SOFTWARE REQUIREMENTS ANALYSIS AND SPECIFICATION

2. SOFTWARE REQUIREMENTS ANALYSIS AND SPECIFICATION

2.1. RELATED WORK

Cloud storage has revolutionized data storage by providing scalable, flexible, and accessible solutions for individuals and businesses. However, it also presents security risks, such as unauthorized access, data breaches, and privacy concerns. This section should discuss the types of security challenges commonly associated with cloud storage, along with standard security practices, such as encryption, access controls, and audits.Encryption plays a pivotal role in ensuring data confidentiality. This section explores common encryption techniques used in cloud security, focusing on their strengths and weaknesses. Discuss traditional Public Key Infrastructure (PKI), symmetric encryption, and how they are used in cloud environments. Emphasize the complexity and scalability issues that can arise when using these techniques in a distributed environment.

2.1.1 LITERATURE SURVEY

[1] Xiaoping Li, Sesnior Member, IEEE, Lihua Qian, and Ruben Ruiz, "Cloud Workflow Scheduling with deadlines and time slot Availability," IEEE Transactions on Service Computing, Issue. 2, pp. 455–466, 2016.

Allocating service capacities in cloud computing is based on the assumption that they are unlimited and can be used atany time. However, available service capacities change with workload and cannot satisfy users' requests at any time from the cloudprovider's perspective because cloud services can be shared by multiple tasks. Cloud service providers provide available time slotsfor new user's requests based on available capacities. In this paper, we consider workflow scheduling with deadline and time slotavailability in cloud computing. An iterated heuristic framework is presented for the problem under study which mainly consists ofinitial solution construction, improvement, and perturbation. Three initial solution construction strategies, two greedy- and fair-basedimprovement strategies and a perturbation strategy are proposed. Different strategies in the three phases result in several heuristics.Experimental results show that different initial solution and improvement strategies have different effects on solution qualities.

[2] A. Verma and S. Kaushal, "Deadline constraint heuristic–based genetic algorithm for workflow scheduling in cloud," International Journal of Grid and Utility Computing, vol. 5, no. 2, pp. 96–106, 2014.

Task scheduling and resource allocation are the key challenges of cloud computing. Compared with grid environment, data transfer is a big overhead for cloud workflows. So, the cost arising from data transfers between resources as well as execution costs must also be taken into account during scheduling based upon user's Quality of Service QoS constraints. In this paper, we present Deadline Constrained Heuristic based Genetic Algorithms HGAs to schedule applications to cloud resources that minimise the execution cost while meeting the deadline for delivering the result. Each workflow's task is assigned priority using bottom-level b-level and top-level t-level. To increase the population diversity, these priorities are then used to create the initial population of HGAs. The proposed algorithms are simulated and evaluated with synthetic workflows based on realistic workflows. The simulation results show that our proposed algorithms have a promising performance as compared to Standard Genetic Algorithm SGA.

concept of **[3] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," Future Generation Computer Systems, vol. 29, no. 1, pp. 158–169, 2013.**

Effective use of elastic heterogeneous cloud resources represents a unique multi-objective scheduling challenge with respect to cost and time constraints. In this paper we introduce a novel deadline constrained scheduling algorithm, Deadline Constrained Critical Path (DCCP), that manages the scheduling of workloads on dynamically provisioned cloud resources. The DCCP algorithm consists of two stages: (i) task prioritization, and (ii) task assignment, and builds upon the Constrained Critical Paths to execute a set of tasks on the same instance in order to fulfil our goal of reducing data movement between instances. We evaluated the normalized cost and success rate of DCCP and compared these results with IC-PCP. Overall, DCCP schedules with lower cost and exhibits a higher success rate in meeting deadline constraints.

[4] S. Abrishami and M. Naghibzadeh, “Deadline-constrained workflow scheduling in software as a service cloud,” Scientia Iranica, vol. 19, no. 3, pp. 680–689, 2012

The advent of Cloud computing as a new model of service provisioning in distributed systems, encourages researchers to investigate its benefits and drawbacks in executing scientific applications such as workflows. In this model, the users request for available services according to their desired Quality of Service, and they are charged on a pay-per-use basis. One of the most challenging problems in Clouds is workflow scheduling, i.e., the problem of satisfying the QoS of the user as well as minimizing the cost of workflow execution. In this paper, we propose a new QoS-based workflow scheduling algorithm based on a novel concept called Partial Critical Paths (PCP), which tries to minimize the cost of workflow execution while meeting a user-defined deadline. This algorithm recursively schedules the partial critical paths ending at previously scheduled tasks. The simulation results show that the performance of our algorithm is very promising.

2.1.2 EXISTING ALGORITHMS

Grid technologies have progressed towards a service-oriented paradigm that enables a new way of service provisioning based on utility computing models, which are capable of supporting diverse computing services. It facilitates scientific applications to take advantage of computing resources distributed world wide to enhance the capability and performance. Many scientific applications in areas such as bioinformatics and astronomy require workflow processing in which tasks are executed based on their control or data dependencies. Scheduling such interdependent tasks on utility Grid environments need to consider users’ QoS requirements. In this paper, we present a genetic algorithm approach to address scheduling optimization.

