



UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS
MSc Computer Science

**Architecture for Cloud-Based Rapid Application
Development Framework**

BY

Isaac Gachugu Gathia

P58/73002/2009

Mobile: 0721 709 390

Email: igachugu@gmail.com

Supervisor: Prof. Elijah Omwenga.

**Submitted in partial fulfillment of the requirements of Master of Science in Computer
Science**

DECLARATION

This project as presented in this report is my original work and has not been presented for any other University award. All the work in this project is my own except where acknowledged in the text.

Sign:  _____

Date: 31/05/2012

Isaac Gachugu Gathia
P58/73002/2009

This project has been submitted as partial fulfillment of the requirements for Master of Science degree in Computer Science of the University of Nairobi with my approval as the University supervisor.

Sign:  _____

Date: 31/5/12

Prof. Elijah Omwenga
Project Supervisor
School of Computing and Informatics
University of Nairobi

ACKNOWLEDGEMENTS

I wish to sincerely thank the almighty God for giving the wisdom, knowledge, courage and the will to carry out this project. It has not been all smooth but through His blessings, I have managed against all odds to come through.

I also wish to immensely thank my supervisor Prof. Elijah Omwenga whose invaluable support, challenge and guidance made all the steps worth making. Along with my supervisor were the panel members Daniel Orwa and Eric Ayienga whose contributions and encouragement made all the difference. To Andrew Mwaura, who was always there whenever I needed him, I will always be grateful. To all my colleagues at Safaricom and classmates, especially Charles Maina who was always the shove I needed, I appreciate you!

Finally I thank my lovely wife Irene and son Dylan for their encouragement and understanding during the strenuous fight to complete the project in time, sometimes at the expense of their precious time. Without their support, it would all have been worthless. Same goes to my parents John Gathia and Alice Wanjiku whose humble prayers worked miracles.

May God bless you all for your different contributions to this worthy course!

ABSTRACT

Application developers currently have a wide selection of rich application development tools to use. These tools have a common limitation in that they require some hardware, which is mostly high end, and base software to run. While these may not appear like major challenges to software development companies, startup companies and individuals, students and trainers among others may find this requirement prohibiting. In a world where most innovative ideas have come from campus rooms, it is critical to have development environments that are easily accessible and whose cost is minimal or at least dependent on the usage. Other than startup cost, developers are also faced with the nightmare of managing development platform migration every time they switch development computers or re-install the operating system. This goes hand in hand with the fact that development is limited to the single development computer that has the development tools installed. Developers are hence not able to make changes on the fly without having to access their development computers.

In this report we look at a viable architecture of a cloud based rapid application development framework. Cloud platform transforms the way computing resources are utilized avoiding the need for upfront purchase of fundamental hardware and software. Having a development environment that runs off the cloud and accessible through the browser helps address the challenges faced under the current development environments. Accessibility of such platforms as well as data security become the immediate problems that a developer is likely to be faced with. These have however been mitigated by use of rich client and asynchronous server access that reduces traffic requirements when developing on the platform. By using single sign-on for user authentication based on known and reliable authentication services, the environment addresses the security issue.

The results of the surveys conducted have clearly indicated willingness by developers to migrate to a cloud based development environment if guaranteed to have features that match locally installed development environments. This dissertation demonstrates the practicality of a cloud based application development framework through the proposed design, prototype development and user surveys.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Research Questions	3
1.4 Research Objectives	3
1.5 Significance of the study	4
1.6 Project Scope.....	5
1.7 Research Assumptions and Limitations	5
CHAPTER 2: LITERATURE REVIEW	7
2.1 Towards cloud computing.....	7
2.2 Cloud computing stacks	7
2.2.1 SaaS – Software as a Service	7
2.2.2 PaaS – Platform as a Service	8
2.2.3 IaaS – Infrastructure as a Service.....	8
2.3 Application development Trends	8
2.3.1 Rapid Application Development.....	8
2.3.2 Programming paradigms	9
2.3.3 Automating code generation and interface layouts.....	11
2.3.4 Integrated Development Environments	11
2.3.5 Interface based IDE programming.....	12
2.3.6 Template based code generators	12
2.3.7 Typical Java Integrated Development Environment.....	13
2.3.8 Sample RAD IDE Architecture	15
2.4 Cloud Based Development.....	16
2.4.1 Existing options	16
2.4.2 PaaS Architecture.....	17
2.4.3 Common PaaS Architecture Components	21
CHAPTER 3: RESEARCH METHODOLOGY	23
3.1 Research Process.....	23

3.1.1	Literature Review.....	23
3.1.2	Design and development process.....	23
3.1.3	Interviews and review meetings.....	24
3.1.4	Architectural Design and Prototype Build Increment.....	24
3.1.5	User Acceptance Testing	25
3.1.6	Data collection and analysis.....	26
3.1.7	Data collection methodology	26
3.1.8	Data Analysis	27
3.1.9	Limitations of methodology.....	28
3.2	Conceptual Framework	28
3.2.1	Base Cloud Platform as a Service (PaaS)	29
3.2.2	Database Management System	29
3.2.3	Database Application Development	29
3.2.4	Authentication Model and Data Security.....	30
3.2.5	Multi-User Support and User Space Separation	30
3.2.6	Rich User Client and Asynchronous Server Communication.....	31
3.2.7	Application Build and Deploy	31
3.2.8	Plugin Architecture	32
3.2.9	User Collaboration	32
3.2.10	Version Management	32
3.2.11	Conceptual design.....	34
3.3	System Analysis	35
3.3.1	System Overview	35
3.3.2	System Objectives / Requirements	35
3.3.3	Scope and limitations	36
3.3.4	Google technologies selection	36
3.4	Prototype Design and Development.....	36
3.4.1	Design Concepts	36
3.4.2	High level architecture	45
3.4.3	Architectural Design	45
3.4.4	Functional Testing	59

CHAPTER 4: RESULTS AND DISCUSSIONS	64
4.1 Functional Tests	64
4.2 Survey details	64
4.3 Summary of observations.....	70
CHAPTER 5: CONCLUSIONS AND FUTURE WORK.....	71
5.1 Conclusion.....	71
5.2 Contribution to the Research.....	72
5.2.1 Cloud IDE Architecture	72
5.2.2 Creating awareness of PaaS cloud stack.....	72
5.2.3 Virtual file system.....	72
5.2.4 Non-Relational Databases for Cloud Computing	73
5.2.5 Use of JSON File Exchange Formats for Asynchronous User Interface.....	73
5.3 Future Work	74
5.3.1 Hosted compilers	74
5.3.2 Cloud Data Access Components.....	74
5.3.3 Cloud based runtime environments	74
5.4 Research Questions Answered	75
CHAPTER 6: REFERENCES	77
6.1 Books and Journals.....	77
6.2 Internet references	78
CHAPTER 7: APPENDICES	81
7.1 Key Servlet Functions	81
7.1.1 Common Java Code Library (iceUtils.java)	81
7.1.2 HTTP Get / Post Methods.....	82
7.1.3 Saving a File to Virtual File System	84
7.1.4 Loading Projects using JSON objects.....	84
7.1.5 Project Download to OS File System	85
7.2 Javascript Code	87
7.2.1 IDE Code Completion.....	87
7.2.2 JavaScript File Management.....	89
7.2.3 Javascript Save Code (AJAX)	90

7.3 User Survey 91

7.3.1 Survey Summary..... 91

7.3.2 Survey Responses Summary..... 92

LIST OF FIGURES

Figure 3.1.1: Sample cloud representation from VMware (Adapted from Davis, 2008)	7
Figure 3.2.1: Cloud stacks (adapted from Sait, 2011)	8
Figure 3.3.1: Typical java development environment (Adapted from Sonu, 2009).....	14
Figure 3.3.2: Inside the Eclipse SDK (Adapted from Eclipsepluginsite.com, 2008)	15
Figure 3.4.1: Aurora SDK Architecture (Adapted from Ou, Najaran and Rouf, 2007)	17
Figure 3.4.2: PaaS Layers	18
Figure 3.4.3: OrangeScape runtime architecture (Adapted from OrangeScape, 2010).....	19
Figure 3.4.4: Wolf platform architecture (Adapted from Wolf, 2011).....	20
Figure 3.4.5: Windows AppFabric services (Adapted from Raz, 2011)	20
Figure 3.4.6: Windows Azure services platform (Adapted from Iyogi, 2011).....	21
Figure 3.4.7: Comparison of two cloud architectures (Windows and Amazon) (Adapted from Kaefer, 2009)	22
Figure 4.3.1: Scrum Development methodology.....	24
Figure 5.10.1: Cloud IDE Conceptual Design	34
Figure 6.1.1: Simplified view of Eclipse IDE (Adapted from Eclipse, 2010).....	37
Figure 6.5.1: Apps Datastore Entity view.....	41
Figure 6.6.1: JSON object (JSON, 2010)	42
Figure 6.6.2: JSON array (JSON, 2010).....	42
Figure 6.6.3: JSON value (JSON, 2010).....	43
Figure 6.6.4: JSON String (JSON, 2010)	43
Figure 6.6.5: JSON Number (JSON, 2010)	44
Figure 7.4.1: High level prototype structure.....	45
Figure 7.5.1: Prototype architectural design	46
Figure 7.5.2: Prototype user interface.....	47
Figure 7.5.3: Development mode user authentication	47
Figure 7.5.4: Project tree.....	48
Figure 7.5.5: Dynamically generated JSON project tree object	50
Figure 7.5.6: Prototype (Icecloud) Editor	53
Figure 7.5.7: Prototype database architecture.....	54
Figure 7.5.8: File types JSON object.....	55

GLOSSARY

Acronyms

Acronym	Meaning
IDE	Integrated Development Environment
PaaS	Platform as a service
SaaS	Software as a service
IaaS	Infrastructure as a service
RAD	Rapid Application Development
REST	Representational State Transfer
JSP	Java Server Pages
JSTL	JSP Standard Tag Library
AJAX	Asynchronous Java and XML
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
RDBMS	Relational Database Management System
JSON	Java Script Object Notation
API	Application Programming Interface
CRUD	Create Retrieve Update Delete – These are basic database tasks expected from a database management system (DBMS)

Proprietary Terms and Technologies

Any proprietary terms and technologies used in this report belong to the trademark owners even where no bibliography reference has been made. Key terms used throughout this report include:

Term / Technology	Description
Google	Google is a cooperate company that offers a myriad of computer technologies, some of which have been referenced in this report. These include Google AppEngine, Google Accounts, Google Datastore and Google web toolkit.
Microsoft	Microsoft is a cooperate company that offers a myriad of computer technologies, some of which have been referenced in this report. These include Windows Azure, SQL Azure, Appfabric and windows operating system
Amazon	Started off as an online shopping solution, the company has grown to be a major technology player offering various cloud solutions.
OrangeScape	OrangeScape provides Platform as a Service (PaaS) to build domain rich solutions. OrangeScape uses a modeling driven visual development environment for creating business applications and can be deployed as SaaS or on-premise applications.
Wolf	WOLF is an Online Database Application Platform architected to help design, deliver and use Software-as-a-Service (SaaS) database applications, using only a web browser.
VMware	A company providing virtualization software that's at the heart of cloud offerings
Eclipse	Open source application development environment from Eclipse foundation
Java	Programming language originally developed by Sun Microsystems
DOJO	An open source Javascript library that delivers powerful performance, and scales with development process. It's the toolkit developers turn to for building superior desktop and mobile web experiences
Code Mirror	An open source code-editor component that can be embedded in Web pages. It provides <i>only</i> the editor component but with a rich API that allow for feature extension to accomplish common IDE tasks.

CHAPTER 1: INTRODUCTION

1.1 Background

Many software developers today are still using the old way of "write your own code" strategy. Although it may seem right for a developer's perspective in terms of skill enhancement and originality, when it comes to developing large-scale applications on a tight deadline, this may not seem to be the best approach. Right now, many software development companies are adopting "Framework-Driven Development". This is a strategy of using pre-coded "frameworks" to speed up the development of a certain application (Nacion, 2009).

An object-oriented framework is a reusable design together with an implementation. The design represents a model of an application domain or a pertinent aspect thereof, and the implementation defines how this model can be executed, at least partially. A good framework's design and implementation is the result of a deep understanding of the application domain, usually gained by developing several applications for that domain. The framework represents the cumulated experience of how the software architecture and its implementation for most applications in the domain should look like. It leaves enough room for customization to solve a particular problem in the application domain.

Developers who apply a framework reuse its design and implementation. They do so to solve an application problem that falls into the domain modeled by the framework. By reusing the design, application developers customize (hopefully) well understood software architecture to their own specific application problem. This helps them get the key aspects of the architecture right from the beginning. By reusing the implementation, application developers get up to speed more quickly. Through design and code reuse, frameworks help developers achieve higher productivity and shorter time-to-market in application development (Riehle, 2000).

Software development frameworks play a pivotal role in the move to cloud computing. These are presented in the platform as a service (PaaS) constituent of cloud computing spectrum in which hosting providers allow users to build custom applications out of component services located in their data centers. The other two are software as a service (SaaS) whereby whole

applications are delivered over the Internet, and infrastructure as a service (IaaS) whereby raw computing resources such as processing and storage are delivered online.

A recent Gartner survey found that 95% of organizations intend to maintain or grow their SaaS usage in the coming year. But the same is not yet true for the other two constituents, IaaS and PaaS (Swabey P. 2010).

One reason for the disparity in the adoption of the various flavors of cloud may be their different switching costs. For end-users, SaaS offerings are easy to adopt, as they typically employ user interfaces that are similar in style to consumer websites. But for software developers, the target audience of IaaS and PaaS offerings, working with the cloud can require a different set of skills and know-how, especially for those used to building traditional applications on client-server platforms.

This explains why software development frameworks – toolkits that allow programmers to be more productive by simplifying the development process – may well prove to be pivotal in the cloud era.

1.2 Problem Statement

Rapid application development (RAD) has been defined as a methodology that enables organizations to develop strategically important systems faster while reducing development costs and maintaining quality. This is achieved by using a series of proven application development techniques, within a well-defined methodology. (CaseMaker, 2000) This is becoming even more real with the rise of cloud computing and specifically PaaS stack.

A true RAD environment should enable application development without having to worry about non value adding pre-requisites into the actual product delivery. This is what PaaS is providing to the industry where one just needs to invest in the development skills that support the development and with minimum investment, they are ready to deliver the solution. Currently, even developers for cloud services (SaaS) rely heavily on on-premise development tools. This is changing fast to cloud development and this creates an urgent need to migrate to cloud based application development frameworks.

Cloud development also invites more people into the development arena with the minimum capital requirements to start off. This creates a need for well-defined application development methodology that enables the developers to carry out their tasks seamlessly while complying with the standards of cloud computing. This is where cloud based integrated development environments and software development frameworks come in handy.

Current cloud development environments offer very little to the traditional developer compared to the desktop based environments. Web based IDEs lack the runtime environments while the environments with runtime environments do not offer real development experience and are best suited for super users and business oriented professionals. Examples of runtime development environments are OrangeScape and Wolf.

1.3 Research Questions

- 1 What are the trends supporting rapid application development in cloud computing?
- 2 What are the major challenges facing the adoption of cloud application development and PaaS in general and what are the solutions?
- 3 How can framework based development increase the uptake of cloud application development?
- 4 Can cloud (web) environments support fully fledged IDEs and framework based application development for programmers?

1.4 Research Objectives

The objectives of this study are as follows:

1. Review how cloud computing trends affect current application development methodologies
2. Review how *Framework Based Application Development* methodology improves the uptake of PaaS
3. Analyze the impact of framework based cloud development on organizations and software developers
4. Implement an architecture to support framework based cloud development

1.5 Significance of the study

The study was aimed at coming up with an architecture for a rapid application development framework on the cloud. This was achieved by taking a closer look at framework based development in the traditional on premise development as well as the current trends in cloud computing and cloud based development. The research targeted application developers who are expected to use their inherent programming and software development skills to develop SaaS applications on the cloud.

From this study, one is able to visualize the feasibility, requirements and methodology of implementing a cloud based application development framework. The key benefits in cloud based development environment proposed here are:

- Zero foot print development environment: Developers will no longer need to invest in hardware and pre-requisite software to start application development. One can easily use community infrastructure e.g. cyber cafes, school or college computer centers etc., to start application development.
- Best in breed infrastructure at usage cost: Most cloud environments charge users based on their usage of the cloud resources. With the cloud providers availing reliable security, uptime, bandwidth and antivirus protection, developers get the best of all these and only pay for what they use.
- Round the globe availability: Online applications are available independent of location as long the internet is available. This means developers don't have to keep travelling with their development computers and will be able to access their work from anywhere as long as they can access the internet. With advancement in mobile technology, this could as well be their mobile phones or other data enabled mobile devices.
- Seamless migration: The proposed architecture supports code porting between cloud and on premise development environments. Developers will therefore be able to download their code from the cloud, make changes on local development environments and upload the same when done back to the cloud seamlessly.
- Cloud computing adoption: With PaaS being the least accepted stack of cloud computing, this research has worked towards demystifying this particular stack. It is expected that

pioneers of PaaS cloud offering will leap huge profits and also boost the use and uptake of cloud computing in general.

- Seamless development collaboration: The proposed architecture supports development teams' collaboration which is critical in managing large development projects where more than developer is involved.
- Cost savings: All these benefits are consolidated in cost savings where companies are able to realize huge gains by not investing in expensive development hardware and software, along with ease of product development and deployment to the cloud.

1.6 Project Scope

Due to the limited time and resources, the research scope was limited to the following:

1. Analysis of various components of a development framework for cloud application development environment
2. Standard architectural design for a cloud based rapid application development framework on the PaaS layer
3. Technical design documentation for the various modules required to implement a development framework on a cloud operating system
4. A prototype of a cloud based rapid application development framework

1.7 Research Assumptions and Limitations

The research was conducted under the following assumptions and limitations:

1. The research focused on n-tier development environment but the prototype is based on a standard 2-tier web application development on the cloud.
2. Billing and resource monitoring are assumed to be provided by the core cloud hosting environment and the development environment is not required to implement a separate model for the same.
3. The study was limited to the available core cloud features at the time. All possible effort has been made to bring out similar and standard features across related development environments and platforms.

4. Google Appengine was selected for prototype development as it offered seamless integration to Eclipse development environment for Java as well as rich cloud libraries, free resources and reliable test environment which largely influenced the overall design.

CHAPTER 2: LITERATURE REVIEW

2.1 Towards cloud computing

Cloud computing refers to the hardware, systems software, and applications delivered as services over the Internet. When a cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud. The term Private Cloud is used when the cloud infrastructure is operated solely for a business or an organization. A composition of the two types (private and public) is called a Hybrid (Antonopoulos and Gillam, 2010)

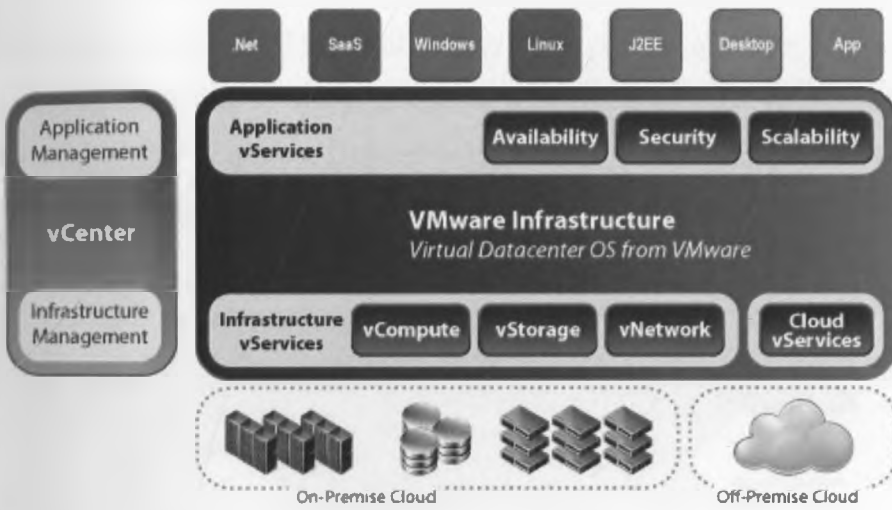


Figure 2.1.1: Sample cloud representation from VMware (Adapted from Davis, 2008)

2.2 Cloud computing stacks

2.2.1 SaaS – Software as a Service

This is essentially based on the concept of renting application functionality from a service provider rather than buying, installing and running software yourself. Offerings within this range from services include Salesforce.com at one end, delivering the equivalent of a complete application suite to players like MessageLabs at the other, whose services are designed to complement operational infrastructure, Google apps and others.

2.2.2 PaaS – Platform as a Service

This is about providing, a platform in the cloud, upon which applications can be developed and executed. Players like Google, again Salesforce.com (this time with Force.com), and Microsoft (with Azure) exist in this space. Facilities provided include things like database management, security, workflow management, application serving, and so on.

2.2.3 IaaS – Infrastructure as a Service

The proposition here is the offering of compute power and storage space on demand. The difference between this and the other two categories of cloud is that the software that executes is essentially yours. In practical terms, the model is based on the same principles of virtualization that we are all familiar with in the context of server partitioning or flexible storage. Rather than running a virtual image on a partition existing on a physical server in your data center, you spin it up on a virtual machine that you have created in the cloud. Virtual disks can be created in a similar manner, to deal with the storage side of things. (Schulz, 2009)



Figure 2.2.1: Cloud stacks (adapted from Sait, 2011)

2.3 Application development Trends

2.3.1 Rapid Application Development

Traditional lifecycles devised in the 1970s, and still widely used today, are based upon a structured step-by-step approach to developing systems. This rigid sequence of steps forces a user to “sign-off” after the completion of each specification before development can proceed to

the next step. The requirements and design are then frozen and the system is coded, tested, and implemented. With such conventional methods, there is a long delay before the customer gets to see any results and the development process can take so long that the customer's business could fundamentally change before the system is even ready for use.

