Specific Design Considerations for Cloud Accounting Mobile Apps

Laura Elena Tudoran

Academia de Studii Economice, Bucuresti

DOI: 10.29322/IJSRP.8.7.2018.p7917
http://dx.doi.org/10.29322/IJSRP.8.7.2018.p7917

Abstract- In the era of rapid application development which emphasizes heavily the rapid prototyping and iterative delivery, it is imperative that software developers take the time to properly plan the software development phases. With the emergence of cloud computing and mobile cloud computing software developers are tempted to jump in and write code without planning. This practice can lead to resource waste, thus making the use of software development methodologies an imperative. This paper proposes five UML diagram models for a cloud accounting app.

Index Terms- cloud computing, SOA, QoS, IaaS, PaaS, SaaS, MDE, DSML

I. INTRODUCTION

Considered by authors such as Huang et al. (2013), a consequence of virtualization development and software oriented architecture (SOA), cloud computing revolutionized the IT industry by facilitating access to infrastructure and software based on a subscription (Garg et al., 2012). As a result, more and more IT companies have begun to offer cloud services to their customers. The difficulty consumers face when choosing a cloud computing service provider is amplified, on the one hand, by the multitude of existing providers on the market, and on the other by the lack of a framework that allows them to evaluate different vendors based on type of service offered and its quality (Quality of Service - QoS).

As early as 2011, cloud computing technology has confirmed its promising platform status for providing the following services: service-based infrastructure (IaaS), PaaS and SaaS, cloud serving centers state-of-the-art data with a virtual service network architecture (hardware, database, interface, applications) so developers have the ability to deploy and access applications globally at competitive prices correlated with user requirements specified in QoS (Ferry et al., 2013).

II. SOFTWARE DEVELOPMENT METHODOLOGIES COMPATIBLE WITH CLOUD COMPUTING TECHNOLOGY

Service-Oriented Software Architecture (SOA) is a design and implementation model based on discrete software features that provide application functionality as services to other applications. SOA is independent of any vendor, product, or technology.

SOA and cloud computing share some similar features, both focusing on the concept of service. The basic element of SOA is a service-oriented software, cloud computing being an extension of the IT infrastructure for the SOA concept. Because it has all the computing resources (including hardware and software), cloud computing is a much more efficient and economical choice than the SOA architecture. SOA is much more strategic and abstraction, while cloud computing is more tactical and specific. Cloud features enrich and extend content based on cloud computing resource and service sharing.

On the other hand, SOA and cloud computing are complementary. Cloud computing provides remote cloud services that are available for SOA, and SOA provides the way for composing cloud services that are required for business applications. At the same time, points of interest are different. SOA focuses on designing the system that adopts service architecture, focusing on how services are being processed, reuse, agility, while cloud computing focuses on providing and using services, focusing on how services are delivered and virtualized, on their dynamic prolongation on demand, on resources and services.

Because the current context of cloud computing includes a multitude of cloud providers as well as several IaaS and PaaS solutions, Ferry et al. (2013) support the idea that cloud infrastructure is exposed to application developers through software packages located in layers of SaaS and PaaS designed to support the creation and deployment of applications.

To serve this purpose, the development of scalable architectures and application development environments for creating, accessing, managing, deploying, and maintaining cloud applications in a way that is as accessible to developers has become critical because of the ability to run and manage multi- cloud lets you exploit the specificity of each cloud solution and therefore optimize the performance, availability and cost of applications.

Cloud computing technology is attractive, especially for small and medium sized businesses, because it allows them to focus on the consumption or service delivery on the top of the cloud infrastructure. At a high level, cloud computing does not seem radically different from the other existing paradigms: WorldWideWeb, grid computing, service computing and cluster computing. However, the key differentiators of cloud computing are its technical features, such as on demand-shared or fast elasticity, self-service, almost infinite scalability, end-to-end virtualization support, and robust support for resource monetization and billing used. In addition, non-technical differentiators include the services that are offered in the pay-as-you-go model, service level agreement (SLA), rapid implementation, lower pre-cost, low maintenance and reduced environmental impact.

