© Communications of the Arab Computer Society, Vol.9, No.1, December 2016, ISSN:2090-102X

Pattern Oriented Analysis for Web Based Applications on Cloud

Amar Ibrahim E.Sharaf Eldein*

*College of Graduate Studies, Sudan University of Science and Technology

Khartoum, Sudan

amarcoibrahim@hotmail.com, amarsharaf@sust.edu.sd

Abstract

Cloud computing has emerged as important technology that requires new ways of architecting web applications on cloud. Migration web based applications to clouds faced requirements changes during developments, which let to design complexity problem. Pattern oriented analysis approach for web base application in goal simplified requirements analysis and refinement, facilitate selecting cloud design patterns to provide better design solutions. This approach help developers to know how web application structured and use quality requirements with selected patterns applied on cloud application development process. This paper focuses on highlighting the cloud base architectural patterns for building cloud Applications and introduces approach for ensuring web based cloud application requirements components mapped to scenario based, decide what patterns that can be selected to use. A Student Academic Records Result System for student Transcript has been used as a case study.

Keywords: Web application; Cloud Patterns; Cloud Application; Requirements Analysis; Pattern oriented Analysis;

1. Introduction

Web based applications architecture was constructed based on client- server patterns. New trend for web applications migrate to the clouds [1]. To reduce effort of web application and easier development for cloud applications, require understanding requirements components [2]. Requirements analysis is the first stage for architecting web based applications development, analysis structures requires a lot of effort from all stakeholders.

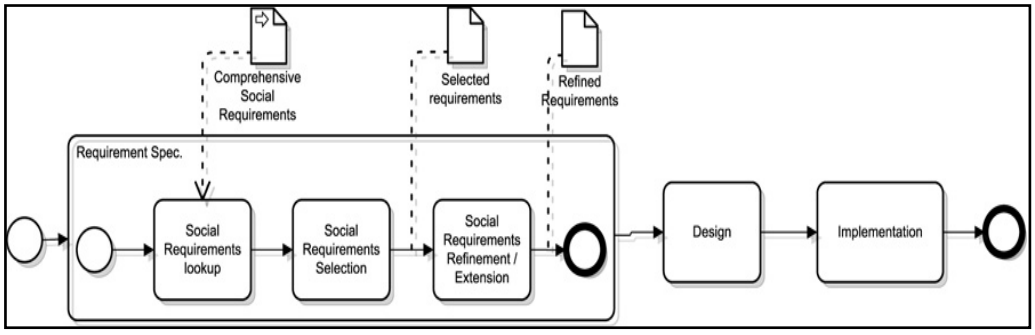

Figure 1. Overview of pattern- based development approach for Web application[3].

Cloud applications are recognized as a trend for the next generation of existing Web applications, and hence how to migrate these Web applications to the clouds becomes a desired field [4].Cloud changing the way of developing web applications, requirements and analysis are major challenges of software development; main problem is to describe the requirements changes during analysis process for web applications and selecting cloud design patterns to be developed. One of existing

cloud solution approach for web based applications on cloud is the use of patterns. Cloud Patterns based on requirements analysis used to describe cloud application design goals. The core contributions of this paper is proposed methodology for pattern oriented analysis, used to find suitable cloud design patterns from identify application components requirements; This paper addressed cloud computing design patterns; analysis of existing web based requirements; patterns selection from requirement specification as a process upon this taxonomy; refine and analyze requirements with patterns; create lower level design of abstraction from selected cloud design patterns; and proposed method in a case study. Pattern oriented analysis approach specifying and classifying web application requirements, then select based design patterns (Acquaintance) from patterns repository, next from specific requirement scenario process select suitable related patterns (Retrieval).Refinement for cloud patterns with application component requirements get candidates cloud design patterns selection to build, deploy and change several processes for cloud applications.

The paper is organized as follows: Section2 describes cloud application architecture design patterns taxonomy. Section 3 describes the research method used to support select required patterns. In section 4 describes the case study. Finally in section 5 we conclude the paper with future work and the appendix that contain the general cloud overview design patterns.

2. Cloud Application Architecture Design Patterns

The following section address design patterns taxonomy for cloud application architecture.

