

**A Modeling based Architectural Framework for
Multi-Tenant SaaS Web Application Development**

Tharaka de Alwis

108256D

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

October 2011

A Modeling based Architectural Framework for Multi-Tenant SaaS Web Application Development

Tharaka de Alwis

108256D

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

October 2011

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Tharaka de Alwis

Date

The above candidate has carried out research for the Masters thesis under my supervision.

Dr. Chandana Gamage

Date

Department of Computer Science and Engineering

University of Moratuwa

Abstract

In the era of cloud computing, Software as a Service (SaaS) has been coined as the next generation of web application software solutions. In this model software needs to allow multiple users organizations (tenants) to use the same application instance in a highly reliable and secure manner. Users will be given access to the application on a subscription basis and the application is customized according to their needs by changing user interfaces, work-flow and business processes. SaaS gives end user organizations an opportunity to engage in business without making a significant investment on buying expensive hardware or software. The end users' customized solution is hosted over the Internet giving them the opportunity to access the application as a utility service, on-demand basis.

From a functional perspective SaaS web applications require the capabilities of customization and configurability to support the evolution of the product. Since multiple end user organizations make use of the application, high attention towards security, scalability, performance and availability is mandatory from an architectural perspective. This makes SaaS application development, deployment, migration and maintenance extremely challenging and sometimes considered as an engineering nightmare. The research community has come up with different frameworks and modeling approaches to develop SaaS applications. Yet there are very few architectural tools to model SaaS solutions to help architects develop practically feasible SaaS solutions.

Our intention from the project was to identify the challenges in SaaS application development, maintenance and migration. We propose a modeling based SaaS framework with a set of tools and an effective methodology to develop the SaaS solutions efficiently while meeting the critical SaaS architectural requirements. The authors have come up with a framework to develop SaaS applications using third party web application frameworks and UML 2.0 based Profile named SaaSML. Using the proposed modeling based SaaS framework approach, Software Architects will be able to easily build SaaS web applications and address the challenges of SaaS application development life cycle.

Keywords: Multi-Tenant SaaS web application, SaaS Framework, Software Architecture tools, UML and XML based modeling tool

Acknowledgements

I would like to present my deepest thanks to my research supervisor Dr. Chandana Gamage for his valuable support, supervision and useful suggestions throughout this thesis study. His guidance enabled me to complete my work successfully. Special thanks to Dr. Shehan Perera, Mrs. Vishaka Nanayakkara, Dr. Shahani Weerawarana and Dr. Srinath Perera for providing valuable insight into topics on Software Design, Software Architecture, Distributed Computing and Computer Science Research which helped me extensively to complete my work.

Special thanks go to the CEO of Sabre Technologies, Mr. Chandima Cooray who gave me insight into SaaS five years back and opportunities work on SaaS application development projects. I am also grateful for the support & assistance provided by the Senior Management of Sabre Technologies, who provided me with research facilities and opportunities to attend conference events on Cloud Computing. I would like thank the development teams and clients who worked with me at Sabre on SaaS projects during the last couple of years. The practical experience that I gained from them was invaluable to lay the foundation and derive solutions for this thesis study.

I would also like to thank my colleague Kokila De Silva for reviewing my work and providing valuable feedback & suggestions for improvement.

Finally, I wish to express my gratitude to my parents for their love, support, guidance and education they provided throughout my life.

Table of Contents

Declaration	iii
Abstract	iv
Acknowledgements	v
List of Tables	xi
List of Figures	xiii
1 Introduction	1
1.1 Introduction to Cloud Computing	1
1.2 Software as a Service (SaaS)	4
1.3 The Principle of Multi-tenancy	4
1.4 Requirements for SaaS Applications	5
1.4.1 Functional requirements	5
1.4.2 Non-functional requirements	6
1.5 Challenges of SaaS	7
1.5.1 Maintainability issues of SaaS applications	7
1.5.2 A model of challenges of SaaS applications	8
1.6 Summary	12
2 Literature Review	13
2.1 SaaS Architectural Design Overview	14
2.2 SaaS Architectural Design Approaches	17
2.2.1 SaaS Architecture through purpose built platforms	17
2.2.2 SaaS Architecture through SOA based platforms	22
2.2.3 By extending ASP based web application frameworks	23