The following limitations are usually encountered when programming apps for a cloud computing environement:

- Knowledge of different types of cloud resources is required and this usually relies on procedural programming or scripting languages;
- Interaction with cloud resources is primarily accomplished through low-level APIs and command line interfaces;
- SaaS deployment depends on the programming environments supported by IaaS and PaaS providers;
- Lack of flexibility and efficiency to support generic applications that can run simultaneously on multiple cloud infrastructures. Therefore, it is clear that the development of system architecture and application development environments that can simplify and improve the cloud programming task are essential to harnessing their capacity.

Cloud software distribution is subject to security breaches, privacy abuses, and access control violations. Illegal copying, malware manipulation and other violations of cloud-based services, applications, and data are examples of security threats, intimacy, and trust.

To address access control issues, Yu et al. (2011) in the paper titled „A Novel Watermarking Method for Software Protection in the Cloud", identifies an internal access control threat that is not completely eliminated by common encryption techniques, cryptographic signatures, and access control labels. The authors address this threat by using the software watermarking.

Ensuring privacy is essential and has become one of the most relevant issues because in the absence of this, users may eventually lose confidence and passion for implementing virtualized cloud-based applications (CPUs, storage, databases, etc.).

However, cloud-based IT solutions are usually deployed on heterogeneous clouds and the features offered are often incompatible. This diversity prevents proper exploitation of the full potential of cloud computing as it hinders interoperability and promotes provider lock-in, and increases the complexity of developing and managing multi-cloud systems. Applying techniques and techniques based on advanced engineering models (MDEs) would be appropriate to tamper with this complexity.

MDE (Model-Driven Engineering) is a software engineering branch that aims to improve the productivity, quality and cost-effectiveness of software development by switching from the code-based paradigm to the model-based modeling. This approach, which is usually summed up by the expression "model once, generates from anywhere," is particularly relevant to reduce the complexity of the development methodology of complex systems such as multi-cloud systems. Models and modeling languages as the main artifacts of the development process allow developers to work at a high level of abstraction, focusing on cloud issues rather than on implementation details. Transforming the model as a primary technique to wholly or partly generate software discourages developers from repetitive and error-prone tasks.

Domain-specific modeling languages (DSMLs) provide abstractions and notations that allow for straightforward and understandable expression of domain concepts instead of encoding them in a lower-level programming language.

Unified Modeling Language (UML) is a family of graphical notation supported by a single meta-model that helps in the design of information systems, especially systems using object oriented programming languages. UML's usefulness in the development of information systems is that it is independent of any programming language.

III. DESIGNING A CLOUD ACCOUNTING APP USING UML

According to Wikipedia, UML is "a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system". Its creation was initially motivated by the desire to standardize notation systems and various software design approaches.

UML is not a development method by itself, but it was however designed to be comparable with OMT, the Booch method and RUP. UML has two major categories of diagrams. The first category represents structural information while the other category represents behavior and interactions.

This paper's focus will be the presentation of the behavior and structure diagrams, with examples for a cloud computing app.

A. USE CASE DIAGRAM

A use case diagram shows a collection of usage cases and actors that:

- provides a general description of how the system will be used
- provides an overview of the functionality that the system wants to offer
- shows how it interacts with the system with one or more actors

When designing a cloud accountig app, the development team should take into consideration two different end user perspectives. The two end user perspectives the author is referring to are the app manager and the day-to-day user.