Rapid application development has been made possible by adaptive and reusable programming techniques. These are best brought out by analyzing the growth of programming paradigms behind the various application developments.

2.3.2 Programming paradigms

A programming paradigm is way of conceptualizing what it means to perform computation and how tasks to be carried out on a computer should be structured and organized. The paradigms are not exclusive, but reflect the different emphasis of language designers. Most practical languages embody features of more than one paradigm. There have been four common paradigms and a new one (Aspect Oriented) as detailed below

2.3.2.1 Imperative paradigm

This is based on commands that update variables in storage. The Latin word *imperare* means "to command". The language provides statements, such as assignment statements, which explicitly change the state of the memory of the computer. This model closely matches the actual executions of computer and usually has high execution efficiency. Many people also find the imperative paradigm to be a more natural way of expressing themselves

2.3.2.2 Functional programming paradigm

In this paradigm we express computations as the evaluation of mathematical functions. Functional programming paradigms treat values as single entities. Unlike variables, values are never modified. Instead, values are transformed into new values. Computations of functional languages are performed largely through applying functions to values eg (+ 4 5).

2.3.2.3 Logic programming paradigm

In this paradigm we express computation in exclusively in terms of mathematical logic. While the functional paradigm emphasizes the idea of a mathematical function, the logic paradigm focuses on predicate logic, in which the basic concept is a relation.

Logic languages are useful for expressing problems where it is not obvious what the functions should be. For example consider the uncle relationship: a given person can have many uncles, and another person can be uncle to many nieces and nephews.

2.3.2.4 The Object-Oriented Paradigm

Object oriented programming paradigm is not just a few new features added to a programming language, but it a new way of thinking about the process of decomposing problems and developing programming solutions. Alan Kay characterized the fundamental of OOP as follows:

- Everything is modeled as object
- Computation is performed by message passing: objects communicate with one another via message passing.
- Every object is an instance of a class where a class represents a grouping of similar objects.
- Inheritance: defines the relationships between classes.

The Object Oriented paradigm focuses on the objects that a program is representing, and on allowing them to exhibit "behavior". Unlike imperative paradigm, where data are passive and procedures are active, in the O-O paradigm data is combined with procedures to give objects, which are thereby rendered active. (Bellaachia, 2010)

2.3.2.5 The Aspect-Oriented Paradigm

Although the object-oriented paradigm provides a rich set of tools for abstraction and modularization, it cannot address the problem with so called cross-cutting concerns. One can think the cross-cutting concerns as functionality that when implemented, will scatter around the final product in different components. Since this kind of functionality will cut through the basic functionality of the system, it is hard to model even with the object oriented programming. Good examples of the cross-cutting concerns are authorization, synchronization, error handling and transaction management. Aspect-oriented programming tries to address the problem by modularizing the crosscutting functionality into more manageable modules – aspects. Unlike the object-oriented programming, aspect-oriented programming does not replace previous programming paradigms. Therefore it can be seen as a complementary to the object-oriented paradigm rather than a replacement.

In aspect-oriented programming the system is divided into a two halves: the base program and the aspect program. The base program will contain the main functionality of the system and can be implemented using object-oriented programming. The aspect program will consist of the cross-cutting functionality that has been modularized away from the base program. This leads to a more concise structure since the functionality of the cross-cutting concerns is contained within well defined modules (Laukkanen, 2008).

2.3.3 Automating code generation and interface layouts

Object oriented development and introduction of code components have made a developers work a lot much easier and organized. Components enable very well structured object encapsulation with most of the components integrating to the development environments for visual manipulation of properties and event based code framework generation.

Automated code generation and interface layouts call for use of predefined templates that enable dynamic content replacement within the templates to create dynamic objects and code. Most modern IDEs with visual interface design have this feature as a key strength in speeding up application development.

In this section we look at the developments towards automated code generation and interface layouts as the key principle towards rapid business application development.

2.3.4 Integrated Development Environments

An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a compiler, a debugger, and a graphical user interface (GUI) builder. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications. The BASIC programming language, for example, can be used within Microsoft Office applications, which makes it possible to write a WordBasic program within the Microsoft Word application. IDEs provide a user-friendly framework for many modern programming languages, such as Visual Basic, Java, and PowerBuilder. IDEs for developing HTML applications are among the most commonly used. For example, many people designing Web sites today use an IDE (such as

HomeSite, DreamWeaver, or FrontPage) for Web site development that automates many of the tasks involved (Furey A. and Pottjewijd A. 2001).

Most of the current IDEs support multiple programming paradigms to give the developers flexibility while others enforce use of one paradigm, e.g. Java IDEs enforce use of Object Oriented Programming. Delphi and Visual Studio allow use of several programming paradigms. Most IDEs ship with basic wizards for interface and code generation. What has been missing is the information necessary to enable developers to automate the development tasks they do by creating their own wizards, underlying code templates and functional objects.

2.3.5 Interface based IDE programming

In both component and service oriented development, the design of the interfaces is done such that a software entity implements and exposes a key part of its definition. Therefore, the notion and concept of “interface” is key to successful design in both component-based and service-oriented systems. An interface defines a set of public method signatures, logically grouped but providing no implementation. It defines a contract between the requestor and provider of a service. Any implementation of an interface must provide all methods. (Brown, Johnston and Kelly, 2004)

Researchers in the area of automated design of user interfaces have shown that layout of an interface can, in many cases be generated from the applications data model using an intelligent program that applies design rules. The specification of interface behavior however has not been automated in the same manner and is mostly a programmatic task. Mecano is a model based user-interface development environment that extends the notion of automating interface design from data models (Puerta, et al, 1994)

2.3.6 Template based code generators

Template code generators rely on predefined templates that allow for dynamic string replacements to generate code based on some configurations. The aim of templates is to set the coding standard for all repetitive code snippets and leave out variables that will be automatically filled in based on the wizard actions. Just like the object oriented programming paradigm where

developers use objects to organize and reuse their code, wizard based code generators help create coding standards and save developers from repeating similar tasks. More advanced code generators provide wrappers to database objects allowing for a seamless integration to databases for full-fledged applications. Good examples of code generators in the market are Codebreeze, CodeSmith and T-sharp all of which are commercial packages for .Net languages. Just like object components, developers can choose to get the packages or develop their own tools and integrate into the IDEs for a seamless development experience.

In this study, we focus on developing code generators and wizards as a programming methodology. Just like aspect oriented programming tries to separate code duplication for code that cuts across objects, wizard based programming separates the code that gets the objects and makes them functional and packages it on to a template. This ensures full code reuse and flexibility in programming tasks. We use Delphi IDE and language to explore what has been done to enable this kind of development.

2.3.7 Typical Java Integrated Development Environment

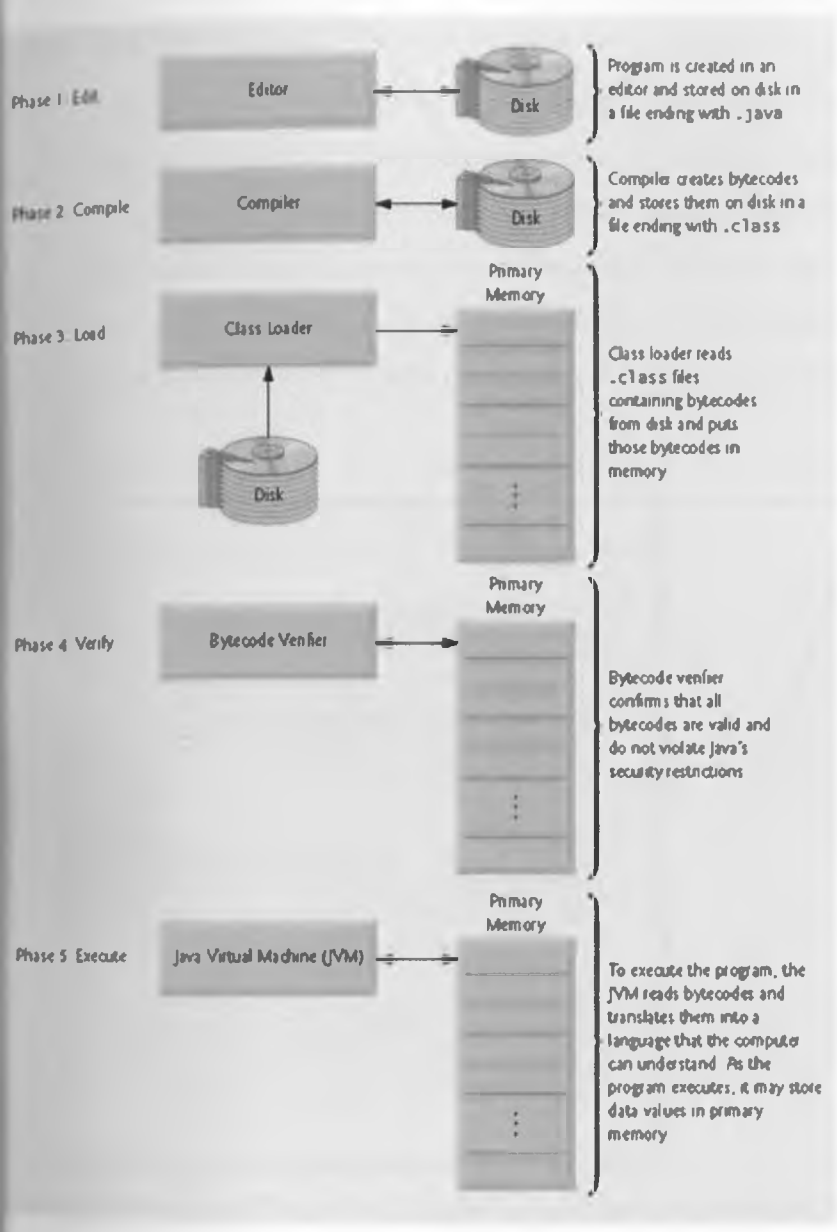


Figure 2.3.1: Typical java development environment (Adapted from Sonu, 2009)

A typical integrated development environment enables developers to create a program, compile the program, load into memory and execute. An application development framework sits on the top layer of program creation making it easier for developers to create code and interfaces in a simpler and more structured manner.

2.3.8 Sample RAD IDE Architecture

1) Eclipse is an open platform. It is designed to be easily and infinitely extensible by third parties. At the core is the eclipse SDK, we can build various products/tools around this SDK. These products or tools can further be extended by other products/tools and so on. For example, we can extend simple text editor to create xml editor. Eclipse architecture is truly amazing when it comes to extensibility. This extensibility is achieved by creating these products/tools in form of plug-ins (Eclipsepluginsite.com, 2008)

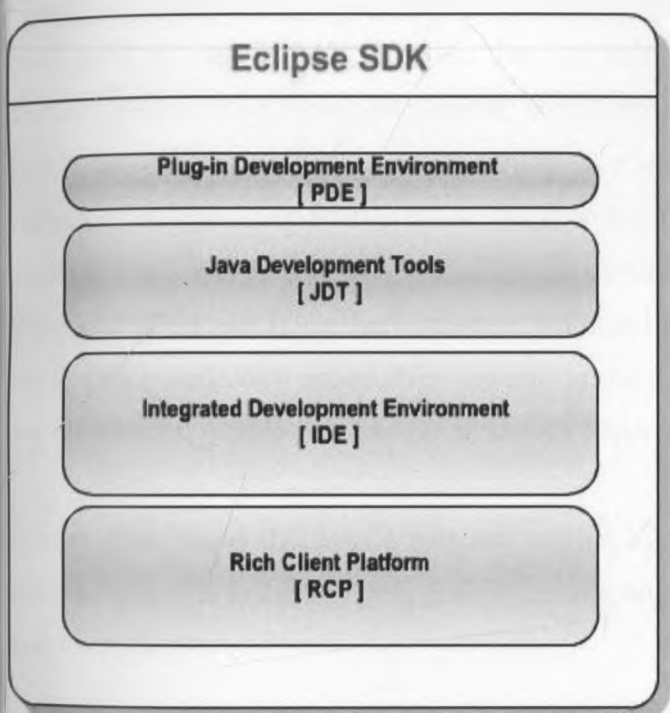


Figure 2.3.2: Inside the Eclipse SDK (Adapted from Eclipsepluginsite.com, 2008)

RCP: On the bottom is RCP which provides the architecture and framework to build any rich client application.

IDE: It is a tools platform and a rich client application itself. We can build various form of tooling by using IDE for example Database tooling.

JDT: It is a complete java IDE and a platform in itself.

PDE: It provides all tools necessary to develop plug-ins and RCP applications. This is what we will concentrate on the course of this tutorial.

2.4 Cloud Based Development

2.4.1 Existing options

The startling growing software sizes and hardware consumption (e.g. memory and CPU) of IDEs as well as their plug-ins have gradually become a headache. Moreover, programmers have to ensure that their favorite IDEs and development toolkits (e.g. JDK) are installed and properly configured in their computers before they are able to start working, which takes a substantial amount of time. Even running a properly configured IDE takes a long time to load. It seems like that IDEs will add troubles to programmers these days instead of aiding them. Therefore, some interesting solutions are emerging and can keep the hundreds of megabytes away from disk. (Ou, Najaran and Rouf, 2007). These solutions are coming in the form of cloud based IDEs, with the true cloud IDEs (not IDEs that connect to the cloud from traditional development environments) being web based. Web based IDEs are still in the growth phase and slowly gaining acceptance for commercial production. Examples include PhpAnywhere, CodeRun, Aptana and Bepin.

Aurora Web based IDE was a research project by a team of students from the University of British Columbia in 2007. The high level architecture diagram was as shown below (Ou, Najaran and Rouf, 2007).

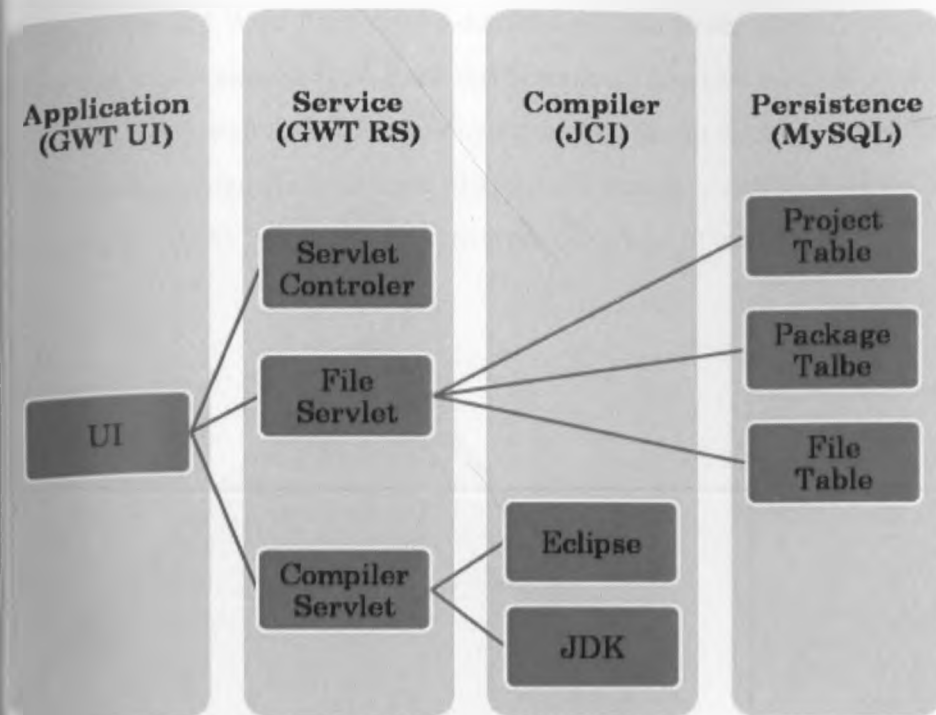


Figure 2.4.1: Aurora SDK Architecture (Adapted from Ou, Najaran and Rouf, 2007)

Other framework based IDEs have been optimized for cloud development and deployment through well-defined software frameworks. Spring Framework for Java is making cloud development an easier process for Java developers. Cloud Foundry is the open platform as a service project initiated by VMware. It can support multiple frameworks, multiple cloud providers, and multiple application services all on a cloud scale platform. (VMware, 2011). Spring framework is still available as a legacy software package installed on desktops.

2.4.2 PaaS Architecture

2.4.2.1 PaaS Stack Overview

PaaS contains 2 layers: Cloud OS and Cloud Middleware. Google App Engine and Azure are examples of Cloud OS. OrangeScope and Wolf are examples of cloud middleware or in Gartner's terms aPaaS i.e. Application Platform as a Service. In essence, aPaaS is a stack that runs on top of cloud OS.

OrangeScape and Wolf PaaS are on-demand browser based platforms for rapidly designing and delivering multi-tenant Cloud SaaS applications. They can be used as a 5 GL platform which automates many standard software development tasks to simplify application development. The unique selling proposition of these platforms is that they can be used by business analysts (non developers) as well. You don't need to write one piece of code!

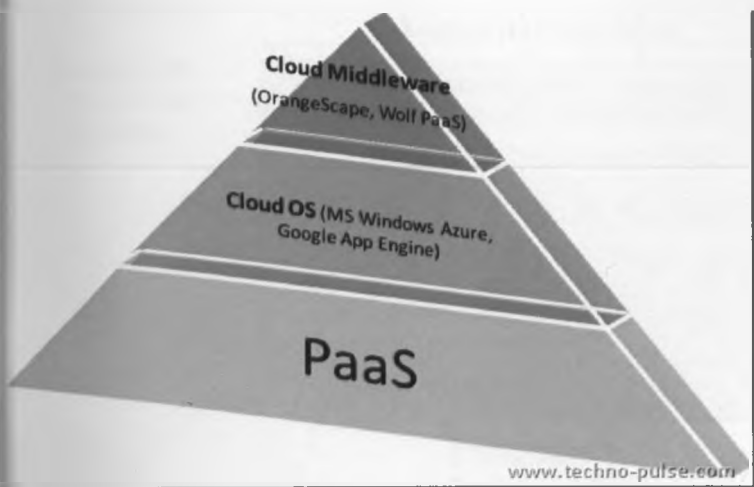


Figure 2.4.2: PaaS Layers

2.4.2.2 Sample aPaaS Architecture

OrangeScape run-time platform has pre-integrated four-tier architecture on SOA foundation.

Key components of the runtime environment are:

Component Service: The core the application around which the other services operate and exposes application models as business components.

Workflow service: Takes care of activity assignment and automatically provides queue configuration for each activity in the process design.

Persistence service: Automatically persists the data into either RDBMS (MySQL, SQL Server, Oracle, Ingres or DB2) or even NoSQL database like Google's Big Table.

Web service: Each data model entity is exposed as a REST style web service with 5 default invocation methods; one each of CRUD – Create, Read, Update & Delete and Submit for Workflow State transition.

Presentation: The front-end application used by end-users is an AJAX application running on the browser interacting with the run-time using the REST style web services.

The following diagram depicts a simplified high level architecture of the runtime platform (OrangeScape, 2010).

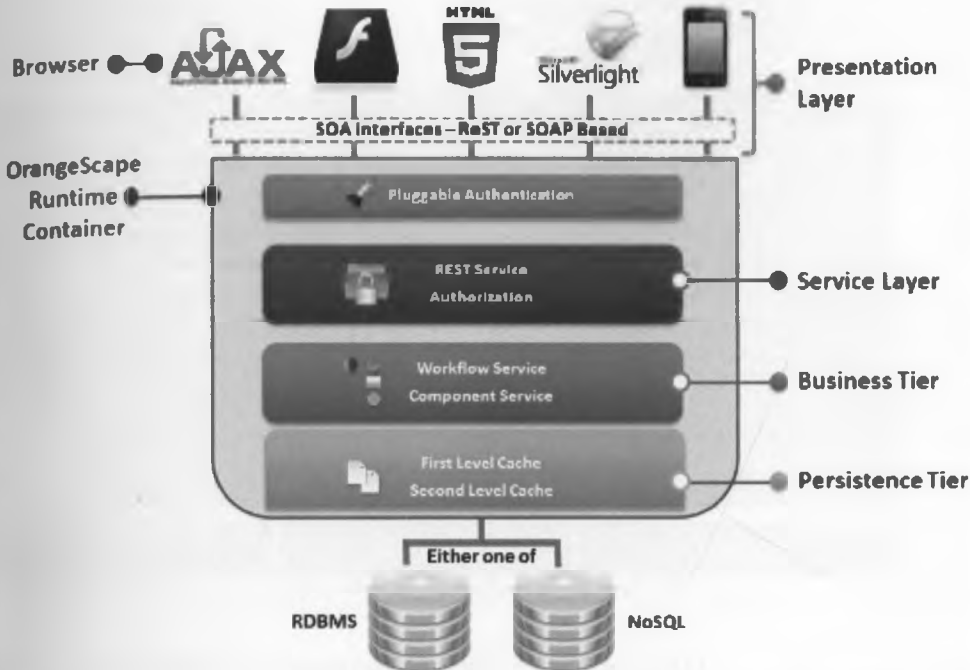


Figure 2.4.3: OrangeScape runtime architecture (Adapted from OrangeScape, 2010)

WOLF is a code-free Cloud Computing Platform which serves as a complete business and technology framework to manage the design and delivery of your SaaS applications.

