large Cloud systems now leverage historical information about the data evolutionary processes correlated into better judgements, recommendations, support or decisions [18].

Last, but definitely not least, customization is an ever-present requirement in the ERP world, especially bound to Cloud environment challenges [19, 20]. Concepts like User Defined Fields (UDFs), Tables (UDTs) or Objects (UDOs) are quite common functionalities as well as triggers, data constraints, database relations, etc.

II. OBJECTVE & EXISTING SOLUTION COMPARISON

Motivated by commercial global adoption and determined to achieve ERP vendor independence, the solution's main objective is the to design an integration strategy that empowers internal and external development alike to quickly integrate one ERP after the other.

The integration topic is a quite debated one both in the research and the development worlds and there are already a wide range of integration solutions available, on premise or cloud based, proprietary or open source, targeting large or small enterprises.

Putting side by side the most common integration options pertinent to this use-case against some of the most important solution Key Performance Indicators (KPIs), Table I. outlines a high level comparison.

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.

Manuscript received August 17, 2016; revised September 1, 2016

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 & Simple Prototyping	✖	✖	?	✔
Meta-Service Support	✔	✔	✖	✖

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 significand drawback for ESBs, as their overall complexity typically exceeds significantly the required functionality of a single integration scenario.

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 data system outside the company firewall, a lot of modern organizations choose the private cloud as a stepping stone, compromising responsibility externalization in favor 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 comes for free in the built-in ERP mobility case, this can or cannot be well supporter in an ERP, even cloud-based, but it is definitely not a walk in the park at the ESB level, at stated before.

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.

None of the above solutions types are able accomplish an integration framework that is ERP vendor independent, requires no or little external know-how and additional infrastructure overhead, provides fast and simple prototyping and supports meta-services.

Therefore, as furtherly shown in this article, the solution was required to designing its own strategy, validated throughout the development in a world-wide productive environment.

III. SOLUTION DEVELOPMENT

In order to exemplify and demonstrate these previously mentions concepts, we would like to introduce a software solution we have been actively developing and maintaining for the last four years, along with my colleagues at Coresystems [21], a young and ambitious international company.

Coresute.com is a cloud based FSM (Field Service Management) solution designed to offer a variety of integrated tools and workflows meant so serve the field workforce, the service center and management alike in field service oriented companies.

A. Mobility

Coresuite started little over half a decade ago, as a pure mobility solution, meant to provide fast access to critical data for companies working with SAP Business One (SAP B1) in their back office – the ERP software solution offering from SAP for small and midsize businesses. As a young startup concept, its functionality was fairly limited, offering capabilities for just a couple of business objects and of course, with read-only abilities.

The opportunity arose first and foremost from the licensing side. SAP B1, like all the other ERPs out there, are indispensable to large enterprises and their license cost reflects just that. They prefer to use "channel sales" or reseller partners, so the price might vary depending on volume, country of purchase, commercial level of partner (silver, gold, etc.) and more, but on average one named-user license costs more than $1000 [22], while the mobile license for coresuite.com used to cost around $30 and today – in an improved license structure – the price is still comparable.

In addition, on top of mobility, coresuite.com was offering a great feature from the very start: offline synchronization. To use the SAP B1 client for an automated interaction with the back office, you would need to provide your employee – on top of a portable computer – with internet connection and probably Virtual Private Network (VPN) access, so a good number of companies were still using pen and paper.

Luck also plays its part in business. Although this was not a use case in the initial design phase, some of the customers were servicing equipment in screened or non-wireless-friendly environments (like specialized industrial areas), thus offline sync fitted them like a glove.

After having the mobility pillar in place, the team continued to extend its functionality adding further business objects and capabilities for each of them, but they didn't stop there, they kept looking for ways to enhance the enterprise experience beyond mobility.

B. Process Automation

The following two additions to the coresuite.com suite were the "Resource Planner" and the "Checklist Designer". The first of the two is a tool for scheduling and dispatching servicing activities, which at the core has a timeline view that allows drag and drop of "service calls" onto "technician" lanes. Of course, today's version of product has much evolved, offering full linked access to all related business objects, a wide flexibility via configuration, but also neat features like automatically reserving spare parts in the associated warehouse when assigning a service call to a technician or version management for all changes made to the business object.