2.1 Design Patterns

Design patterns are the core of a solution to a problem, capturing best practices on how system applications should be designed. The benefits of using patterns include development of reuse of software architecture [5], support to make good design decisions. Also describe how different application components can be implemented, capture expert knowledge, reduced costs, and improve developer communication. In addition patterns help complex resources solution. Deploying these solutions to develop complex systems involves integration issues and iterative development [6].

2.2 Cloud Application Architecture Patterns

Cloud application architecture patterns describe how applications have to be designed to benefit from a cloud environment, also described how applications themselves can be offered as configurable cloud services [7]. In addition pattern approach provide solutions to cloud application challenges and requirements [8], that benefit cloud application development, reduce develop time, deploy multiple applications, test, configure and integrated solutions [9].

Based on **Christoph Fehling e.tal** [10]cloud design patterns based taxonomy as following phases, which describe abstract solutions to problems in how cloud application can build on top of an elastic platform.

2.2.1 Fundamental Cloud Architecture

Cover the fundamental architectural styles that architects and developers have to be aware of when building a cloud native application categorize to:-

A. ***Loose Coupling Architecture*:** communication separates application functionality from concerns of communication partners regarding their location, implementation platform, the time of communication, and the used data format. By using Broker solution that can communicate components and multiple integrated applications to decoupled from each other's.

B. ***Distributed Application Architecture*:** describe how application's functionality may decompose to be distributed among several resources. Cloud application solutions rely on divides provided functionality among multiple application components that can be scaled out independently,

redundant resources ensure that the unavailability of one resource does not affect the application as a whole.

2.2.2 Cloud Application component patterns

Patterns of this category refine how functionality of a cloud application can be implemented in separate components. Applications components are developed specifically for cloud offering and requirements because they are not explicitly specified [11].Cloud patterns characterized by three central patterns.

- ***User interface components:*** provide application functionality to users.
- ***Processing components:*** handle computational tasks.
- ***Data access components:*** handle data stored in storage offerings. They can deal with storage offerings at different cloud providers with different consistency levels.
  Data access components can further be adjusted to inherently support eventual consistency by abstracting data to hide that there may be data inconsistencies.

2.2.3 Multi-tenancy patterns

Describe how cloud applications and individual components can be shared by multiple customers on different levels of the application stack. Can support multiple client tenants simultaneously, that achieve the goal of cost effectiveness.

- ***The shared component***: provides functionality to different tenants without maintaining a notion of tenants itself.
- ***The tenant-isolated component***: does the same but ensures that tenants do not influence each other while they access shared functionality.
- ***The dedicated component pattern***: enables some functionality to be provided exclusively to tenants without sharing it with others.

2.2.4 Cloud integration patterns

Describe special application components to enable the communication across cloud boundaries, as applications are often not standalone and must be integrated with other cloud applications and non-cloud applications [12].

- ***A restricted data access component***: extends the functionality of the data access component to incorporate data obfuscation if sensitive data may not be retrieved completely from a less secure environment. Alternatively, data may be replicated between environments.