SaaS applications developed on the platform are endowed with a single instance multi-tenant architecture. They are instantly integrated into an automated provisioning system which handles the allocation of network resources, database creation and account information. The robust User-Role management system makes it easy to create users & roles and assign rights to various modules of your application (WOLF 2011)



Figure 2.4.4: Wolf platform architecture (Adapted from Wolf, 2011)

Windows Azure AppFabric provides pre-built, higher level middleware services that raise the level of abstraction and reduce complexity of cloud development. These services are open and interoperable across languages (.NET, Java, Ruby, PHP...) and give developers a powerful pre-built "class library" for next-gen cloud applications. Developers can use each of the services stand-alone, or combine services to provide a composite solution (Microsoft 2011).



Figure 2.4.5: Windows AppFabric services (Adapted from Raz, 2011)

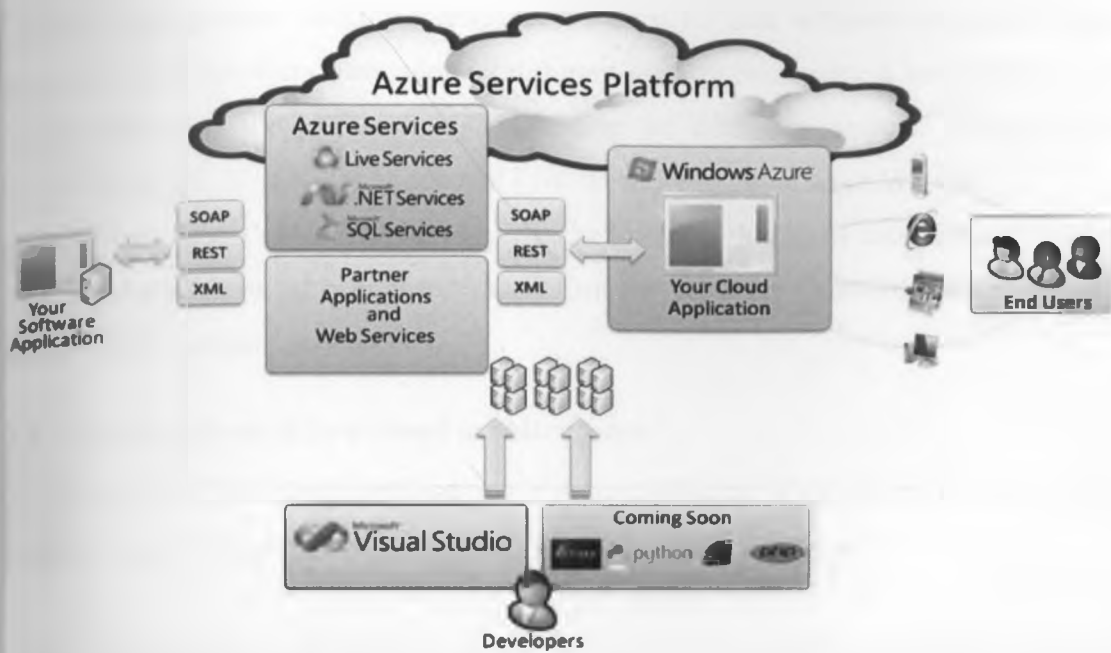


Figure 2.4.6: Windows Azure services platform (Adapted from Iyogi, 2011)

2.4.3 Common PaaS Architecture Components

Based on the dominant players in the current cloud PaaS, it becomes clear that some components are common among the PaaS implementations.

Operating system: This forms the base of other software components. It acts as the link between the hardware compute components and other software components required to offer cloud services. Examples are Windows Azure and Amazon EC2.

Data Storage: Cloud environments just like their on-premise counterparts require persisting data for computational requirements. Cloud environments implement both file-system based and database structures for data persistence. Data storage based on databases is implemented as standard relational databases, binary data storage (BLOB) as well as unstructured storage which is currently being preferred for quick access and horizontal scaling among the cloud providers. Examples under this section include SQL Azure, Blob storage, Table service from Microsoft and Relational Database service, S3 and Simple DB from Amazon for implementation of relational, blob and unstructured storage respectively.

Messaging: Messaging services are used for inter-process communication among the various components of the PaaS layer. Main messaging methods used are message queues and message notification services.

Identity Management: Identity management is used for user authentication and authorization. Different cloud providers have adopted different identity management methods, with common identity becoming the preferred method among the dominant players. Google has adopted Google Accounts; Microsoft has adopted Live ID while Amazon has AWS identity.

Content Delivery: Content delivery layer is the top layer that hosts the end user applications as well as client frontend. Microsoft uses Content Delivery Network while Amazon uses CloudFront for content serving.

2.4.3.1 Comparison of two cloud architectures

A comparison of two cloud architectures is shown below to bring out the common components and how each of the architecture compares for the various components.

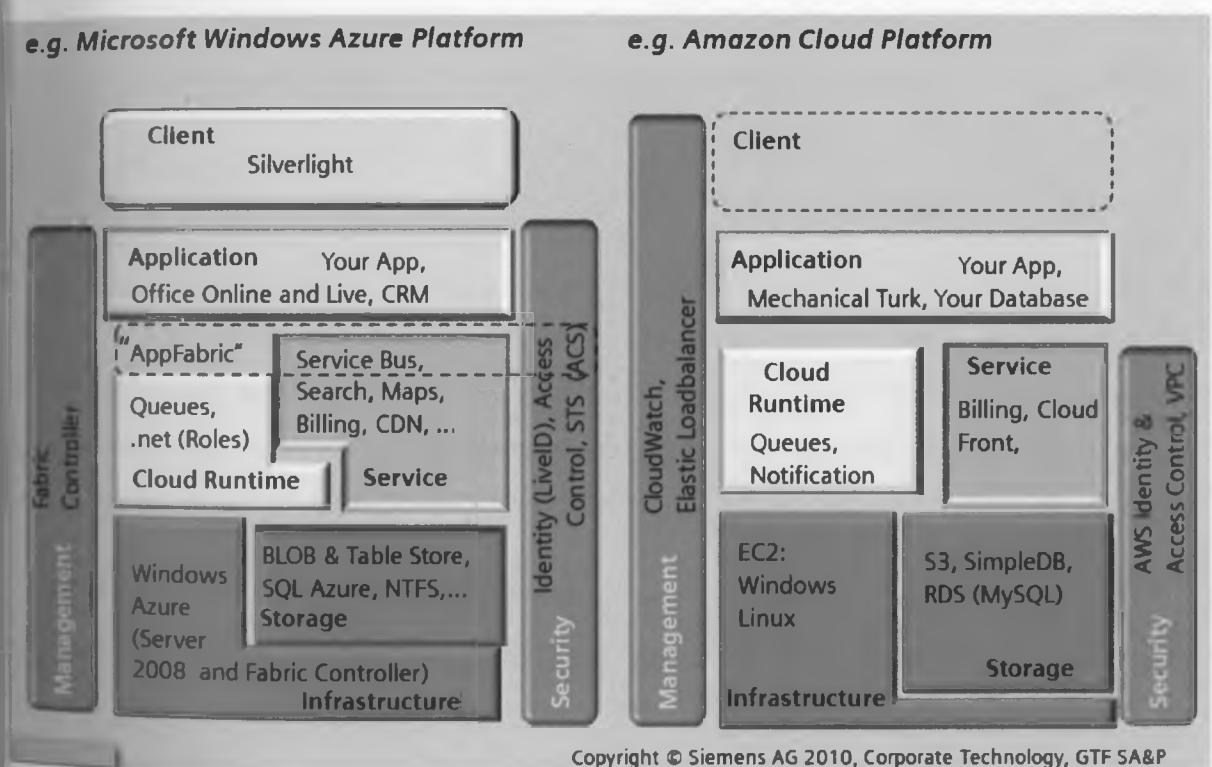


Figure 2.4.7: Comparison of two cloud architectures (Windows and Amazon) (Adapted from Kaefer, 2009)

CHAPTER 3: RESEARCH METHODOLOGY

This section outlines the activities that were carried out throughout the duration of the research.

3.1 Research Process

During the duration of the research, various existing IDEs architectures were explored to review their feature sets. More emphasis was put on features supporting basic file manipulation, program editing, program compiling, rapid application development and framework based development. The feature sets were mapped to a cloud environment to enable design of a cloud based RAD framework

The prototype development followed standard software development lifecycle by iterating analysis, design, development, testing until the final product was realized.

3.1.1 Literature Review

This involved further study in the work done with regard to achieving rapid software development and code reuse, with specific focus on *framework based development*. This was reviewed with the cloud computing and development in mind to bring out the power of framework based development for the cloud environment. The review also looked into the architectures imposed by cloud developments and the basics that support cloud development as well as obstacles against it that need to be overcome.

3.1.2 Design and development process

This research process entailed frequent changes within the architectural design of the system, thus an agile method of software development was used. This allowed for necessary changes arising during the development process to be incorporated in the next iteration or development phase. SCRUM method of project management was used to ensure shorter intervals, called sprints, during which objectives for the next design and development phase were set and reviewed before the next iteration. Due to the time constraints, each sprint lasted for a week for the entire duration of the project.

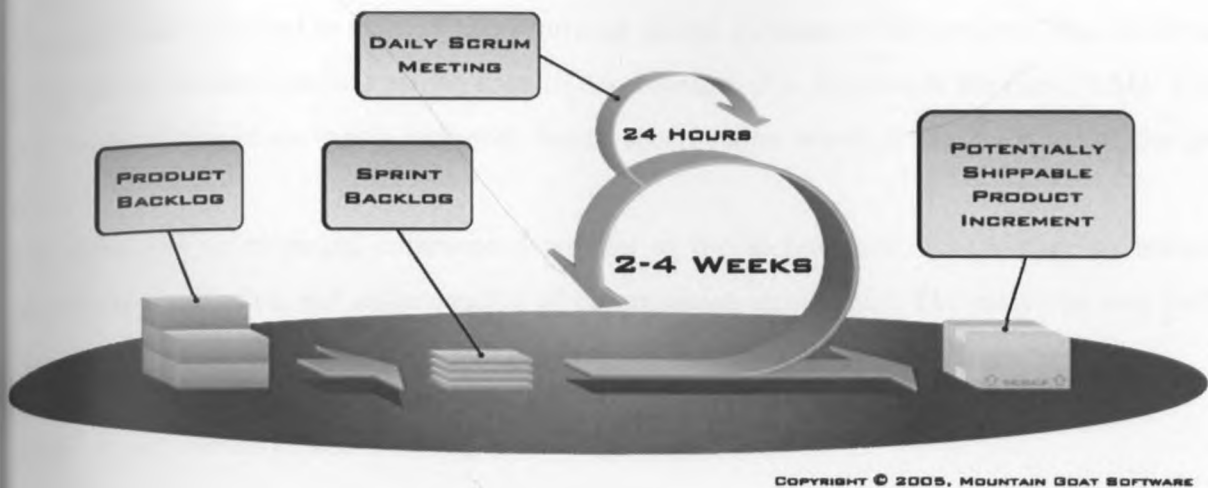


Figure 3.1.1: Scrum Development methodology

3.1.3 Interviews and review meetings

To ensure the critical feature sets of a RAD environment were captured in the design, a series of interviews were held involving development teams. This primarily focused on known development teams from selected companies, who were involved in enterprise and customer systems development. The interviews were based on the stage of design and prototype development and lasted at most one hour. The process was documented and views used to guide the feature sets to incorporate in the design stage.

3.1.4 Architectural Design and Prototype Build Increment

This research includes prototype development. By definition, the implementation aspect of the prototype covers the range of prototyping the complete product (or system) to prototyping part of, or a sub-assembly or a component of the product. The complete prototype, as its name suggests, models most, if not all, the characteristics of the product. It is usually implemented full-scale as well as being fully functional. One example of such prototype is one that is given to a group of carefully selected people with special interest, often called a focus group, to examine and identify outstanding problems before the product is committed to its final design. On the other hand, there are prototypes that are needed to study or investigate special problems associated with one component, sub-assemblies or simply a particular concept of the product that requires close attention (Chua, 2003).

The prototype was used to identify more intricate design elements of the project. This involved design of an architecture that can be used in development of a framework for cloud RAD. The end result of this phase was a technical design specification which is presented in the design section.

The prototype development commenced parallel to the architectural design stage to ensure proper communication and understanding of the proposed architecture. The prototype was built incrementally with each build undergoing testing for validation against the design.

3.1.5 User Acceptance Testing

User acceptance testing (UAT) is a critical phase in any system development and implementation. This research used UATs mapped out of the system requirements to ensure critical items of an IDE are delivered in the prototype. Key functions subjected to the UAT are:

- User management
 - User registration
 - User login
 - Multi user projects management
- Project management:
 - Create project
 - Manage files in a project
 - Inline syntax check (code parsing)
 - Deploy to the cloud – static files only binary compilation not supported
 - Download to local computer
- File management
 - Create source code files
 - Assign files to project
 - Manage different file types based on selected programming languages
 - Delete files
 - Move files across projects
 - Download to local computer
- Code management
 - Template based code generation

- Syntax highlighting
- Code completion
- Inline syntax parsing (compilation)
- User interface
 - Natural look and feel (desktop based interface)
 - Asynchronous file management
- Accessibility
 - Cloud based interface
 - Local client side functions

3.1.6 Data collection and analysis

This involved collection of data on software development using the traditional development methods and cloud based application development. The analysis was based on a parameterized model and user survey covering the following basic inputs:

- 1 Willingness of developers to use cloud based development environments as the preferred method of application development.
- 2 Feature requirements in a cloud based application development environment
- 3 Challenges to the adoption of cloud based application development environments
- 4 Cost of development environment setup for traditional as well as cloud based development environments.
- 5 Development effort – This took into consideration the man-hours required in the two setups both in the long run and short run with regard to the application development team
- 6 Ease of product deployment and supportability with focus on SaaS as the end product.

3.1.7 Data collection methodology

Qualitative methods: grounded in the assumption that individuals construct social reality in the form of meanings and interpretations, and that these constructions tend to be transitory and situational. Qualitative research typically involves qualitative data, i.e., data obtained through methods such interviews, on-site observations, and focus groups that is in narrative rather than numerical form.

Quantitative methods: Quantitative inquiries use numerical and statistical processes to answer specific questions. Statistics are used in a variety of ways to support inquiry or program assessment/evaluation. Descriptive statistics are numbers used to describe a group of items. Inferential statistics are computed from a sample drawn from a larger population with the intention of making generalizations from the sample about the whole population. The accuracy of inferences drawn from a sample is critically affected by the sampling procedures used. (Wholey, Hatry and Newcomer 2004, Cited in InSites 2007)

During the duration of this research process, both methods were used to analyze the results of interviews with the various development teams. This was done inform of a web survey with specific questions as detailed below:

1. Do you use any Integrated Development Environment (IDE) for Rapid Application Development (RAD)?
2. Do you use any software frameworks for enterprise development?
3. Are there any situations that call for traditional application development methodologies (non RAD)?
4. Would you consider using a cloud based development environment?
5. What would be your concerns in using cloud based development environment?
6. What features of an IDE would you want to see in a cloud based IDE?
7. What software frameworks would you want implemented in the cloud based IDE?
8. Do you have any plans to develop a cloud based application (SaaS)?
9. What current tools would you consider for cloud application development?
10. Given similar features on the cloud development platform as your computer based development environment, would you switch fully to a cloud based development environment?

3.1.8 Data Analysis

The survey was based on a targeted audience group, based on application development experience. Social media, email messaging as well as phone calls will be used to reach out to potential participants for the survey. This was to ensure a wide coverage of the responses. All the

users who participated in this research have been in software development for at least three years with diverse skills in development platforms and programming languages. Notable companies where developers were involved include Safaricom, Telkom Kenya (Orange), Kenya Tea Development Agency (KTDA) among others.

Survey monkey was used for this as it offers a trial survey with usable limitations. The actual survey was hosted on <http://www.surveymonkey.com/s/57H5NP3> and remained open for the duration of this research. Detailed research questions, responses and analysis are provided in chapter 4 of this report.

The prototype was subjected to a cycle of tests in a user acceptance testing phase to judge the success of the prototype in bringing out the cloud IDE architecture. This focused on functional requirements and how they are met in the resulting prototype.

3.1.9 Limitations of methodology

The selected methodology focused on current development trends, with specific interest on known development teams from various local companies in Kenya as well as developers on social media, www.facebook.com. The teams were used to review and give suggestions but were not involved in the development process, hence may have exposed a different opinion before and after the prototype development process, depending on their expectations.

The views of the development teams were also largely limited to the work done in this study and not in comparison to any similar more mature and commercial initiatives which could be happening in parallel. This may have the results of data analysis not indicating the reality if it was presented from a commercial software development house.

3.2 Conceptual Framework

This section discusses relevant concepts used in the course of designing the architecture of a cloud based IDE, and which directly dictates the core functionality offered.

3.2.1 Base Cloud Platform as a Service (PaaS)

Platform as a service (PaaS) layer of the cloud architecture is divided into two main areas, cloud operating system and cloud middleware. Cloud Operating System offers the core interface to the underlying infrastructure and compute resources in a transparent manner. These include memory management, disk operations and process management. Examples of base PaaS include Google App Engine, Windows Azure and Amazon Elastic Compute Cloud (EC2).

In addition to provision of basic operating system functions, the base PaaS need to have low level middleware components available to make it feasible to run an IDE on top of the operating system. These include a database management system, user authentication model, file management and required application servers, eg. Java Virtual Machine and servlet containers. These need to have been already provisioned and accessible to the developers of the IDE as well as the end users of the IDE depending on the features exposed by the Cloud IDE.

3.2.2 Database Management System

A cloud based database management system (DBMS) is critical in the design of a cloud IDE. This allows for storage of user settings, project details and file contents. The DBMS need to be structured such that it supports fast CRUD (Create, Retrieve, Update and Delete) operations for transparent file management operations. This can help implement a virtual file management system as an option to use of physical files on disk and converted as required to assure users of portability across various cloud hosting providers for the final deployment. Examples are Google App Engine Datastore, Windows SQL Azure and MysqI for Amazon EC2.

3.2.3 Database Application Development

The development environment also needs to support database development and integration features. This will enable two and three tiered application development with database integration. The requirement assumes availability of the database management system will be provided on the cloud environment by the cloud provider and the application will only be required to use available APIs to perform common CRUD database tasks. From a cloud perspective, the application development environment is expected to factor in database API wrappers to enable

integration to different database systems and architectures. These include support for standard relational databases as well as the cloud common no-sql databases. The environment users will then be able to configure the application to connect to the target database depending on database management system availability and perform CRUD operations comparable to traditional development environments.

3.2.4 Authentication Model and Data Security

Cloud users expect high level of data and operational security. This is critical in the uptake of the cloud offerings at all levels and is a key design consideration in the Cloud IDE. Currently, various cloud PaaS providers have pre-integrated single sign on model, which requires users to use one account across multiple sites and applications. The user authentication service is maintained in a central authentication service that's readily integrated into applications developed on top of the base cloud PaaS. Examples are Google Accounts, Windows Live ID.

The base cloud PaaS then must expose the relevant Application Programming Interfaces (APIs) to enable the cloud IDE call authentication services on demand. This takes off the burden of user management from the core IDE and uses best practice approach to user account management. Private data stored in the database must be encrypted to guarantee confidentiality and data integrity.

3.2.5 Multi-User Support and User Space Separation

Multi-user access allows for common usage of the same application instance in a way that creates a feeling of private ownership on the solution. Several users or entities sharing the same application and data storage are completely separated from each other and one user has a view of any other user's data. This allows the users if IDE to manage their projects and files independently of any other user, while maintaining confidentiality and data integrity. The key to achieving this is maintaining unique user accounts and mapping user projects and files under workspaces defined per user.

3.2.6 Rich User Client and Asynchronous Server Communication

A cloud IDE runs on the web and is likely to be affected by server communication especially on slow connections. It is therefore critical to have a wide range features running on the client machines, through use of rich client applications and asynchronous server communication for server side processing. This calls for wide use of advanced JavaScript and related technologies that enable creating application logic on the client side. Core features that can be incorporated on the client side include:

- Code completion
- Syntax highlighting
- Basic edit features
- Inline syntax compilation.

Communication to the server side cannot be avoided but should be reduced to critical file and project management operations. Asynchronous JavaScript and XML (AJAX) technologies make asynchronous communication to the server side feasible. This technology enables part of the web page to communicate with the server without refreshing the entire web page, which gives the user same feeling as when using a desktop platform.

Standard features that will be handled at server side include:

- Project management
- File management
- Template file management and code generation
- Team collaboration

3.2.7 Application Build and Deploy

Any IDE is expected to provide features that allow for application verification and deployment. This may include code compilation, application build or code parsing depending on the platform and runtime architecture.

A cloud based IDE should also support deploying of the application to the cloud to run as a Software as a Service (SaaS). Another key feature is project download for the complete project,

which should happen transparently to the developer (cloud user). This would allow the developer to use a cloud IDE and migrate to a locally hosted development environment as required.

3.2.8 Plugin Architecture

Plugin architecture has become widely expected among most developers as a way of enriching the IDE with additional features. These range from basic editor support features to core development components and classes developed by third party community. A cloud based IDE design should incorporate plugin architecture for multiple development options and language support.

3.2.9 User Collaboration

User collaboration is a growing requirement among developers. Traditional IDEs supported this through team development, where each team member was able to check out and work on a project file and later check it in. Other members would be notified of the file status and would not be able to work on it at the same time. Modern collaboration features allow collaborating users to work on the same document at the same time. Google Docs has implemented this in real time, where each edit is tagged with the user editing and this is visible on the document. The implementation handles conflicts in editing. This advanced collaboration feature would be a good value add to any cloud based IDE but comes with additional load on connectivity and bandwidth utilization.

