Software Engineering for the Cloud: a Research Roadmap

Elias Adriano Nogueira da Silva
Federal University of São Carlos, São Paulo, Brazil
Email: elias.silva@dc.ufscar.br

Daniel Lucrédio
Federal University of São Carlos, São Paulo, Brazil
Contact: http://www.dc.ufscar.br/~daniel

Abstract—Having arisen from the industry, cloud computing is receiving increasing attention from the academy. We are witnessing an increasing number of papers dedicated to experience reports and proposal of methods, processes and tools focused on cloud-related technology. Researchers are also attempting to identify and formalize the main concepts behind cloud computing, forming a basis for future investigations about how this approach can be used to promote advances in many fields of computer science. However, while the main research challenges and opportunities related to cloud computing in general are already well-known, in the software engineering area the scenario is less clear. In this paper, we present the results of a systematic literature review where we tried to understand how researchers from the software engineering field are viewing the cloud computing paradigm. We identified ten software engineering research opportunities focused specifically on cloud computing, which are presented and discussed in terms of related work. We also present a discussion on some practical issues related to the development of software for the cloud, trying to make some obscure points clearer and aiming to facilitate the work of software engineering researchers and practitioners.

I. INTRODUCTION

The concept of cloud computing has gained recent attention mainly because of the industry’s initiative. Companies attempted to optimize and even profit from their computing infrastructure, and thus a plethora of related concepts and proprietary technology started to appear, aiming mostly at the potential market. It was only recently that the academic community started to look at these concepts more carefully. Although it is possible to track down earlier related work describing virtualization, grid computing and clusters [1], [2], [3], which are concepts related to clouds, the first academic papers dealing explicitly with cloud computing started to appear around 2008-2009 [4], [5], [2]. Most of these were attempts to propose a proper definition for cloud computing and its underlying concepts and technologies [4], [2], [6]. However, up to this date, no definition is widely accepted by the community [2], [7], [8].

One study that stands out is a 2009 technical report from the University of California at Berkeley [9], a prominent academic group in the field. The report discusses the main concepts behind cloud computing, and presents some upcoming challenges and research opportunities for the academic community. Soon this report and its authors became widely cited, laying down the ground for different research directions in the cloud computing field.

However, the approach followed by the Berkeley report was more general. It pointed out some major challenges that relate to cloud computing from a higher perspective, leaving some unanswered questions in different research areas. One of these areas is software engineering, which is the subject of this paper. In this scenario, some of the first questions that need to be answered by software engineering researchers and practitioners are: what are the main challenges and research opportunities in developing software for the cloud, in terms of methodologies, languages, frameworks, tools and other software engineering concerns? What has already been done in these few years of cloud research, from the software engineering perspective?

To try to answer this question, we performed a systematic review in the available literature, including academic and industrial publications. Our objective was to elicit some gaps, challenges and opportunities for software engineering researchers, as well as to provide guidance to practitioners in terms of describing what is involved in the adoption of current cloud technologies.

Our research method and findings are described in the remainder of this paper. First, we discuss a brief background (Section II) on cloud computing and related concepts. Next (Section III), we describe the systematic review process, including the search protocol, the adopted criteria and some interesting data related to the study. In Section IV we present the results of the review and our main findings. We identified ten different research opportunities, which we believe need to be further investigated by the software engineering community. These may serve as a basis for future work in the field. In Section V we present a secondary result of this study, which includes some observations related to software engineering for the cloud, but from a more practical perspective. We try to clarify some points that we found to be a little confusing in the literature, in order to help software engineering researchers and practitioners in the development of their work. Section VI discusses related work. Finally, in Section VII we present some concluding remarks.

II. BACKGROUND

The concept of cloud did not arise as a new technology model, but as the integration of technologies from the past [8], which resulted in a new way to use and provide computing power as a service through the Internet.
The main ideas behind this concept are: to allow companies to acquire computing resources by demand; to enable payment according to the utilization volume; and to make it possible for a company to completely ignore the knowledge about where these resources come from [9].