2.2.4	Other approaches	26
2.2.5	Summary of Architectural approaches	27
2.3	SaaS Application Development Through Modeling	29
2.4	SaaS Modeling Tools	30
2.5	SaaS Architectural Evaluation Approach	36
2.6	Summary	37
3	Methodology	38
3.1	Anatomy of a SaaS Architectural Framework	39
3.1.1	SaaS Architectural goals and constraints	40
3.1.2	Business usecase view	41
3.1.3	SaaS logical architecture	45
3.1.4	SaaS framework strategies to solve SaaS functional requirement challenges	47
3.1.5	SaaS framework strategies to solve SaaS non-functional requirement challenges	48
3.2	SaaS Modeling	49
3.2.1	Model driven architecture	49
3.2.2	MDA tools	51
3.2.3	The Unified Modeling Language (UML)	52
3.2.4	Domain specialization through UML Profiles	54
3.2.5	MDA tools based on UML 2.0 profile to develop SaaS web applications	54
3.2.6	SaaSML overview	56
3.2.7	Generation of a domain model based on SaaSML diagrams	64
3.3	SaaS Application Development	65
3.3.1	Third party web application frameworks for each layer	67
3.3.2	Tool chain selection criteria	67
3.3.3	Application framework usage to solve SaaS functional requirement challenges	70
3.3.4	Application framework usage to solve SaaS non-functional requirement challenges	72
3.3.5	Summary of application framework approach to derive SaaS architecture	73
3.4	Other Methodologies to Develop SaaS Web Applications	74

3.4.1	SOA based Architecture modeling and development	74
3.4.2	Software product line based modeling and development	75
3.5	Summary	76
4	Implementation	77
4.1	Process View	77
4.2	Logical View	84
4.2.1	Structuring of Business Components	84
4.2.2	MDA approach to generate SaaSML based components	85
4.2.3	Tenant aware services	87
4.3	Implementation View	88
4.3.1	Tenant specific UI branding	89
4.3.2	Tenant specific Workflow customization	93
4.3.3	Tenant specific labels, messages and errors	95
4.3.4	Tenant specific UI fields in HTML forms	99
4.3.5	Tenant specific business logic, rules, services and DAO execution .	104
4.3.6	SaaS billing module and subscription plans	107
4.3.7	Service usage metering	108
4.3.8	Performance allocation based on Tenants billing plan	112
4.3.9	Fault monitoring based on Tenants using logs	115
4.3.10	Tenant security for SaaS application	117
4.3.11	Tenant specific extensions to the data model	120
4.4	SaaS Modeling Tool	127
4.4.1	Goals and context	127
4.4.2	Business use case	128
4.4.3	Generate diagrams based on SaaS-ML profile	128
4.4.4	A tool to develop a build environment keeps the core application and tenant specific components	129
4.4.5	A tool to test business logic on core application and tenant's com- ponents	129
4.4.6	Viewpoint description of the tool	129
4.5	Summary	134
5	Analysis	135
5.1	SaaS Application Testing Methodology	135

5.2	Performance Testing Methodology	141
5.2.1	Requirements for performance study	141
5.2.2	Goals of the performance study	141
5.2.3	Assumptions and known limitations	141
5.2.4	Performance Metrics Selection	141
5.2.5	Monitoring Tool	142
5.2.6	Evaluation Technique and Experiment method	142
5.2.7	Workload	143
5.2.8	Design Experiment	143
5.2.9	Analysis Test Results	144
5.3	ATAM Finding on the SaaS Framework	147
5.3.1	Quality Scenarios	147
5.3.2	Quality attribute utility tree	158
5.3.3	Risks	160
5.3.4	Non-risks	161
5.3.5	Sensitivity points	162
5.3.6	Trade-off points	162
5.4	Summary	163
6	Conclusion	164
6.1	Key Findings	164
6.2	Future Work	165
	References	167
	List of Abbreviations	175
A	Technologies Used	177
A.1	Java Application Server platform	177
A.2	Java Web Server platform	177
A.3	Servlet	178
A.4	Java Server Pages (JSP)	178
A.5	Struts	178
A.6	Tiles	178
A.7	Spring Framework	179
A.8	Hibernate	179