3.2.10 Version Management

A version control system (or revision control system) is a system that tracks incremental versions (or revisions) of files and, in some cases, directories over time. Of course, merely tracking the various versions of a user's (or group of users') files and directories isn't very interesting in itself. What makes a version control system useful is the fact that it allows you to explore the changes which resulted in each of those versions and facilitates the arbitrary recall of the same (Sussman, Fitzpatrick and Pilato, 2011). A cloud IDE should implement a version control system to offer developers the benefits of version control and enable more explorative development and safe collaborative editing where users do not accidentally overwrite other members' changes. Version control happens in two ways:

3.2.10.1 The lock-modify-unlock solution

Many version control systems use a lock-modify-unlock model to address the problem of many authors clobbering each other's work. In this model, the repository allows only one person to change a file at a time. This exclusivity policy is managed using locks. Harry must "lock" a file before he can begin making changes to it. If Harry has locked a file, Sally cannot also lock it, and therefore cannot make any changes to that file. All she can do is read the file and wait for Harry to finish his changes and release his lock. After Harry unlocks the file, Sally can take her turn by locking and editing the file.

3.2.10.2 The copy-modify-merge solution

In this model, each user's client contacts the project repository and creates a personal working copy. Users then work simultaneously and independently, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately, a human being is responsible for making it happen correctly through conflict resolution for any conflicting changes (Sussman, Fitzpatrick and Pilato, 2011).

3.2.11 Conceptual design

Based on the research work and prototyping experience, the below design was arrived at as a standard design that can be adopted to implement a fully-fledged cloud based integrated development environment. This is a consolidation of the components described in this section.

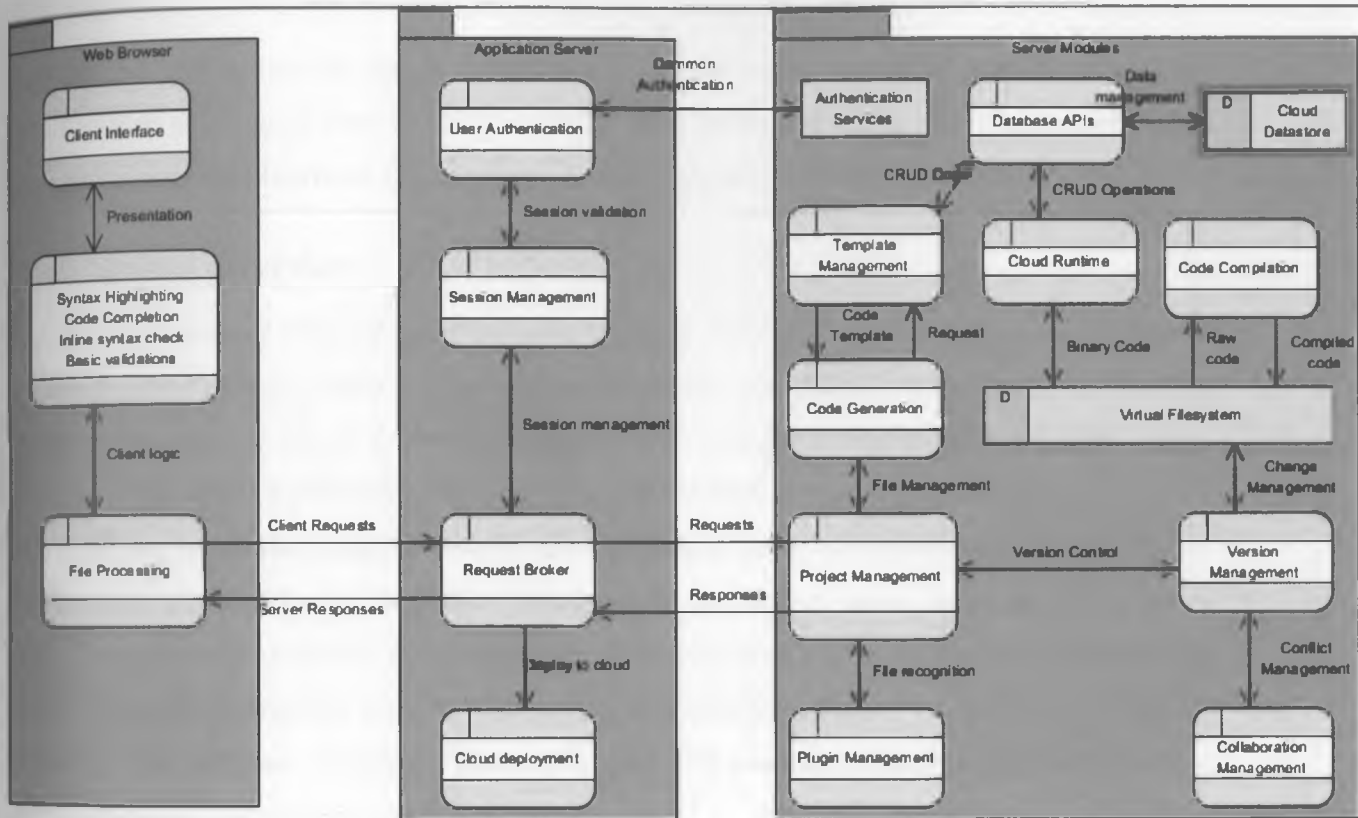


Figure 3.2.1: Cloud IDE Conceptual Design

The design is broken down into three key components representing the three logical layers implemented. These are:

- 1 Web frontend: This represents the user interface and rich client features that users have direct access to. These are implemented using client side technologies and most do not require calls to the server.
- 2 Application server: This layer processes client requests and channels the same to backend layer for fulfillment. The layer also handles overarching requirements including session management and user authentication by integrating to common authentication protocols.

- 3 The backend layer handles most intricate details of the IDE. These include virtual file management, code template management and code generation, version management, collaboration features, database integration, modules plugin and core cloud platform API integration.

3.3 System Analysis

This section will outline the system requirements that the complete product will be required to fulfill as well as a logical view of how the features will be implemented. The system (prototype) has been codenamed **Icecloud IDE** for ease of communication with the focus group.

3.3.1 System Overview

The system (*Icecloud IDE*) is a platform that provides developers with a web-based multi-user integrated development environment for applications, specifically web applications. The environment is multi-user in the sense that many users can access and use the IDE at the same time with strict space separation based on user-based workspace management module.

The platform also allows users to manage the file system that their projects use with the use of explorer like tools in the user interface. Managing the file system means being able to perform basic IO operations on the file system within a project under a user's workspace. This includes creating, renaming, deleting, copying and moving files across projects.

Once a project has been completed, users of the platform can immediately deploy their projects onto Google AppEngine to run as SaaS.

3.3.2 System Objectives / Requirements

The specific system objectives are

1. Provide a multiuser cloud based integrated development environment.
2. Provide syntax highlighting for a limited number of preconfigured programming languages
3. Provide plugin architecture for additional features and language support
4. Provide a virtual file system for projects and source code file management
5. Provide a facility for users to deploy their applications to a cloud server (Google Appspot cloud)

6. Provide user aid tools, specifically intelligent code generation

3.3.3 Scope and limitations

Icecloud IDE will primarily focus on Google App engine cloud environment with the aim of shaping the general architecture of a cloud based IDE. The concepts exposed by this research can very well be applied to other cloud environments with some modifications on the backend system. User interface components are client based and can ship with any cloud provider as long as the data and communication mechanisms are maintained.

3.3.4 Google technologies selection

Google technologies have been selected for this project due to:

1. Free resources available for testing on the cloud infrastructure. These include bandwidth, compute (CPU) and storage
2. Mature integration to legacy development tools and Java language support in Eclipse
3. No known cloud based IDE that runs on Google cloud services
4. Wide support for Google and related development tools under open source technologies
5. Reliable cloud infrastructure and a wealth of services to support this kind of project

3.4 Prototype Design and Development

This section provides the details of prototype design and development. The prototype is code named *Icecloud IDE* for identification purposes. Key details covered in this section include relevant concepts on core components used, application specifications, architectural design and development and application testing.

3.4.1 Design Concepts

The system design has used several existing technologies to create a wealth of features as dictated by the system requirements. These are explored in detail in this section.

3.4.1.1 Eclipse development environment

Eclipse is a universal, Multilanguage software development environment an open, extensible, integrated development environment (IDE). Eclipse represents one of the most exciting initiatives to come from the world of application development, and it has the support of leading

companies and organizations in the technology sector. Eclipse is gaining widespread acceptance in both commercial and academic arenas. (McAffer, Gamma and Wiegand, 2010)

The Eclipse platform itself is structured as subsystems which are implemented in one or more plug-ins. The subsystems are built on top of a small runtime engine. The figure below depicts a simplified view.

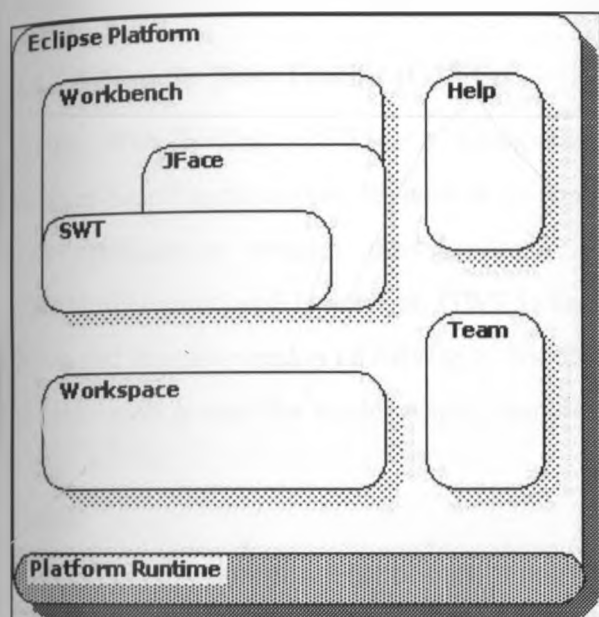


Figure 3.4.1: Simplified view of Eclipse IDE (Adapted from Eclipse, 2010)

The term *Workbench* refers to the desktop development environment. The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources.

Each Workbench window contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars. More than one Workbench window can exist on the desktop at any given time (Eclipse, 2010).

Eclipse has been selected to be the IDE for prototype and development, with Google AppEngine and Google Web Toolkit plug-ins for integration to the cloud environments.

3.4.1.2 Google AppEngine

Google App Engine is a system that exposes various pieces of Google's scalable infrastructure so that you can write server-side applications on top of them. Simply this is a platform which allows users to run and host their web applications on Google's infrastructure. These applications are easy to build, easy to maintain and easy to scale whenever traffic and data storage is needed. By using Google's App Engine, there are no servers to maintain and no administrators needed. The idea is user just to upload his application and it is ready to serve its own customers.

3.4.1.3 Google Web Toolkit (GWT)

Google Web Toolkit (GWT) is a development toolkit for building and optimizing complex browser-based applications. Its goal is to enable productive development of high-performance web applications without the developer having to be an expert in browser quirks, XMLHttpRequest, and JavaScript. GWT is used by many products at Google, including Google Wave and the new version of AdWords. It's open source, completely free, and used by thousands of developers around the world. Application development features include:

Write

The GWT SDK provides a set of core Java APIs and Widgets. These allow you to write AJAX applications in Java and then compile the source to highly optimized JavaScript that runs across all browsers, including mobile browsers for Android and the iPhone. Constructing AJAX applications in this manner is more productive thanks to a higher level of abstraction on top of common concepts like DOM manipulation and XHR communication. You aren't limited to pre-canned widgets either. Anything you can do with the browser's DOM and JavaScript can be done in GWT, including interacting with hand-written JavaScript.

Debug

You can debug AJAX applications in your favorite IDE just like you would a desktop application, and in your favorite browser just like you would if you were coding JavaScript. The GWT developer plugin spans the gap between Java bytecode in the debugger and the browser's JavaScript. Thanks to the GWT developer plugin, there's no compiling of code to JavaScript to view it in the browser. You can use the same edit-refresh-view cycle you're used to with

JavaScript, while at the same time inspect variables, set breakpoints, and utilize all the other debugger tools available to you with Java. And because GWT's development mode is now in the browser itself, you can use tools like Firebug and Inspector as you code in Java. GWT enables

Optimize

Google Web Toolkit contains two powerful tools for creating optimized web applications. The GWT compiler performs comprehensive optimizations across your codebase — in-lining methods, removing dead code, optimizing strings, and more. By setting split-points in the code, it can also segment your download into multiple JavaScript fragments, splitting up large applications for faster startup time. Performance bottlenecks aren't limited to JavaScript. Browser layout and CSS often behave in strange ways that are hard to diagnose. Speed Tracer is a new Chrome Extension in Google Web Toolkit that enables you to diagnose performance problems in the browser (Google 2011a).

3.4.1.4 Google Accounts

Google Accounts is a centralized user authentication management solution. It allows access to various Google services through a common authentication model.

Third-party applications often require limited access to a user's Google Account for certain types of activity. To ensure that user data is not abused, all requests for access must be approved by the account holder. Access control has two components, authentication and authorization. Authentication services allow users to sign in to your application using a Google Account. Some services also allow users to sign in using another account, such as an OpenID login. Authorization services let users provide your application with access to the data they have stored in Google applications. Google takes privacy seriously, and any application that requires access to a user's data must be authorized by the user. Authentication and authorization services are often referred to collectively as *auth* (Google, 2011b).

3.4.1.5 Google AppEngine Datastore

The App Engine datastore is a schema less object data store, with a query engine and atomic transactions. The Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces, as well as a low-level datastore API.

The High Replication datastore provides an even more reliable storage solution with no planned downtime, higher availability of reads and writes, eventual consistency for all queries except ancestor queries, and strong consistency for reads and ancestor queries (Google, 2011c).

App Engine provides two different data storage options differentiated by their availability and consistency guarantees:

Master/Slave datastore: Uses a master-slave replication system, which asynchronously replicates data as you write it to a physical data center. Since only one data center is the master for writing at any given time, this option offers strong consistency for all reads and queries, at the cost of periods of temporary unavailability during data center issues or planned downtime. This option also offers the lowest storage and CPU costs for storing data.

High Replication datastore: Data is replicated across data centers using a system based on the Paxos algorithm. High Replication provides very high availability for reads and writes (at the cost of higher-latency writes). Most queries are eventually consistent. Storage quota and CPU costs are approximately three times those in Master/Slave option.

The App Engine datastore saves data objects, known as entities. An entity has one or more properties, named values of one of several supported data types. For instance, a property can be a string, an integer, or even a reference to another entity.

The datastore can execute multiple operations in a single transaction. By definition, a transaction cannot succeed unless every operation in the transaction succeeds. If any of the operations fail, the transaction is automatically rolled back. This is especially useful for distributed web applications, where multiple users may be accessing or manipulating the same data at the same time. Unlike traditional databases, the datastore uses a distributed architecture to automatically manage scaling to very large data sets. It is very different from a traditional relational database in how it describes relationships between data objects. Two entities of the same kind can have different properties. Different entities can have properties with the same name, but different value types. While the datastore interface has many of the same features of traditional databases, the datastore's unique characteristics imply a different way of designing and managing data to take advantage of the ability to scale automatically (Google, 2011c).

Appengine Datastore primarily uses keys to manage relationships in the data, which is represented by entities. This can be best represented in a tree form:

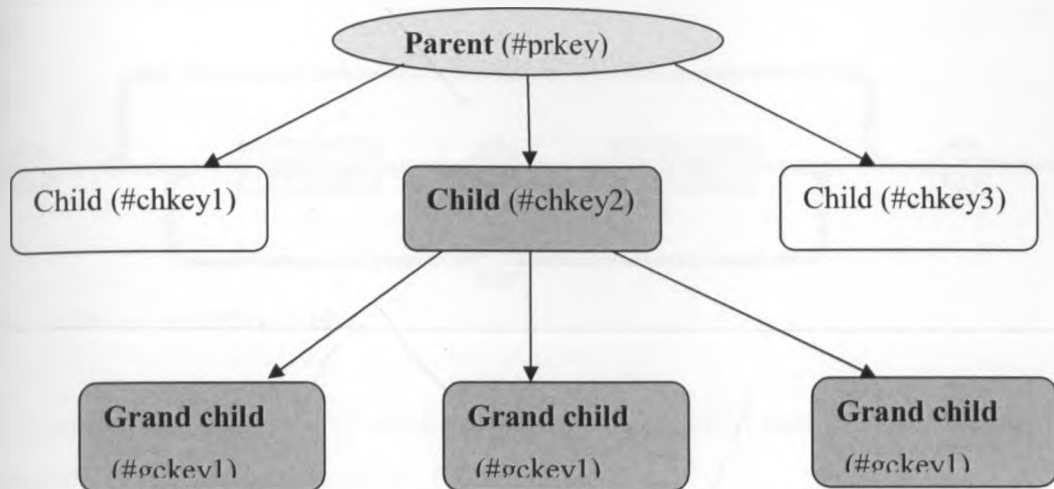


Figure 3.4.2: Apps Datastore Entity view

3.4.1.6 JavaScript Object Notation (JSON)

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write, and for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).

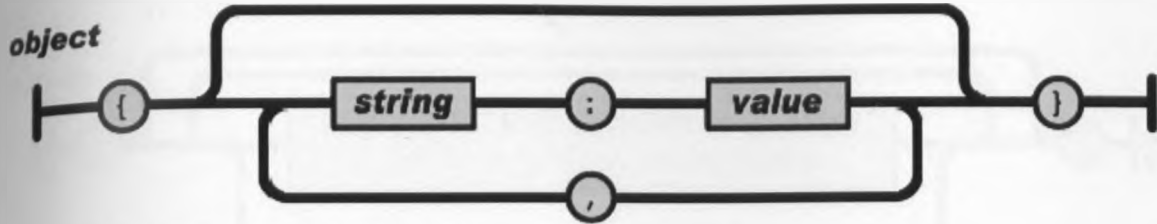


Figure 3.4.3: JSON object (JSON, 2010)

An *array* is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).

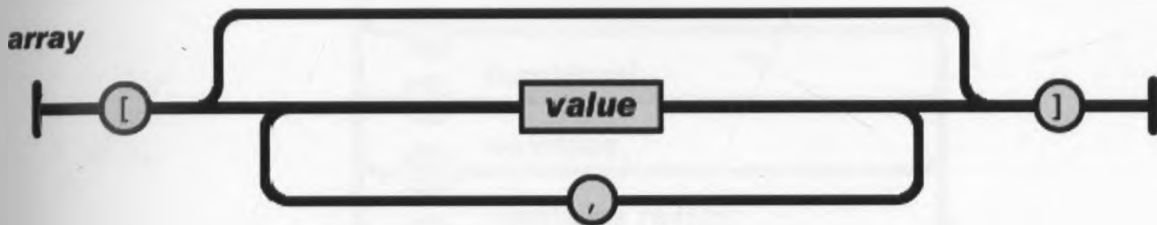


Figure 3.4.4: JSON array (JSON, 2010)

A *value* can be a *string* in double quotes, or a *number*, or true or false or null, or an *object* or an *array*. These structures can be nested.

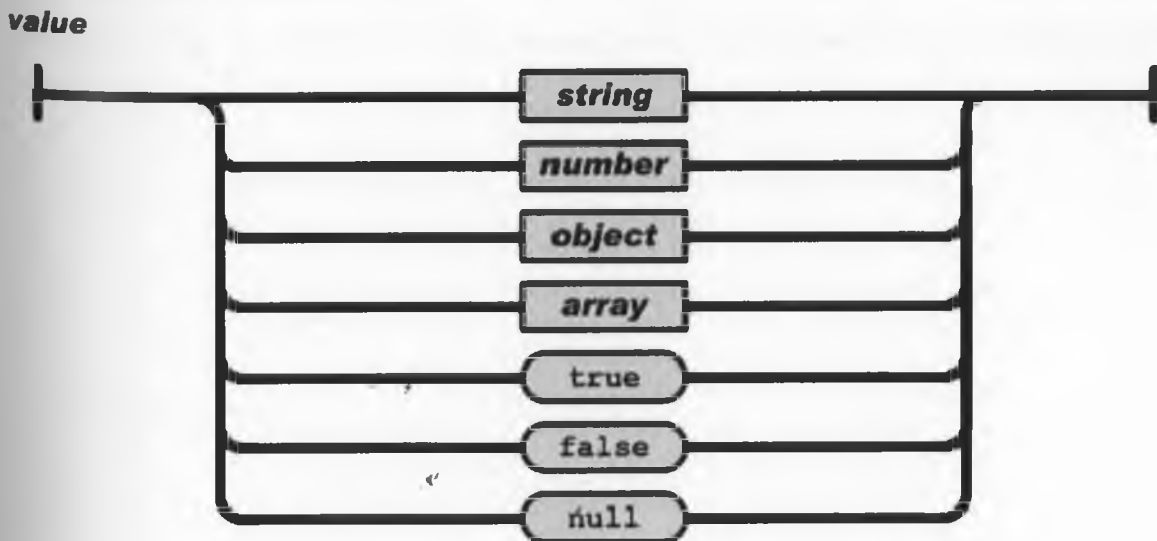


Figure 3.4.5: JSON value (JSON, 2010)

A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.