As can be observed in Figure 1, from the cloud-based application manager's perspective, the use case diagram can be represented as follows:

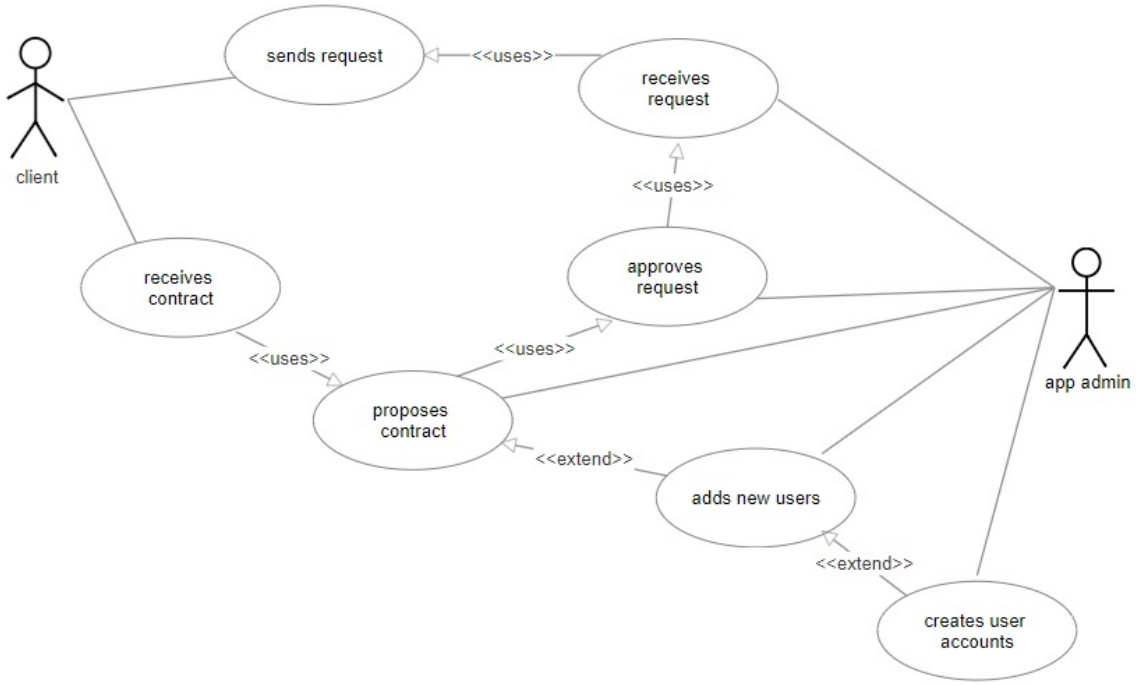

Figure 1:Use case diagram from app admin perspective

The client interacts with the app admin when sending requests and when receiving the contract. The app admin is the actor that receives all client requests, evaluates them and then approves or rejects them. Also, he is the one that can add new users and create their account once the contract proposal has been accepted and signed by the client.

In the case of a cloud accounting app, the „client" that the author referred to above becomes the "accountant". The "accountant" actor is the one that interacts with the partners, which can be internal – such as employees – or external – such as suppliers and customers. The accountant's role is to receive documents from the partners, verify them, return them if they do not meet all validity criteria or initiate the creation of all applicable accounting operations. The "accountant" actor is the one that adds all the new partners, the one that updates the chart of accounts, the one that creates accounting operations which will represent the basis for the monthly reports and financial statements. The management can fundament the decisional process after consulting the financial statements.

From the user's perspective, the use case diagram can be presented as follows in Figure 2:

