Modelling Financial SaaS as Service Components

Muthu Ramachandran and Victor Chang

School of Computing, Creative Technologies and Engineering, Leeds Metropolitan University, Leeds, LS6 3QS, U.K.

Abstract. Financial applications demand better performance and accuracy in a cloud than the traditional computing platforms. Therefore, designing financial software as a service (FSaaS) requires engineering and systematic approach. This paper has designed a financial SaaS component model and a service architecture supporting required flexibility, scalability, and allowing change in environment. This paper has proposed an integrated service-oriented architecture and a SaaS component model for financial domain that provides trequired scalability, flexibility and customisation. We have also demonstrated the design and customisation of service component interfaces to the financial simulation so that it provides automatic prediction models for investors to know accurate results in buy and sale prices. Therefore, large-scaled simulations can be achieved within a matter of seconds (13.5 seconds).

1 Introduction

Financial applications require on-demand services that are offered by cloud computing with cost-benefits. Therefore, financial domain has begun to reap these benefits with emerging financial SaaS such as FinancialForce which is developed by SalesForce, NetSuite, Intacct, and Oracle's financial SaaS. NetSuite [11] claims helped companies to increase their revenues by 95% by moving to their financial SaaS. Accenture [1] report on financial technology trends and high performance computing predicts:

- Leveraging technology to address new & change in regulations
- Reliable and globally harmonised financial systems
- Add value through strategic applications
- Harvest benefits from technology

Accenture [1] report also claims SaaS to be a simple, efficient, engaging, accessible, clear structure, intuitive, and supportive. With keeping this set of requirements as design criteria, this paper has designed a SaaS component model and a service architecture supporting required flexibility, scalability, and allowing change in environment. This paper has proposed an integrated service-oriented architecture and SaaS component model for financial domains which provides required scalability, flexibility and customisation that are at the heart of a financial SaaS.

We demonstrate the system design and the implementations. Examples include the results of running software that can compute call and put prices. Results of simula-

Ramachandran M. and Chang V..
Modelling Financial SaaS as Service Components.
DOI: 10.5220/0004979000130020
In *Proceedings of the International Workshop on Emerging Software as a Service and Analytics* (ESaaSA-2014), pages 13-20
ISBN: 978-989-758-026-0
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

tions also support the fact that all computations can be completed within seconds. To illustrate our key messages, the breakdown of this paper is as follows. Section 2 presents the overall view about SOA approach an FSaaS application. Section 3 discusses the service component model. Section 4 shows the results of running FSaaS and large-scaled simulations in seconds. Section 5 sums up issues covered and provides conclusion.

2 SOA Approach for FSaaS Applications

FSaaS applications require open, flexible, interoperable, collaborative and integrated architecture to provide services to integrate various stakeholders such as citizens and financial services (e-shares, e-stockbrokerage, e-fund-management, QoS, FSaaS cloud simulation services, e-health, e-tax, e-national security, e-pension, e-payment, e-education and training, e-work and employment, e-funding, etc), business organisations and vendors (e-procurement), and government agencies (both inter and intra governments). Therefore, designing architecture for FSaaS applications pose a design challenge. Software architectural design needs to be verified, assessed and evaluated for its quality before its development. There are well known generic design criteria such as flexibility, maintainability, testability, reusability, etc. The key to achieving good architectural quality for the system being developed is to extract key characteristics of this system from various sources such as non-functional requirements, customer requirements, existing systems, experts, research literatures [2-3 & 6-16] and so on. This paper identifies such characteristics from various perspectives supporting emerging technologies which are shown in Figure 1. These FSaaS application characteristics are the backbone for developing service-oriented architecture and components.

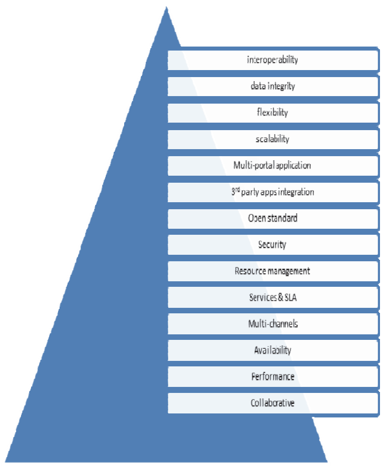

Fig. 1. Characteristics of FSaaS Applications.