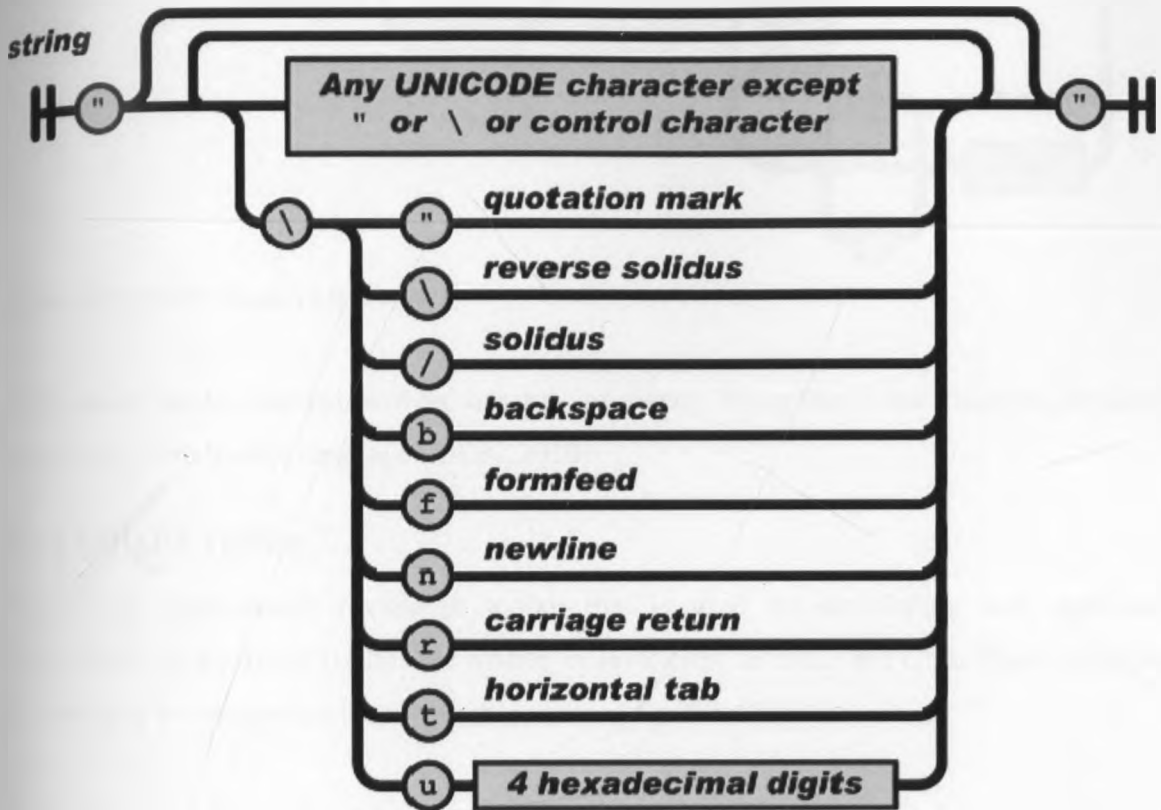


Figure 3.4.6: JSON String (JSON, 2010)

A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

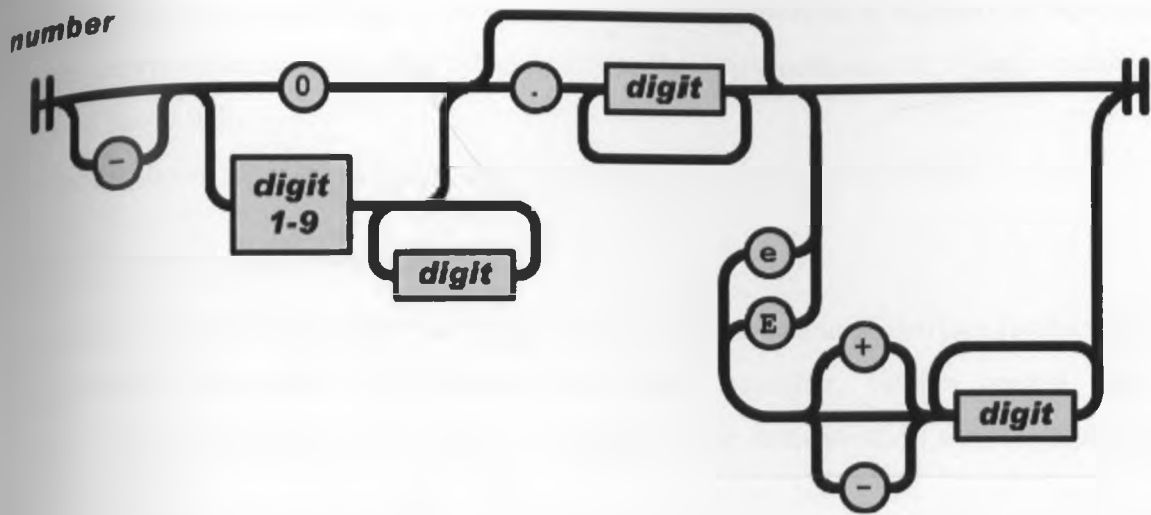


Figure 3.4.7: JSON Number (JSON, 2010)

Whitespace can be inserted between any pair of tokens. Excepting a few encoding details, that completely describe the language (JSON, 2010)

3.4.1.7 DOJO Toolkit

Dojo is an open-source JavaScript toolkit that is used for developing web applications. Components in the Dojo Toolkit are written in JavaScript, HTML, and CSS. These components are aimed to be encapsulated and reusable components for widgets.

The Dojo Toolkit is divided into three main components: the Dojo core, Dijit, and Dojox. Its responsibilities include manipulating the Cascading Style Sheets (CSS), supporting animation features and drag-and-drop functionality, among others.

Dijit is Dojo's widget library. Some widgets that it offers include menus, trees, tabs, closable tabs, sliding tab containers, and other different kinds of containers which are useful for developing the user interface of an IDE.

3.4.1.8 CodeMirror Editor Library

CodeMirror is a JavaScript library that can be used to create a relatively pleasant editor interface for code-like content. These include computer programs, HTML markup, and similar. The

library supports plugin architecture by use of *modes*, which provide a standard architecture for support incorporating new languages. This is key to the implementation of a plugin architecture for cloud based IDE.

The library allows for customizations and extensions to fit the situation at hand.

3.4.2 High level architecture

The system is fundamentally structured in a 3 tier setup, with end user interface (project and file manipulation), application server (http servlet, code compiler, version control and file management) and Google services backend (database, user authentication service, deployment) as shown below

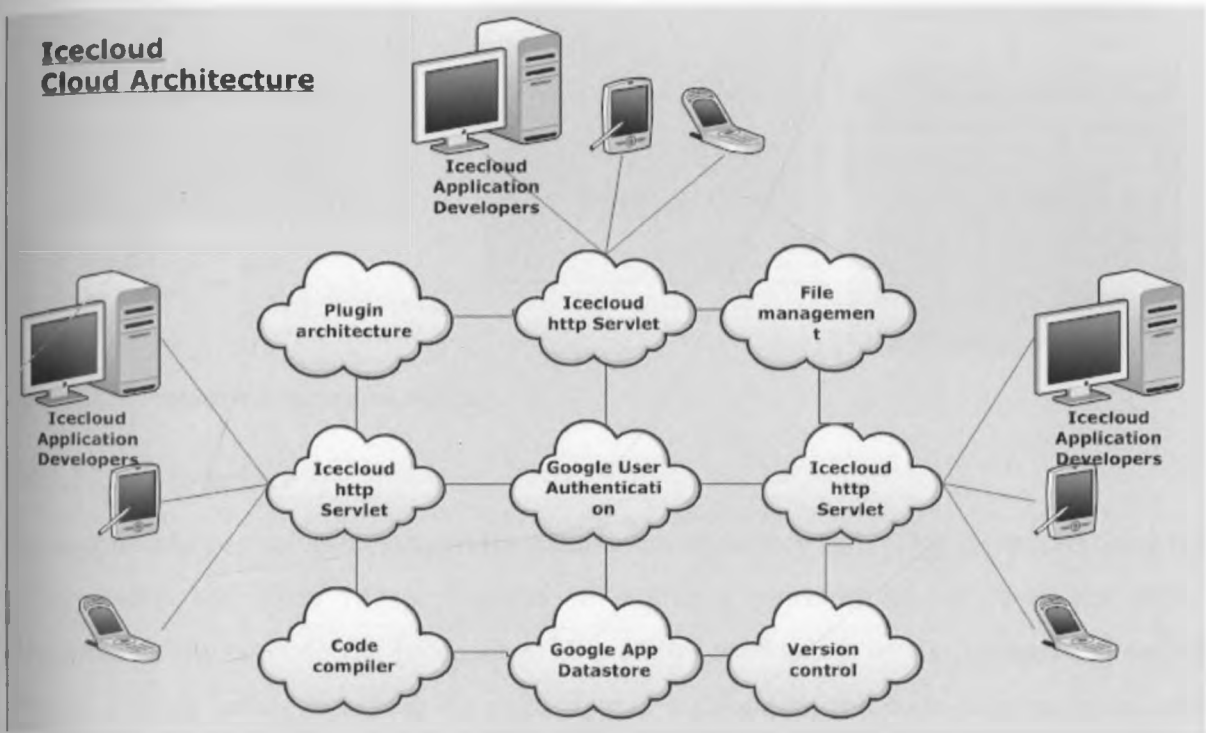


Figure 3.4.8: High level prototype structure

3.4.3 Architectural Design

This section provides the complete architecture of the cloud based IDE prototype. The architecture assumes implementation on Google App engine and draws the design from the conceptual design covered in section 3.2.

Icecloud IDE Core Architecture

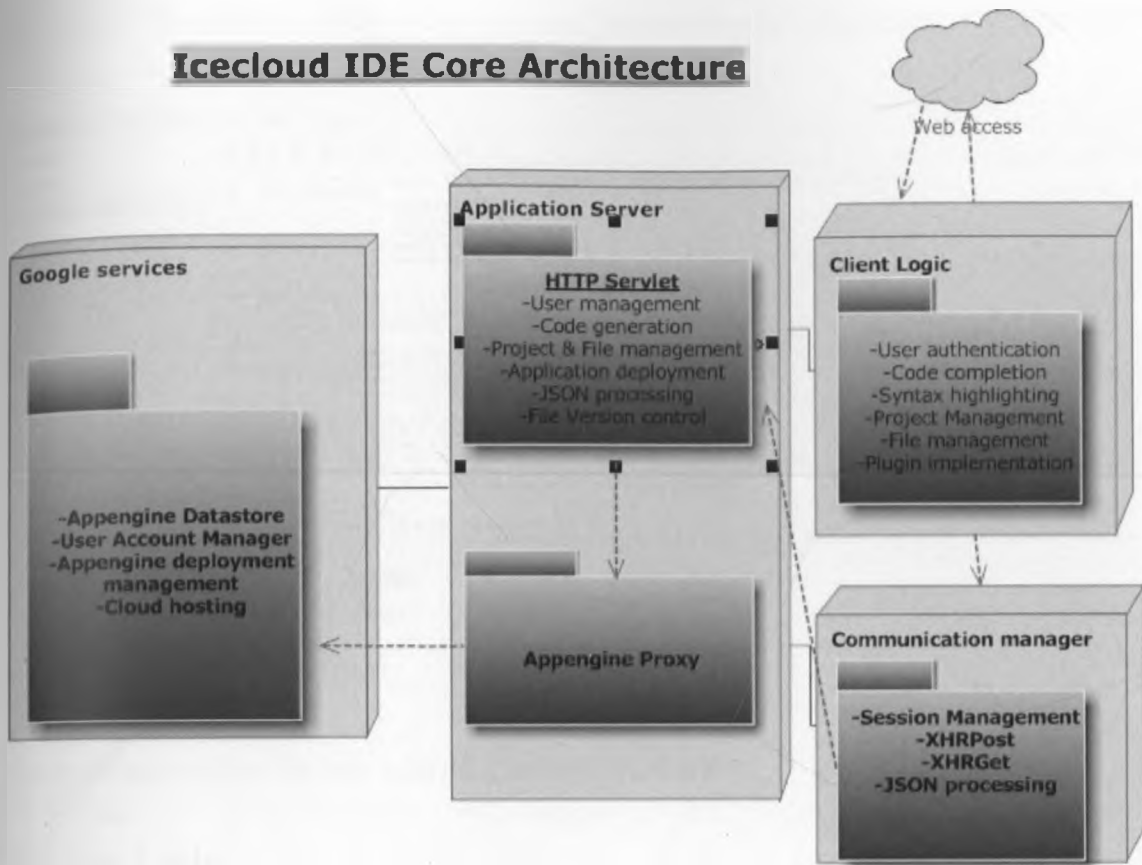


Figure 3.4.9: Prototype architectural design

3.4.3.1 User interface

The user interface is strives to support the natural feel of desktop IDE. This is realized using the DOJO toolkit and Code Mirror libraries to realize a powerful set of JavaScript based asynchronous source code file processing. This allows for loading of file content and saving changes without having to refresh the page. Syntax highlighting and code completion are also handled from the client side using an extensible plugin model. This is best represented by the diagram below:

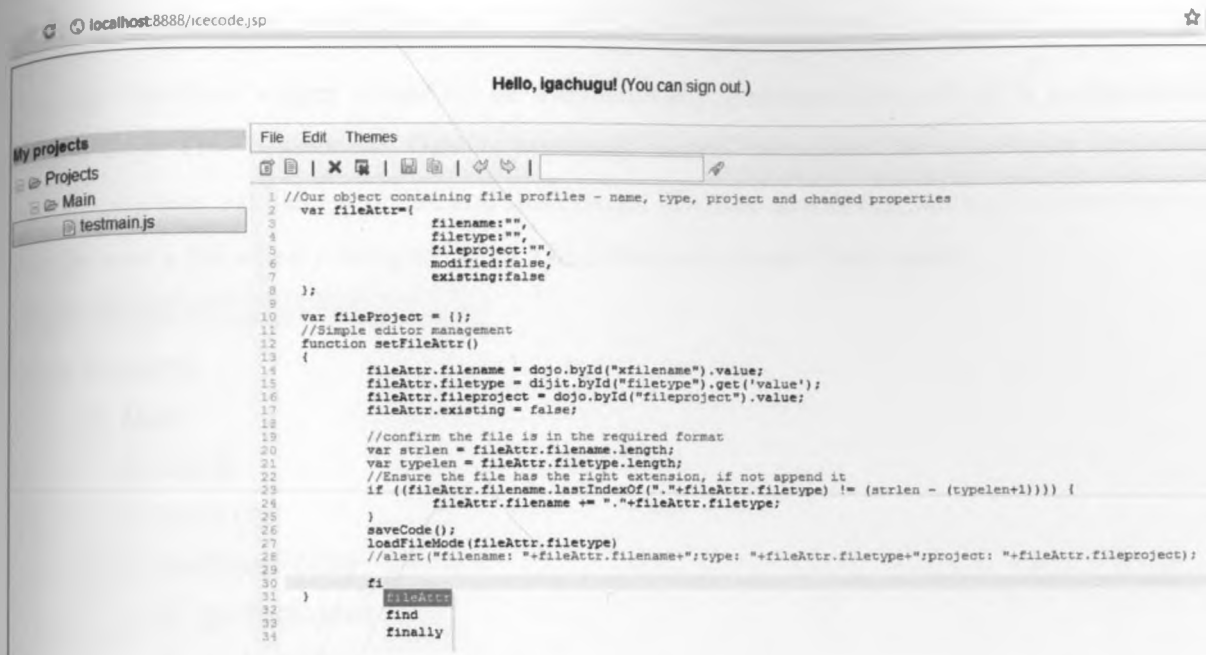


Figure 3.4.10: Prototype user interface

The main components of the user interface are explained below:

3.4.3.2 User Login

User login is fully managed by Google’s common user management *auth* service. This eliminates the security concerns most cloud based users would have about the security of their work. In development mode, this is simulated by a basic user authentication module that mimics how the system works with real integration to Google user authentication service. On the hosted environment, Google user management service is responsible for session and all user related details taking the burden off the core system. This creates a feeling of security and familiarity among millions of existing Google service customers.

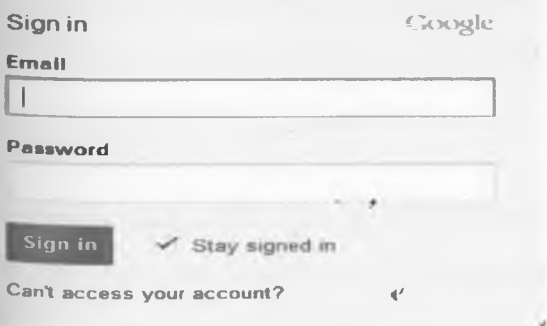


Figure 3.4.11: Development mode user authentication

3.4.3.3 Projects management tree

This user interface widget comprises of a dynamically generated tree which is implemented using a Dojo Tree component. Data is populated in real time from the Appengine Datastore representing user's project structure and file system. The tree adapts the famous Explorer look to give the user a feel of easy navigation and a high interactive web client model.

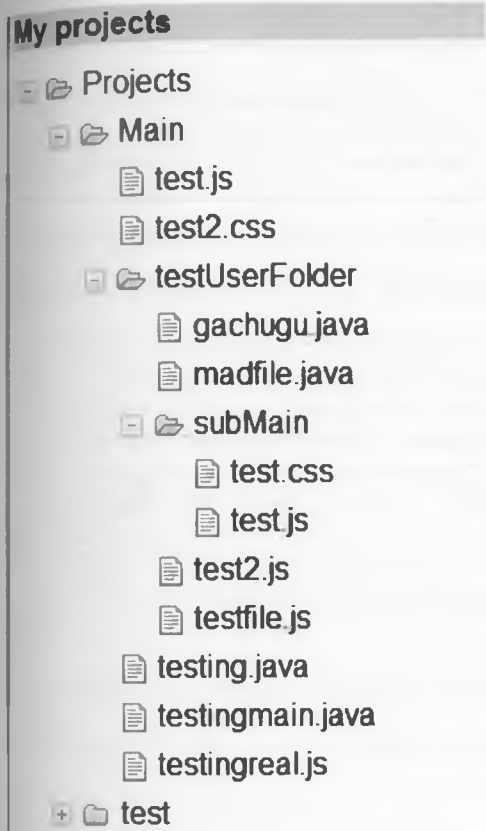


Figure 3.4.12: Project tree

The tree allows for a visual representation of the virtual file system in the familiar explorer style. Users can easily switch between projects without losing time and crowding the working area. The interface also allows access to the file being edited which remains highlighted to aid in active file recognition.

Tree data is generated from the backend servlet once a user is authenticated. All the projects under that user are loaded on to the project tree using a JSON object, which is generated

dynamically from the java http servlet handling the client requests. Sample code is as shown below:

Main function handling http get options to determine the request being serviced. For projects, command option L and C are used.

```
else if ((option.equals("L")) || (option.equals("C"))) //load projects tree L or Combo C
{
    List<Entity> projects = getProjects(user);
    respdata = loadProjects(projects,option);
}
//send the response data back - Either code value or projects JSON object
log.info("JSON class returned for processing:\n"+respdata);
resp.getWriter().println(respdata);
```

A method getProjects(User user) that takes in a Google User object and returns all the projects under that user in an array list of Entity objects. Each entity contains project meta data.

```
private List<Entity> getProjects(User user){
    // view of the projects belonging the user
    Query query = new Query("Projects", getWorkspaceKey(user));
    query.addSort("Owner", SortDirection.ASCENDING);
    query.addSort("projectName", SortDirection.ASCENDING);
    query.addFilter("Owner", Query.FilterOperator.EQUAL, user);

    DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
    return datastore.prepare(query).asList(FetchOptions.Builder.withLimit(5));
}
```

And finally a method loadProjects takes in the entity and loads the files under that project. Load option parameter is used to separate projects loaded for a lookup combo used in file creation and the project tree.

```

private String loadProjects(List<Entity> projects, String loadOption)
{
    String respdata = ""; //this will contain our JSON data
    int seq = 0;
    //return the list of files in JSON format for DOJO tree loading
    //initialize the JSON structure
    respdata = "{\nidentifier:'id',\nlabel:'fname',\nitems:{\n";
    //get all projects
    for (Entity project : projects) {
        String projectName = (String) project.getProperty("projectName");
        String projectType = (String) project.getProperty("projectType");
        if (seq>0) respdata += ",";
        respdata += String.format("{id:'%s',fname:'%s',ftype:'%s',type:'project',\nchildren:[",
            projectName, projectName, projectType);
        //load the files under the project
        if (loadOption.equals("C"))
            respdata += "]\n"; // ignore child references
        else
            respdata += loadProjectFiles(project);
        seq++;
    }
    return respdata + "\n]\n";
}
}

```

The resulting JSON objects is shown below:

```

{
  identifier:'id',
  label:'fname',
  items:[
    {id:'Icepad',fname:'Icepad',ftype:'Web based notepad',type:'project',
  children:[({_reference:'Icepad.datamodel.sql'},({_reference:'Icepad.uMain.java'})]},
    {id:'Icepad.datamodel.sql',fname:'datamodel.sql',filetype:'sql',type:'file'},
    {id:'Icepad.uMain.java',fname:'uMain.java',filetype:'java',type:'file'},
    {id:'Main',fname:'Main',ftype:'Testing main project',type:'project',
  children:[({_reference:'Main.testmain.js'})]},
    {id:'Main.testmain.js',fname:'testmain.js',filetype:'js',type:'file'}
  ]
}

```

Figure 3.4.13: Dynamically generated JSON project tree object

On the client side, this structure is processed and loaded into a DOJO widget using a JavaScript function. The code is shown below:

```

//load projects and contained file in a tree
function loadProjects()
{
    //build the projects data model
    var treeModel = new dijit.tree.ForestStoreModel({
        store: store,
        query: {
            "type": "project"
        },
        rootId: "root",
        rootLabel: "Projects",
        childrenAttrs: ["children"]
    });
    //create the tree in place of user project
    var projtree = new dijit.Tree({
        model: treeModel,
        onClick:function(item){
            var ufile = store.getValue(item, "id");
            var ftype = store.getValue(item, "filetype");
            umode = loadFileMode(ftype);
            //umode = "application/x-httpd-php";

            loadFile(ufile, umode); }
        }, "prtree");
    //use projtee as required
}

```

3.4.3.4 Main Menu and Toolbar

The system maintains a menu that mimics desktop based IDEs, with an accompanying toolbar. This is implemented using a Dojo Menu and Dojo Toolbar respectively, both of which are fully maintained from client side JavaScript. A sample of each is shown below:

Main menu (menu.js), Tool Bar (toolbar.js)

```
//Create the menu bar here
```

```
var pMenuBar;  
dojo.addOnLoad(function() {  
    pMenuBar = new dijit.MenuBar({});  
  
    var pSubMenu = new dijit.Menu({});  
    pSubMenu.addChild(new dijit.MenuItem({  
        label: "New Project",  
        onClick: function(){showProjectDialog();  
    }  
}));  
    pSubMenu.addChild(new dijit.MenuItem({  
        label: "New File",  
        onClick: function(){  
            editor.setValue("");  
            showFileDialog();  
        }  
}));  
});
```

```
//Icecloud toolbar
```

```
var toolbar;  
dojo.addOnLoad(function() {  
    toolbar = new dijit.Toolbar({},  
    "toolbar");  
    //Add toolbar actions - New project  
    toolbar.addChild(new dijit.form.Button({  
        label: "New Project",  
        showLabel: false,  
        iconClass: "dijitIcon dijitIconNewTask",  
        onClick: function(){showProjectDialog();}  
}));  
    //New file  
    toolbar.addChild(new dijit.form.Button({  
        label: "New File",  
        showLabel: false,  
        iconClass: "dijitIcon dijitIconFile",  
        onClick: function(){  
            editor.setValue("");  
            showFileDialog();  
        }  
}));  
});
```

3.4.3.5 Code Editor

The code editor forms the core of the IDE. This enables users to edit their source code and benefit from other IDE features, notably code completion and syntax highlighting. The editor is implemented using Code Mirror library, which has default editor with syntax highlighting and line numbering. The editor is purely coded in JavaScript and can be extended as required to

accommodate any function that can be implemented on JavaScript. Cascading style sheets are also used to ensure the editor properties meet user expectations and support for multiple editing themes. These include search highlighting, active line marker, bracket matching, code indenting and dynamic change of syntax highlighting mode depending on the file loaded.