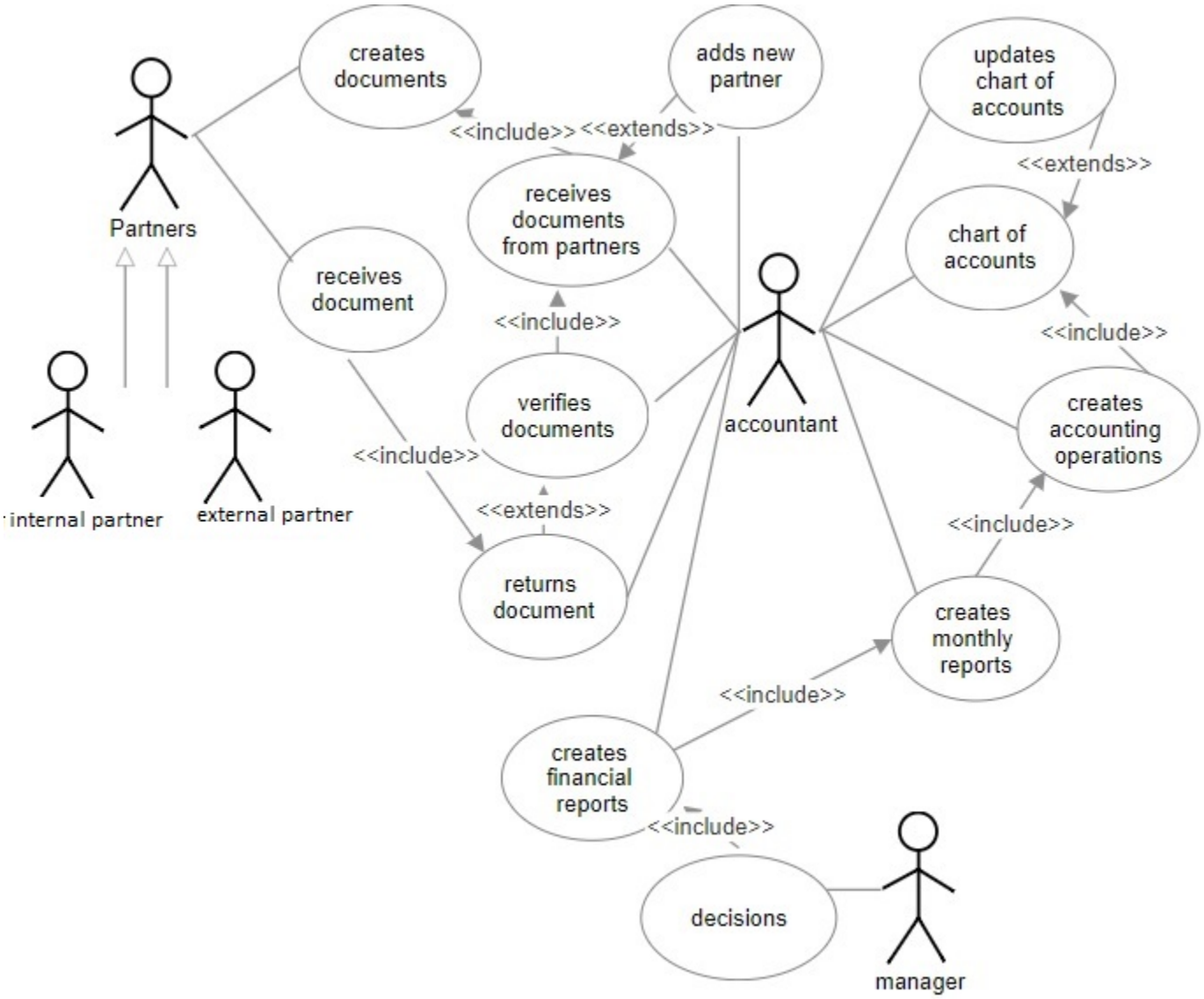

Figure 2:Use case diagram from the user's perspective

B. CLASS DIAGRAM

The class diagram describes the static structure of the information system by presenting the classes of the system, its attributes, methods and relations between objects.

Figure 3 present the class diagram for the accounting operations. The classes are represented by boxes which contain three compartments, as follows:

- the top section contains the name of the class
- the middle section contains the attributes of the class
- the bottom section contains the operations that can be executed by the class

The “AccountingOperation” and “Operation” classes are associated. The same applies for the “Document” and “Operation” class and for the “Account” and “Initial balance” classes. The “Ledger” class cannot exist without the “Operation” class, the relationship between those two classes is known as “Composition”. The same applies for the “Credit” and “Operation” classes and for the “Debit” and “Operation” classes.