To further expand on concepts in Figure 1, FSaaS applications need to work and interface with multi-platform and multi-vendor applications and therefore this needs to be designed for interoperability. Multi-channel delivery of services has been recommended to deliver FSaaS services through various and emerging communication channels such as mobile phones, digital TV, cloud services, call centres, kiosks, emails, PCs, teleconferencing, e-gov apps, web, and webinars - some of these are not been used by any government at present. Each characteristic shown in Figure 1 represents not only the application attribute but it also provides a set of key design criteria and quality attributes for developing architecture that support emerging technologies for FSaaS. For example, the design criteria interoperability is paramount for FSaaS applications as they are multi-platform and multi-channel based. Interoperability is supported in the service-oriented architecture design for FSaaS applications by means of a service bus and loose coupling of the other components which is shown in Figure 2.

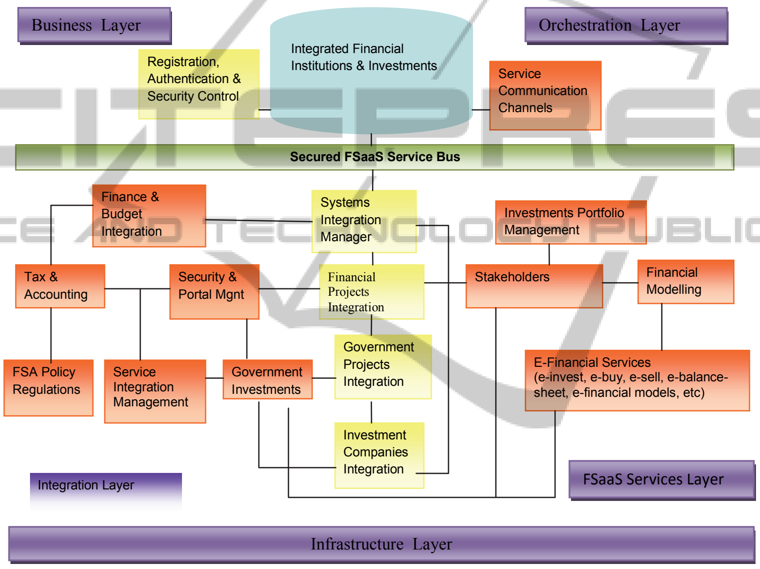
Business Layer
Orchestration Layer
Registration, Authentication & Security Control
Integrated Financial Institutions & Investments
Service Communication Channels
Secured FSaaS Service Bus
Finance & Budget Integration
Systems Integration Manager
Investments Portfolio Management
Tax & Accounting
Security & Portal Mgnt
Financial Projects Integration
Stakeholders
Financial Modelling
FSA Policy Regulations
Service Integration Management
Government Investments
Government Projects Integration
E-Financial Services (e-invest, e-buy, e-sell, e-balance-sheet, e-financial models, etc)
Investment Companies Integration
Integration Layer
FSaaS Services Layer
Infrastructure Layer

Fig. 2. Service-Oriented Architecture for FSaaS.

These services can be made available as stand alone, integrated, componentised, web based service component, composite service (a set of interconnected services), virtualised services (cloud based), and dynamically re-configurable services. The architecture presented in this paper is then based after a critical review and analysis of a number of existing architectures for FSaaS applications. As mentioned before, the SOA based architecture consists of four distinct levels of abstraction layers which are connected and communicated by messages through a core communication channel known as a service bus or a central bus. These layers are: 1) a business layer with a dedicated set of services, 2) an orchestration layer with a set of services where new services can be composed, 3) an FSaaS layer that supports integration of services, government departments and local governments, and 4) an e-business layer that supports new businesses and integration of data. The SOA based architecture for FSaaS services then ensures that it achieves the expected service-oriented design factors such as customisation, cost-effectiveness, availability, etc.

Referring to Figure 2, at the business and orchestration layers provide high level service composition based on new business perspective and policies (both political and economical factors). Mostly, the customisation and the new business needs arise from these two key variables. The sub-systems such as registration control, security control, integrated services for FSaaS applications control, and communications channels help to achieve customisation at a higher level of abstraction without affecting underlying business logic services. These are communicated and connected to layers below using a concept of service bus known as FSaaS secured service bus. The layer below the business layer provides services from various FSaaS departments, and external suppliers (e-Business layer).