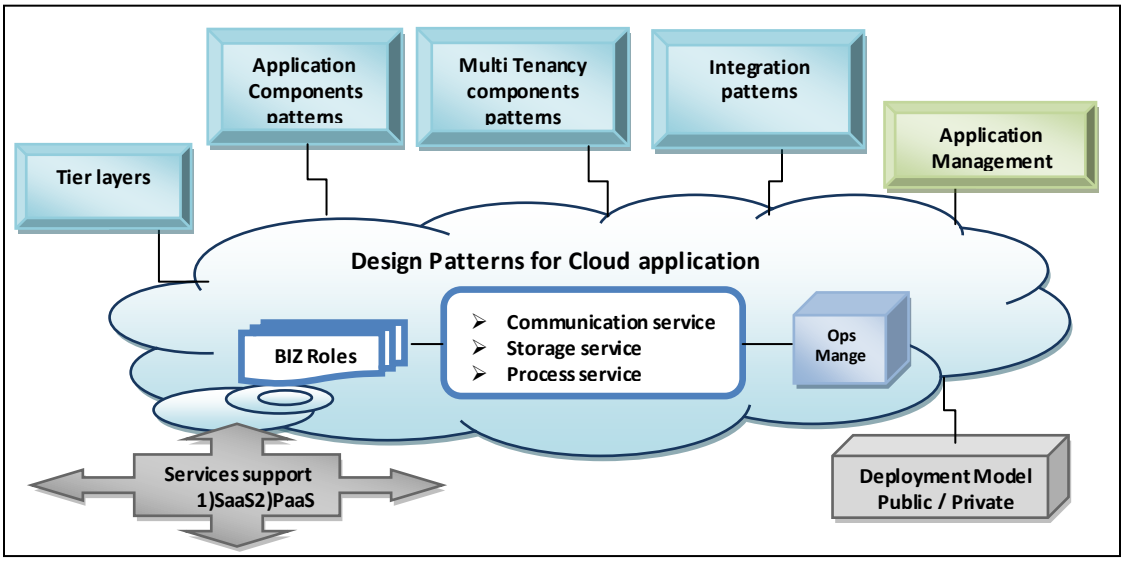

Figure 2. Design patterns for cloud applications architecture overview.

3. Methodology

Pattern based approach has been used to capture best practices as how applications should be designed [13].Pattern oriented analysis approach based on requirements analysis is used. Pattern oriented analysis consists of major steps, start from identify of existing web application architecture requirements; requirements analysis for defining a problem , scenario based for gathering specific requirements to identify set of relevant cloud patterns; adapted cloud include refinement to trace candidate selected cloud patterns to lower level of abstraction to give scalable design.

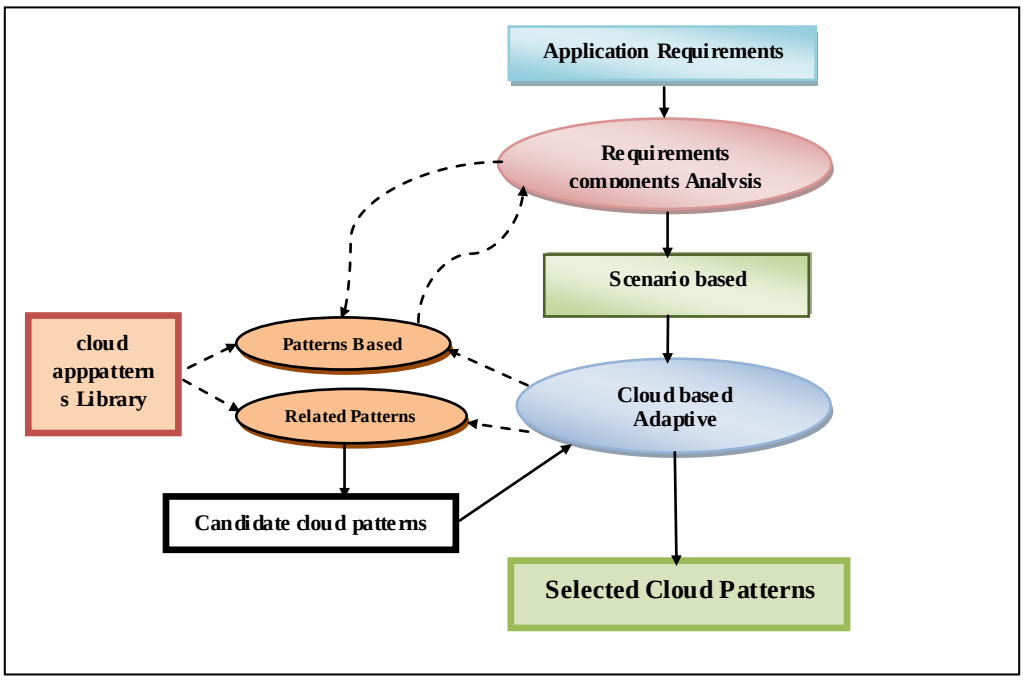

Figure 3.Overview of pattern oriented Analysis for scenario based.

The following subsections will discussed each phases of the proposed method with the help of UML diagram and using examples. Patterns oriented analysis approach steps as following:-

1) **Application Requirements:** Application requirements is a details descriptions of application services can be requires from a customers, system and software for a design or implementation.
2) **Requirements Components Analysis:** Analyze the existing application requirements to identify problems to be solved and to determine conceptual components.

Process: finding components to identify functionality identify problem and generate use case diagrams. Identify application component (logical level), if application is large analyzed into subsystems and subsystem analyzed into more details. Need to iterate with other analysis activities:-

- ***Acquaintance activity:*** to identify set of cloud design patterns will be used in application design (from pattern library) patterns data base activity.
- ***Retrieval***: retrieve from patterns repository related patterns, process focus on how to select given patterns, matching if related patterns should be selected, to set as candidate design patterns.

