Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 64 (2015) 977 – 983

Procedia Computer Science

Conference on ENTERprise Information Systems / International Conference on Project MANagement / Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2015 October 7-9, 2015

Cloud Configuration Modelling: a Literature Review from an Application Integration Deployment Perspective

Inma Hernández[a]*, Sandro Sawicki[b], Fabricia Roos-Frantz[b], Rafael Z. Frantz[b]

[a]*Department of Languages and Information Systems, ETSI Informática, University of Seville, Avda. Reina Mercedes, s/n, 41012. Seville, Spain.*
[b] *Department of Exact Sciences and Engineering, Unijuí University. Rua do Comércio, 3000. Ijuí 98700-000, RS, Brazil.*

Abstract

Enterprise Application Integration has played an important role in providing methodologies, techniques and tools to develop integration solutions, aiming at reusing current applications and supporting the new demands that arise from the evolution of business processes in companies. Cloud-computing is part of a new reality in which companies have at their disposal a high-capacity IT infrastructure at a low-cost, in which integration solutions can be deployed and run. The charging model adopted by cloud-computing providers is based on the amount of computing resources consumed by clients. Such demand of resources can be computed either from the implemented integration solution, or from the conceptual model that describes it. It is desirable that cloud-computing providers supply detailed conceptual models describing the variability of services and restrictions between them. However, this is not the case and providers do not supply the conceptual models of their services. The conceptual model of services is the basis to develop a process and provide supporting tools for the decision-making on the deployment of integration solutions to the cloud. In this paper, we review the literature on cloud configuration modelling, and compare current proposals based on a comparison framework that we have developed.

Peer-review under responsibility of SciKA - Association for Promotion and Dissemination of Scientific Knowledge

Keywords: Enterprise Application Integration; Optimisation; Cloud-computing; Cloud Service Model; Service Variability.

* Corresponding author. Tel.: +34 95 455 27 70; fax: +34 95 455 27 70.
E-mail address: inmahernandez@us.es

Peer-review under responsibility of SciKA - Association for Promotion and Dissemination of Scientific Knowledge
doi:10.1016/j.procs.2015.08.616

1. Introduction

Usually companies need to use their software ecosystems[9] to support and improve their business processes. Ecosystems comprise a number of applications, which are usually designed without considering their possible future integration. Within the Software Engineering area, the field of study known as Enterprise Application Integration[5] seeks to provide methodologies, techniques and tools for designing and implementing integration solutions. In general, an integration solution aims at orchestrating a series of applications to keep them synchronized or providing new functionalities that can be created from the existing ones. As shown in Figure 1, an integration solution comprises a number of processes that contain the integration logic and a number of communication ports that connect processes and applications from the software ecosystem to the integration solution.

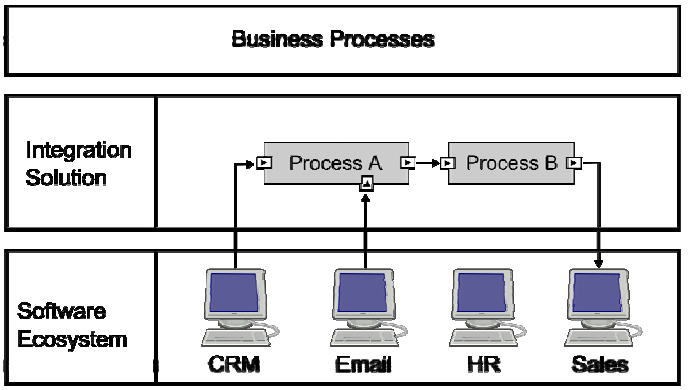

Fig. 1. Typical Integration Solution.

Cloud-computing[7] is another field of research that has drawn the attention of the scientific community. This field is transforming existing software ecosystems and revolutionizing the way companies provide computer support to their business processes. Cloud-computing allows companies to hire service packages greatly reducing their costs on IT infrastructure without having to sacrifice the quality of the IT support provided to their business processes. One of the most interesting features in cloud-computing is the charging model practiced by providers, and the availability of a high-capacity IT infrastructure at a low cost. This new model charges clients based on the amount of computing resources (memory, CPU time, data transfer, network bandwidth, etc.) they consume.