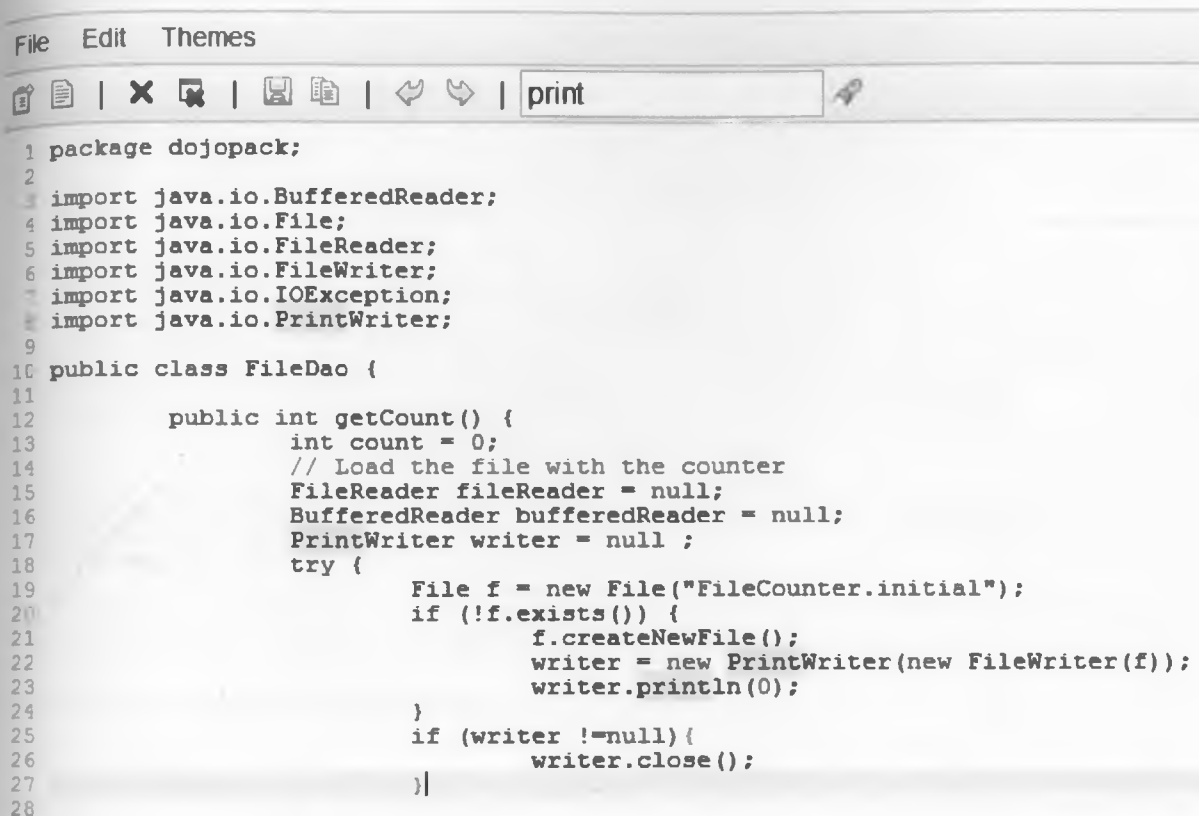


Figure 3.4.14: Prototype (Icecloud) Editor

3.4.3.6 Database structure

Icecloud IDE runs on Appengine Datastore, which is structured to handle fast CRUD (Create, Retrieve, Update and Delete) operations. The entities are maintained in a tree view, with all the entities falling under the user parent, represented by the workspace key. The actual user entity is managed by Google under the common user authentication model.

Icecloud Database Structure

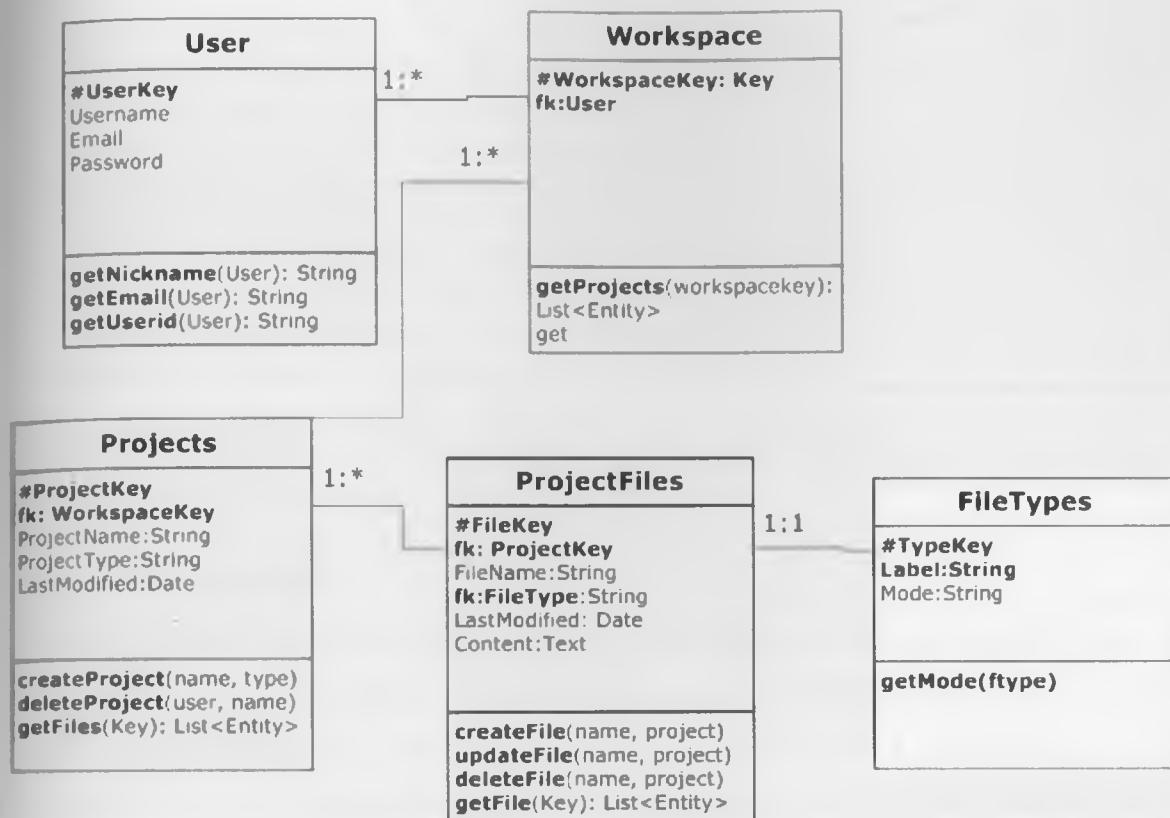


Figure 3.4.15: Prototype database architecture

File types are maintained as a JSON object on the client side for ease of plugin management.

This contains three elements with different purposes for each.

1. Name: Used for file creation as the user visible label
2. ID: Internally used for file and mode management
3. Mode: Used for syntax highlighting and plugin management in the editor

The structure is as shown below:

```

{
  identifier: "id",
  label: "name",
  items: [
    {name: "Javascript", id: "js", mode: "javascript"},
    {name: "CSS", id: "css", mode: "text/css"},
    {name: "PL/SQL", id: "sql", mode: "text/x-plsql"},
    {name: "XML", id: "xml", mode: "application/xml"},
    {name: "HTML", id: "html", mode: "text/html"},
    {name: "PHP", id: "php", mode: "application/x-httpd-php"},
    {name: "Java", id: "java", mode: "text/x-java"},
    {name: "C++", id: "cpp", mode: "text/x-c++src"},
    {name: "Python", id: "py", mode: "text/x-python"}
  ]
}

```

Figure 3.4.16: File types JSON object

3.4.3.7 Servlet Functions

The system uses the services of a java servlet that handles standard http get and post. These are implemented to handle Asynchronous request issued using XMLHttpRequest (XHR) functions, XMLHttpRequestGet and XMLHttpRequestPost. These enable a part of the web page to handle a standard http Post or a Get without refreshing the page. The respective commands are handled by the servlet as standard commands, with parameters differentiating the requests. The key user functions handled by the servlet are:

1. Retrieving user projects based on the logged in user (User object)
2. Retrieving a project files for each user project using project key
3. Retrieving file content for the selected source code file based on file key
4. Saving a new project for a specific user
5. Saving a new source code file under a specific project
6. Version control for the source code files
7. Template code generation for new files with predefined templates based on development language
8. Deleting a project based on a project key
9. Deleting a file based on a file key.
10. Loading user settings for the workspace for the given user
11. Virtual file management and conversion to standard file system for download

Sample code for the method handling http Post and an xhrGet JavaScript functions are shown below:

```
313  /** Servlet methods for handling standard/ajax post and get form commands**/
314  //Save the code or project to appengine datastore
315  public void doPost(HttpServletRequest req, HttpServletResponse resp)
316-  throws IOException {
317      //We implement one entity group user using AppsDatastore
318      String postOption = req.getParameter("postOption");
319      ///Post option P: Project, F: File U: User Settings
320      if (postOption.equals("P")) {
321          String projectName = req.getParameter("dlname");
322          String projectType = req.getParameter("dltype");
323
324          Key key = saveProject(projectName, projectType);
325          resp.setContentType("text");
326          String keyString = KeyFactory.keyToString(key);
327
328          resp.getWriter().println(keyString);
329      }
330
331      else if (postOption.equals("F")){
332          String fileKey = req.getParameter("fileKey");
333          String fileContent = req.getParameter("code");
334          if (fileKey != null)
335              updateFile(fileKey, fileContent);
336          else {
337              String fileName = req.getParameter("filename");
338              String fileType = req.getParameter("filetype");
339
340              String projectName = req.getParameter("projectName");
341              String parentFolder = req.getParameter("folderKey");
342              //if content is null load the template file and save
343              if ((fileContent.trim().length() == 0)){
344                  //%% (fileType.equalsIgnoreCase("java") || fileType.equalsIgnoreCase("sql"))
345                  FileTemplate flTemplate = new FileTemplate(fileName);
346                  fileContent = flTemplate.loadTemplate();
347              }
348
349              saveFile(projectName, fileName, fileType, parentFolder, fileContent);
350              //load file template as defined
351              if (fileContent.length() > 0){
352                  resp.setContentType("text");
353                  resp.getWriter().println(fileContent);
354              }
355          }
356      }
357      else if (postOption.equals("FL")){
358          String folderName = req.getParameter("folderName");
359          String projectName = req.getParameter("projectName");
360          String parentFolder = req.getParameter("folderKey");
361
362          saveFolder(projectName, folderName, parentFolder);
363      }
364  }
```

```

496 //load specific files
497 function loadFile(fname, fileKey, umode)
498 {
499     //AJAX file retrieval on demand to reduce the load on the browser
500     var xhrArgs = {
501         url: "/save?p=F",
502         //timeout:2000,
503         handleAs: "text",
504         content:{fkey:fileKey},
505         preventCache: true
506     }
507     //Call the asynchronous xhrPost
508     var deferred = dojo.xhrGet(xhrArgs);
509     //Now add the callbacks
510     deferred.addCallback(function(data) {
511         //load the file elements appropriately
512         editor.setValue(data);
513         //load the name in the filename box
514         fileAttr.filename = fname.substring(fname.indexOf('.')+1);
515         fileAttr.fileproject = fname.substring(0, fname.indexOf('.'));
516         fileAttr.filetype = fname.substring(fname.lastIndexOf('.')+1);
517         //if (!fileAttr.filetype == 'js')
518             editor.setOption("mode", umode);
519         fileAttr.modified = false;
520         fileAttr.existing = true;
521     });
522     deferred.addErrback(function(error) {
523         targetNode.innerHTML = "Error loading file: " + error;
524     });
525 }
526

```

The above JavaScript function calls a servlet method that responds with the source code file contents from Appengine Datastore.

```

42 //Load the file contents given the file key
43 public static String loadFileKey(String fileKeyStr) {
44     DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
45     Key fileKey = KeyFactory.stringToKey(fileKeyStr);
46     //load the file contents
47     try {
48         Entity pFile = datastore.get(fileKey);
49         return ((Text) pFile.getProperty("fileContent")).getValue();
50     }
51     catch(EntityNotFoundException e){
52         log.info("Entity not found: " + e.getMessage());
53         return "";
54     }
55 }
56 }

```

Both `xhrGet` and `xhrPost` work under the same premise, and we explore `xhrGet` for simplicity.

The Dojo `xhrGet()` function is the cornerstone function of AJAX development. Its purpose is to provide an easy to use and consistent interface to making asynchronous calls to retrieve data. This API is an abstraction atop the browser's `XMLHttpRequest` object and makes usage the same regardless of which browser your application is running on. This makes it much simpler to write cross-browser compatible AJAX style applications. This function, in essence, implements making an HTTP 'GET' call.

`dojo.xhrGet` (and other functions in the same line: `dojo.xhrPost`, `dojo.xhrDelete`, `dojo.xhrPut`), are bound by the 'same domain' security policy of the browser. This means that they can only establish a connection back to the same server that served the HTML page. If you wish to use this API to talk to servers other than the one that originated your page, then you will have to use a proxy on your originating server and have it forward the requests.

The `xhrGet()` function takes an object as its parameter. This object defines how the `xhrGet` should operate. Minimally, this object must contain a 'url' attribute so that the function knows where to send the request. Having just a 'url' attribute isn't the most useful approach to calling the function, though. You can also embed information such as how to handle the return data (As XML, JSON, or text), and what to do when it completes. It also accepts other useful parameters such as 'preventCache', and 'sync', which alter its behavior slightly (dojo, 2011).

3.4.4 Functional Testing

Functional tests are carried out to ensure that results from system usage meet the set objectives and requirements. The same tests are used in refining the design which in turn creates another loop of testing. This aims at having the final product conforming to the system specifications and requirements.

Specific test cases are used for functionality testing, mapping each of the requirements to a test case. Below is a summary of the actual data used in functional testing, which can be mapped to a quality assurance system, Mercury Quality Center by HP. This allows multiple users to execute the tests and fill in their results as well as track any defects raised. Test cases used are as shown below:

Subject	Test Case	Objective	Step Name	Description	Expected Results	Actual Results	Status
Accessibility	TC-001 IDE Accessibility	Test cloud system access across multiple browsers	Step 1	Open Icecloud IDE from location http://iceclouddev.appspot.com/	IDE opens and requests the user to login to use. If user is logged into any Google account, user is automatically granted access to IDE	IDE opens and requests the user to login to use. If user is logged into any Google account, user is automatically granted access to IDE	Pass
Accessibility	TC-001 IDE Accessibility	Test cloud system access across multiple browsers	Step 2	Click on the Sign In link to sign in to Icecloud IDE. The system uses Google account management services.	IDE redirects to Google's user accounts manager for authentication. User is requested to allow Icecloud access to account information. Once logged in, user accesses the IDE interface	IDE redirects to Google's user accounts manager for authentication. User is requested to allow Icecloud access to account information. Once logged in, user accesses the IDE interface	Pass
Accessibility	TC-001 IDE Accessibility	Test cloud system access across multiple browsers	Step 3	Run steps 1 and two on Mozilla, Google chrome and Internet explorer	The IDE has a similar interface across the different browsers mentioned.	The IDE has a similar interface across the different browsers mentioned.	Pass

Subject	Test Case	Objective	Step Name	Description	Expected Results	Actual Results	Status
Accessibility	TC-001 IDE Accessibility	Test cloud system access across multiple browsers	Step 4	Click on signout	The browser redirects to google accounts URL and shortly redirects back to the home screen for Icecloud IDE	The browser redirects to google accounts URL and shortly redirects back to the home screen for Icecloud IDE	Pass
IDE Functionality	TC-001 Project Management	Test how the IDE handles different project functions	Step 1	From the toolbar or main menu, click on New project.	New project dialogue is displayed	New project dialogue is displayed	Pass
IDE Functionality	TC-001 Project Management	Test how the IDE handles different project functions	Step 2	Without click on OK button without entering the project name	Project dialogue highlights the required fields - Project name	Project dialogue highlights the required fields - Project name	Pass
IDE Functionality	TC-001 Project Management	Test how the IDE handles different project functions	Step 3	Enter the project name and description and click on the OK button	New project is created successfully and information shown on the status bar at the bottom of the editor window. The new project is added to the project tree.	New project is created successfully and information shown on the status bar at the bottom of the editor window. The new project is added to the project tree.	Pass
IDE Functionality	TC-001 Project Management	Test how the IDE handles different project functions	Step 4	Using a different user account, create a new project with a different name	The new project is reflected under the current user. Previous projects under different user are not shown.	The new project is reflected under the current user. Previous projects under different user are not shown.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 1	On the main menu or the toolbar, click on the New File.	A new file dialogue is displayed	A new file dialogue is displayed	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 2	Click on the OK button to save the file	Dialog marks the required fields with a red ! Mark indicating a required field.	Dialog marks the required fields with a red ! Mark indicating a required field.	Pass

Subject	Test Case	Objective	Step Name	Description	Expected Results	Actual Results	Status
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 3	Add the file details and specify file type JavaScript (default), then click OK	The file is created, loaded with default code if defined and saved. The status bar displays file save message the file is added under the right project.	The file is created, loaded with default code if defined and saved. The status bar displays file save message the file is added under the right project.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 4	Start typing on the editor, or copy paste some JavaScript code on to the editor	The editor performs syntax highlighting on the code with key words recognition	The editor performs syntax highlighting on the code with key words recognition	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 5	Add a new line on the editor and press CTRL+Space	The editor pops an auto complete box for code completion	The editor pops an auto complete box for code completion	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 6	Make several changes to the editor and click save on the toolbar.	File is saved without affecting the editor and a message displayed on the toolbar.	File is saved without affecting the editor and a message displayed on the toolbar.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 7	Create a file of any other of the supported formats (PHP, Java, html, PL/SQL etc)	The editor changes syntax highlighting to reflect the new language.	The editor changes syntax highlighting to reflect the new language.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 8	Add different files to the different projects	Each file is mapped correctly on the project tree.	Each file is mapped correctly on the project tree.	Pass

Subject	Test Case	Objective	Step Name	Description	Expected Results	Actual Results	Status
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 9	Click on the various files under different projects	The selected file is loaded in the editor with the appropriate syntax highlighting	The selected file is loaded in the editor with the appropriate syntax highlighting	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 10	On the main menu or the toolbar, click on the Save As icon.	File dialog is displayed with the current file details populated	File dialog is displayed with the current file details populated	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 11	Type a different file name and click on OK	The new file is saved appropriately	The new file is saved appropriately	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 12	Make some changes on the file and test undo and redo buttons.	Undo and redo operations are reflected on the editor	Undo and redo operations are reflected on the editor	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 13	Type some text to search in the search box on the toolbox	The found text is highlighted in yellow for all instances of the text.	The found text is highlighted in yellow for all instances of the text.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 14	On the Themes menu, click on the various themes.	The editor changes syntax mode and background color differently for the various modes on the options.	The editor changes syntax mode and background color differently for the various modes on the options.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 15	Click on delete file on the toolbar or main menu	A confirmation to delete the file appears with the [project.filename] format.	A confirmation to delete the file appears with the [project.filename] format.	Pass
IDE Functionality	TC-002 File Management	Test how the IDE handles different source	Step 16	Click on Cancel button	The dialog closes without affecting the file	The dialog closes without affecting the file	Pass

		code files					
IDE Functionality	TC-002 File Management	Test how the IDE handles different source code files	Step 17	Click on OK button	The file is deleted and text on the editor cleared.	The file is deleted and text on the editor cleared.	Pass

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Functional Tests

Functional tests were carried out by members of the focus group who also participated in the analysis and design interviews. The detailed listing of test results is covered in section 3.44 under functional testing. Tests conducted only related to the functional requirements covered by the prototype and are not meant to reflect features expected from commercial products implementing the proposed design.