A.9 EMF	180
-------------------	-----

List of Tables

1.1	Categories of Cloud Computing delivery models	3
2.1	Summary SaaS Architecture development approaches	27
2.2	Summary of SaaS tools research work done by academics	35
3.1	Architectural goals for Multi-Tenant Architectural design	40
3.2	SaaS framework strategies to solve SaaS functional requirement challenges	47
3.3	SaaS framework strategies to solve SaaS non-functional requirement chal- lenges	48
3.4	Summary of UML2 based diagrams	53
3.5	Application framework usage to solve SaaS functional requirement chal- lenges	70
3.6	Application framework usage to solve SaaS non-functional requirement challenges	72
4.1	Architectural goals for SaaS Modeling tool framework	127
4.2	Add, Modify and Remove Model (Java files)	131
4.3	Model source artifacts based on Tenant	132
4.4	Ensure the build is not broken though out the life cycle of project	132
5.1	Selection parameter hierarchy	136
5.2	Converted global weight of attributes	137
5.3	Product raw score with respect to attribute	137
5.4	Product raw score with respect to attribute justification	138
5.5	Ranking of products	140
5.6	Modifiability Scenario - Add, Modify and Remove User Interface views . .	148
5.7	Modifiability Scenario - Vendor organization's developer change components	148

5.8	Modifiability Scenario - SaaS Vendor organization’s developer or Tenant organization developer change components (UI, Work-flow, business process, rules or service) in tenant specific area in a shorter duration (compared to a new development)	149
5.9	Security Scenario - Imposter trying to login as a Tenant User to user portal .	150
5.10	Security Scenario - Tenant user trying to access unauthorized page through URL submission	150
5.11	Security Scenario - Past employees, clients with expired accounts try to get unauthorized access	151
5.12	Performance Scenario - Concurrent user computation load increases an response time exceeds threshold	152
5.13	Performance Scenario - Data losses when connecting with external systems	152
5.14	Performance Scenario -High computation usage by a Tenant organization (or multiple) causing computation load increases an response time exceeds threshold level for other Tenants	153
5.15	Scalability Scenario - Allow any number of concurrent user access system managing infrastructure cost	154
5.16	Usability: Tenant user personalize the layout	155
5.17	Usability : Ensure Tenant Users experience to make it a utility service to use it on a regular basis	155
5.18	Availability : SaaS application system crash or service outage	156
5.19	Availability : SaaS application authentication user authentication Federated Identity Management system (Google or Salesforce.com) is down	156
5.20	Availability : SaaS application connection sub system not available (Database, Mail Server)	157

List of Figures

2.1	SaaS logical Architecture given in the article Architecture Strategies for SaaS applications (Catching the Long Tail)-MSDN	15
2.2	Force.com Architecture	18
3.1	Usecase diagram of a typical SaaS web application	44
3.2	SaaS Architecture	45
3.3	Models and transformations in MDA	50
3.4	SaaSML and UML representation Venn diagram	56
3.5	Core Usecase Diagram of a SaaS web application	58
3.6	Tenant Usecase Diagram of a SaaS web application	59
3.7	Core Domain Diagram of a SaaS web application	60
3.8	Tenant Org Diagram of a SaaS web application	60
3.9	User Hierarchy Diagram of Tenant of a SaaS web application	61
3.10	User Provisioning Diagram of a SaaS web application	62
3.11	Billing and Metering Diagram of a SaaS web application	63
3.12	Governance Rules Diagram of a SaaS web application	64
3.13	Global Settings Diagram of a SaaS web application	65
3.14	Detailed Implementation Architecture	67
4.1	Process flow of the login service for Browser based user access of the SaaS application	78
4.2	Process flow of the rest of the SaaS application through Browser based user access	81
4.3	Process flow of the login service for Web Services access within the SaaS application	82
4.4	Process flow of transactional web services of the SaaS application	83
4.5	SaaSML based Component Diagram	85

4.6	SaaSML based MDA approach to generate sources	86
4.7	Detailed logical architecture	88
4.8	Eclipse Architecture	130
4.9	Eclipse Architecture to support SaaS Architectural framework	133
5.1	JMeter Test Plan thread execution criteria	144
5.2	JMeter Overall Test Plan with Thread Group, Web Requests and Reporting listeners	145
5.3	Configuration option to abandon the test automatically if error rate goes beyond 0.1%	146
5.4	Graph that represents concurrent user sessions (threads) Vs Test duration . .	146
5.5	Graph that represents Response time Vs Test duration	146
5.6	Graph that represents Throughput Vs Test duration	147
5.7	Quality Attribute Utility tree of Multi-Tenant SaaS application	159