Enterprise Service Bus (ESB)[1] is a core technology to build application integration solutions. Its main elements are a set of adapters, a language, and an orchestration engine. The adapters allow software engineers to abstract the details of the different technologies for communicating the integration solution with the applications being integrated. The orchestration language allows creating models that describe the integration solutions at a high level of abstraction. The orchestration engine, also known as integration engine, provides all the necessary support to implement integration solutions.

The current trend to migrate applications from companies' software ecosystem to the cloud must be supported by technologies to deploy and run integration solutions in the cloud as well. Therefore, cloud-computing providers should provide service orchestration. The Orchestration-as-a-service (OaaS) is closely related to the business model that seeks to provide Platform-as-a-service (PaaS), in which cloud-computing providers provide their customers a computing environment in which their applications can be deployed and run, including its orchestrations. According to Wlodarczyk et al.[11], the key to progress in Enterprise Application Integration is to provide ESBs that run within the cloud. This motivates the creation of more efficient orchestration engines for the cloud, since the more efficient the orchestration engine, the less computational resources will be consumed and consequently the less companies will spend on the adoption of cloud-computing. As a complement, the more efficient the orchestration engine, the more the clients can be served by a cloud-computing provider.

Currently there are several companies providing services for cloud-computing. Each provider offers different plans that can be selected in accordance with the computational needs of each client. The cost of computing resources varies not only between different providers, but also between different service plans from the same provider. Decision-making related to which provider/plan best fits customer needs result in significant time and cost saving for companies regarding the deployment and execution of integration solutions in the cloud.

The demand for computing resources of an integration solution can be computed either from the implemented integration solution running on the orchestration engine, or from the conceptual model that describes the integration solution. The first option entails the implementation of the model and the ability of the orchestration engine to measure the computing resources consumed by the integration solution. The second option does not require the implementation of the integration solution, since it uses discrete-event simulation techniques to estimate the amount of computational resources demanded by the solution. An estimation of the computational demand of integration solutions is essential for the decision-making process.

The choice of a cloud-computing service provider depends on the services it provides, the combination of these services, and their costs. Therefore, it is desirable that providers supply conceptual models that describe in detail the variability of services and constraints between them so that these models can be taken as input to the decision-making process.

The review of the technical and scientific literature shows that there are no methodologies, techniques or tools to estimate the demand of computational resources consumed by integration solutions from their conceptual models. In addition, cloud-computing providers do not have or supply conceptual models of their services. This is the motivation to devise a supporting tool and process for decision-making on the deployment of integration solutions to the cloud, as shown in Figure 2.

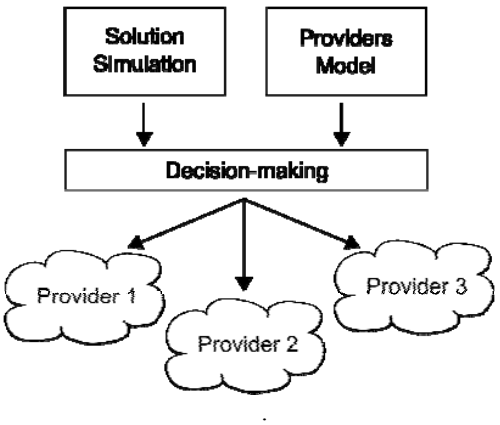

Fig. 2. Integration Solution Deployment.

The rest of this paper is organized as follows: Section 2 discusses the related work in cloud configuration modelling and proposes a framework to analyse these former proposals. Section 3 presents our main conclusions.

2. Related work

The latest trends in cloud configuration modelling are focused on feature models, although other approaches have been proposed as well, such as UML or mathematical models. In the following subsections we provide an overview of the latest proposals in this field, and we define a framework that allows comparing these proposals side by side.