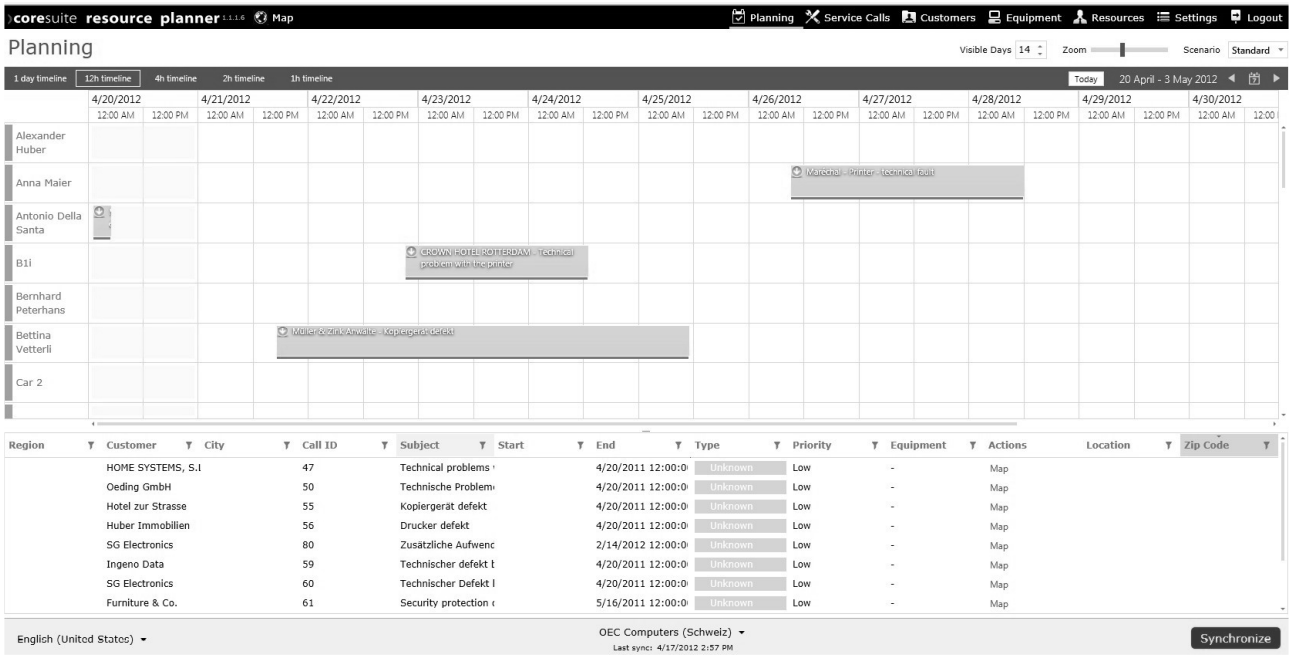

Figure 1: Coresuite's Resource Planner [23]

The "Checklist Designer" in return ascended from a client requirement that proved valuable enough to be included in the standard solution. To meet his requirement, Coresuite.com implemented a web client that allows designing a list of complex UI elements (text areas, radios, checkboxes, picture boxes, etc.) with associated descriptions and instructions and then mapping it with a specific equipment, so that every time a technician needs to service it, it will be forced to complete the entire checklist before moving on.

Finally, a later addition to the family was "Service Me" or "The Self-Servicing Client". This was meant for end-users providing them with the capability to identify their product by scanning a barcode and integrate with the service & support workflows in a digital manner: download manuals and related information, check & order available upgrades, manage & schedule service revisions and more. The application is also highly customizable allowing the Servicing Company to upload their logo on the interface, change color theme, change form structure, etc.

IV. INTEGRATION PROJECTS

A. Foundations

During the efforts of expanding the solution on the process automation side, interest stated being spanned for the solution by customers that were using different ERPs in the back office, so Coresuite stated equipping itself with more connectors: "Microsoft CRM Connector" and "File Connector". In a simplified network-diagram view, a Connector is a background service, typically one-time configured, that synchronize business data between the ERP and the Coresuite Cloud.

Recently, coresuite.com introduced its standalone mode that generated the need of implementing the Excel Importer which enables mapping of CSV columns to Cloud Data Transfer Object (DTO) fields and via a simplistic UI perform a one-time upload of data.

Thereby, the Coresuite solution generously rounded up its portfolio with all the above mentioned applications and more, including some third party Connectors developed by partners on top of the Coresuite Application Programming Interfaces (APIs). However, this implicitly rose the integration complexity between the solution components and implied significant architectural changes that will also be discussed later on. A simplified view of the solution today can be found in Fig. 2.