There is still no widely accepted academic definition for cloud computing [7], [8]. Vaquero et al. [2] made a thorough study of the various definitions found in the literature to achieve a more complete definition. Although it is still a proposal, their definition is consistent and complete, and therefore is the one used in this work: “Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs” [2].

This definition contains the main elements of cloud computing, but it is too general regarding the resources. Apparently, any kind of resources could be virtualized. To make this definition more concrete, we could look at the many taxonomies that attempt to characterize which kinds of resources can be made available in a cloud. Many were conceived under a business perspective [6], and include terms such as HaaS (Hardware as a Service), PaaS (Platform as a Service), DaaS ([Development, Database, Desktop] as a Service), IaaS (Infrastructure as a Service), BaaS (Business as a Services), TaaS (Testing as a Service [10], [11]), FaaS (Framework as a Service), and even the generic XaaS (Everything as a Service). The definitions behind those terms are very wide and sometimes misunderstood [12], but there are three categories that seem to be more well established:

- **Software as a Service - SaaS**: refers to applications that are delivered to customers in the form of services over the Internet [9], [13]. A SaaS normally operates over a cloud infrastructure, but this is not necessary. Examples of SaaS include: Google Drive\(^1\), Flickr\(^2\), Picasa\(^3\), SkyDrive\(^4\), among others.

- **Platform as a Service - PaaS**: the idea is to provide a full software development platform as a service, including an execution environment, frameworks and different useful integrable services [6], [13]. Normally, the service also includes hosting, testing and a distribution channel or market. Some examples are: Google App Engine\(^5\) and Microsoft Azure\(^6\), among others.

- **Infrastructure as a Service - IaaS**: consists in delivering computational infrastructure as a service [6], [13]. The customer acquires - and pays by demand - services for processing, storage and other fundamental computing resources, making it possible to deploy and run any software, including operating systems and applications. It is normally a virtual computer that runs on a real infrastructure maintained by a provider. The customer has full control over the (virtualized) operating system and the running applications. The most interesting part of this approach is the possibility to rapidly increase the computational power, which can sometimes be done automatically by the provider. Some examples of IaaS include: Amazon\(^7\), GoGrid\(^8\), IBM SmartCloud\(^9\), among others.

Another classification that can be made considers aspects such as ownership and sharing of the cloud infrastructure. A cloud environment that is shared among different customers is called a “public” cloud [9], [13]. Such environment is normally owned, hosted and maintained by an organization selling cloud services, and is provided to the public in general under a contract where many users share the same infrastructure and pay for what they use [13], [6]. Amazon, Google App Engine and Azure are examples of public clouds.

Most public cloud providers use many security mechanisms, but if for some reason an organization does not want to deploy its data or business logic into a public cloud, it may build or rent its own cloud infrastructure. This is called a “private” cloud, i.e. operated solely for an organization. It may be hosted and managed by the organization itself or by a third party, and may be deployed on premise or off premise [13], [6]. This allows organizations to obtain the benefits of cloud computing, such as server virtualization\(^10\) and elasticity, without having to share the same physical infrastructure with other organizations. However, differently from public clouds, an upfront investment on infrastructure needs to be made.

A compromise between these models, where parts of an application run in a public cloud and other parts run in a private cloud, is also possible. This is known as a “hybrid” cloud [6], [13].

### III. The Systematic Review

In this study, we followed a formal systematic literature review process [15], [16], [17]. A systematic review proposes a fair assessment of the research topic as it uses a rigorous and reliable review methodology, together with auditing tasks to reduce the researcher bias [18]. To enable the documentation of the systematic review and to facilitate the process, the START\(^11\) tool was used [19]. As part of the process, we developed a protocol that provided a plan for the review in terms of the method to be followed. Table I shows a synthesis of the systematic review protocol.