3) **Scenario based:** Set of conceptual components, use cases and scenarios diagram.
4) **Adapted cloud Patterns:** Select a set of cloud design patterns that will be used in the cloud architecture design of each component. Process defined solution matching by set of candidate patterns, need to:-
   - ***Iteration***: the acquaintance and retrieval activities.

- ***Specific cloud patterns library***: contain related patterns to the specific problem.
- ***Study relationship between patterns***: to know which patterns use other patterns; refines other patterns and conflict with another one.

Overall output selection of set of cloud design patterns will be used in design for the various parts of the system.

4. Case study: Migration a Web based applications to the Cloud

Cloud computing provides scalable and reliable computing services that can be benefit to migrate existing applications to the cloud. There is a little concern for migrating existing applications to cloud [14].This case study motivated by own experience with Students Academic Records Management System (SARMS) run on desktop application need to be migrate to cloud, to produce Students Result graduate certificates processes with different users functionality in the College with two branches. Application lacks of scalability and resource sharing capabilities [5].Goals to benefits from cloud design patterns, establish new patterns for application requirements to design a cloud based application that delivering Students Academic Results services. In addition allow features of cloud application such as accessibility, scalability and portability.

4.1 Application Requirements

Student information system is a software application for education establishment to manage student data, include Student Information Management System, Student Records System, Student Result System; these systems encompass a wide range of functions and capabilities, and therefore vary in size, scope, and capability [15].

Need to understand the requirement of Student Academic Records for online Results services. College branches should not focus on providing integrated branches student academic records on line but also getting student's requirements from applications.

The following requirements are described below, include:

- Providing up- to -date student data base that can be easily access.
- Providing an efficient to collect students' results error-free;
- Enabling processing and releasing of transcripts document in real time.
- Providing effective secure and protect students data against unforeseen disaster;
- providing decision making information about students' academic Result;
- Providing a seamless communication interface between students and the college branches [15].
- Providing portability for college branches applications.

4.2 Requirements Analysis

Optimized business organizations established from requirements analysis procedures and activities [16]. Requirements analysis are the most important and difficult activities [17]to identifying business tasks and methods. First need to redesign desktop application to be web based application, to be cloud native application, and then migrated to the cloud SaaS, to enables students to browse the Academic Result Records online.

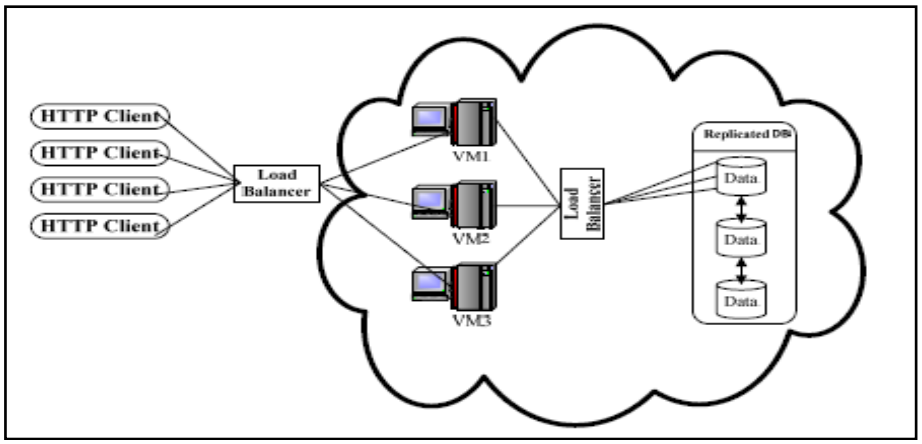

Figure 4. Cloud Application Architecture [5].

Referring to figure 4, requirements components for migration application on cloud computing classified to:-

(A)Stakeholders Identification

1) **Academic Affairs:** is a head quarter, responsible for aggregating different branches students result data base to help graduation transcript certificate procedures.
2) **Branch user:** who are entering student Exams marks and producing different Results services in each college branch.
3) **Academic Assistance:** responsible for prepare student academic Records in specific required styles.
4) **Students:** use the system to query Academic result details[18].

(B)Stakeholders Requirements

Requirements from different stakeholders changes during the development process.