B. Concept

The Cloud Integration started as a simple concept. Data needed to be transferred back and forward between an ERP (SAP B1) database and the Cloud one. This implied three high level layers for the Connector, visible in Fig. 3.

The Data Interface (DI) is the layer responsible for the communication with the ERP. Most of the ERPs out there support plugins or extensions and in order to do that, the ERP needs to provide the plugin software with a notification system. The DI layer will hook into this notification system and will receive information about data content and structure as it is being modified by a client application user or a different plugin application.

On the other end, the Cloud Interface (CI) layer is symmetrically responsible for the communication with the Cloud. Based on the implementing Cloud API, the CI layer will receive changes performed on the cloud mobile clients and store them in the ERP DB, while pushing up changes performed by the back office users in the ERP clients.

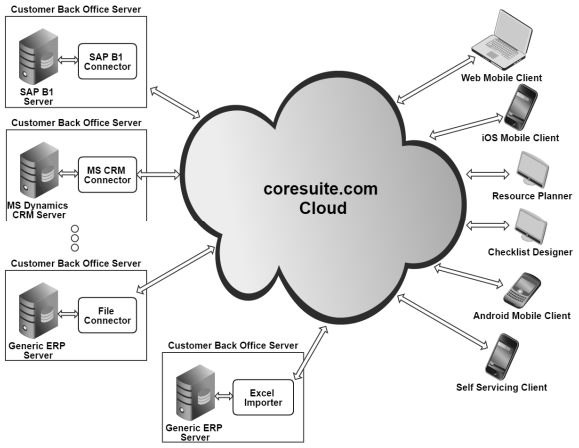

Figure 2: Coresuite's Component Architecture Nowadays

This leaves us with the Connector Service layer, responsible to managing the synchronization process between the two entities. This implies data and configuration responsibilities alike:

Data-wise, the Connector needs to implement data mapping – two-ways between the Cloud and ERP DOMs, upload object priorities, possible handshake protocols (2/3-way confirmation), triggers, data migrations, etc.

Configuration-wise, it needs to pull off a fist-time-use configuration, store Cloud and ERP credentials and sync information, process and execute Cloud remote commands, handle messaging between applications, manage backend information request, implement security protocols, etc.

Depending on the complexity of the requirements, a Connector can go anywhere between a very thin layer and a feature-bundled, full-fetched application.

C. Evolution

As drafted in the previous sub-chapter, the integration was a topic that had to suffer a lot of rethinking throughout time. It started from the solution proof of concept, where the only job of the SAP B1 Connector was to acquire data in a raw format, based on an object type and id and pass it to the Cloud. Since the proof of concept only included a couple of business objects – in read-only mode, there was no extra logic for the connector to implement: no triggers, data migration, etc. Moreover, since the data was acquired on the Connector side but then the mapping was done in the cloud, the connector layer in the on premise side was very thin.

Soon other business objects were added to the solution and the first limitation came into play: SAP B1 was not handling data correctly. There were – and still are – two type of limitations here:

The first one is bugs in the ERP data interface software. Unfortunately, the bigger the ERP, the more seldom the releases – including hotfixes. In addition, while an ERP can survive the market with a buggy release for 6 months or more, a smaller software solution, like Coresuite, would not. If the product is unusable, the customer will just terminate the contract and move on.

The second, more infrequent but at the same time more severe, is miss-design. One of the simple examples is mutable (naming convention based) database primary keys. In this scenario, changing a key would generate duplicated data on the cloud, since the notification system is unable to provide any detail about the DB record history.