A total of 122 articles were analyzed, of which 100 were selected and their data extracted and analyzed according

---

5. Google App Engine: [http://code.google.com/appengine](http://code.google.com/appengine)
10. Server Virtualization: is the technique of execution of one or more virtual servers over one physical server [14]
11. [http://lapes.dc.ufscar.br/ferramentas/start-tool](http://lapes.dc.ufscar.br/ferramentas/start-tool)
to the recommendations of the review protocol. The entire systematic review was documented, from its conception to the extraction of results, to allow its repetition and auditing by other researchers interested in the study.

The study was conducted in a period of eight months, by a single researcher (one of the authors of this paper). Whenever there was doubt regarding the inclusion/exclusion criteria, analysis or summarization, other researchers from the group were consulted.

As mentioned before, the literature related to software engineering for the cloud is recent. Figure 1 shows the distribution of the selected papers by publication year. As it can be seen, all papers included in this study were published after 2008.

Another point of interest in the study is the low distribution of studies per author. The collection of the selected 100 papers included more than 300 different authors, and no author published more than two papers, what indicates that there is not a clear formation of specialized research groups on software engineering for cloud computing. One exception (although not from the software engineering area) is the aforementioned technical report from University of California at Berkeley [9], which was invited to be published as a paper [12], being frequently cited in the cloud literature\footnote{By the time this paper was written, there were 302 citations, according to http://www.scopus.com/}.

\section*{IV. SOFTWARE ENGINEERING FOR THE CLOUD: A RESEARCH ROADMAP}

In this section we discuss the main findings of the review, i.e. what are the main challenges and research opportunities related to software engineering and cloud computing.

\subsection*{A. Data Lock-In}

Data lock-in is the difficulty developers face when having to port their applications from one cloud platform to another \cite{9}. Such problem can be seen mostly in the PaaS scenario: in order to take advantage of a very flexible cloud architecture, the applications have to be specifically developed for the chosen platform. For example, in order to offer great elasticity, Google App Engine – a PaaS provider – imposes a specific programming style and has its own way to manage data, and thus an application developed for it may not be easily ported to

\begin{table}
\centering
\caption{SYSTEMATIC REVIEW PROTOCOL}
\begin{tabular}{|l|l|}
\hline
Objectives: & To identify the main challenges and research opportunities in developing software for the cloud, and how to adopt the main cloud services, from the software engineering perspective; \\
\hline
Main Question: & What are the main challenges and research opportunities in developing software for the cloud, in terms of methodologies, languages, frameworks, tools and other software engineering concerns and what has already been done in these few years of cloud research, from the software engineering perspective? \\
\hline
Intervention: & Cases study, surveys, definitions and other types of research related to cloud software development; \\
\hline
Control: & We found no study to use as control; \\
\hline
Population: & Research related to cloud software development; \\
\hline
Results: & Case studies, surveys, definitions and research work related to cloud software development; \\
\hline
Application: & Cloud computing software development research; \\
\hline
Keywords: & Cloud computing; software development; software engineering; research gaps; \\
\hline
Source selection criteria: & Availability of the papers over the web; search engines using keywords; \\
\hline
Studies Languages: & English and Portuguese; \\
\hline
Source Search & The string was first defined in Scopus, then adapted to other engines like ACM and IEEE; \\
\hline
Methods: & Automatic search; Search through bibliographic references of selected papers; Search by groups and authors; Search over the related conferences and journals; \\
\hline
Source Engines: & Scopus; IEEE; ACM; Google Academic Search; Springer; \\
\hline
Study inclusion criteria: & The full paper was available; The study involved definitions about cloud computing; The study contained experience reports about cloud computing software development; \\
\hline
Study exclusion criteria: & The full paper was not available; The study was not directly related to cloud computing; The study approached cloud computing without considering software development; \\
\hline
Studies types definition: & Papers and technical reports; \\
\hline
Initial studies selection: & Based on abstract, keywords and title; \\
\hline
Studies quality evaluation: & The quality of studies was evaluated in an ad hoc way, after the reading; \\
\hline
Information extraction fields: & Study types; Search domain; main idea; short summary; \\
\hline
Results summarization: & All the accepted papers were analyzed, and the data were summarized into a single textual report, where the results were described; \\
\hline
\end{tabular}
\end{table}

Fig. 1. Selected Publications by Year
a different PaaS provider, nor can its data. Even if a developer wanted to host an application in his/her own private cloud later, great effort would be necessary to rebuild the code, redeploy it and migrate all the data.