2.1. Modelling proposals

Menzel et al.[8] dealt with the problem of finding a proper cloud infrastructure from a mathematical point of view, defining it as an optimisation problem. They proposed CloudGenius, a framework that provides decision support for application migration to the cloud. It defines a number of parameters, both numerical and non-numerical, that characterise a given cloud infrastructure.

The model on which CloudGenius is based includes a number of configuration elements, i.e., network latency, performance parameters such as CPU, RAM and storage size, popularity, operating system, virtual machine format and pricing parameters, such as hourly price, and license, amongst others. The main limitation of this proposal is its inability to deal with qualitative criteria.

Khajeh-Hosseini et al.[6] proposed the Cloud Adoption Toolkit, a collection of tools that provide decision support for the migration of applications to a cloud environment. They consider a number of factors that may contribute to the impact caused by the migration of an application to the cloud, i.e., cost, energy consumption, stakeholder impact, social and political factors amongst others. However, their proposal is focused only on the cost model, which is based on an UML profile for UML deployment diagrams.

The authors work under the assumption that, in most cases, the application deployment is performed on virtual machines (VM). Their cost model includes a number of configuration elements, i.e., operating system, server specifications (e.g., CPU clock rate, RAM), storage size, applications and data already deployed on the VM, and hosted databases, amongst others. They consider the elasticity, i.e., the ability to define requirements that may vary in time, as the key benefit of cloud environments. Therefore, the model allows defining a baseline usage for each cloud resource, and usage patterns that express variations on this baseline using natural language.

Frey et al.[2] defined another UML model to define cloud configuration deployment options, and they apply this model to the problem of finding near-optimal cloud deployment architectures and runtime reconfiguration rules. They propose a genetic algorithm that analyses the configuration space of a given cloud provider and finds the near-optimal configuration, i.e., one that minimizes costs, average response times and service level agreements (SLA) violations. They leverage their previous proposal of a cloud deployment simulator (CloudSIM) to estimate the costs, response times and SLA violations for each configuration.

The authors work under the same former assumption that, in most cases, the application deployment is performed on virtual machines (VM). Their model includes configuration elements such as the instance type, the number of virtual machines, the conditions under which scaling actions are mandatory to achieve a better use of the environment resources (e.g., when a certain CPU utilization threshold has been reached), and the proper scaling actions to be performed (i.e., to add or remove resources).

Quinton et al.[10] apply feature models (FM) to devise a solution to the problem of supporting the migration of software applications to a cloud-computing environment, which includes the selection and configuration of cloud providers and the deployment of the application on the selected configuration.

Their proposal is based on a cloud knowledge model ontology devised by an expert for each cloud provider, which comprises three types of elements: basic concepts (such as the language the application to deploy has been developed with), countable elements, which are concepts with a certain number of instances (such as the number of application servers), and finally quantifiable elements, which are elements that are defined not only with a number of instances, but also with a unit (such as the RAM, which is expressed in bytes). They also allow defining a number of constraints over the former elements. Finally, the concepts in the cloud knowledge model are related to the features in each FM by means of a mapping. Therefore, the feature models include cloud configuration elements such as language, application server, RAM, CPU, and whether a load balancer is needed, amongst others. In order to extract the information to create the models for each cloud provider, the authors suggest the use of reverse engineering on the web configurator of each cloud provider as a solution.

They implemented their proposal and conducted a number of tests to evaluate the practicality and scalability of their implementation. They concluded that their proposal was well suited to handle large configuration spaces, with a number of features and constraints that would make it burdensome for a human user to perform the selection by hand, and that it was able to find the best configuration in a negligible time.

The authors observed that the main challenge when using FM to find the best cloud configuration is the fast evolution of the configuration spaces, which makes it necessary to automatically update the models after each change.

Garcia-Galan et al.[3] have proposed the most recent application of FM to solve the problem of selecting the most suitable configuration amongst the configuration space offered by a given provider.