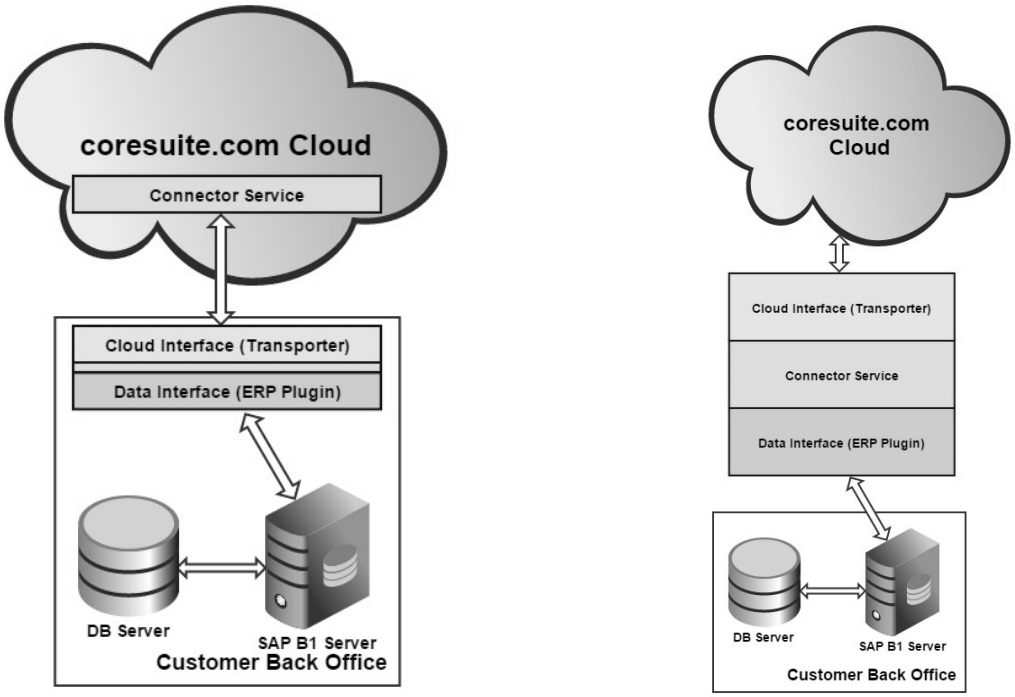

Figure 3: B1 Connector Desired vs. Actual Layers

Therefore, if the ERP was unable to correctly compose reliable raw data for the B1 Connector, it had to get the data itself. Sometimes by extending the DB structure, other times just by querying in a different place, the Connector was now responsible of knowing what data is stored where and how to assemble it.

In time, features came in that required the connector to implement additional complexity, like special services, self-managing updates, injected settings and commands, etc. Meanwhile the MS CRM Connector was under development and the same pattern started reproducing there as well – with similar but duplicated efforts required.