It is not difficult to see why this problem is hindering the adoption of the cloud model [20]. The possibility of becoming “locked in” a particular platform, not being able to choose a different one later (customer lock-in), puts the developers in a difficult position. They mostly fear being charged abusive fees later, or having their applications unavailable due to lack of service quality [9]. There are some initiatives towards solving this problem. They normally revolve around two approaches:

- **API standardization**: if every cloud provider employs some well-known standards in its platform, in theory an application could be easily ported from one provider to another without too much effort. In this sense, the OpenCloud Manifesto [21] is gathering efforts to group companies around the specification of an open standard for cloud computing. However, until the time this paper was written, important companies such as Google and Microsoft had not joined the manifesto. There are other initiatives, such as: (i) DMTF (Distributed Management Task Force), which employs efforts focused on standardizing interactions between cloud environments by developing specifications that deliver architectural semantics and implementation details to achieve interoperable cloud management between service providers and their consumers [22]; (ii) OCCI-WG (Open Cloud Computing Interface Working Group), which is a protocol and API for all kinds of management tasks, and was created to develop a remote management API for IaaS-based services, allowing the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. The current release of the Open Cloud Computing Interface is suitable to serve many other models in addition to IaaS, including e.g. PaaS and SaaS [23]; (iii) IEEE P2301, responsible for developing a standard that will enable portability. To achieve this goal, it will group the different interfaces (templates, formats and operation conventions) of the cloud elements into logical profiles; and (iv) IEEE P2302, which will define the topology, protocols, functionality and governance required to support cloud-to-cloud interoperability and federated (intercloud) operations [24], [25].

- **Model-driven approaches**: the idea is to employ model-driven engineering (MDE) [26] in the development of cloud applications. MDE seems to be a viable solution to the lock-in problem, because through higher level models and automatic transformations, the developers are able to focus on a more conceptual, platform-independent level when developing applications. However, there is no consensus about the set of models, languages, transformations and software processes that could be used to successfully develop cloud applications using MDE [27]. Current efforts revolve mostly around the identification of abstractions that would allow one to specify and create systems independently of the underlying platform, and the automatic code generation for some specific platform or service infrastructure [28], [29].

**B. Decision-making about migration to the cloud model**

In order to take advantage of the benefits of the cloud model, many organizations have the desire and/or intention to embrace the cloud for existent and new applications. This decision is inherently complex, and suffers the influence from multiple factors such as cost, availability, performance, security and QoS, each of which have a major influence on every organization [30].

Before making the decision, an organization must take into consideration the viability of the cloud model in its current context. It must carefully analyze the constraints related to cloud platforms, both technical [31] and organizational, and consider its business requirements [32].

The literature contains research providing support for the decision about migration to the cloud [33], [30], [32], [34], [35]. Ezzat et al. [36] studied the cloud from different perspectives, in order to determine the aspects that mostly affect the decision about cloud adoption. However, in this review we have not found a formal process to specifically support this task.

If the decision about the migration is positive, the IaaS model is the ideal choice to deploy pre-existent systems, while PaaS is more adequate for new applications [37]. An incremental strategy can be adopted, to effectively migrate data and applications to the cloud. For pre-existing systems, the migration itself is a whole new challenge, as explains next section.

