HAL
open science

Cloud Integration Patterns

Danny Merkel, Filippos Santas, Andreas Heberle, Tarmo Ploom

To cite this version:

Danny Merkel, Filippos Santas, Andreas Heberle, Tarmo Ploom. Cloud Integration Patterns. 4th European Conference on Service-Oriented and Cloud Computing (ESOCC), Sep 2015, Taormina, Italy. pp.199-213, 10.1007/978-3-319-24072-5_14 . hal-01757562

HAL Id: hal-01757562

https://inria.hal.science/hal-01757562

Submitted on 3 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Cloud Integration Patterns

Danny Merkel[1], Filippos Santas[2], Andreas Heberle[3] and Tarmo Ploom[2]

[1] Julius Bär, Zürich, Switzerland
danny.merkel@julius-baer.com

[2] Credit Suisse, Zürich, Switzerland
{filippos.santas,tarmo.ploom}@credit-suisse.com

[3] Karlsruhe University of Applied Sciences, Germany
andreas.heberle@hs-karlsruhe.de

Abstract. Enterprises use the cloud for unlimited resource, scalability and elastic provisioning along with being able to use state of the art commodity or specialized solutions available in the cloud. The challenge of this vision is the proper and safe integration of on-premise IT-Landscapes with data and applications in the cloud. To find solutions for integration of classical and cloud environments two approaches, top-down and bottom-up, were used. In the top-down approach cloud integration patterns were specified based on scenarios. In the bottom-up approach cloud integration patterns were based on case study application requirements. Results of this paper are novel cloud integration patterns for various cloud integration scenarios.

Keywords: Cloud Computing, SOA, Integration, Topology, Patterns, SaaS, Public Cloud, Private Cloud, Multi-Cloud

1 Introduction

Cloud computing has emerged as one of the key technologies that are or will be heavily used by companies. According to a study of IDG Enterprises, 42% of the IT decision makers are planning to increase spending on cloud computing in 2015, making cloud computing projects the most important IT initiatives [3]. Enterprises with large application landscapes benefit from the availability of (potentially) unlimited resources and the elastic provisioning of cloud resources. The different service models are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS); Software as a Service (SaaS). Together with the different deployment options (Public Cloud, Private Cloud and Hybrid Cloud) these allow for various integration options to optimize communication and deal with sensitive data. The hybrid cloud is one of the major areas where cloud integration is required and practiced with various levels of success. The question is: how to migrate existing enterprise application landscapes to a cloud computing environment? Existing enterprise application landscapes are usually based on heterogeneous technologies deployed on-premise with a high degree of tight cou-

pling between applications in the form of point-to-point integration and sensitive data flows unconstrained within the application landscape. This makes the adoption of cloud computing challenging.

Contribution: This paper examines how on-premise enterprise application landscapes can be integrated with private and multiple public clouds and with SaaS avoiding point-to-point integration. Various cloud integration scenarios and topologies are described and cloud integration patterns with required middleware are identified. Furthermore, we examine how cloud integration contributes to financial benefits.

This paper is organized as follows: Section 1 gives an overview on existing integration patterns; section 3 focuses on the integration problem in more detail; in section 4 we introduce the approach we used to find suitable integration patterns; section 5 shows our results. The last section concludes our work and discusses future research.

2 Integration Patterns and Related Work

Patterns represent the collective experience of software experts and allow for the cost effective implementation of software's non-functional requirements reducing development cost from 10%-35%, improving time to market up to 20%, and reducing maintenance costs by 15%-20% [6]. Proven message exchange patterns (MEP) patterns are used similarly in enterprise landscapes to integrate services and applications without harming system runtime (e.g. performance) and maintenance requirements. Services can communicate synchronously, or asynchronously and may consume bulk data asynchronously. The commonly accepted standard for MEPs is defined by the Web Service standard [10]. Baros, Dumas and Hofstede discuss interaction patterns for orchestrated web services with BPEL [1]. Hophe and Woolf describe general Enterprise Integration Patterns [4] like Message Routing, Message Transformations and error handling.