4.2 Survey details

Survey monkey is a commercial web based survey tool that allows for design and sharing of online survey. This tool came in handy as it offers free limited survey design and data collection. The survey used for this research was code named Icecloud IDE Survey for identification purposes only and was availed on <http://www.surveymonkey.com/s/57H5NP3>. The survey was open for the entire duration of the project.

As at 13th September 2011, the results were as detailed below:

Icecloud IDE Survey

Education Edit

Design Survey

Collect Responses

Below is a list of the collectors you are currently using to collect responses. To view the details or change the properties of an existing collector, just click the name. To collect more responses for this survey from a different group of people, click "Add New Collector"

Collector Name (Method)	Status	Responses	Last Response
Web Link (Web Link)	● OPEN	16 responses	September 12, 2011 1:05 AM

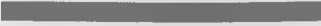

Response Summary

Total Started Survey: 16
Total Completed Survey: 16 (100%)

PAGE: KCECLOUD IDE USER REVIEW



1. Do you use a Integrated Development Environment(IDE)for Rapid Application Development(RAD)?

 Create Chart  Download

		Response Percent	Response Count
Yes		87.5%	14
No		12.5%	2
		answered question	16
		skipped question	0

2. Do you use any software frameworks for enterprise development?

 Create Chart  Download

		Response Percent	Response Count
Yes		93.8%	15
No		6.3%	1
		answered question	16
		skipped question	0

3. Are there any situations that call for traditional application development methodologies (non RAD)?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
Yes		93.8%	15
No		6.3%	1
		answered question	16
		skipped question	0

4. Would you consider using a cloud based development environment?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
Yes		75.0%	12
No		25.0%	4
		answered question	16
		skipped question	0

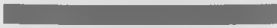

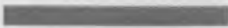


5. What would be your concerns in using cloud based development environment?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
None		6.7%	1
Data Security		66.7%	10
Connectivity issues		73.3%	11
Loss of control		66.7%	10
System Availability		66.7%	10
Limited features		26.7%	4
		Other (please specify) Show Responses	4
		answered question	15
		skipped question	1

6. What features of an IDE would you want to see in a cloud based IDE?

 Create Chart  Download

		Response Percent	Response Count
File management		73.3%	11
Project management		66.7%	10
Task management		60.0%	9
Multiuser environment		73.3%	11
Plugin management		60.0%	9
Visual design		66.7%	10
Code generation		66.7%	10
Code completion		66.7%	10
	Other (please specify) Show Responses		5
		answered question	15
		skipped question	1

7. What software frameworks would you want implemented in the cloud based IDE?

Download

Response Count

Hide Responses 16

Responses (16) Text Analysis My Categories (0)

GOLD FEATURE: Text Analysis allows you to view frequently used words and phrases, categorize responses and turn open-ended text into data you can really use. To use Text Analysis, upgrade to a GOLD or PLATINUM plan.

Learn More

Upgrade »

Showing 16 text responses

No responses selected

restful/soap web services integration
9/12/2011 11:05 AM View Responses

Android development enviroment
9/12/2011 8:57 AM View Responses

eclipse
9/12/2011 8:17 AM View Responses

Rapid Application development
8/24/2011 10:12 AM View Responses

JAVA
8/23/2011 8:32 AM View Responses

AJAX
8/19/2011 9:59 AM View Responses

answered question 16

skipped question 0

8. Do you have any plans to develop a cloud based application - Software as a service(SaaS)?

Create Chart Download

	Response Percent	Response Count
Yes	81.3%	13
No	18.8%	3

answered question 16

skipped question 0

9. What current tools would you consider for cloud application development?

Download

Response Count

Hide Responses 16

Responses (16) Text Analysis My Categories (0)

GOLD FEATURE: Text Analysis allows you to view frequently used words and phrases, categorize responses and turn open-ended text into data you can really use. To use Text Analysis, upgrade to a GOLD or PLATINUM plan.

Learn More

Upgrade »

Showing 16 text responses

No responses selected

Mysql java ssh ftp tomcat webserver hard disk space
9/12/2011 11:05 AM View Responses

Mobile applications
9/12/2011 8:57 AM View Responses

simple apps
9/12/2011 8:17 AM View Responses

GWT
8/24/2011 10:12 AM View Responses

n/a
8/23/2011 9:32 AM View Responses

IBM Application Development Services for Cloud
8/19/2011 9:59 AM View Responses

answered question 16

skipped question 0

10. Given similar features on the cloud development platform as your computer based development environment, would you switch fully to a cloud based development environment?

Create Chart

Download

Response Percent Response Count

Yes  68.8% 11

No  31.3% 5

answered question 16

skipped question 0

4.3 Summary of observations

From the survey and oral discussions with interested reviewers drawn from the focus group, it was evident that there is high need to have commercial production version of a cloud based IDE. Users interviewed expressed the desire to carry out their development with the same flexibility as they access their favorite internet mail.

Key concern for using a cloud based development environments (IDE) was connectivity which may limit access to their work when internet services are not accessible. However, with the increasing internet providers and reliability of cloud vendors like Google, potential users expressed some level of comfort in migrating to cloud based development environments. A comfort level also lies in the fact that the users can download their projects and continue using any editor that supports the same features as an on premise development environment.

The key functionalities expected from the cloud based IDEs match the common on premise development environments, with speed to deliver being the key driver. Key differentiator from the on-premise versions is the ability to have multiple users work on the same project seamlessly with fluent collaboration features. A close comparison to Google Docs, in which users can edit the same document with teams from remote machines as if they were doing it on the same machine but with access to specific sections of the document, kept coming up.

With the need to develop Software as a service (SaaS) applications growing every day, Cloud IDEs are key drivers to the shift from on premise application developments as well as on premise applications to cloud based offerings. Current developments in the cloud world are more aligned to using a cloud based IDE as opposed to on premise development environment as this makes it seamless to develop, deploy and apply patches without being tied to a static development platform. The solution offers a zero footprint installation which is ideal for on-the move application development.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

Cloud computing as a concept has matured with SaaS and IaaS layers being the most widely used. Platform as a Service is gaining footing based on the stability of the lower stack (Infrastructure as a Service) and the demand for the higher stack (Software as a Service). Application development on the cloud based frameworks is the next item on developers mind and it is highly expected that cloud vendors will put a lot of investments on this development.

The architecture highlighted here is expected to apply for commercial developments with some level of commercial and high usage considerations. Current application developers are very ready to start developing on the cloud as long as they feel it will make application development faster and seamless to migrate to on premise platforms to mitigate against known risks of connectivity and access.

It will also be expected that with commercial implementation of cloud based application development environments, operating systems dependency will be less for web based applications as the key consideration will shift to the runtime environment, which will be the browser.

In conclusion, I confidently state that the next five years will see an explosion in the growth of commercial cloud based IDEs, which will be zero foot print. Developers will hence be more concerned about application development and not the underlying infrastructure which limits starters and students from exploring and innovating with thin budgets. This will also see the rise in the SaaS applications as more developers will be able to easily develop and deploy to the cloud.

5.2 Contribution to the Research

This study has achieved the objectives set out at inception by bringing out the practical design and implementation of the cloud based development environment. It has also helped unearth some limitations that exist in the current cloud environments. Some of the key achievements and my own contributions into the research area include:

5.2.1 Cloud IDE Architecture

As envisioned in the objectives, this research has come up with a standard architectural design that will enable any interested parties to develop a standard cloud based IDE. This is based on extensive study in the existing development environments as well as current developments in cloud computing. A major shift from traditional application development platforms is expected to happen with the growing demand for cloud based applications, which are being offered as Software as a Service (SaaS) as well as stability and reliability of the cloud infrastructure. A key differentiator from the current on premise development environments is the collaborative development and seamless application deployment to the cloud as SaaS.

5.2.2 Creating awareness of PaaS cloud stack

As brought out in the research and literature review, various stacks of the cloud have gained acceptance in varying proportions with the PaaS stack being the worst performer. While most developers have plans to develop SaaS applications, very few had considered developing the same application from the cloud in the same manner that they expect their end users to access the final product! This was largely due to the awareness created by the industry players who are mainly cloud providers, who have focused more on deploying applications developed from on-premise environments with no much emphasis on the ability to develop and deploy from the cloud. This research helped developers to realize they can as easily develop and deploy from the cloud without much concern for static development environments.

5.2.3 Virtual file system

Based on the selected cloud platform for prototype development, Google AppEngine, it was not possible to store source code files in the traditional file system format. This required an

innovative way of managing the user files in a manner that hides the actual implementation of the file system to the user. While the user files are stored in the database as BLOBs, the user will not notice this limitation as they are able to download their source files in ordinary file system formats. The prototype uses in memory file streams to convert stored BLOB data to conventional file system for user download and is able to map the project structure to a file system folder format.

5.2.4 Non-Relational Databases for Cloud Computing

The research brought to focus a major shift that is happening in the cloud computing arena with regard to database management systems. Most of the cloud platform providers have moved away from the famous Relational Database Management Systems (RDBMS) to non-relational databases. This has come with ease of object management for light weight databases as well as performance and speed enhancements.

The implementation of Google Datastore is done through Google's Big Table, which hosts many internal and external Google services. The DataStore is a non-relational database. This means unlike traditional databases, developers don't build normalized tables then JOIN them for results. Instead, the DataStore is optimized for Read speed. The JOIN command is not supported so developers need to architect the database as they would for a high volume reporting database, meaning more columns and fewer tables. The prototype was developed on this database, which made it easy to develop the virtual file system explained above.

5.2.5 Use of JSON File Exchange Formats for Asynchronous User Interface

The research also brought to focus the use of Java Script Object Notation (JSON) as a preferred method of data interchange for asynchronous user interfaces. Its counterpart, XML has been heavily used and popularized, but from this research, it was evident that JSON offers more ease of use as it's easy to parse and form. Most of the AJAX (Asynchronous JavaScript and XML) interfaces exposed by the prototype rely on use of JSON objects for data interchange. Full details have been analyzed under theoretical framework.

5.3 Future Work

Based on the prototype development experience, the below items fit perfectly for further work in this area to enable full use of cloud based application development:

5.3.1 Hosted compilers

This will involve developing a hosted language specific compilers and parsers accessible from the web where developers will submit the source code and get the compiled versions, including compilation messages. This should then be integrated to the cloud based IDE for transparent compilation. The envisioned versions will work similarly to current web services with AJAX and related technologies playing a major role.

5.3.2 Cloud Data Access Components

Cloud data access objects and components will be a very rich addition to any cloud development environment. Application developers using the cloud environment will be able to specify the database their end application will use, and the development platform will provide configurable components that will enable users to connect to the database from the development environment and access the database. The same components will be used in runtime mode to enable the end application perform CRUD operations at database level. One way of achieving this is to have libraries that support several hosting environments and databases or to have dedicated components for specific database providers on the cloud. One will need to cater for the two existing models of relational and No-SQL databases.

5.3.3 Cloud based runtime environments

Users of cloud based application development should be able to test their source code by executing the programs and getting a feel of the production environment. An end to end solution need to include runtime environment with the necessary security and sand box features to ensure developers do not execute malicious code on the target environments. This could be developed as part of the IDE or with third party providers to enable variant targets to be tested. While this may appear a basic feature in any IDE, the nature of current cloud providers requires more innovative approaches.

Google AppEngine currently does not offer any file based services which are key in developing file based applications (IO Libraries in Java and related languages). This means that for one to handle files in the traditional manner, virtual file management features exposed under this research will be very important. One such application is compilation of code stored in the database as opposed to physical files.

5.4 Research Questions Answered

Below is a confirmation on how this study answered the set research questions:

- 1 What are the trends supporting rapid application development in cloud computing?
 - a. The sections on literature review, system analysis and system design brought out technological developments that have made it possible to utilize cloud environments for rapid application development.
- 2 What are the major challenges facing the adoption of cloud application development and PaaS in general and what are the solutions?
 - a. From the design view as well as user survey, it was clear internet connectivity was the key to using cloud services. This also presents a single point of failure which is a major challenge in the adoption of cloud application development. Use of asynchronous communication can help reduce the challenge but not eliminate it. This being an infrastructural challenge, the solution relies on having many options around internet access. Another key challenge is data security which can be mitigated by using mature single sign-on features provided by the cloud hosting environments. This has been clearly brought out in the design.
- 3 How can framework based development increase the uptake of cloud application development?
 - a. Provision of an IDE is only the first step in building cloud application development uptake. Users will be more drawn towards the environment if it provides value addition and faster application development and deployment compared to the traditional environments. Use of rich software frameworks and code templates will be critical in moving the users to the cloud environments. This was a key requirement in the requirements gathering and user surveys.

4 Can cloud (web) environments support fully fledged IDEs and framework based application development for programmers?

- a. The answer is a clear YES. This has been clearly demonstrated using the prototype that was developed in a limited time frame and using limited resources. All possible features on the local installation of development environments are possible on cloud environments. All the key requirements highlighted in the system analysis section were implemented in the prototype development.

CHAPTER 6: REFERENCES

6.1 Books and Journals

- 1) Abdelghani Bellaachia (1999). *Advanced Software Paradigms. CSCI 210 Programming paradigms*, pp.6-10 [pdf], George Washington University. Available at http://www.seas.gwu.edu/~bell/csci210/lectures/programming_paradigms.pdf, accessed on 26th Feb 2011.
- 2) Alan Brown, Simon Johnston, Kevin Kelly (2003). Rational software whitepaper from IBM. *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*, p.5 [pdf]. Available at [http://www.sysedv.tu-berlin.de/intranet/kc-kb.nsf/eab33685727bd57fc1256979005d9f28/D40B52D747C3A224C1256DB30072AB6F/\\$File/TP032.pdf?OpenElement](http://www.sysedv.tu-berlin.de/intranet/kc-kb.nsf/eab33685727bd57fc1256979005d9f28/D40B52D747C3A224C1256DB30072AB6F/$File/TP032.pdf?OpenElement), accessed on 25th Mar 2011.
- 3) Angel R Puerta, Henrik Eriksson, John H Gennari and Mark A Musen (1994). *Beyond Data Models for Automated User Interface generation*, p.1 [pdf]. Stanford: Stanford University. Available at http://bmir.stanford.edu/file_asset/index.php/441/SMI-94-0539.pdf, accessed on 25th Mar 2011.
- 4) Antonopoulos N. and Gillam L. (Eds.) (2010). *Cloud Computing: Principles, Systems and applications*, p.3. London Dordrecht New York Heidelberg. Springer.
- 5) C. K. Chua (2003). *RAPID PROTOTYPING - Principles and Applications* (2nd Edition), p.3. World Scientific Publishing Co. Pte. Ltd.
- 6) CASEMaker Inc (2000). *Rapid Application development: What is Rapid Application Development (RAD)?*, p.1 [pdf]. Available at: www.casemaker.com/download/products/totem/rad_wp.pdf, accessed on 20th Mar 2011.
- 7) Dirk Riehle (2000). *Framework Design: A Role Modeling Approach*. Ph.D. Thesis, No. 13509, p.2. Zürich, Switzerland, ETH Zürich.
- 8) Gerald Kaefer (2009). *Cloud Computing Architecture*, pp.4-5 [pdf]. Siemens AG, CT T DE IT1 Corporate Technology. Available at <http://www.sei.cmu.edu/library/assets/presentations/Cloud%20Computing%20Architecture%20-%20Gerald%20Kaefer.pdf>, accessed on 14th May 2011.

- 9) Jingwen Ou, Mahdi Tayarani Najaran, Mushfiqur Rouf (2007). *Communication Protocols Project. Aurora SDK: A Web Based Integrated Development Environment* pp.1-4. [pdf]. University of British Columbia. Available at <http://www.cs.ubc.ca/~nasarouf/projects/cpsc527/AuroraSDK.pdf>, accessed on 24th Mar 2011.
- 10) Jyri Laukkanen (2008). *Aspect Oriented Programming* p2. Seminar Paper, University of Helsinki [pdf]. Available at: www.cs.helsinki.fi/u/paakki/laukkanen.pdf , accessed on 26th Feb 2011.
- 11) McAffer E., Gamma E. and Wiegand J. (eds) (2010). *Eclipse Rich Client Platform, 2nd edition* p.ii. Boston: Pearson Education, Inc.
- 12) Sussman B.C., Fitzpatrick B.W and Pilato C.M (2011). *Version Control with Subversion For Subversion 1.6*, p.22. Stanford, California: Creative Commons Attribution License.
- 13) Wholey, J., Hatry, H., & Newcomer, K. (eds). (2004 cited in InSites, 2007, *Tips on Qualitative and Quantitative Data Collection Methods* p.2). Handbook of practical program evaluation. San Francisco, CA. Jossey-Bass.

6.2 Internet references

- 1) David Davis (2008). In a move to Cloud computing – VMware announces Virtual Data Center OS (VDC-OS). Available at <http://www.vmwarevideos.com/in-a-move-to-cloud-computing-vmware-announces-virtual-data-center-os-vdc-os>, accessed on 24th April 2011.
- 2) Dojo (2011). Dojo Toolkit Documentation: `dojo.xhrGet`. Available at <http://dojotoolkit.org/reference-guide/dojo/xhrGet.html#dojo-xhrget>, accessed on 12th July 2011.
- 3) Eclipse contributors (2010). Eclipse documentation: Eclipse platform overview. Available at <http://help.eclipse.org/helios/index.jsp>, accessed on 4th April 2011.
- 4) Eclipsepluginsite.com (2008). Eclipse Plugin Development TUTORIAL. Available at <http://www.eclipsepluginsite.com/index.html>, accessed on 24th Mar 2011.
- 5) Furey Anne. and Pottjewijd Arjan (2001). Integrated Development Environment (IDE) definition: Available at <http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment>, accessed on 15th Mar 2011.

- 6) Google (2011a). Google Web Toolkit: Developing with Google Web Toolkit. Accessed at <http://code.google.com/webtoolkit/overview.html>, accessed on 4th April 2011.
- 7) Google (2011b). Google code: Authentication and Authorization for Google APIs. Available at <http://code.google.com/apis/accounts/>, accessed on 4th April 2011.
- 8) Google (2011c). Google code: Datastore Overview. Available at <http://code.google.com/appengine/docs/java/datastore/>, accessed on 4th April 2011.
- 9) Itai Raz (2011). Introduction to Windows Azure AppFabric blog posts series – Part 3: The Middleware Services Continued. Available at <http://blogs.msdn.com/b/windowsazureappfabric/archive/2011/02/23/introduction-to-windows-azure-appfabric-blog-posts-series-part-3-the-middleware-services-continued.aspx>, accessed on 28th June 2011.
- 10) Iyogi Technical Services (2011). Windows® Azure: A Cloud Computing Medium. Available at <http://windows7.iyogi.com/news/windows-azure-cloud-computing/>, accessed on 2nd July 2011.
- 11) JSON (2010). Introducing JSON. Available at <http://www.json.org/>, accessed on 24th June 2011.
- 12) Microsoft (2011). Windows Azure AppFabric: Next-Generation Cloud Middleware Platform. Available at: <http://www.microsoft.com/windowsazure/AppFabric/Overview/default.aspx>, accessed on 27th Mar 2011.
- 13) Nacion Arjay (2009). Speed-Up Software Development: Framework-Driven Development. Available at: <http://www.sourcecodester.com/blog/speed-software-development-framework-driven-development.html>, accessed on 26th Feb 2011.
- 14) OrangeScape (2010). OrangeScape runtime architecture. Available at <http://www.orangescape.com/platform/architecture/>, accessed on 25th Mar 2011.
- 15) Yousuf Sait (2011). Cloud Computing Concepts and Migration Strategies of an Application to Cloud. Available at <http://ysf1991.wordpress.com/2011/07/11/cloud-computing-concepts-and-migration-strategies-of-an-application-to-cloud/>, accessed on 24 Mar 2011.

- 16) Sonu (2011). Typical Java Development Environment. Available at <http://churmura.com/technology/computer-science/typical-java-development-environment/19503/>, accessed on 18th April 2011.
- 17) Swabey Pete (2010). Features in Development & Integration: In The Frame. Available at <http://www.information-age.com/channels/development-and-integration/features/1260983/in-the-frame.html>, accessed on 26th Feb 2011.
- 18) VMware (2011). Cloud Foundry. No Obstacles: Introducing Cloud Foundry. Available at <http://www.cloudfoundry.com/>, accessed on 25th Mar 2011.
- 19) Wayne Schulz (2009). What is SaaS, Cloud Computing, PaaS and IaaS? Available at: <http://www.s-consult.com/2009/08/04/what-is-saas-cloud-computing-paas-and-iaas/>, accessed on 19th Mar 2011.
- 20) WOLF Frameworks (2011). WOLF Application Platform Technology. Available at <http://www.wolfframeworks.com/platform.asp>, accessed on 25th Mar 2011.