3 Service Component Model for FSaaS Applications

This section proposes an approach to developing FSaaS applications which is based on developing a financial service component which has provides required customization extensibility. As stated before, e-government applications require open, flexible, interoperable, collaborative and integrated architecture to provide services. These services can be made available as stand alone, integrated, componentised, web based service component, composite service (a set of interconnected services), virtualised services (cloud based), and dynamically re-configurable services. This vision is similar to the Open Group's Service Integration Maturity Model (OSIMM) [13]. The OSIMM provides:

- A process roadmap for attaining key practices with metrics
- Seven levels of maturity to improve
- A quantitative model for assessing current practices and to improve with recommended practices

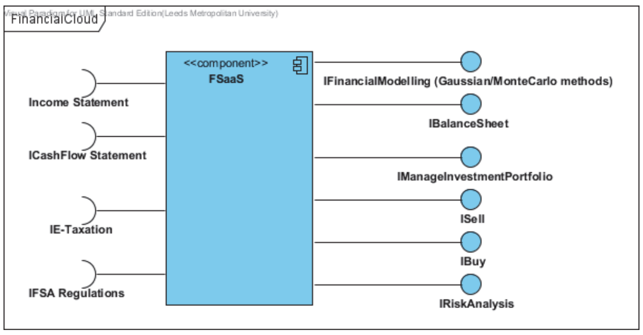
Visual Paradigm for UML Standard Edition(Leeds Metropolitan University)
FinancialCloud
<<component>>
FSaaS
Income Statement
ICashFlow Statement
IE-Taxation
IFSA Regulations
IFinancialModelling (Gaussian/MonteCarlo methods)
IBalanceSheet
IManageInvestmentPortfolio
ISell
IBuy
IRiskAnalysis

Fig. 3. FSaaS Service Component Model.

Figure 3 shows a design of software as a service component model for financial applications (FSaaS) where it shows two types of services such as provide (lollypop notation) and require services (arc notation and the naming convention starts with I). For example, FSaaS services include Income statement, ICashFlow statement, Ie-taxation, IFSA regulations). IFSA refers to providing interface service integration for Financial Authority regulations which any investment service providers should adhere to and is flexible for change in their regular change thus providing required scalability and flexibility of FSaaS characteristics discussed earlier.

Similarly, providers service interfaces are Ifinancial modelling, IBalancesheet as a service which is automatically produced from income and cashflow statements. The other provider interfaces include ImanageInvestment Portfolio, Isell, IBuy, and IRiskAnalysis services. In order to achieve the recommended process and key practices, we need to have interoperability, reconciliations and resiliency between systems, where interface linkages are appropriate to other services such as those of emerging technologies interfaces (e.g. Ie-taxation linked to E-Gov applications, and IFSA regulation service linked to FSA). Similarly, such reconciliations services must be automated for the sake of cost-efficiency. To develop an integrated FSaaS service-oriented architecture system, it is critical that service analysts and software engineers identify and address the key challenges when implementing e-government services. Some of relevant questions are listed below:

- How to adopt a new process and to identify the scope of the business services to be developed and supported?
- How data integrity will be achieved and secured?
- How data will be protected when needed to support management decisions?
- How systems will fit together to support service choreography and orchestration layers?
- How systems will fit together to support the enabling emerging technologies (system architecture needs to be service-oriented thereby any new emerging technologies in the future should easily be integrated more cost-effectively than with traditional system architectures)?
- How data and services will be safeguarded and secured to ensure the integrity and re-configurability of service operations and personal data (information assurance)?
- How new services can be composed and re-configured at run-time?

This paper has address some of these issues such as achieving business process using BPMN for identifying FSaaS component services, management decision are intergrted with service component interfaces as governance, SOA architecture model has been designed to illustrate other issues. Currently, we are working on developing a WSDL to link to relevant MATLB code or any other simulation services as shown in Table 1 (discussed in the following section).

4 Demonstrations and Simulations for FSaaS

This section demonstrates how FSaaS can be used and explains the code and results during and after using FSaaS services. This mainly involves Monte Carlo Methods (MCM) and Black Scholes Model (BSM). Chang et al., [4] describe how to use MCM and BSM for financial computation. They demonstrate the use of risk analysis and financial modelling on Clouds based on MATLAB and Mathematica, which offer benefits such as performance, accuracy and integration with security. This includes the selection of Linear Square Method (LSM) that can compute 100,000 simulations in one go, which takes between 4 to 25 seconds depending on the number of time steps. It can also work with IBM Fined-Grained Security Model, and it can provide a safer environment for FSaaS on Clouds.