D. External Integration

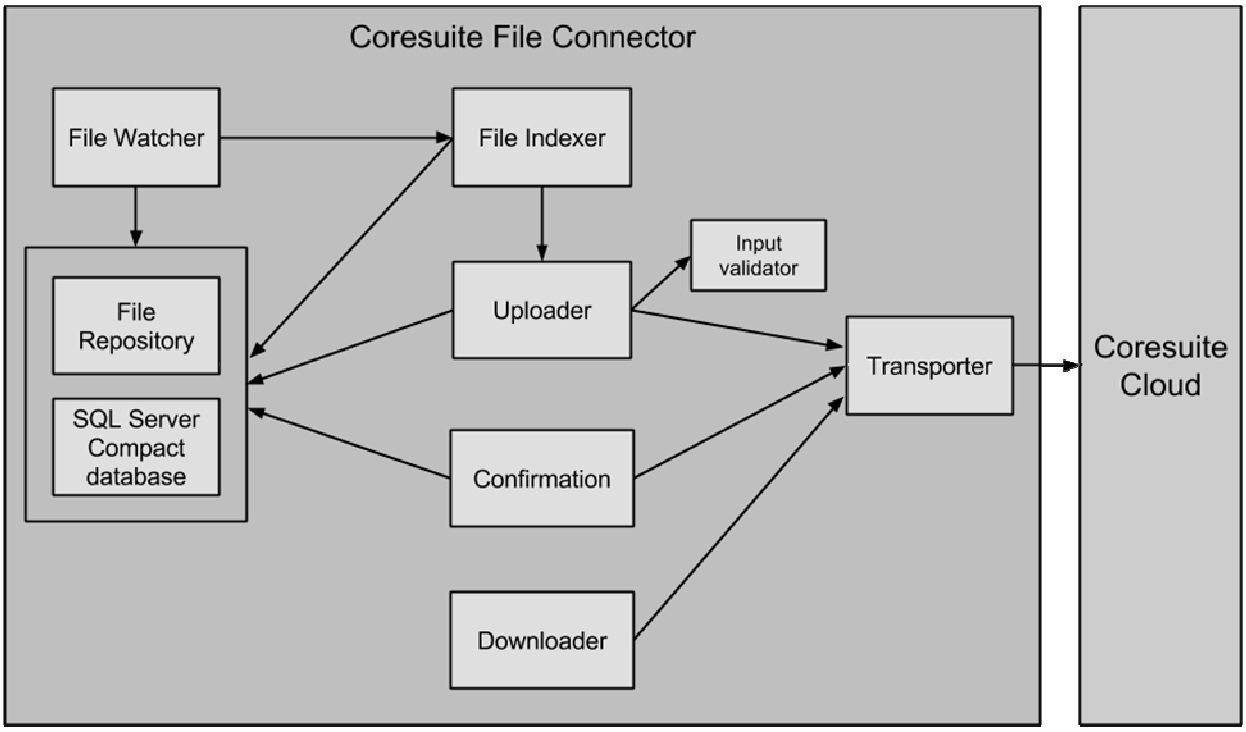

Figure 4: B1 Connector Desired vs. Actual Layers

Like every big system, the solution had to provide a synchronization API. Coresuite actually exposes a bunch of them: Data, Bulk and Sync. However, these are most of the time hard to prototype on top and require a decent amount of internal knowledge to master cross solution flows. Because of this, but also because of duplication effort reasons, we stated to provide several abstraction layers on top of the ERP Sync API.

This first one was the Transporter library, previously referred to as the Cloud Interface, capable of internally managing the communication with the Cloud while protecting it at the same time by developer miss-use.

The second, with its high level architecture visible in Fig. 4, was the File Connector application, as well mentioned above, which is a full connector (protocol) implementation, but with an open DI endpoint.

This is actually an xml file interface, meant to interface with an Extract Transform Load (ETL) or Object Relational Mapper (ORM) endpoint or and equivalent consultant level tool that requires no or limited software development knowledge to interact with.

While each Connector implementation is different and unique, most of the differences will be seen in the DI area, where ERP specific technologies and protocols with be a requirement. However, all existing Connector implementation and possibly future implementations based on the same API will share a great deal of the protocol bound components, like the Uploader, Downloader and Confirmator, as well as the Transporter library.

V. ADOPTION RESULTS

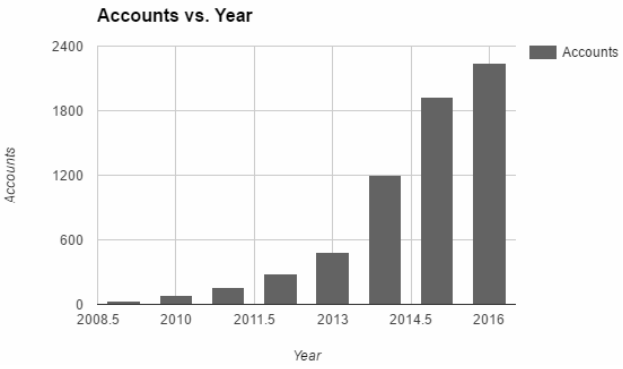

Figure 5: Curesuite Adoption Trend

Coresuite's integration options did have an impact on commercial adoption, shaping an exponential growth in both accounts and users between 2009 and 2016, as shown in Fig. 5.

However, most of the existing productive account today are not due to external integration options, but rather the original SAP B1 Connector, who to this day is the most used and generates the most revenue, as visible in Fig. 6.

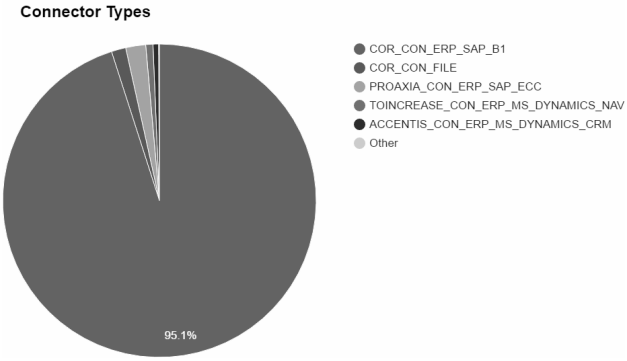

Figure 5: Connector Types Distribution

V. CONCLUSIONS

In respect to the case study presented in this article, we can see that a full Connector implementation for every available ERP is not a scalable solution in terms of time or costs, thus the requirement for an integration platform capable of quickly encompassing new systems is imminent for a software solution that aims for expansion and innovation.

Also, an integration project for business software goes far beyond data mapping and replication to priorities, handshake protocols, triggers or data migrations. At the same time, it needs to account for configuration, persistence, remote capabilities, messaging, security and privacy. Therefore, the architecture of such an application needs to be carefully designed to support at any time changes and extensions with ease.