There is extensive literature on how to build a SOA and clouds using design patterns. T. Erl examines commonly applied SOA patterns [5], such as Enterprise Service Bus (ESB) including service broker, asynchronous messaging, etc., and Service Design Patterns for security, messaging, service implementation, etc. Cope, Erl et al. identified Cloud Computing Design Patterns to support scalability, reliability, or monitoring of cloud environments and applications deployed in a cloud [7], [18]. Fehling et-al introduce patterns for cloud offerings and design and management of cloud applications [15]. We observe that middleware patterns related to SOA are applicable to the cloud. We identified the following categories of patterns applicable in SOA and cloud architectures:

1. Patterns related to the Service Loose Coupling principle. These are implemented by the middleware messaging at the communication layer between services. Common implementations include the components of an Enterprise Service Bus.
2. Patterns related to the Service Autonomy principle. These involve storage and data replication and resource redundancy.
3. Patterns related to the Statelessness principle, which allows for a state repository for improving the availability and reliability of services.

Cloud usage in an existing enterprise landscape requires more topology-oriented patterns to integrate cloud capabilities with existing applications. Such topology patterns are not covered by the existing approaches.

3 Problem Statement

Enterprises invested vast amounts of money and resources in the last 40 years in building their own IT infrastructure, services and applications. With the advancement of the software industry many of these solutions are nowadays implemented by 3rd party providers in clouds and are available for reuse. The locally implemented CRM system or payments infrastructure that evolved over time and cost millions, no longer bring any competitive advantage and may become a cost factor that does not allow for prompt satisfaction of new business requirements. Using applications in the cloud is a way to increase organizational agility and reduce operational costs.

Multi-Cloud

Access to cloud implementations is constrained by regulations, company policies and billing policies. National laws may prescribe encryption for storage of data and disallow the replication of sensitive data to other countries. Other laws may prescribe that the operational data of a company related to a country lie within the borders of the country, even if the server hubs are in another country. SaaS offerings do not always guarantee confidentiality or controlled access for the data and where and how temporary state data is stored. Further, there are limitations in the products and policies of cloud providers, in particular billing.

Enterprises have the choice to use multiple cloud providers. The sensitive data may be hosted in one cloud (potentially private), the CRM system may be supported by a public (3rd-party) cloud provider, the payments by another public cloud provider and the archives may be hosted in a third cloud that has better prices for storage of large amounts of data, provides discovery and encryption, and guarantees storage within national borders. A fourth provider may provide data warehouse services with big data analysis for marketing purposes. Further, a provider may charge only for retrieval of data and not for the amount of data stored, while another provider charges only for storage of data but not for CPU consumption or file download. Assuming that everything else is equal, it is financially beneficial for an organization to use the former to store huge amounts of data and to use the latter to process data. In order to maintain decent operational costs and take advantage of the modern implementations of applications in the cloud, an international enterprise will decide to use different cloud providers for different applications.

Fig. 1. Potential Architecture

This distributed solution (Figure 1) is not an extreme scenario. Financial institutions, pharmaceutical companies, big travel agencies, almost all companies in developed countries face this dilemma. If they do not use technologies on the cloud they are confronted with high IT development and operational costs, while usage of these technologies from a single cloud provider will most probably result in legal issues, non-optimal operational costs and vendor locking costs. Statistical results are provided in [16]: the majority of the companies in the UK that use cloud purchased offerings from three vendors, although they prefer to use only one vendor.

Coupling in Enterprise Landscapes

Traditionally, the main implementations in the cloud involved single isolated applications with minimal dependencies to other applications; provisioning of market data; exchange rate calculations; auction systems; real estate databases; time plans; etc. Such systems are typically isolated from other applications, can be managed online, or fed asynchronously with data. Their migration to the cloud (or reuse of a SaaS offering) requires mainstream technologies and affordable effort.

But what happens if a single application is already integrated with several other applications that require a number of mandatory components to function properly? Migration of all dependent applications at once is associated with high risks and lack of any ROI (Return of Investment) in the first years after the project starts. A step-by-step migration reduces the risk and may show ROI shortly after each application has been migrated. This is however architecturally challenging.

The first step is the migration of internally hosted applications to a single cloud environment. This results in Single-Cloud Integration. Compared to Simple SaaS Integration, the integration effort is substantially higher. Due to dependencies, the externally hosted applications cannot operate without the depending internal systems. If more parts of this application ecosystem are deployed externally, reliable intra-cloud communication is necessary.

Multi-Cloud Integration

Services deployed and integrated in different clouds must be orchestrated so that they provide support for end-to-end business processes or for support of different kinds of processing. Credit applications may be created in a CRM system in one cloud; rated by experts in an in-house application with proprietary rating models; sent back to the CRM application; formalities and other documents are collected and archived in a third cloud; and the account opens in the private cloud.

These activities involve separate systems where each system does not know about the other. The CRM does not know the rating models, and the archive is agnostic to either of them. A separate orchestration mechanism must be available that allows for the flow of data and consistent state transition in the objects across the clouds. Notice that this orchestration manages long running transactions that tend to accumulate substantial state during their execution, including confidential data.