Mathematical models such as MCM are used in Risk Management area, where they are used to simulate the risk of exposures to various types of operational risks. Monte Carlo Simulations (MCS) in Commonwealth Bank Australia are written in Fortran and C#. Running such simulations may take several hours or over a day [4, 5]. The results may be needed by the bank for the quarterly reporting period. MCM is suitable to calculate best prices for buy and sell, and provides data for investors' decision-making [4, 5]. MATLAB is used due to its ease of use with relatively good speed. While the volatility is known and provided, prices for buy and sale can be calculated. Part of the code (fareastmc.m) to is used to present formulas in MCM and demonstrate coding algorithm presented in Table 1.Variables are explained in [4].

Table 1. Coding algorithm in Monte Carlo in MATLAB for best buy/sell prices.

```
dt=T/(NSteps-1);
vsqrdt=sigma*dt^0.5;
drift=(r-(sigma^2)/2)*dt;
x=randn(NSimulations,NSteps);
Smat=zeros(NSimulations,NSteps);
Smat(:,1)=S;
for i=2:NSteps,
   Smat(:,i)=Smat(:,i-1).*exp(drift+vsqrdt*x(:,i));
end
```

The following demonstrates running the code and the calculated prices. Call prices are to buy and put prices are for sale. The program calculates the lower limit, the MCPrice value and the upper limit for each buy and sale category.

```
> fareastmc (this is the name of the MATLAB file)
            [LowerLimit MCPrice UpperLimit]
Call Prices: [4.196694 4.248468 4.300242]
Put Prices:  [7.610519 7.666090 7.721662]
```

Outliers are common while during computation and they do not fall into the expected results. To achieve a high level of accuracy, outliers need to be removed. Figure 4 is a good example. Two screenshots on the left half with "original gamma variables" show that financial indexes have outliers. In other words, they are not smooth and outliers need to be filtered out. The other two screenshots on the right half with "stratified gamma variables" means that all outliers are removed and the results are smooth and usable for predictive modelling or other financial computation. Removal of outliers can ensure a high quality of data analysis can be offered to investors. How removal of outliers relates to the architecture is as follows. It ensures numerical computations are as accurate as possible and do not provide results that may make investments or decisions that incur in the loss of money and reputation.

We then perform large-scale simulations to show that all computations can be achieved within seconds. The hardware infrastructure is as follows. The desktop has

2.67 GHz Intel Xeon Quad Core and 4 GB of memory (800 MHz) with installed. Two Amazon EC2 public clouds are used. The first virtual server is a 64-bit Ubuntu 8.04 with large resource instance of dual core CPU, with 2.33 GHz speed and 7.5GB of memory. The second virtual server is Ubuntu 7.04 with small resource of 1 CPU with 2.33 GHz speed and 1.5 GB of memory. There are two private clouds set up. The first private cloud is hosted on a Windows virtual server, which is created by a VMware Server on top of a rack server, and its network is in a network translated and secure domain. The virtual server has 2 cores of 2.67 GHz and 4GB of memory at 800 MHz. The second private cloud is a 64-bit Windows server installed on a rack, with 2.8GHz Quad Core Xeon, 16 GB of memory. Our FSaaS can work on both Octave or MATLAB. The experiment began for running the Octave 3.2.4 on desktop, private cloud and public cloud and started one at a time due to the licensing issue. Table 1 summarises the benchmark for the execution time, which involves running FSaaS simulations in five different platforms. It took longer time to run simulations in the public cloud with small instances, which is not recorded in Table 2. This is due to their low CPU and memory requirements resulting in longer completion time. The key advantage of designing FSaaS as a service component is any change in the financial models which can be changed without any ripple effect to the entire FSaaS architecture.