Their focus is on IaaS, and they propose a model that is based on FM, and apply Automated Analysis of Feature Models (AAFM) as a reasoning technique over the models. Their model can be graphically represented using a tree-like notation, in which features are organized hierarchically. The information to create the model is automatically extracted from the provider web site using an ad-hoc web scraper. Therefore, this proposal is only applicable to a provider (Amazon EC2), although the authors plan to include different providers in the future, hence enabling cross-provider comparison in the configuration selection. Their model includes cloud configuration elements such as instance type (which determines the RAM and number of cores), operating system, storage capability, geographic location, and billing information, amongst others. Their model includes as well some attributes that allow to define more precisely each configuration, such as customer usage data. Finally, their proposal allows defining constraints on the features and attributes of the model, using plain-text rules.

They implemented their proposal and compared their implementation against two well-known commercial tools (Amazon TCO and CloudScreener), and they concluded that their proposal provides a wider range of configuration options and that it chooses the most suitable configuration that fulfils every user requirement, while achieving reasonable execution times. However, for the time being it is only able to deal with the Amazon EC2 configuration space.

Gherardi et al.[4] explored the use of Extended Feature Models (EFM) to provide design and configuration support in the migration of robotics applications to a cloud environment. EFM is an evolution of FM which aims at enriching the features in a model by means of a number of attributes, enhancing its semantic content.

The EFM proposed by the authors includes configuration elements such as the number of containers (cloud environment instances), number of connections, database, and the ability of several robots to share a number of common deployed functionalities. The resulting models are highly dependent on the technology used to implement the robots, i.e., the programming language, the communication protocols, and the use of multi-threading, amongst others. Their models can be automatically transformed into XML and JSON configuration files that provide support for the application migration to the cloud.

2.2. Comparison Framework

In this subsection, we propose a framework to analyse the former proposals. We have studied and compared a number of features, namely:

- Cloud service: either Infrastructure-as-a-service (IaaS), Platform-as-a-service (PaaS) or Software-as-a-service (SaaS).
- Application types: in some cases the proposals focus on a specific type of applications, whereas in other cases it is not specified.
- Cross-provider: represents whether a proposal is able to deal with different providers at the same time or not.
- Model type: specifies the type of model used.
- Model elements: enumerates the configuration elements included in each model.
- Provider info source: indicates where the information to create each model was extracted from.
- Other: Other useful information about each model.

Table 1: Comparison framework

PROPOSAL	CLOUD SERVICE	APPLICATION TYPES	CROSS - PROVIDER	MODEL TYPE	MODEL ELEMENTS	PROVIDER INFO SOURCE	OTHER
Khajeh-Hosseini et a.	IaaS	Not specified	Yes	UML Profile + natural language	OS, server type, server specifications, storage, applications/data deployed, hosted database, billing	-	Elasticity requirements SLA's
Mentzel et al.	IaaS	Single-tier Web application	No (prototype)	Mathematical	CPU, RAM, Disk, Network latency, OS, implementation language, popularity, virtualization format, billing	-	
Frey et al.	IaaS	Enterprise	Yes	UML	Node configurations: Instance type, num of VMs, scalling conditions (CPU Utilisation threshold, timePeriod, scope), scalling actions (shrink/grow)	Manual	Takes into account SLA's
Quinton et al.	PaaS, IaaS	Not specified	Yes	FM+Ontologies	Language, Application Server, RAM, CPU, Load balancer, Other (Provider-Specific)	Manual	Constraints, User requirements
Gherardi et al.	SaaS	Robotics	No	EFM+XML/JSON	Containers, Connections, Database, Sharing mode	-	
Garcia-Galan et al.	IaaS	Not specified	No	FM	Instance type, SO, location, RAM, storage, not of core, dedication, usage, cost, billing	Automated - Provider web page	Model size: 81 features, 17 attributes, >20000 constraints

Table 1 reports on the results of our analysis regarding the former features. A dash symbol (‘-‘) in a cell indicates that the proposal does not provide information about the corresponding feature. The results have demonstrated that Infrastructure-as-a-Service is the most analysed type of cloud service in the context of modelling service variability in cloud providers, except for two proposals that analyse service variability in Platform-as-a-Service and Software-as-a-Service.