Access Control, Integrity, Confidentiality

Public access to the clouds and exchange of data across geographic locations introduce difficulties that must be resolved. The communication between CRM and an electronic archive of customer documents, or between CRM and payments systems, must have confidential customer info either stripped or encrypted, otherwise confidentiality may be compromised. Customer or transactional data can be patient data, the subjects of a medical experiment, the composition of a new substance, the client of a bank, etc.

Cross border constraints may further impose constraints on solutions. In several industries in the EU and the US, there are restrictions on the storage of customer data in other countries. For example medical data or certain formalities in the financial industry may not even leave the country from where they have been created.

Many organizations check security constraints only in the simple context of the applications that need to exchange data. Most organizations do not check security in solutions involving different applications in different clouds. In the cloud however, security requirements like authentication and authorization have to be implemented for the whole business process, i.e. over several clouds with different technical infrastructure.

4 Methodology

In this section we describe the process for identifying cloud integration patterns. The landscape is within Credit Suisse AG, a large organization in the financial services. The variety, variance and criticality of the requirements across domains in this industry allow us to examine a lot of useful and challenging use-cases involving the cloud. The analysis is done in three steps:

1. current state analysis
2. top-down analysis
3. bottom-up analysis

Each analysis step includes the results of the previous step to improve the evaluation results. The following sections describe these steps in detail.

4.1 Current State Analysis

The current state analysis identifies the progress in cloud adoption in the enterprise. Internal information was collected by interviewing subject matter experts at Credit Suisse and combined with statistical data from a Federated Identity provider. Further, the current integration architecture of these solutions was analyzed.

The results are used to define suitable integration challenges in the top down analysis. The current state analysis reuses the results of different initiatives in the bank. Some of these analyze the potential of integrating applications in the private cloud, public cloud and multi-cloud to align these concepts with the IT strategy.

4.2 Top-Down Analysis

In the top-down analysis significant cloud integration challenges are identified and analyzed from a bird's eye view. The service and deployment models from the NIST [13] are used within the study. From the possible integration scenarios we selected those that clarify major aspects of multi-cloud integration.

The selected scenarios are: SaaS Integration, Public Cloud Integration and the Centralized and Decentralized Multi-Cloud Integration within Hybrid Clouds. The variations for PaaS or IaaS integration are not significant for the integration architecture at this level of abstraction. For simplicity in the presentation, without loss of accuracy, the private cloud is considered as a special case of the public cloud.

The common problem examined in all scenarios is how service integration challenges can be solved by integration patterns. Well-known integration patterns (see [4], [5]) are reused and combined together. Different combinations of patterns have different advantages and disadvantages and can solve an integration challenge in different ways. It is important that we select combinations that have a high score with respect to the quality requirements of the overall solution, including financial benefits. How do we rate such combinations? Each combined pattern (called a topology pattern) is rated by standardized evaluation criteria. As evaluation criteria we consider the system qualities [17], that are, non-functional such as Performance, Latency, Availability, Reliability, Extensibility, Maintainability, Security, Integrity, Scalability, and Portability across clouds. The SOA principles [5] contribute to the overall success of the architecture within an organization. It is imperative that the application of integration patterns respects these principles.

4.3 Bottom-Up Analysis: Two Applications

Different architectures are typically applied in different business domains or application areas. The one size fits all is associated with increased risks, high complexity and difficulty to obtain results in big organizations. In the bottom-up analysis the impact of different, real-life application types and architectures is analyzed. In our study we selected two application types that are as diverse as possible in order to identify the disparities in integrating cloud architectures in different contexts. One scenario focuses on a data intensive application and another on a computation intensive application.

As data intensive applications we consider applications that handle hundreds of terabytes of data, require data integrity and consistency and allow for discovery of information in very tight time constraints. As an example of a data intensive application we consider a large archive system in the ECM (Enterprise Content Management) domain, subject to tight regulatory conditions and able to satisfy requirements for investigations and litigation.

Computationally intensive applications must satisfy real time performance requirements, thousands of transactions per second and with very high availability and reliability. Business criticality of such applications increases the importance of these requirements. An example application that is also business critical for a financial institution is a trading system in the Securities domain.

The applications in these two distinct domains (ECM and Securities) help us to identify suitable patterns and combine them in an enterprise cloud architecture. The differences in these architectures are then compared against each other. New patterns are listed and put into context. Further, we examine the applications in these two domains in an orchestrated environment for providing end-to-end business processes.