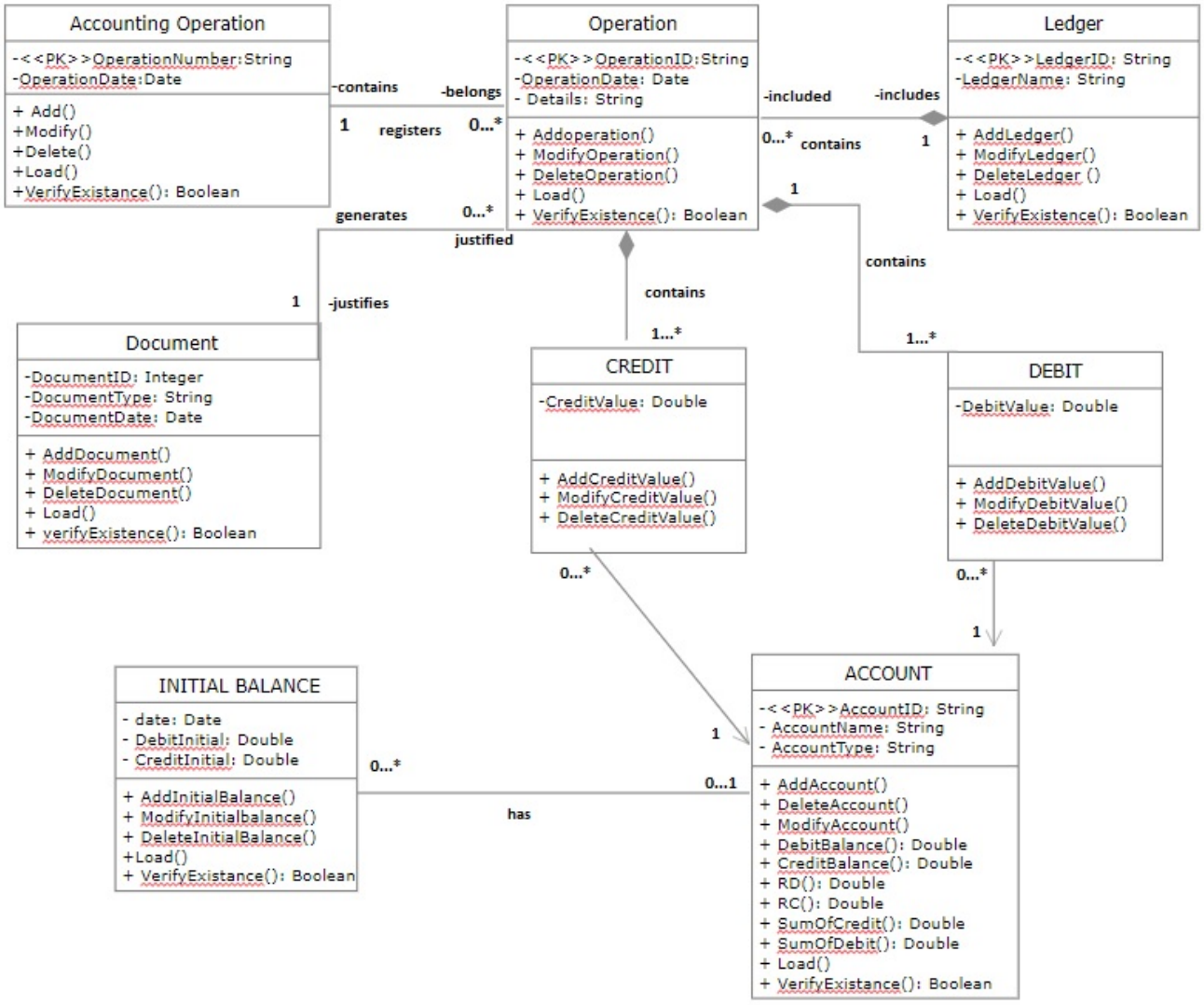

Figure 3: Class Diagram

C. ACTIVITY DIAGRAM

The activity diagram describes the sequence of operational activities of the components of a system.