**C. Legacy software migration**

There are many systems still in use today that became outdated, either because they make use of unsupported technology or earlier versions of an operating system or platform. However, the costs for their modernization and the fact that they may still provide essential services keep them active. These are commonly known as legacy systems. As expected, most of these are not ready to be migrated into a cloud environment, and this is true even for some more recent pre-cloud web applications. One way of solving this problem is a complete reengineering (i.e. a reconstruction of the software for a new platform), but this process is normally too expensive. An alternative would be an strategy to migrate legacy software into a cloud infrastructure and to ensure its correct functioning in the new environment, without the need for a complete reengineering. There are many experience reports that focus on this problem [38], [39], [40], [41], [42], [43], [44], [45], [31], [46], but more research is still needed, mainly to formalize a strategy that defines a step-by-step process to

---

13The list with all members of the OpenCloud Manifesto can be found at [http://www.opencloudmanifesto.org/supporters.htm](http://www.opencloudmanifesto.org/supporters.htm)
14IEEE P2301: [http://grouper.ieee.org/groups/2301/](http://grouper.ieee.org/groups/2301/)
15IEEE P2302: [http://grouper.ieee.org/groups/2302/](http://grouper.ieee.org/groups/2302/)
perform the migration, considering all available cloud services. Such strategy would constitute a very important contribution towards the systematic migration to the cloud model.

D. A reengineering process for the cloud

New technologies, standards, languages and frameworks related to cloud computing may facilitate the development of applications, the evolution of current software and make companies more competitive. In these cases, migration alone (as discussed in previous section) is not enough, and companies find themselves in a crossroads between reconstructing their applications to make complete use of new cloud technologies and becoming out of the market.

The literature contains some successful examples of software reengineering for the cloud. For example, Zhou et al. [47] proposed a novel approach for reengineering enterprise software for cloud computing. Following a five-step ontology development process, the enterprise software ontology is created by generating an integrating code ontology, database ontology and Hibernate framework ontology. This approach can help to identify potential service candidates in the legacy system.

However, studies are still needed, to formalize a complete reengineering process focusing on cloud infrastructures, considering the particularities and the different service models offered in this paradigm.

E. Mechanisms to facilitate the adoption of the hybrid model

As discussed in Section II, the hybrid model is the alternative found by some companies to avoid large upfront investments and still maintain some critical parts of its applications under more rigid control. However, the interoperability between public and private cloud infrastructures is not easy to achieve [24]. The standardization can help to improve this model, but as discussed in section IV-A, there is still no standard for the development of interoperable applications for cloud computing.

The literature has some efforts in this direction. One example is the group IEEE P2302, which has been researching this question. Other examples can also be found in the literature [48], [49].

The existence of frameworks, tools and/or process to facilitate the implementation of the hybrid model could be the solution many companies are waiting for before definitively adopting cloud computing.

F. Implementation of Modeling as a Service - MaaS

Bruneliele, Cabot and Jouault [27] proposed the concept of Modeling as a Service - MaaS. Similar to SaaS, MaaS would allow the deployment and execution of modeling and other model-driven services, such as transformations and code generation, over the Internet. The idea is to provide a model-driven engineering (MDE) environment in the cloud, including a repository of models and model-driven engineering tools that could be used by development teams in the form of services.

MaaS has the potential to leverage the adoption of MDE among software developers, making use of the cloud to facilitate some of the tasks involved in model-driven engineering [50]. The implementation of the MaaS model can lead to different research opportunities, such as collaborative development issues [51], concurrent modification issues [52], management of large model repositories, among others. Bruneliele, Cabot and Jouault [27] highlight the following possibilities of MaaS research:

- One possibility is the creation of a tool for collaborative and distributed modeling [53]. In this context, the cloud would serve as an instrument that would allow the specification and sharing of software models [54] among distributed team members.
- Definition of modeling mash-ups as a combination of MDE services from different vendors.
- Availability of model transformation engines in the cloud to provide platform-independent model management services.
- Code generation in the cloud: the transfer of code generation and simulation services for the cloud can facilitate the deployment and evolution of software applications, increasing productivity. Developers would not need to worry about having to setup and maintain an infrastructure to develop model-based applications.
- Distributed Global model management. Complex MDE projects involve several models (possibly conforming to different metamodels), model transformations, model injectors and projectors, etc. MaaS would facilitate the manipulation of all these modeling artifacts in a distributed environment.