5 Results

In this section we describe the patterns that have been identified in our three-step analysis. The results are obtained in the order defined in the methodology section, in several iterations within our big organization. It is not necessary to complete the state analysis in order to continue with the top-down and bottom-up analysis. We proceed to the next step with partial analysis results from the previous step.

5.1 Results from Current State Analysis

The internal research in the current state analysis indicated an ongoing and growing use of cloud service offerings even in a risk averse industry.

SaaS Integration

The SaaS integration has been the first integration challenge for our enterprise. The initial use of cloud services started with isolated mainstream SaaS applications for currency converters, legislation documentation (for the Legal and Compliance departments), registries of companies and business at national level (for checking the status of clients), intranet repositories, payments applications, Lombard credit rating models, price comparison data, financial instrument databases, etc. SaaS applications can be easily used and don't require long and expensive internal software provisioning processes. SaaS is often used for less critical software demands.

Initially, a SaaS application is designed to provide application functionality isolated from the enterprise network; we call this Simple SaaS Integration. The main concern for integrating the cloud applications with the applications of the internal environment was the accessibility and usability of the systems using strong authentication. Thus the integration effort was limited to SSO (Single Sign On) and PKI (Public Key Infrastructure) integration to enable implicit but secure login. In these cases, further integration into an existing application landscape was neither possible nor necessary. No major data flows were required from the applications in the private environment to the isolated SaaS in the public clouds. Data could be imported from the public SaaS, but there was no confidential out-going traffic, except for user credentials and certificates. This result is very intuitive and supports the problem statement of section 3.

As the functionality of this SaaS application is extended the integration effort increases. SaaS market leader Salesforce indicates this through several available service interfaces for their application [9]. We call Advanced SaaS Integration any SaaS integration that goes beyond isolated applications and targets the integration of a web of applications across organizations or across clouds. To enable usage of the entire func-

tionality, these applications must be integrated with the services that are currently provided by the private enterprise landscape.

Complexity of Integration

Our current state analysis exposed the hard reality that integration in the SaaS context is almost always direct Point-to-Point Integration. This covers SaaS to SaaS integration as well as SaaS to enterprise environment integration. The Point-to-Point Integration is beneficial only if it is limited to a small number of nodes. In this case it may have a straightforward implementation and results in higher efficiency and availability due to the direct communication between the nodes. On the other hand, this pattern has disadvantages in SaaS service management and maintainability. Each node of the service communication is typically proprietary and involves transformations that are implemented redundantly in other nodes. The amount of transformations for each node has polynomial complexity. Without direct support from a centralized cloud infrastructure the result is increased management and maintenance effort (Figure 2).

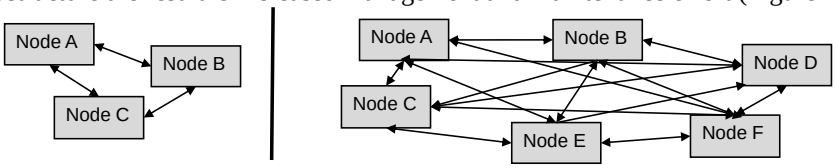

Fig. 2. Point-To-Point complexity for 3 and 6 Nodes

Overall, the current state analysis counted 23 simple and 2 Advanced SaaS Integration nodes. However, we expect that there are nodes hidden behind undocumented or external workflows and thus the number of integration nodes is higher in each category. A concept for standardizing the integration is needed at an architecture level in order to manage the high complexity of point-to-point integration.

5.2 Results from Top-Down Analysis

After concluding the current state analysis, a three step top-down analysis was performed. Each step examines the problem: How can the services communicate with each other and how do the solution topologies look like? During the analysis, we identified different patterns for each scenario. Due to space restrictions we present only selected patterns in more detail. These are SaaS Integration, Single-Cloud Integration, Centralized Multi-Cloud Integration and Decentralized Multi-Cloud Integration. The full description can be found in [14] along with information on the method we used for assessing the benefit of the patterns. In the following we examine each of these steps separately.

SaaS Integration

Problem: The integration of SaaS applications with applications in the private cloud results in Point-To-Point integrations with tight coupling and growing complexity.

Solution: With the SaaS Broker Integration (Figure 3) we introduce an intermediate layer in a cloud environment layer by applying the ESB (Enterprise Service Bus) pattern [11]. In this pattern the ESB is deployed to a public cloud. The ESB operates as a broker between the SaaS applications and the enterprise environment. It controls the communication between the applications in the different SaaS environments. No direct communication between the SaaS applications is allowed.