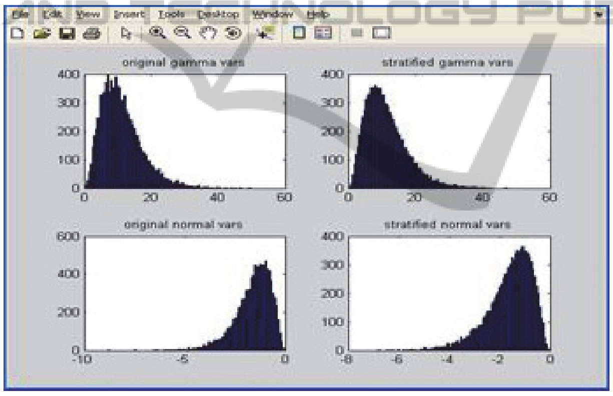

Fig. 4. Removal of outliers.

Table 2. Timing benchmark to run FSaaS code on Octave 3.2.4.

Number of simulations and time taken (sec)	5,000	10,000	15,000
Desktop	11.08	11.92	12.71
Public cloud (large instance)	11.95	12.30	13.15
Private cloud (default VMs)	11.31	12.13	12.90
Private cloud (large VMs)	9.63	10.51	11.48

5 Conclusion

Financial applications demand better performance and accuracy in a cloud than the traditional computing platforms. Therefore designing financial software as a service

(FSaaS) requirerss engineering and systematic approach. This paper has demonstrated a systematic approach to developing large scaled financial applications with current technology such as SOA which supports SaaS features to be integrated with cloud solutions. This paper has also demonstrated the design and customisation of service component interfaces to the financial simulation so that it provides automatic prediction models for investors to know accurate results in buy and sale prices. Therefore, large-scaled simulations can be achieved within a matter of seconds (13.15 seconds).

References

1. Accenture (2011) http://www.slideshare.net/fullscreen/ramblingman/accenture-financial-saa-s-external-presentation-final/3
2. Baresi, L., Di Nitto, E., and Ghezzi, C (2006) Towards open-world software: issues and challenges, Computer, Special Issue on Software Engineering: the past and the future, October 2006.
3. Cartwright I and Doernenburg E (2006) Time to jump on the SOA bandwagon, IT Now, British Computer Society, May 2006
4. Chang, V., Li, C.S, De Roure, D., Wills, G., Walters, R.J., Chee, C.: The Financial Clouds Review, International Journal of Cloud Applications and Computing, 1 (2), (2011) 46-63.
5. Chang, V., Business Intelligence as Service in the Cloud, Future Generation Computer Systems, (2014).
6. Erl, T (2005) Core principles for service-oriented architectures, August 2005, Retrieved Oct 2013, from www.Looselycoupled.com/opinion/2005/erl-core-infr0815.html
7. Groves, D (2005) Successfully planning for SOA, BEA Systems Worldwide, 11 Sept 2005
8. Helbig, J (2007) Creating business value through flexible IT architecture, Special Issue on Service-oriented Computing, IEEE Computer, V.40, No.11, November 2007.
9. Hohpe, G (2002) Stairway to Heaven, Software Development, May 2002
10. Nano, O. and Zisman, A. (2007) Realizing service-centric software systems, Special issue on SoC, IEEE Software, November/December 2007.
11. NetSuite (2014) http://www.netsuite.co.uk/portal/uk/seo-landing-page/accounting-2/main.shtml?gclid=CLK9k5q-37sCFTHLtAodikoAzw, accessed 2014
12. Open Group (2009) OSIMM, Retrieved Oct 2013, from https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=12450
13. Ramachandran, M (2012) Software Security Engineering: Design and Applications, Nova Science Publishers, New York, USA, 2012. ISBN: 978-1-61470-128-6, https://www.novapublishers.com/catalog/product_info.php?products_id=26331
14. Ramachandran, M (2013) Business Requirements Engineering for Developing Cloud Computing Services, Springer, Software Engineering Frameworks for Cloud Computing Paradigm, Mahmmood, Z and Saeed, S (eds.), http://www.springer.com/computer/communication+networks/book/978-1-4471-5030-5
15. Sellami, M. and Jmaiel, M (2013) A Secured Service-Oriented Architecture for FSaaS in Tunisia, www.redcad.org/PDFs/C33.pdf, accessed on 15th September.
16. Wauters, P., Declercq, K., van der Peijl, S., and Davies, P (2011) Study on cloud and service-oriented architectures for FSaaS, FP7 report ref. Ares(2012)149022 - 09/02/2012, Deloitte.