1) **Academic Affairs**: in main college, will be given more powers (enable/disable/ update) than other users to ensure that the information entered is correct.
2) **Academic staff:** manage every branch students account, update and maintain students result. Also view student Academic result details for better understanding the student performance. The staff also gets the updates from the main college regarding any events occurring from exam section in the college.
3) **Academic Assist:** required replica of results data for managing student academic records.
4) **Student:** require authentication to handles the student login process, query to view Academic result by semester or current semesters.

(C) Requirements Category:

Requirements can be classified into following category to make easier design.

1) **Functional requirements analysis (FR):** explain what to be done by identifying the necessary activity to be accomplished by end user's perspective.
   A. **Cloud Platform:** provides various control surfaces such as process management, routing, logging, scaling, configuration, and deployment for building and operating an application.
   B. **Virtual Machine (VM)**: contains both the application server and the web server.
      1) **The web server**: handles the http requests, passes request to an appropriate program also send response viewing in the web browser.
      2) **The application server**: provides access to business logic for use by the client application programs.
   C. **Storage spaces:** are replicated geographically to avoid problems caused by failure of the storage space.

D. **NOSQL storage**: Dynamic simple DB, which is accessed via API.

2) Operational requirements

To keep process functioning over a period of time. Web based access for 24/7 by students and academic staff; API to gain complete control of the application.

3) Technical requirements (TCR)

Define issues that considered implementing the process, or creating the product.Migration technique: for moving Web application data to the cloud; Replication: The different web services runs on a virtual machine (VM).Deploy student information system with other tools; Resource management and configuration tools.

4) Transitional requirements (TSR)

Steps needed to implement the process. Indicates how the requirements are behave as the consequence of external requirements. HTTP protocol: improve various branches communication with the web server, via load balancer; load balancer: Exterior: intercepts requests from clients to distribute over web server (VM instance); Interior: to spread traffic over app server (stored to available data storage).

5) Non Functional requirements

Scalability: Application need to add/delete new resources as and when required; **Availability:** should be available for access not only in the college branch but outside of the college branches (deploying application on public cloud); **Security:** to enable secure communication to deploy Student Academic Records database on public cloud; to enable secure transfer of college branch to/ from platform; **Privacy:** Apart from security available on public cloud, restricted access to cloud service for some data is private for especial branch; **Usability:** Interface of Student Academic should be user friendly[19]; **Performance:** for helping the college in managing the whole database of the student studying in all branches; **Portability:** move application on different cloud platforms.

4.3 Scenario Based Cloud Migration

To better understand the needs of migration to cloud for case study, requirements mapped to UML scenarios, described a high level includes actors, use cases and sequence diagrams. Student records for final results collated using Microsoft Excel. Amount of time taken to manually prepare academic records through the Excel file to approved for transcript certificate procedures.

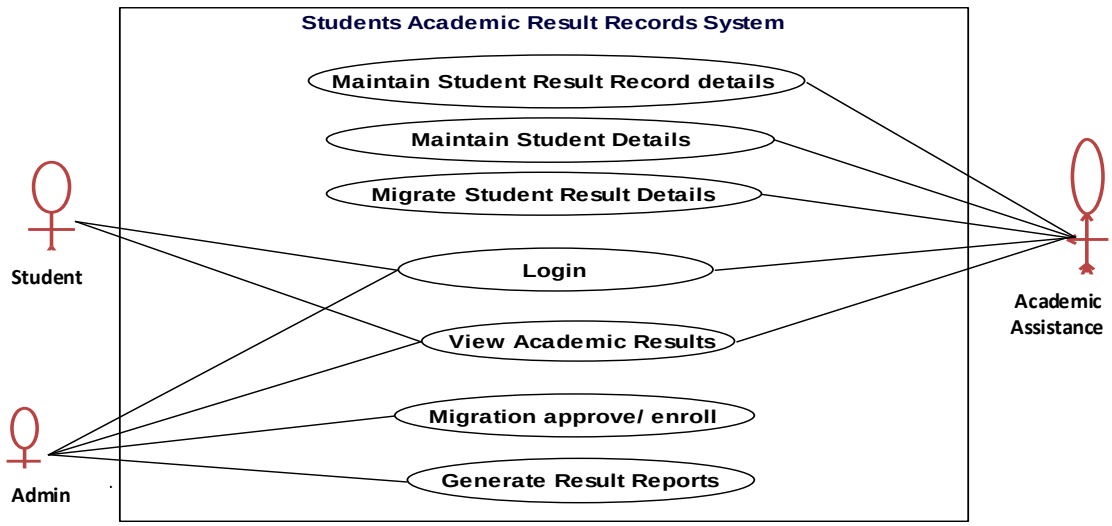

Fig 5. Use case diagram of the university Academic Result system.