Figure 4 presents the activity diagram for the entire app. The rounded rectangles represent the actions, the diamonds represent the decisions, the bars represent the start or the end of concurrent activities. The black circle at the start represents the initial node of the workflow and the encircled black dot represents the final node of the workflow. In this case, the workflow starts when documents are received. The next action is to verify the documents. This step is followed by a decision: if the document does not meet the standards, it must be returned and if the document does meet the standards the system can proceed to register the operation.

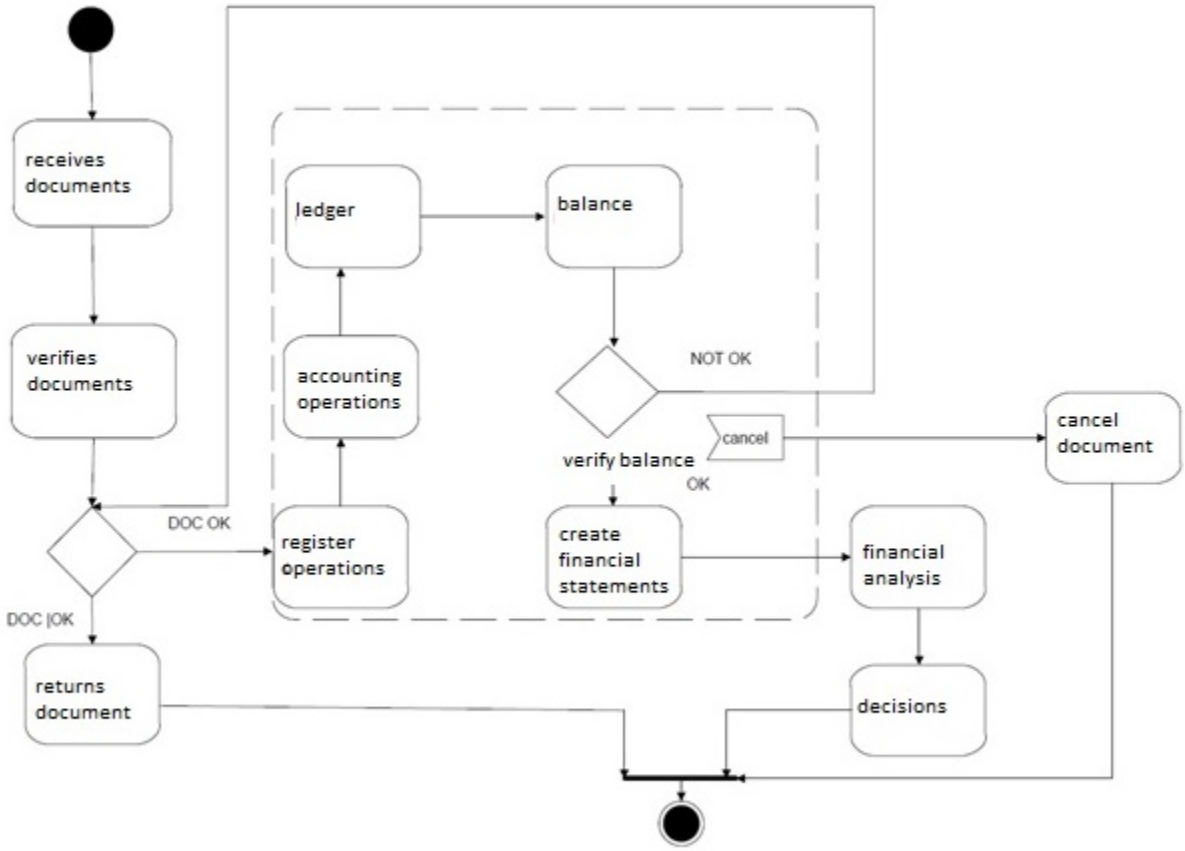

Figure 4: Activity Diagram for the app

Figure 5 presents the activity diagram for the “SumOfCredit” operation. After the initial node, the app faces a decision. If the SumOfCredit is grater or equal than SumOfDebit, then the next activity is to calculate the CreditBalance which will be equal to the difference between SumOfCredit and SumOf Debit. If SumOfCredit is however lesser then SumOfDevit, then the CreditBalance is zero. After the app computes the CreditBalance value it also reaches the final node.

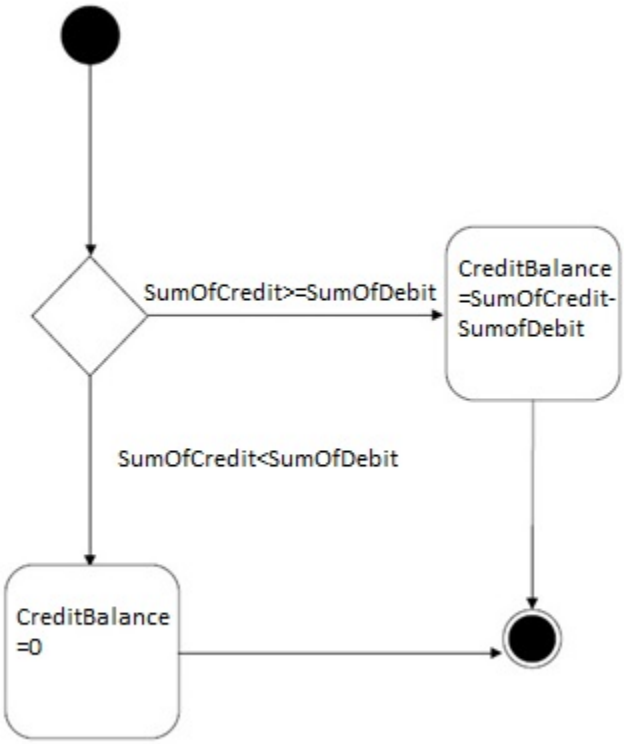

Figure 5: Activity Diagram for "SumOfCredit" operation

IV. CONCLUSION