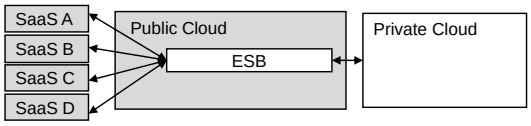

Fig. 3. SaaS Broker Integration Pattern

Consequences: The SaaS Broker Integration pattern enables centralized control of all SaaS communication and scales better compared to Point-to-Point integration. All service calls to the internal environment can be filtered or transformed to company standards. The single point of failure has negative impact on the availability, even though load balancing and failover may cover this risk. A drawback, especially for time critical services, is additional latency through transmission of the ESB. This effect can be significant if SaaS solutions are distributed in datacenters worldwide. If latency in transmission is critical, then direct communication should be preferred.
A similar solution using intermediate layers for integration among clouds is offered by several providers as Integration Platform as a Service (IPaaS) [8]. The difference between an IPaaS and the SaaS Broker Integration pattern is the self-hosted ESB in the Public Cloud versus a standardized IPaaS platform.

Single-Cloud Integration (Simple Hybrid Cloud)
Problem: A single public cloud is integrated with the private cloud. Services are deployed to both clouds and need to communicate with each other.
Solution: By applying the Distributed ESB pattern (Figure 4) we introduce two ESBs, one in each cloud. The intra-cloud communication is handled by the corresponding cloud ESB. Cross-cloud communication is steered over both ESBs.

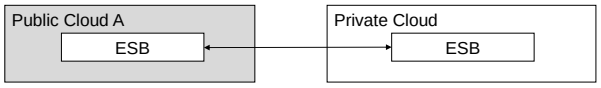

Fig. 4. Distributed ESB Pattern

Consequence: This pattern enables direct intra-cloud communication for each of the environments which has positive effects on the latency and availability of the overall topology compared to a single ESB in one of the environments This pattern shows advantages through the distribution of the ESB infrastructure which enables intra-cloud and cross-cloud communication. Further, this pattern allows for nodes to operate independently of the availability of other nodes. A disadvantage is the additional management effort for the ESB in the cloud.

Centralized Multi-Cloud Integration

Problem: Multiple Clouds need to be integrated in an enterprise network. Services are deployed to all environments and need to communicate with each other.

Solution: The Internal Cross-Cloud ESB is applied (Figure 5) resulting in an architecture where each of the multiple clouds has its own ESB. All cross-cloud communication is routed over the internal ESB. The intra-cloud communication is managed by each ESB separately.

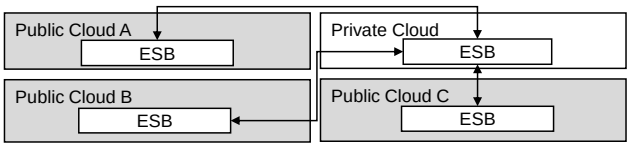

Fig. 5. Internal Cross-Cloud ESB

Consequence: This pattern supports integration of different platforms across cloud providers and decoupling within each cloud. Each cloud provider can select the appropriate brokerage and messaging platform that is available for all the services deployed within the cloud. The integration with the ESBs of each cloud provider is done by the organization's ESB in the private cloud, which manages the internal, private applications storing confidential data. This approach scales very well for different cloud providers; big organizations that take advantage of the platforms of each cloud provider. Another advantage of this topology is that is allows for centralized cross-cloud service communication management. The Internal Cross-Cloud ESB pattern is useful when the cross-cloud communication needs to be centrally controlled and restricted. Disadvantages are introduced through the additional transaction time through the cross-cloud communication.

Decentralized Multi-Cloud Integration

Problem: Same problem as in Centralized Multi-Cloud Integration where a centralized ESB is not possible or is not desired.

Solution: The Peer to Peer Multi-Cloud Integration Pattern (Figure 6) is applied, resulting in an architecture where each cloud has its own ESB. The ESB of each cloud is able to communicate directly with the ESBs of the other peer clouds. The intra-cloud communication is managed by each ESB separately. The direct communication between each cloud has positive impact on the latency of cross-cloud service communication in time critical cross-cloud communication and avoids the single point of failure in integration.