Referring to figure 5, the use case diagram consists of actors, such as student, administrator, and Academic Assistance. The following describes the use cases components:-

- **Cloud platform**: hosts college branches (subsystems) components and implementing different services.
- **Login:** access Interface for administrator to view, verify Academic Records that need approval for enrollment; Academic Assistance: is a branch user maintain academic records functions deployed on a cloud; Students log on to the system and can browse the Result Record.
- **Academic Result Services:** responsible for most of the application's functionality, displays the current Academic Results offerings and their details, details include the semester subjects and degree.
- **Result inquiring agent:** manage workflow service components for subsystems. Forwards queering requests from user to Academic Result services and return back the result information; catches the preferred Academic Result information for students based on their preferences for their queering decisions.

4.4 Patterns Selection

Based on pattern oriented analysis process, set of design patterns solution selected for case study to development cloud application architecture. Acquaintance activity focus on cloud application component patterns and multi tenancy patterns, with concern on shared component pattern with different users requirements.

4.4.1 Cloud application component patterns

Retrieval activity candidate the following design patterns:-

- ***User Interface component:*** serve as bridge between different users and cloud environment, to access application and provide data from other sources. Benefits users are, Academic Affairs: mange system functionality; Branch user: understand academic result demand and can access specific Academic result; Student used interface to view Academic result details; In addition to cloud provider and operator can manage application services.
- ***Processing component:*** made configuration to support different requirements, handled by different academic result component. Provide relationship with user interface component (received users request), e.g. **Academic assistance:** can analyze the academic result of students to generate different reports styles which will be used for graduation certificate.
- ***Data access pattern:*** to store and access different data source provided by component services, related with user interface e.g. **Academic Record** of students will be access using user interface to public cloud, generated and reflected into cloud web application.

4.4.2 Multi tenancy patterns

To create an extensible, stable and robust multi tenancy application, developers have to understand how web application is structured and how it request is handled for each user of specific tenant [2]. Shared component patterns used to improve resource sharing and user's accessibility to the system, also reduce resource usage. Candidate related design patterns are:-

- ***The dedicated component pattern*** provides exclusive access to application components that provide critical functionality. Used by branch users to maintain Student Academic Records.
- ***The tenant-isolated component pattern*** represents implementation between the shared component and dedicated component approaches. Need to configure application to support students functionality access level for Academic Record services, need load balancer for request from user interface and spread over cloud services. Also need cloud provider to provide high degree of isolation for running process to branch application.

Table1. Summary for cloud application patterns selection.

#	Acquaintance	Retrieval	Solution
1	**Fundamental Architectures**	**Loose Coupling**	Makes the system flexible to run different components on different cloud. Broker used for communicating components ad decouple integrated applications.
2	**Cloud Application Components**	**Stateful Component**	A synchronized internal state; Replication Internal state.
3		**Stateless Component**	Storage Offerings (external); Increase Elasticity.
4		**User Interface Component**	Reduce Dependencies; e.g. Elastic Load Balancer: to spread traffic to web server auto scaling group.
5		**Processing Component**	Processing functions meet different customers Split into separate function blocks (stateless).
6		**Data Access Component**	Integrity of data and coordinates. Allow multiple customers access single instance of DB.
7		**Transaction-based Processor**	Ensure messages receives are processed successfully and altered data successfully after processing. Deliver services to enhance portability e.g. REST full API.
8	**Multi-Tenancy**	**Shared Component**	To improve resource sharing and user's accessibility to the system. Deploying components on selected clouds; Deploying solution mechanism over clouds; Optimize requirements for clouds.
9		**Tenant-isolated Component**	Ensure isolation between branches by controlling tenant access, processing performance used, and separation of stored data.
10		**Dedicated Component**	Dedicated application components are provided exclusively for each tenant using the application.

Selected patterns for application components and multi tenancy with related patterns facilitate developing and migrated Student Academic Records system to native cloud application based on methodology.