All reliable software solutions require a solid and thorough foundation pf planning and clear, precise communication among the team members and the final users during all stages of the product's development. The main point of this paper is to highlight the importance of using UML diagrams to better visualize user interactions, processes and the system's structure when designing apps that will run in a cloud environment.

REFERENCES

[1] Z. Yu, C. Wang, C. Thomborson, J. Wang, S. Lian, AV. Vasilakos, "A Novel Watermarking Method for Software Protection in the Cloud," in A Novel Watermarking Method for Software Protection in the Cloud , 2011, DOI: 10.1002/spe.1088.

[2] G. Zhang, Y. Yang, D. Yuan, J. Chen, "A Trust-based Noise Injection Strategy for Privacy Protection in Cloud", in . Software: Practice and Experience 2011. DOI: 10.1002/spe.1052.

[3] N. Ferry, A. Rossini, F. Chauvel, B. Morin, A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems", 2013 IEEE Sixth International Conference on Cloud Computing, pp888-894

[4] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Special Publication 800-145, September 2001.

[5] E. Brandtzæg, M. Parastoo, and S. Mosser, "Towards a Domain-Specific Language to Deploy Applications in the Clouds," in *CLOUD COMPUTING 2012: 3rd International Conference on Cloud Computing, GRIDs, and Virtualization*. IARIA, 2012, pp. 213–218.

[6] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *GRID 2009: 10th IEEE/ACM International Conference on Grid Computing*. IEEE, 2009, pp. 17–25.

[7] X. Li, X. Yang, S. Kandula, M. Zhang, "CloudCmp: comparing public cloud providers," in *IMC 2010: 10th ACM SIGCOMM Conference on Internet Measurement*, M. Allman, Ed. ACM, 2010, pp. 1–14.