G. Billing and auditing mechanisms

Auditing, usage reports and billing are functionalities that are normally associated to SaaS [55]. The provider of such systems must include administrative functions to support this kind of tasks, which are not necessary in the traditional software model.

In our review we found no study focusing specifically on these problems, and thus we believe there is space for research and implementation on such types of mechanisms, to discover better ways to achieve the necessary billing and auditing functionalities, according to the reality of different companies. Case studies are also a possibility to help in the identification of the most important issues related to these tasks.

H. Cloud Service Composition

As in other fields of computer science, the dynamic nature of cloud-based technologies lead to constant changes in terms of platforms, tools and technologies. In this scenario, it is not uncommon to see different applications or sets of services having to cooperate for mutual benefit. Service composition, in the context of Service-Oriented Architecture, is still an important research subject in software engineering [56], [57].

In the cloud context, the terms “intercloud”, “multi-clouds” and “cloud-of-clouds” are used to refer to the composition
of cloud providers as an integrated (or federated) cloud environment [58]. In such environments, users can (transparency or not) leverage the resources from different (either public or private) cloud providers [25], [59], [60], [61], [62], [63]. These terms are similar and suggest that cloud computing should not be restricted to a single cloud [59], but a “cloud-of-clouds”.

In such environment, it would be possible to have application components deployed and co-existing on multiple clouds. This raises challenges for the developer, who has to worry about architectural and development issues that consider this multi-cloud scenario. In particular, the composition of services needs an effective communication between providers (interoperability). Abstractions for efficient service orchestration and choreography [64] also need to be conceived or adapted for the cloud.

There are some efforts related to cloud service composition in the literature [65], [66], [67], [68], [69]. However, the related technologies are still under development and more research is necessary. Standardization efforts, such as those described in Section IV-A, could offer a partial solution, but these are not yet established.

I. Case studies

Organizations are normally afraid to adopt new technologies until relevant success cases are presented. In this sense, to reinforce the benefits of the cloud, to show the importance of this paradigm, and its potential to revolutionize the market, it would be interesting to have a large number of case studies.

The literature already has many examples of such efforts. Chauhan & Babar [38] report their experience in the migration of service-oriented systems to the cloud computing model. Khajeh-Hosseini et al. [44] show a case study of migrating an enterprise application to IaaS. Tran et al. [40] show a taxonomy of critical factors in migration to cloud computing model. We should continue to publish such type of results, and case studies and experience reports should always be on the agenda of software engineering researchers. The more case studies available, the more confident researchers and practitioners will become regarding the benefits of the cloud, and the validity of the research.

J. Open source platforms

This last issue may not be considered as an open research question by itself, but it presents a challenge for many researchers. The most important cloud computing platforms are proprietary and, as a consequence, not open for investigation, modification and/or extension. This poses a problem for researchers that want to experiment with different platform architectures and setups. In this sense, the existence of open source alternatives could be of great help to the software engineering – and cloud computing in general – community.

There are some open source options available (like Eucalyptus16 and Nimbus17), but these are mostly focused on the IaaS model. We still need more research dedicated to produce a more comprehensive model that can be improved and used to support cloud adoption in its different possibilities.

V. OTHER FINDINGS OF THE STUDY

The choice of a cloud service depends on the type of application that will be used. Some options target developers, some target end users and some target both. During the review, we found some interesting information on studies describing the benefits of cloud computing, and success stories involving the paradigm. These are practical issues regarding the use of cloud services and, more specifically, the development of software for the cloud paradigm. In the following sections, we try to compile this information, hoping it will be of practical value to researchers and practitioners interested in developing software for this emerging field.

A. SaaS vs SOA

Reviewing the literature, we noticed that there is some confusion between the terms SaaS and Service-Oriented Architecture (SOA). Laplante et al. [70] also pointed out such confusion.