5. Conclusion and Future work

Web applications have permeated every aspect of people's life. Many developers migrates their Web applications to cloud platforms. Research achieved: Taxonomy for cloud application design patterns introduced; proposed pattern oriented analysis method based on requirements analysis. Approach to trace functional requirements with patterns to extract candidate cloud design patterns; Moreover, case study describe architecture requirements and present a method for the migration of Web based applications requirements into cloud application components, to identify selected design patterns discussed. Solution(s) drawbacks Acquaintance and retrieval patterns depends on analyst knowledge to select suitable design patterns; combine some functional requirements component, may increase the complexity of the application[14].Further study intend As future work shared selected patterns to get reviews and feedback , refined it and applied to case study; Patterns contributes to design phase, propose oriented design method to develop cloud application benefits from selected design patterns; In addition, will continue to deploy migration case study to suitable cloud platforms. Finally, requirements analysis is a key to select candidate cloud patterns for building, developing and deploying applications to cloud computing. Patten oriented analysis method expected to improve the requirements quality of migrated to cloud application, since few sentences are stated about it. Also help developers to design best cloud application in efficient manner.

References

[1] Pandey, Dhirendra, et al. "A Framework for modelling software requirements." International Journal of Computer Science 8 (2011): 1694-08141.

[2] Bien, Ngo Huy, and Tran Dan Thu. "Multi-tenant web application framework architecture pattern." Information and Computer Science (NICS), 2015 2nd National Foundation for Science and Technology Development Conference on. IEEE, 2015.
[3] Brambilla, M. A. R. C. O. "From requirements to implementation of ad-hoc social Web applications: an empirical pattern-based approach." IET software 6.2 (2012): 114-126.
[4] Lin, Jyhjong, LendyChaoyu Lin, and Shiche Huang. "Migrating Web Applications to Clouds with Cloud-Based MVC Framework." Computer, Consumer and Control (IS3C), 2016 International Symposium on. IEEE, 2016.
[5] Adewojo, A. A., et al. "Cloud Deployment Patterns: Migrating a Database Driven Application to the Cloud using Design Patterns." Proceedings of the World Congress on Engineering and Computer Science. Vol. 1. San Francisco USA 2015.
[6] Yacoub, Sherif M., and Hany H. Ammar. "UML support for designing software systems as a composition of design patterns." International Conference on the Unified Modeling Language. Springer Berlin Heidelberg, 2001.
[7] Zhao, Liang, et al. "An architecture framework for application-managed scaling of cloud-hosted relational databases." Proceedings of the WICSA/ECSA 2012 Companion Volume. ACM, 2012.
[8] Erl, Thomas, Robert Cope, and Amin Naserpour. Cloud computing design patterns. Prentice Hall Press, 2015.
[9] Chiara Brandle, Vanessa Grose, Mi Young Hong, Jeffrey Imholz, PrashanthKaggali, Marco Mantegazza." Cloud Computing Patterns of Expertise" IBM, January 2014.
[10] Fehling, Christoph, et al. "An architectural pattern language of cloud-based applications." Proceedings of the 18th Conference on Pattern Languages of Programs. ACM, 2011.
[11] Fehling, Christoph, et al. "Pattern-based development and management of cloud applications." Future Internet 4.1 (2012): 110-141.
[12] Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, Peter Arbitter,Springer. "Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications" Springer, 2013.
[13] Malcher, Viliam. "Design Patterns in Cloud Computing." 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). IEEE, 2015.
[14] Han, Dan, and EleniStroulia. "Federating Web-Based Applications on a Hierarchical Cloud." 2014 IEEE 7th International Conference on Cloud Computing. IEEE, 2014.
[15] Obasi, Nmaju, E. O. Nwachukwu, and C. Ugwu. "A Novel Web-Based Student Academic Records Information System." West African Journal of Industrial and Academic Research 7.1 (2013): 31-47.
[16] DhirendraPandey, UgrasenSuman., A.K. Ramani. "A Framework for Modelling Software Requirements", International Journal of Computer Science Issues,Vol. 8, Issue 3, No. 1, May 201, ISSN (Online): 1694-08141, www.IJCSI.org, 2011.
[17] Lin, Jyhjong, LendyChaoyu Lin, and Shiche Huang. "Migrating Web Applications to Clouds with Cloud-Based MVC Framework." Computer, Consumer and Control (IS3C), 2016 International Symposium on. IEEE, 2016.
[18] Bharamagoudar, S. R., R. B. Geeta, and S. G. Totad. "Web based student information management system." International Journal of Advanced Research in Computer and Communication Engineering 2.6 (2013): 2342-2348.
[19] Kolekar, Sucheta, Radhika M. Pai, and ManoharaPai MM. "Adaptive user interface for e-learning applications based on learning styles using Web Logs analysis: A hybrid cloud architecture." TENCON 2015-2015 IEEE Region 10 Conference. IEEE, 2015.
[20] Adewojo, A. A., J. M. Bass, and I. K. Allison. "Enhanced cloud patterns: A case studyof multi-tenancy patterns." Information Society (i-Society), 2015 International Conference on. IEEE, 2015.
[21] Adikara, Fransiskus, BayuHendradjaya, and BenhardSitohang. "Requirements refinements and analysis with case-based reasoning techniques to reuse the requirements." Electrical Engineering and Informatics (ICEEI), 2015 International Conference on. IEEE, 2015.