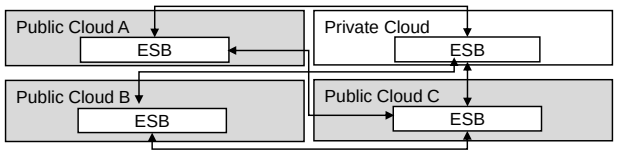

Fig. 6. Peer to Peer (P2P) Multi-Cloud Integration

Consequence: A disadvantage of this approach is that cloud providers with different platforms must implement broker functionality supporting all the format and protocol transformations required for the communication with other cloud providers. This introduces development and integration overhead. Further, it is very unlikely that cloud providers are able to support data model transformations required for the exchange of data across applications in different clouds. Additionally, high effort is necessary to manage this environment centrally. Monitoring of the communication requires additional components; this increases the complexity and results in additional development and maintenance effort.
This pattern can be implemented through a separate ESB management component. Alternatively an existing ESB is the master ESB and collects monitoring information from other ESBs.

5.3 Results from Bottom-Up Analysis

As mentioned in the Methodology section (4), the patterns identified by the top-down analysis have been applied in the bottom-up approach in the two domains. Nonfunctional requirements influence the architectural decisions; e.g. direct communication for increased performance vs. hub communication for reduction of costs and increased maintainability. Further enhancements include the support of mission critical integration aspects like authentication and authorization, monitoring (including load balancing, and usage monitoring) as well as reporting (including billing).

Topology Description
In all scenarios the topology consists of three environments. The private environment and two public clouds. The internal environment is necessary, because several applications cannot move to the public cloud. Two different public cloud environments are used to ensure the availability of the financial transaction system and to prevent any possible data loss and integrity issues in the archive system. Each important application layer is redundantly implemented. In the archive system (Figure 7) only the data layer is replicated over the clouds. The public cloud B runs as a secondary backup solution of the data layer in standby mode. Since business does not require very high availability for the online access of documents, the other layers are not replicated.

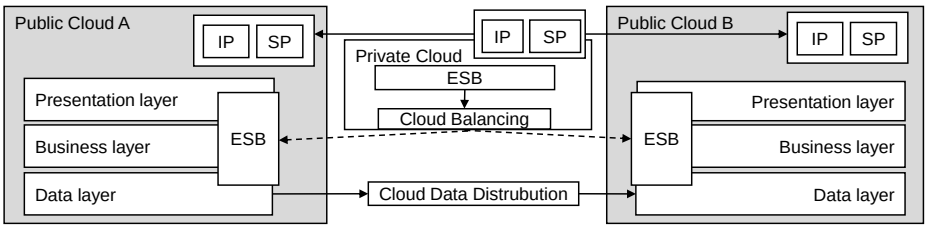

Fig. 7. Topology View Financial Transaction System

The financial transaction system must be fully replicated in two fully operational clouds for high availability (Figure 8). Eventual unavailability of one cloud solution is recognized by the cloud balancer which routes all service calls to the other cloud.

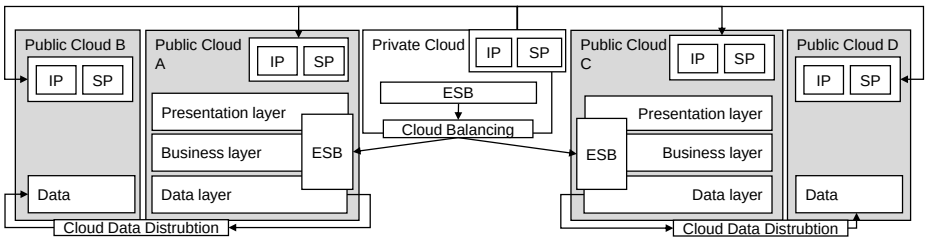

Fig. 8. Topology Archive System

Cloud Balancing and Cloud Data Distribution

We used the Distributed ESB pattern together with the Cloud Balancing pattern [7] for integrating different clouds with potentially different loads. Cloud balancing allows for IT resources to be load-balanced across multiple clouds and must not be confused with the cloud load balancing which distributes load in a cluster of servers within a single cloud. Cloud balancing requires all the mechanisms of the load balancing and workload distribution, along with the existence of redundant storage and servers in another cloud. Services can be cloud balanced if they have been provisioned in different clouds and user and service provisioning are supported across clouds.

In our implementation we are using clusters of servers (and hypervisors) within each cloud, but we are not clustering across clouds. When a service is load balanced to another cloud, then the temporary state of the service persists and can be migrated to the other cloud. Additionally, we are using Cloud Data Distribution to distribute queries on data across clouds. Data is distributed and replicated across clouds according to legal constraints along with requirements for performance and availability.