For example, Sharma et al. [71] do not distinguish between the terms SOA and SaaS, proposing an approach focused on interoperability that uses MDE to generate web services descriptors (WSDL). The authors then refer to the approach as SaaS development, but in fact it is closer to the concept of SOA.

Laplante et al. [70] made a wide study of the characteristics of these models and concluded that the fundamental difference is: SOA is a model for building software and a SaaS is a model for delivering it. While SOA is concerned with the creation of services to compose a more flexible architecture for an application, SaaS is concerned with the delivery of software functionality in the form of services, which may be provided independently of the underlying architectural style.

To better understand the difference between SaaS and SOA, we can cite Google Apps [72]. Google makes its applications (Google mail, Google calendar, Google sites, etc) available to companies that do not wish to share their information, as happens with normal users. To use such services, companies make a paid subscription and do not need to worry about the internal architecture of the applications, nor with their deployment and maintenance in a private infrastructure. They simply use the service and pay for what they use. In a comparison, we can cite the Application Server Provider (ASP) model, where the customer must acquire a license, and install the software in some particular infrastructure. Oracle’s JD Edwards ERP [73] is an example of ASP.

As an example of SOA, we can cite systems that use service composition to provide some final functionality. For example, e-shopping web sites may use services from delivery companies to calculate shipping fees and conditions and present them to the user. Other services may help to compose the final functionality. The distinguishing feature is that the internal architecture makes explicit use of services to achieve more

16http://www.eucalyptus.com/
17http://www.trynimbus.com/
flexibility and allow more specialized modules to be built [74], [57].

As we can see, SaaS and SOA are distinct concepts. But they are frequently used together, because these two models complement each other. SaaS helps to provide the building blocks for SOA, and SOA helps to implement SaaS more rapidly [70], and this may be the source of the confusion.

B. Developing SaaS

From the software engineering point of view, the implementation of the SaaS model requires some specific concerns. These arise from the service-oriented business model that is normally adopted. As a first concern, the system providers must implement some way to register the use of the services, in order to allow billing, auditing, usage and accounting reports [55], in order to support the Cloud Computing business model [75]. Additional administrative functions are required to support these new kinds of tasks, which are not necessary in the traditional stand-alone software model. More importantly, these extra tasks must not cause problems to the performance or correctness of the software [76].

Another concern is the existence of multiple customers sharing the same infrastructure, each one with his/her own needs in terms of information, functionality and interface. This is known as multi-tenant architecture [77], [78], and is a concern that many SaaS developers have to worry about. Some authors [79], [80], [81] have even proposed a SaaS maturity model to categorize SaaS according to its support for multiple tenants.

SaaS is normally accessed over the Internet [9]. In this sense, developers also have to worry about availability, performance, offline access to the software, and other concerns that arise from this online nature of SaaS. These concerns add extra complexity in relation to traditional stand-alone software.

C. Developing Software for the IaaS model

The IaaS model provides computational resources that lie on the provider’s infrastructure. The target audience of this model are normally infrastructure architects. These architects do not need to worry about issues such as dynamic demands, elasticity and scalability, which are handled by the provider [6].

Regarding the developer, the IaaS model adds little to his/her list of concerns. In this model, the development team can create applications just as if it were developing for their own dedicated private infrastructure. Differently from the PaaS model (presented next) here the software usually does not have any particular feature that demands special knowledge about the provider’s infrastructure. In other words, any application can be deployed in an infrastructure that is provided as a service, as most IaaS providers have many options of virtual machines, using the most common application servers and other deployment technologies. This task also happens much like the traditional way of deployment, because normally the developers have unlimited access to the virtual machines, just as if they were real, private machines.

However, some specific concerns arise, such as load balancing, automatic scalability, data storage services and other functionality that may be used by applications and that must be developed specifically for a particular provider. Open standards and open source implementations, such as JClouds (an OCCI-WG implementation) [82] can help to reduce some problems, by providing a common layer that can communicate with different providers.