Appendix

Table 2. Cloud application architecture patterns summary.

#	Category	Pattern Name	Problem	Solution
1	Fundamental Cloud Architectures	**Loose Coupling**	Reduce dependencies between Distributed Applications and between individual components.	Broker: to communicating components and decouple multiple integrated applications from each other.
2		**Distributed Application**	How can application functionality be decomposed to be handled by separate application components	The functionality of the application is divided into multiple independent components that provide a certain function
3	Cloud Application Components	**Stateful Component**	A synchronized internal state	Replication Internal state
4		**Stateless Component**	Increase Elasticity(Failures)	Storage Offerings(external)
5		**User Interface Component**	Reduce Dependencies	Elastic Load Balancer
6		**Processing Component**	Processing functions meet different customers' requirements	split into separate function blocks (stateless)
7		**Batch Processing Component**	asynchronous processing delayed	stored (asynchronous) until conditions are optimal for their processing
8		**Data Access Component**	Hide & isolate data while ensure data configurability	Integrity of data and coordinates
9		**Data Abstractor**	How consistent data be presented and inconsistencies hidden from other application (components and users)?	Consistent state is unknown by approximating values or abstracting them into more general ones, such as change tendencies (increase / decrease).
10		**Idempotent Processor**	presented of consistent data Duplicate function execution	adjusted to allow retrieved data to be consistent (inconsistency detection)
11		**Transaction-based Processor**	Ensure messages receives are processed successfully and altered data successfully after processing	Transaction-based Delivery
12		**Timeout-based Message Processor**	process messages guaranteeing all messages handled are processed at-least-once	Timeout-based message processor sends acknowledgement after has successfully processed the message.
13		**Multi-Component Image**	Virtual server provide functionality of multiple application components	Multiple application components hosted on a single virtual server (to ensure running virtual servers may be used for different purposes without making provisioning or decommissioning operations necessary).
14	Multi-Tenancy	**Shared Component**	shared between multiple tenants(individual configuration)	Optimize portion of the app_ stack and app components deployed equal to all tenants.
15		**Tenant-isolated Component**	Shared between multiple tenants enabling individual configuration and tenant-isolation regarding performance, data volume, and access privileges	Ensure isolation between tenants by controlling tenant access, processing performance used, and separation of stored data.
16		**Dedicated Component**	Not shared components be integrated into a multi-tenant app	Dedicated application components are provided exclusively for each tenant using the application.

17	**Cloud Integration**	**Restricted Data Access Component**	Need component alter provide DB on Accesses restriction on different environments	Defined privileges for each data element, separate restriction data access
18		**Message Mover**	message queues of different providers be integrated	Message Mover integrate message queues hosted in different environments , receiving messages from one queue and transferring it to a queue in other environments.
19		**Application Component Proxy**	Application component be accessed direct to its hosting environment is restricted.	Synchronous and asynchronous communication with proxy component is initiated and maintained from the restricted environment access the unrestricted environment directly.
20		**Compliant Data Replication**	Replicated between environments, how some environments, handle subsets of the data due to laws and corporate regulation?	Message filters are used to delete and obfuscate certain data elements in these messages. Information about the data manipulations stored in a storage offering.
21		**Integration Provider**	How can application components in different environments, belonging to different companies, be integrated through a third-party provider?	Using integration components offered by a third party provider.