In the financial transaction system, the cloud balancer decides whether to use the service capabilities in public cloud A or public cloud B. The internal environment controls the cloud balancer and is not affected by the unavailability of either cloud.

In the archiving solution, all document searches and retrievals are balanced by the cloud balancer. Along with availability requirements, the documents of different customers may be located in different clouds due to legal and regulatory reasons. Appropriate mechanisms based on business and technical logic decide which cloud to access for different documents. Further, the archives require replication and integrity of the data layer in another location. Therefore Cloud Data Distribution mechanisms synchronize the data between different cloud environments.

Notice that the data distribution across clouds for the archiving solution must satisfy the data integrity requirements of the archive. A typical backup solution may periodically copy data from primary storage to secondary storage at binary level without knowledge of the objects represented by these bytes. But, if a document has been imported to the primary storage of the archive, and after that has been accidentally deleted, the backup to the secondary storage may not notice this import and deletion.

The cloud data replication includes checks and business rules that guarantee the integrity of the archive in all environments.

Security Topology Related Patterns
In order to establish the trust relation to externally hosted environments in a multi-cloud topology, we introduced authentication and authorization components in the top-down analysis. In the two applications analyzed in the bottom-up approach, all nodes use the same authentication and authorization mechanisms. In all clouds we had similar LDAP directories and PKI infrastructure and compatible authorization rule engines. The authentication and authorization data is synchronized across the external environments. This approach centralizes the management and control over the authentication and authorization system and improves the governance of security data. However, the synchronization requires some implementation effort. In the financial transaction system an identity and service provider is deployed to each cloud. In the archive system an identity and service provider is only deployed to the pubic cloud A.
The authentication and authorization system in each node must be connected with the internal master system and the synchronization jobs need to be configured and kept up to date. This cost is, negligible compared to the costs of maintaining different security mechanisms in each cloud and provisioning users separately in each cloud.

5.4 Important Insights from Bottom-Up and Top-Down Analysis

The examination of two real-life business scenarios in the bottom-up analysis gave us an interesting insight: Even with significant differences in the requirements, the resulting cloud architectures appeared to be very similar. The important layers were redundantly implemented; in the data archive the data layer and in the financial management system all layers were replicated. In both cases the Distributed ESB, Cloud Balancing and Cloud Data Distribution patterns have been applied. Both cases profit from the Cross-Cloud Monitoring and Cross-Cloud Security patterns. The former allows for performance and load control of the cloud services and the latter enables end-to-end security over different clouds. Currently, cross-cloud monitoring requires self-developed infrastructure because the monitoring capabilities of the different clouds are not (yet) standardized and the clouds use proprietary implementations.

6 Future Work

Our paper scratched the surface of cloud integration. Several aspects of cloud integration need further elaboration.
Security: Public clouds require significant effort in security measures. Beyond authentication and authorization topology patterns there are numerous other ways to secure public clouds like content encryption, key management, homomorphic encryption, data splitting, computing with encrypted functions, anonymization, data masking, encrypted virtual machines, etc. Future work may define combinations of security patterns to secure a public cloud for targeted trust level.

Cloud Middleware: Cloud middleware is emerging (Amazon, RedHat, Mule, etc.). To provide cloud elasticity there are slight shifts in cloud middleware. Our analysis showed that there is emphasis on asynchronous integration: push asynchronous integration is replaced with pull asynchronous integration, and there is higher emphasis on replication patterns. Further we saw emergence of completely new cloud middleware elements like Cross-Cloud Balancer, Cross-Cloud Data Distributor, etc.

To facilitate migration of existing IT landscapes to the cloud, a mapping of "old" on-premise integration patterns to "new" cloud integration patterns has to be worked out along with the definition of standards in balancing, distribution and monitoring.

Cross-Cloud Monitoring: Today, monitoring capabilities of clouds are limited to a single cloud. In a redundant implementation over different clouds, a cloud monitoring mechanism is necessary to control the load and performance of cloud services. As usual, the main challenge is the lack of standardization of monitoring capabilities, formats, and protocols, but also the lack of standard tools for this activity. A monitoring pattern requires a centralized management component which is linked with the surrounding environments. The monitoring must deliver reliable and agreed service quality during changing demands and be as cost effective as possible. Negative peaks and load throughputs must be addressed for checking cost savings. Scaling strategies are necessary based on application types (e.g., data intensive: transaction time minor relevance, computationally intensive: transaction time highly important).