[8] B. Huang, Y. Xu, M. Yuan, G. Wu, "Formal Behavior Modeling: Business Processes Based on Cloud Platform", JOURNAL OF NETWORKS, VOL. 8, NO. 6, JUNE 2013, pp 1417 – 1424

[9] S. Zhang , J. Zhao. „Feature Aware Multiresolution Animation Models Generation[J]". *Journal of Multimedia*, 5(6): 622-628, 2010.

[10] B Huang, G. Q. Wu, L Wan, L. Li, J. Wang, "A visualization method of requirement checking based on software behavior", *Wuhan University Journal of Natural Sciences*, SpringerVerlag, vol.16, no.6, pp. 507-512, 2011.

[11] D. Harel, R. Marelly, „Come, Let's play: Scenario-Based Programming Using LSCs and the Play-Engine", Springer-Verlag, 2003, GER.

[12] F. Moller, Chris Tofts. „*A temporal calculus of communicating systems"*. Springer Berlin Heidelberg, 1990.

[13] L. Guoyuan, H. Shan, H. Hao, W. Jiyi, C. Wei, "Access control security model based on behavior in cloud computing environment," *Tongxin Xuebao/Journal on Communications*, vol. 33, no. 3, pp. 59-66, March 2012.

[14] J. Damasceno, F. Lins, R. Medeiros, B. Silva, A. Souza, D. Aragão, P. Maciel, N. Rosa, B. Stephenson, J. Li, "Modeling and executing business processes with annotated security requirements in the cloud," *In Proceedings of the 2011 IEEE 9th International Conference on Web Services*, ICWS 2011, pp. 137- 144, 2011

[15] Z. Jia, L. Yeli, "The model design of print on demand business process facing the cloud," *International Review on Computers and Software*,vol. 7, no. 6, pp. 3157-3161, November 2012.

[16] A. Tobias, L. Frank, M. Ralph, S. Steve, "Towards BPEL in the cloud: Exploiting different delivery models for the execution of business processes," *In Proceedings of the 2009 IEEE Congress on Services*, SERVICES 2009, pp. 670-677, 2009.

[17] H. Mamoun, "Realizing business agility requirements through SOA and cloud computing," *In Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, pp. 379-380, 2010.

[18] L. Shuang, "The design and realization of cloud computing framework model based on SOA," *Advanced Materials Research*, vol(171-172), 2011, pp. 696-701, 2011.

[19] Z. Huimin, Y. Xiaolong,"Cloud computing architecture based on SOA,"*In Proceedings of the 2012 5th International Symposium on Computational Intelligence and Design*, pp.369-373, 2012.

[20] G. Vladimir, C. William, C. William. R, S. George. O, Y. Stephen. S, "Challenges towards the global adoption of cloud computing," *In Proceedings of the International Computer Software and Applications Conference*, pp.369-372, 2012.

[21] W. Lingyan, L. Aimin, "The study on cloud computing resource allocation method," *Applied Mechanics and Materials*, vol(198-199), pp.1506-1513, 2012.

[22] E. Brandtzæg, P. Mohagheghi, S. Mosser, "Towards a Domain-Specific Language to Deploy Applications in the Clouds", CLOUD COMPUTING 2012 : The Third International Conference on Cloud Computing, GRIDs, and Virtualization, ISBN: 978-1-61208-216-5, pp 213 – 218

[23] Garg, S. K., Versteeg, S., Buyyaa, R., 2013, "A framework for ranking of cloud computing services", Future Generation Computer Systems volume 29 (2013), pp 1012–1023

AUTHORS

First Author – Laura Elena Tudoran, PhD Student, Academia de Studii Economice, tudoranlaura@gmail.com
Correspondence Author – Laura Elena Tudoran, tudoranlaura@gmail.com, +40748024955