The external integration options do not fully raise to their expectations on the business side. While this can be a usability problem, the result of this implementation definitely raises a question about the Coresuite's partnership model and integration strategy.

A future research direction will drill deeper into the integration highlights of the above presented Coresuite solution while drawing the first steps that have already been taken toward a modular, decoupled and highly scalable integration framework.

ACKNOWLEDGEMENTS

Coresystems AG, a Swiss based international company, offering could-based field service management products, supports the work of this paper [21].

REFERENCES

[1] F. Robert Jacobs, F.C. 'Ted' Weston Jr., "Enterprise resource planning (ERP)—A brief history", Journal of Operations Management, Volume 25, Issue 2, March 2007, Pages 357–363

[2] Mohammad A. Rashid, Liaquat Hossain, Jon David Patrick, "The Evolution of ERP Systems: A Historical Perspective", Copyright © 2002, Idea Group Publishing

[3] Kevin Prouty, "Aging ERP - When Old ERP is Too Old", June 2011, AberdeenGroup

[4] Heather Herald, "Extended ERP reborn in b-to-b" InfoWorld, August 27–September 3 2001, Vol. 23 Issue 35/36, p21, Trade Publication

[5] Piotr Soja, "Success factors in ERP systems implementations: lessons from practice", Journal of Enterprise Information Management, Vol. 19 No. 6, 2006, pp. 646-661

[6] "What Is ERP?" http://www.netsuite.com/, Copyright © 1998 - 2013 NetSuite Inc.

[7] "ERP History", http://www.saptech.8m.com/erp_history.htm, 2013

[8] Shahneel Baray, Shafqat Hameed, Atta Badii, "Analyzing the Effectiveness of Implementing Enterprise Resource Planning in the Printing Industry", European and Mediterranean Conference on Information Systems (EMCIS), July 6-7 2006, Costa Blanca, Alicante, Spain

[9] Shashank Jangiti, "SAP BUSINESS SUITE SOFTWARE MODULES", University Of Northern Virginia, June 24th 2010

[10] Ejaz Ahmed Bhatti, "What is Enterprise Resource Planning", GLOBUS Engineering Management, August 8th, 2010

[11] Tim R. Coltman, Timothy M. Devinney, David F. Midgley, "Customer Relationship Management and Firm Performance", Journal of Information Technology 26(3): 205-219 (2011)

[12] Russell S. Winer, "Customer Relationship Management: A Framework, Research Directions, and the Future", University of California at Berkeley, April 2011

[13] Ram Ganeshan , Eric Jack , M. J. Magazine , Paul Stephens, "A taxonomic review of supply chain management research", Quantitative Models for Supply Chain Management, (1999)

[14] Amit Sachan, Subhash Datta, "Review of supply chain management and logistics research", International Journal of Physical Distribution & Logistics Management, Vol. 35 Iss: 9, pp.664 - 705, (2005)

[15] Eswaran Subrahmanian, Sudarsan Rachuri, Sebti Foufou, Ram 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, No. 1, 2005

[16] "Independent, Expert and Balanced ERP Software Analysis, Reviews & Insight", http://www.erpsoftware360.com/, 2013

[17] PointBeyond, "10 Business Benefits of Automating Processes with SharePoint 2010 Workflow Applications", 12.08.2011, http://www.pointbeyond.com/2011/08/12/10-business-benefits-of-automating-processes-with-sharepoint-2010-workflow-applications

[18] Min Chen, Shiwen Mao, Yin Zhang, Victor C. M. Leung: "Big Data Applications", SpringerBriefs in Computer Science pp 59-79, 7.04.2014

[19] Guo Chao Alex Peng, "Cloud ERP: A New Dilemma to Modern Organizations?", The Journal of Computer Information Systems, Vol. 54, No. 4, pp. 22-30, Summer 2014, Stillwater, ISSN: 08874417

[20] Fumei Weng and Ming-Chien Hung, "Competition and Challenge on Adopting Cloud ERP", International Journal of Innovation, Management and Technology, Vol. 5, No. 4, pp. 309-313, August 2014, ISSN: 20100248

[21] "Coresystems AG" Official Homepage, http://www.coresystems.net/

[22] Andrea Vermurlen, "How much does SAP Business One cost?", http://www.aetherconsulting.com/blog/article/935/how-much-does-sap-business-one-cost/, 1.11.2015

[23] Coresuite Helpdesk Docs, https://helpdesk.coresystems.ch/hc/en-us/categories/200394202