CHAPTER 7: APPENDICES

7.1 Key Servlet Functions

7.1.1 Common Java Code Library (iceUtils.java)

```
package icepad;

import java.util.List;
import java.util.logging.Logger;
import com.google.appengine.api.datastore.DatastoreService;
import com.google.appengine.api.datastore.DatastoreServiceFactory;
import com.google.appengine.api.datastore.Entity;
import com.google.appengine.api.datastore.EntityNotFoundException;
import com.google.appengine.api.datastore.FetchOptions;
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;
import com.google.appengine.api.datastore.Query;
import com.google.appengine.api.datastore.Text;
import com.google.appengine.api.datastore.Query.SortDirection;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

public class iceUtils {
    final static Logger log =
Logger.getLogger(FileTemplate.class.getName());
    final static UserService userService =
UserServiceFactory.getUserService();

    public static User getUser(){
        return userService.getCurrentUser();
    }

    //get a list of the files in a project
    public static List<Entity> listChildFiles(Key parentKey){
        DatastoreService datastore =
DatastoreServiceFactory.getDatastoreService();
        Query query = new Query("projectFile", parentKey);
        query.addSort("fileName", SortDirection.ASCENDING);
        return
datastore.prepare(query).asList(FetchOptions.Builder.withLimit(20));
    }

    public static Key getWorkspaceKey()
    {
        User user = getUser();
        return KeyFactory.createKey("Workspace", user.getNickname());
    }

    //Load the file contents given the file key
    public static String loadFileKey(String fileKeyStr){
```

```

        DatastoreService          datastore
DatastoreServiceFactory.getService();
        Key fileKey = KeyFactory.stringToKey(fileKeyStr);
        //load the file contents
        try {
            Entity pFile = datastore.get(fileKey);
            return ((Text)
pFile.getProperty("fileContent")).getValue();
        }
        catch(EntityNotFoundException e){
            log.info("Entity not found: " + e.getMessage());
            return "";
        }
    }
}

```

7.1.2 HTTP Get / Post Methods

```

/** Servlet methods for handling standard/ajax post and get form commands**/
//Save the code or project to appengine datastore
public void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
    //We implement one entity group user using AppsDatastore
    String postOption = req.getParameter("postOption");
    ///Post option P: Project, F: File U: User Settings
    if (postOption.equals("P")) {
        String projectName = req.getParameter("dlname");
        String projectType = req.getParameter("dltype");

        Key key = saveProject(projectName, projectType);
        resp.setContentType("text");
        String keyString = KeyFactory.keyToString(key);

        resp.getWriter().println(keyString);
    }
    else if (postOption.equals("F")){
        String fileKey = req.getParameter("fileKey");
        String fileContent = req.getParameter("code");
        if (fileKey != null)
            updateFile(fileKey, fileContent);
        else {
            String fileName = req.getParameter("filename");
            String fileType = req.getParameter("filetype");
            String projectName = req.getParameter("projectName");
            String parentFolder = req.getParameter("folderKey");
            //if content is null load the template file and save
            if ((fileContent.trim().length() == 0)){
                FileTemplate flTemplate = new FileTemplate(fileName);
                fileContent = flTemplate.loadTemplate();
            }

            saveFile(projectName, fileName, fileType,
parentFolder, fileContent);
            //load file template as defined

```

```

        if (fileContent.length() > 0){
            resp.setContentType("text");
            resp.getWriter().println(fileContent);
        }
    }
}
else if (postOption.equals("FL")){
    String folderName = req.getParameter("folderName");
    String projectName = req.getParameter("projectName");
    String parentFolder = req.getParameter("folderKey");

    saveFolder(projectName, folderName, parentFolder);
}
}

```

```

//Handle get requests for file load and project listing
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
    //get the user logged in
    UserService userService = UserServiceFactory.getUserService();
    User user = userService.getCurrentUser();

    //check if there is a projectfile
    String option = req.getParameter("p");
    String respdata = ""; //this will contain our JSON data

    if (option.equals("F"))//load specific file
    {
        //load the file using file key
        String fkey = req.getParameter("fkey");
        respdata = iceUtils.loadFileKey(fkey);
    }
    else if (option.equals("D"))//Delete the file
    {
        String fname = req.getParameter("fname");
        deleteFile(user, fname);
        return;
    }
    else if (option.equals("DFL"))//delete folder
    {
        String fkey = req.getParameter("fkey");
        deleteFolder(fkey);
        return;
    }
    else if (option.equals("DP"))//Delete project
    {
        String projectName = req.getParameter("pname");
        deleteProject(user, projectName);
        return;
    }
    else if (option.equals("DL"))//Download project
    {
        String projectName = req.getParameter("pname");

        ProjectDownload pDownload = new
ProjectDownload(projectName){
    pDownload.downloadProject(resp);
    return;
}

```

```

    }
    else if ((option.equals("L")) || (option.equals("C"))) //load
projects tree L or Combo C
    {
        List<Entity> projects = getProjects(user);
        respdata = loadProjects(projects,option);
    }
    //send the response data back - Either code value or projects
JSON object
    //log.info("JSON class returned for processing:\n"+respdata);
    resp.getWriter().println(respdata);
}

```

7.1.3 Saving a File to Virtual File System

```

private void saveFile(String projectName, String fileName, String fileType,
String parentFolder, String fileContent){
    DatastoreService datastore =
DatastoreServiceFactory.getDatastoreService();
    Text txContent = new Text(fileContent);
    Entity projectFile = null;
    Date date = new Date();
    //create a new entity
    Key projectKey;
    User user = getUser();
    if (parentFolder.trim().length() == 0){
        projectKey
KeyFactory.createKey(iceUtils.getWorkspaceKey(), "Projects", projectName);
    }
    else {
        projectKey = KeyFactory.stringToKey(parentFolder);
    }
    projectFile = new Entity("projectFile", fileName, projectKey);
    //We implement one entity group per file and reference the
project as the parent
    projectFile.setProperty("Owner", user);
    projectFile.setProperty("lastModified", date);
    projectFile.setProperty("fileType", fileType);
    projectFile.setProperty("fileName", fileName);
    projectFile.setProperty("fileContent", txContent);
    datastore.put(projectFile);
}

```

7.1.4 Loading Projects using JSON objects

```

private String loadProjects(List<Entity> projects, String loadOption)
{
    String respdata = ""; //this will contain our JSON data
    int seq = 0;
    //return the list of files in JSON format for DOJO tree loading
    //initialize the JSON structure
    respdata = "{\midentifier:'id',\nlabel:'fname',\nitems:[\n";
    //get all projects
    for (Entity project : projects) {
        String projectName = (String)
project.getProperty("projectName");
    }
}

```

```

        String                projectName                =                (String)
project.getProperty("projectType");
        if (seq>0) respdata += ",\n";
        respdata
String.format("{id:'%s',fname:'%s',ftype:'%s',type:'project',\nchildren:[",
                projectName, projectName, projectType);
        //load the files under the project
        if (loadOption.equals("C"))
            respdata += "]\n";          // ignore child references
        else
            respdata += loadProjectFiles(project);
        seq++;
    }
    return respdata + "]\n";
}
}

```

7.1.5 Project Download to OS File System

```

package icepad;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Logger;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;

import com.google.appengine.api.datastore.Entity;
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;

//Class to download project files - Maps virtual file system to real OS file
system

public class ProjectDownload {
    final Logger log = Logger.getLogger(FileTemplate.class.getName());
    private String project = "";

    public ProjectDownload(String projectName) {
        project = projectName;
    }

    private ArrayList<String[]> loadProjectPath(
        ArrayList<String[]> xFilePath, String path, Key projectKey)
    {
        ArrayList<String[]> result = xFilePath;
        List<Entity> projectFiles = iceUtils.listChildFiles(projectKey);
        if (projectFiles.isEmpty())
        {
            return result;
        }
    }
}

```



```

//get children name and keys
for (Entity projectFile:projectFiles){
    //skip grandchildren and same entity
    if (!projectFile.getParent().equals(projectKey) ||
projectFile.getKey().equals(projectKey))
        continue;
    String[] fileArray = new String[3];
    fileArray[0] = (String)projectFile.getProperty("fileName");
    fileArray[1] = path+File.separator+ fileArray[0];
    if (projectFile.getProperty("fileType").equals("folder"))
    {
        result = loadProjectPath(result, fileArray[1],
projectFile.getKey());
        continue;
    }
    fileArray[2]
KeyFactory.keyToString(projectFile.getKey());
    result.add(fileArray);
}
return result;
}

//download project - convert files from strings to code files
public void downloadProject(HttpServletRequest response)
{
    Key projectKey = KeyFactory.createKey(iceUtils.getWorkspaceKey(),
"Projects", project);
    ArrayList<String[]> proPath = new ArrayList<String[]>();
    proPath = loadProjectPath(proPath, project, projectKey);
    if (proPath.isEmpty()){
        return;
    }
    //Set the content header
    response.setContentType("application/zip");
    response.setHeader("Content-
Disposition", "attachment;filename="+project+".zip");
    //Initialize main streams
    ServletOutputStream out = null;
    InputStream in = null;
    ZipOutputStream zOut = null;
    StringBuffer sb = null;
    //loop the files in a project -- for each file in the project
    try{
        //project level streams
        out = response.getOutputStream();
        zOut = new ZipOutputStream(out);
        for (String[] fPath: proPath){
            //log.info(fPath[0]+":"+fPath[1]);
            //Write a Zip stream with project files
            String stFile = iceUtils.loadFileKey(fPath[2]);
            sb = new StringBuffer(stFile);
            //Zip process
            in =
new
ByteArrayInputStream(sb.toString().getBytes("UTF-8"));
            byte[] data = new byte[4096];
            //create a new zip entry per file

```

```

        zOut.putNextEntry(new ZipEntry(fPath[1]));

        int bytesRead;
        while((byteRead = in.read(data, 0, 4096)) != -1){
            zOut.write(data, 0, byteRead );
        }
        zOut.closeEntry();
        in.close();
    }
    zOut.flush();
    zOut.close();
}
catch(IOException e){
    log.info("IO Exception: " + e.getMessage());
}
catch(UnsupportedOperationException e2){
    log.info("Unsupported Operation exception: " +
e2.getMessage());
}
}
}

```

7.2 Javascript Code

7.2.1 IDE Code Completion

```

function startComplete() {
    // We want a single cursor position.
    if (editor.somethingSelected()) return;
    // Find the token at the cursor
    var cur = editor.getCursor(false), token = editor.getTokenAt(cur), tprop
= token;
    // If it's not a 'word-style' token, ignore the token.
    if (!/^[^\w$_]*$/ .test(token.string)) {
        token = tprop = {start: cur.ch, end: cur.ch, string: "", state:
token.state,
            className: token.string == "." ? "js-property" :
null};
    }
    // If it is a property, find out what it is a property of.
    while (tprop.className == "js-property") {
        tprop = editor.getTokenAt({line: cur.line, ch: tprop.start});
        if (tprop.string != ".") return;
        tprop = editor.getTokenAt({line: cur.line, ch: tprop.start});
        if (!context) var context = [];
        context.push(tprop);
    }
    var completions = getCompletions(token, context);
    if (!completions.length) return;
    function insert(str) {
        editor.replaceRange(str, {line: cur.line, ch: token.start}, {line:
cur.line, ch: token.end});
    }
}

```

```

// When there is only one completion, use it directly.
if (completions.length == 1) {insert(completions[0]); return true;}

// Build the select widget
var complete = document.createElement("div");
complete.className = "completions";
var sel = complete.appendChild(document.createElement("select"));
sel.multiple = true;
for (var i = 0; i < completions.length; ++i) {
    var opt = sel.appendChild(document.createElement("option"));
    opt.appendChild(document.createTextNode(completions[i]));
}
sel.firstChild.selected = true;
sel.size = Math.min(10, completions.length);
var pos = editor.cursorCoords();
complete.style.left = pos.x + "px";
complete.style.top = pos.yBot + "px";
document.body.appendChild(complete);
// Hack to hide the scrollbar.
if (completions.length <= 10)
    complete.style.width = (sel.clientWidth - 1) + "px";

var done = false;
function close() {
    if (done) return;
    done = true;
    complete.parentNode.removeChild(complete);
}
function pick() {
    insert(sel.options[sel.selectedIndex].value);
    close();
    setTimeout(function() {editor.focus();}, 50);
}
connect(sel, "blur", close);
connect(sel, "keydown", function(event) {
    var code = event.keyCode;
    // Enter and space
    if (code == 13 || code == 32) {event.stopPropagation(); pick();}
    // Escape
    else if (code == 27) {event.stopPropagation(); close(); editor.focus();}
    else if (code != 38 && code != 40) {close(); editor.focus();}
setTimeout(startComplete, 50);
});
connect(sel, "dblclick", pick);

sel.focus();
// Opera sometimes ignores focusing a freshly created node
if (window.opera) setTimeout(function() {if (!done) sel.focus();}, 100);
return true;
}

var stringProps = ("charAt charCodeAt indexOf lastIndexOf substring substr
slice trim trimLeft trimRight " +
    "toUpperCase toLowerCase split concat match replace
search").split(" ");
var arrayProps = ("length concat join splice push pop shift unshift slice
reverse sort indexOf " +

```

```

        "lastIndexOf every some filter forEach map reduce
reduceRight ").split(" ");
    var funcProps = "prototype apply call bind".split(" ");
    var keywords = ("break case catch continue debugger default delete do else
false finally for function " +
        "if in instanceof new null return switch throw true try
typeof var void while with").split(" ");

function getCompletions(token, context) {
    var found = [], start = token.string;
    function maybeAdd(str) {
        if (str.indexOf(start) == 0) found.push(str);
    }
    function gatherCompletions(obj) {
        if (typeof obj == "string") forEach(stringProps, maybeAdd);
        else if (obj instanceof Array) forEach(arrayProps, maybeAdd);
        else if (obj instanceof Function) forEach(funcProps, maybeAdd);
        for (var name in obj) maybeAdd(name);
    }

    if (context) {
        // If this is a property, see if it belongs to some object we can
        // find in the current environment.
        var obj = context.pop(), base;
        if (obj.className == "js-variable")
            base = window[obj.string];
        else if (obj.className == "js-string")
            base = "";
        else if (obj.className == "js-atom")
            base = 1;
        while (base != null && context.length)
            base = base[context.pop().string];
        if (base != null) gatherCompletions(base);
    }
    else {
        // If not, just look in the window object and any local scope
        // (reading into JS mode internals to get at the local variables)
        for (var v = token.state.localVars; v; v = v.next) maybeAdd(v.name);
        gatherCompletions(window);
        forEach(keywords, maybeAdd);
    }
    return found;
}

```

7.2.2 JavaScript File Management

```

//Simple editor management
function setFileAttr()
{
    fileAttr.filename = dojo.byId("xfilename").value;
    fileAttr.filetype = 'dijit.byId("filetype").get('value');
    fileAttr.fileproject = dojo.byId("fileproject").value;
    fileAttr.existing = false;
    //confirm the file is in the required format
    var strlen = fileAttr.filename.length;
    var typelen = fileAttr.filetype.length;

```

```

//Ensure the file has the right extension, if not append it
if ((fileAttr.filename.lastIndexOf(".") + fileAttr.filetype) != (strlen
- (typelen+1)))) {
    fileAttr.filename += "." + fileAttr.filetype;
}
//createFileTplt();
saveCode();
editor.setOption("mode", loadFileMode(fileAttr.filetype));

//Select the theme to display in the editor
function selectTheme(node) {
    var theme = node.options[node.selectedIndex].innerHTML;
    editor.setOption("theme", theme);
}
//status update
function updateStatus(cont) {
    dojo.byId("ajaxstatus").innerHTML = cont;
}

```

7.2.3 Javascript Save Code (AJAX)

```

//Save the editor content to the server
function saveCode()
{
    if ((fileAttr.filename.length == 0) || (fileAttr.filetype.length ==
0) || (fileAttr.fileproject.length == 0)) {
        showFileDialog();
        return;
    }
    //AJAX Save the file
    if (fileAttr.fileKey.length > 0) {
        fContent
        {postOption:"F",fileKey:fileAttr.fileKey,code:editor.getValue()};
    }
    else {
        fContent
        {postOption:"F",projectName:fileAttr.fileproject,folderKey:fileAttr.folderKey
,
filename:fileAttr.filename,filetype:fileAttr.filetype,code:editor.getValue()
};
    }

    var xhrArgs = {
        url: "/save",
        timeout:20000,
        handleAs: "text",
        //form: "fmedit",
        content:fContent,
        load: function(data) {
            if (data.length > 0)
                editor.setValue(data);
            updateStatus("File saved successfully");
            window.setTimeout("updateStatus('')", 10000);
            //update the editors filename to match the current name

```

```

        //fileAttr.filename = fname.value;
        fileAttr.modified = false;
        if (!fileAttr.existing) {
            refreshTree();
        }
    },
    error: function(error) {
        updateStatus("An unexpected error occurred: " + error);}
}
    updateStatus("Saving file.....");
//Call the asynchronous xhrPost
var deferred = dojo.xhrPost(xhrArgs);
}

```

7.3 User Survey

7.3.1 Survey Summary

1. Do you use a Integrated Development Environment(IDE)for Rapid Application Development(RAD)?

Yes

2. Do you use any software frameworks for enterprise development?

Yes

3. Are there any situations that call for traditional application development methodologies (non RAD)?

Yes

4. Would you consider using a cloud based development environment?

Yes

5. What would be your concerns in using cloud based development environment?

Data Security

Connectivity issues

Loss of control

System Availability

6. What features of an IDE would you want to see in a cloud based IDE?

File management

Project management

Task management

Multiuser environment

Plugin management

Visual design

Code generation

Code completion

Team Management Version Control Code folding Love the themes esp night

7. What software frameworks would you want implemented in the cloud based IDE?

restful/soap web services integration

8. Do you have any plans to develop a cloud based application - Software as a service(SaaS)?

Yes

9. What current tools would you consider for cloud application development?

Mysql java ssh ftp tomcat webserver hard disk space

10. Given similar features on the cloud development platform as your computer based development environment, would you switch fully to a cloud based development environment?

Yes

7.3.2 Survey Responses Summary

Response Summary

Total Started Survey: 16
Total Completed Survey: 16 (100%)

PAGE: ICECLOUD IDE USER REVIEW

1. Do you use a Integrated Development Environment(IDE)for Rapid Application Development(RAD)?

Create Chart Download

	Response Percent	Response Count
Yes	87.5%	14
No	12.5%	2
answered question		16
skipped question		0

2. Do you use any software frameworks for enterprise development?

Create Chart Download

	Response Percent	Response Count
Yes	93.8%	15
No	6.3%	1
answered question		16
skipped question		0

3. Are there any situations that call for traditional application development methodologies (non RAD)?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
Yes		93.8%	15
No		6.3%	1
		answered question	16
		skipped question	0

4. Would you consider using a cloud based development environment?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
Yes		75.0%	12
No		25.0%	4
		answered question	16
		skipped question	0

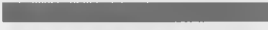







5. What would be your concerns in using cloud based development environment?

[Create Chart](#) [Download](#)

		Response Percent	Response Count
None		6.7%	1
Data Security		66.7%	10
Connectivity issues		73.3%	11
Loss of control		66.7%	10
System Availability		66.7%	10
Limited features		26.7%	4
		Other (please specify) Show Responses	4
		answered question	15
		skipped question	1

6. What features of an IDE would you want to see in a cloud based IDE?

 Create Chart  Download

		Response Percent	Response Count
File management		73.3%	11
Project management		66.7%	10
Task management		60.0%	9
Multuser environment		73.3%	11
Plugin management		60.0%	9
Visual design		66.7%	10
Code generation		66.7%	10
Code completion		66.7%	10
		Other (please specify) Show Responses	5
		answered question	15
		skipped question	1

7. What software frameworks would you want implemented in the cloud based IDE?

Download

Response Count

Hide Responses 16

Responses (16) Text Analysis My Categories (0)

GOLD FEATURE: Text Analysis allows you to view frequently used words and phrases, categorize responses and turn open-ended text into data you can really use. To use Text Analysis, upgrade to a GOLD or PLATINUM plan.

Learn More Upgrade »

Showing 16 text responses

No responses selected

Web application framework Process framework such as ITIL Conceptual framework

8/15/2011 8:27 AM View Responses

None

8/10/2011 11:41 AM View Responses

Development software framework and Service software framework

8/9/2011 8:21 PM View Responses

Agile,REST

8/9/2011 7:55 PM View Responses

Android

8/9/2011 7:36 PM View Responses

PHP JAVA

8/9/2011 7:34 PM View Responses

8. Do you have any plans to develop a cloud based application - Software as a service(SaaS)?

Create Chart Download

	Response Percent	Response Count
Yes	81.3%	13
No	18.8%	3
answered question		16
skipped question		0

9. What current tools would you consider for cloud application development?

Download

Response Count

Hide Responses 16

Responses (16) Text Analysis My Categories (0)

GOLD FEATURE: Text Analysis allows you to view frequently used words and phrases, categorize responses and turn open-ended text into data you can really use. To use Text Analysis, upgrade to a GOLD or PLATINUM plan.

Learn More

Upgrade »

Showing 16 text responses

No responses selected

web based tools

8/18/2011 9:29 PM View Responses

Google App Engine Amazon - EC2

8/15/2011 10:15 AM View Responses

Dot Net; Crystal reporting Data Dynamics

8/15/2011 8:27 AM View Responses

Php, Java

8/10/2011 11:41 AM View Responses

Tools for Agile Application Development

8/9/2011 8:21 PM View Responses

Simple tools

8/9/2011 7:55 PM View Responses

10. Given similar features on the cloud development platform as your computer based development environment, would you switch fully to a cloud based development environment?

Create Chart

Download

	Response Percent	Response Count
Yes	68.8%	11
No	31.3%	5
answered question		16
skipped question		0