Also, the developer must have in mind that, if a public IaaS provider is being used, the software has to be delivered to the end user through the Internet. The SaaS model is normally used, although it is possible to use this infrastructure to host legacy systems [37], to deliver a system under the ASP model, or just to perform some kind of data processing [83].

D. Developing Software for the PaaS model

The PaaS model tries to leverage the flexibility of the cloud model, by providing a complete platform for software development. A cloud platform hides many of the complexities of developing cloud software, and brings scalability and elasticity to a different level. In the PaaS model, the development platform is provided as a service. Applications that are developed for this particular platform can benefit from a specific programming model, that can be fully managed by the platform provider in a fine grained way [84].

The target audience of PaaS are system developers. When developing for PaaS, a developer must adopt a platform and its particular set of tools, libraries, technologies and standards [37]. These may require some initial training, which may lead to an initial loss of productivity. However, they can later help to reduce development time by providing useful functionality, facilitating the development and distribution of cloud applications [84], [85]. For example, Google App Engine (an example of PaaS provider) has many functionalities for image processing, image manipulation, user management and authentication, data storage, location-based services, among others [86]. As in the IaaS model, applications are hosted in the provider’s infrastructure, and are made available through the Internet, following the SaaS or another model.

The problem with this approach is the lock-in, already discussed in Section IV-A, caused by the fixed set of technologies that must be adopted. But other than that, the benefits make PaaS a very interesting approach for cloud application development.

VI. RELATED WORK

The most influential related work in the area is the aforementioned technical report from Berkeley [9]. In this report, Armbrust et al. present the results of a six-month brainstorming effort about the subject. The authors attempt to clarify related terms and provide simple formulas to quantify comparisons between the cloud model and the traditional model. They also identify ten barriers for the success of cloud computing, and present ten opportunities to overcome them, much like we attempted to do in this paper.
Other perspectives of the cloud computing model can be found in the literature: Narasimhan and Nichols [87] present the point of view of cloud platform adopters; Wang et al. [7] present a perspective of the technologies that lie behind the cloud model. Many others explore related concepts [1], [88], [89], [90], [91], [4], [92], [6], [40], [93].

Differently from other works, we attempted to present a software engineering perspective on cloud computing. Like Armbrust et al., we identified some challenges and research opportunities, but focusing specifically on software engineering.

We also attempted to clarify some concepts and present some practical issues related to the development of software for the cloud. A better understanding of cloud computing will allow the community to design more efficient mechanisms which, as a consequence, will facilitate the adoption of this model [4].

As discussed earlier, the cloud concept is more connected to the market. Only recently it has been investigated by the scientific community. There are relatively few publications, although the number of papers related to this subject is increasing. Many of these attempt to propose definitions for cloud computing [4], [2], [6]. However, there is still no widely accepted definition. In this paper, we used the one we considered more complete [2].

VII. CONCLUDING REMARKS

The way cloud services are commercialized can cause deep changes in the software industry, as long as software engineers are able to unleash its full potential. This emerging computing model still needs to be studied, and efforts are needed to solve the current problems and overcome the challenges before the cloud model becomes more well established.

Although cloud-related terms have originated from the industry, the academy has many opportunities to contribute. In this paper we attempted to highlight possible directions in which the software engineering area can approach the cloud computing scenario. We wanted to share our observations because, after a considerable review work, we believe to have reached a point where we should compile the information. We tried to point out other studies that do a similar job, and to compile some interesting points that were not available in a single study yet, at least not for the software engineering area.

In no way we believe the results of our observations are the only and definitive list. The community is welcome to continue place.

ACKNOWLEDGMENTS

The authors would like to thank CAPES and FAPESP (process number 2012/04549-4) for funding this research. We would also like to thank the reviewers for their valuable and insightful suggestions, in particular for pointing out the challenge we describe in Section IV-B.

REFERENCES