Regarding the type of model used, the latest trends in configuration space modelling are focused on feature models and extended feature models, which can be graphically displayed using a tree-like notation.

Regarding the information to create the models, it usually has to be manually collected, since different providers supply the information about their plans heterogeneously; sometimes, a given provider supplies its own information scattered amongst different web pages or tables (e.g., Amazon EC2). This makes it difficult to devise an automated procedure to retrieve this information and use it to compose the models. Therefore, when the provider information changes, e.g., when a provider adds or removes a plan, or modifies the features of a given plan, these changes have to be noticed, in the first place. Then, the models have to be updated, in the best case, or created from scratch, in the worst case, which means that the information has to be manually recollected again. Moreover, the need for manually collecting the provider information is probably the reason behind the inability of some of the proposals to provide a cross-provider analysis. Cross-provider comparison does not seem to be a trend, since only half of the proposals support it.

Finally, we note that none of the proposals deals specifically with integration solutions. Three out of the six proposals focus on a specific type of application (Web Application, Enterprise, and Robotics), whereas the rest of the proposals do not provide information regarding this feature.

3. Conclusions

Cloud-computing is part of a new reality in which both small and large companies have at their disposal a high-capacity IT infrastructure at a low cost, in which enterprise application integration solutions can be deployed and run. Because there is no public detailed conceptual model describing the services and their dependencies, it is not possible to evaluate which cloud-computing provider fits best to an application integration solution. In this paper,

we have surveyed proposals that aim at modelling the variability of services available in cloud-computing providers. We have found that the majority of the analysed proposals focus on a single cloud service type (Infrastructure-as-a-Service) and cross-provider does not seem to be a trend in this research field. Cross-provider comparison would be a very interesting feature, since it would allow clients to compare different cloud-computing providers at the same time and choose the most appropriate provider/plan for a given application integration solution. Usually, the information to create the models has to be manually collected from the provider web site or any other documentation made publicly available by the cloud-computing provider. Furthermore, different providers supply this information heterogeneously. Finally, we have also noticed that the latest trends in configuration space modelling are focused on feature models and extended feature models.

References

1. Chappell, D. (2004). Enterprise Service Bus: Theory in Practice. O'Reilly Media.
2. Frey, S., Fittkau, F., & Hasselbring, W. (2013). Search-Based Genetic Optimization for Deployment and Reconfiguration of Software in the Cloud. ICSE (pp. 512-521). San Francisco: IEEE.
3. García-Galán, J., Trinidad, P., F. Rana, O., & Ruiz-Cortés, A. (2015). Automated Configuration Support for Infrastructure Migration to the Cloud. Future Generation Computer Systems , P.P.
4. Gherardi, L., Hunziker, D., & Mohanarajah, G. (2014). A Software Product Line Approach for Configuring Cloud Robotics Applications. IEEE International Conference on Cloud Computing (pp. 745-752). Anchorage, US: IEEE.
5. Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.
6. Khajeh-Hosseini, A., Greenwood, D., W. Smith, J., & Sommerville, I. (2011). The Cloud Adoption Toolkit: supporting cloud adoption decisions in the enterprise. SOFTWARE – PRACTICE AND EXPERIENCE , 447–465.
7. Mell, P., & Grance, T. (2011). Draft NIST working definition of cloud-computing. Retrieved from http://csrc.nist.gov/ groups/SNS/cloud-computing
8. Menzel, M., & Ranjan, R. (2012). CloudGenius: Decision Support for Web Server Cloud Migration. WWW (pp. 979-988). Lyon, France: ACM.
9. Messerschmitt, D., & Szyperski, C. A. (2003). Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press.
10. Quinton, C., Romero, D., & Duchien, L. (2014). Automated Selection and Configuration of Cloud Environments Using Software Product Lines Principles. IEEE CLOUD. Anchorage, USA: IEEE.
11. Wlodarczyk, T. W., Rong, C., & Thorsen, K. A. (2009). Industrial cloud: Toward inter-enterprise integration. CloudCom, (pp. 460–471).