Cloud Management: Cross-cloud management solutions enable the possibility of optimizing cloud usage and reduce the total cost for the multi-cloud environment based on billing information provided by the cloud providers. In combination with the service SLAs the service provisioning can be optimized to an optimal cost/value ratio. Currently, the cost models of the different cloud providers are not standardized and the prediction of the actual cost is hard and complex. Tools for cross-cloud monitoring and billing do not exist, but will be developed in the context of cross-cloud marketplaces, e.g. provided by Deutsche Börse Cloud Exchange [12].

Cloud Adoption: An adoption of cloud solutions into the enterprise landscape is driven by the offers of cloud providers and software companies. These offerings, especially in the SaaS market, enable the providers to highly standardize their software solution on the one hand and limit customization possibilities on the other. What cloud solutions can replace existing in-house solutions? What tangible steps are needed to migrate existing large-scale application landscapes to cloud based environments? We expect to see more work examining the degree to which Service Orientation and other methodologies need to be applied in order to migrate to the cloud and new patterns that combine SOA with cloud integration.

Multi-Cloud Offers: Enterprise cloud users won't limit their scope to a single cloud scenario. Our research indicated that vendors nowadays still focus on secure intra-cloud solution and don't offer capabilities for the cross-cloud integration. Reasons for the unavailability of cross-cloud support include the lack of advantage for the cloud provider and the fact that the cloud adoption process needs to progress further so that demands for such functionalities grow.

Very high availability requirements, which may not be supported by a single cloud provider, can be covered through redundant implementations over several clouds.

Enabling this scenario through cloud middleware components will address new user groups whose cloud requirements aren't addressed with the existing cloud offerings.

Acknowledgments. This paper is based on a project between the University of Applied Sciences in Karlsruhe and Credit Suisse in Zurich. We thank Prof. Rainer Neumann and Robert Robinson for their feedback, Claus Hagen for his support and Roger Suess, Peter Schnorf and Alain Hsiung for sharing their cloud computing vision.

REFERENCES

1. A. Barros, M. Dumas, and A. H.M. ter Hofstede. Service Interaction Patterns, 3rd Intl Conference on Business Process Management (BPM), Nancy, France, 2005. Springer Verlag.
2. W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns, Distributed and Parallel Databases, vol. 14, issue 3, pp. 5-51, July 2003.
3. IDG Enterprise: Computerworld Forecast Study 2015. http://www.idgenterprise.com/report/computerworld-forecast-study-2015.
4. G. Hohpe and B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, 2004, Pearson Education, ISBN 0-321-20068-3.
5. T. Erl. SOA Principles of Service Design. 2009, SOA Systems inc. ISBN: 9780132344821.
6. F. Buschmann, K. Henney, Pattern-Oriented Software Architecture. Tutorial, OOP 2008, Munich. http://www.sigs.de/download/oop_08/Buschmann%20Mo2%20Patterns_OOP.pdf.
7. R. Cope, T. Erl, and A. Naserpour, Cloud Computing Design Patterns. Prentice Hall/PearsonPTR. To be published June 2015. See also http://cloudpatterns.org
8. Mulesoft: iPaaS: Integration for the Cloud. Cloud. https://www.mulesoft.com/resources/cloudhub/ipaas-integration-platform-as-a-service.
9. Salesforce, Which API should I use? https://help.salesforce.com/HTViewHelpDoc?id=integrate_what_is_api.htm&language=en.
10. W3C, Web Services Description Language (WSDL) Version 2.0 Part 2: Message Core Language. 2007. http://www.w3.org/TR/2007/REC-wsdl20-20070626/.
11. T. Erl et al. SOA Design Patterns. Prentice Hall, 2009, ISBN: 9780136135166.
12. Deutsche Börse Cloud Exchange. The marketplace for cloud resources, 2015. http://cloud.exchange/en/wp-content/uploads/2015/02/20141112_DBCE_Factsheet_EN-1.pdf.
13. P. Mell, T. Grance, The NIST Definition of Cloud Computing, 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
14. D. Merkel, Cloud Integration Patterns. University of Applied Sciences Karlsruhe, 2014.
15. C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. Cloud Computing Patterns. Springer Verlag 2014.
16. Telstra research: Customer Centric Cloud: Hype or Hybrid? March 2015. http://connect.telstraglobal.com/hybrid-customer-clouds.html.
17. L.Bass, P.Clements, R.Kazmann; Software Architecture in Practice, Addison-Wesley, 2007
18. Thomas Erl, Zaigham Mahmood, Ricardo Puttini. Cloud Computing: Concepts, Technology & Architecture by ISBN: 9780133387520, Prentice Hall, May 2013