2.2 RESEARCH METHODOLOGY

2.2.1 SYSTEM ARCHITECTURE:

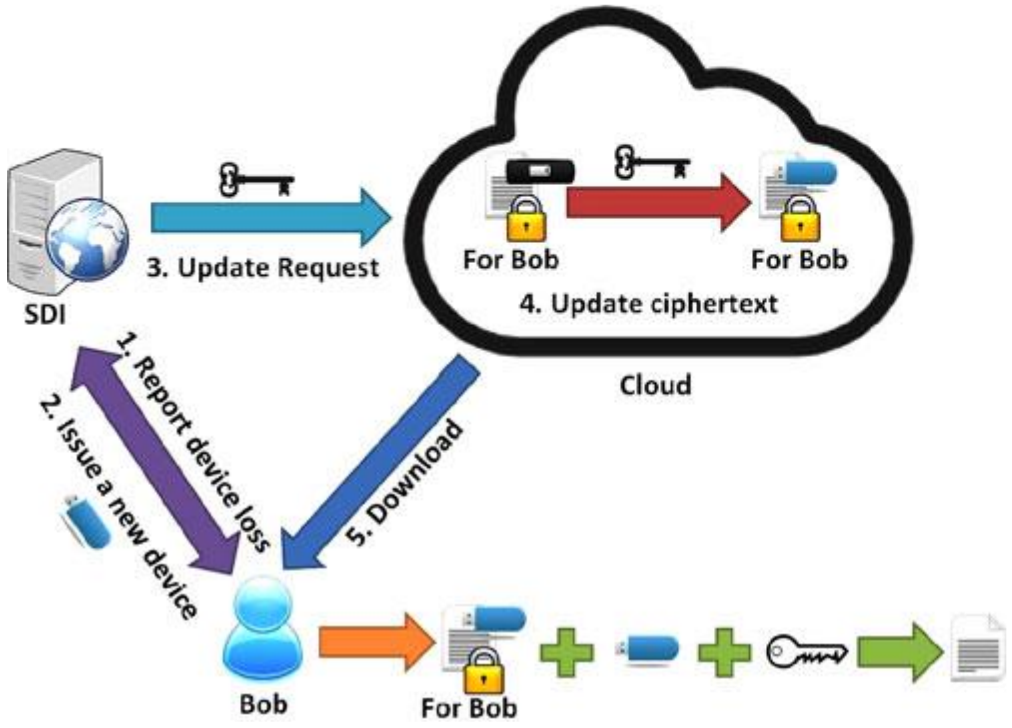

Fig 2.2.1.1: System Architecture

2.2.2 PROPOSED ALGORITHMS

Implementing a two-factor data security protection mechanism for a cloud storage system involves using two different methods to ensure that data is protected. This could be a combination of encryption and a physical or digital key. Here's a step-by-step approach to implementing such a mechanism:

Step 1: Establish Encryption Infrastructure

- Use a secure algorithm to generate bilinear map parameters. This includes two multiplicative cyclic groups.

Step 2: Generate User Keys

- Generate public and private keys for each user based on the bilinear maps. The private key should be kept confidential and the public key made available for encryption operations.

Step 3: Encrypt Data for Cloud Storage

- Before storing data, generate a unique encryption key using a secure random algorithm. This key will be used to encrypt the data before it is stored in the cloud.

Step 4: Set Up a Physical or Digital Second Factor

- Provide users with a physical token (like a smart card, USB key, or hardware token) that stores a unique value or key.

Step 5: Secure Authentication and Data Decryption

- To access the stored data, the user must provide both their credentials (username/password) and the second factor (physical/digital token).

Step 6: Implement Security Measures

- Use secure communication protocols (e.g., HTTPS, SSL/TLS) for data transfer.

2.3PRODUCT FUNCTIONS

Cloud Server Module:

The Cloud user module is developed such that the new users will Signup initially and then Login for authentication. The Cloud user is provided with the option of file search. Then cloud user feature is added up for send the Request to Auditor for the File access. After getting decrypt key from the Auditor, he/she can access to the File. The cloud user is also enabled to download the File. After completion of the process, the user logout the session.

TPA Module:

Will Login on the Auditor's page. He/she will check the pending requests of any User. After accepting the request from the User, he/she will generate master key for encrypt and Secret key for decrypt. After the complete process, the Auditor logout the session.

cloud Provider Module

The Data Provider module is developed such that the new users will Signup initially and then Login for authentication. The Data Provider module provides the option of uploading the file to the Cloud Server using Identity-Based Encryption. Data Provider is provided with the feature of Revocation and

Ciphertext update of the file. Once after completion of this process, the Data Provider logsout of the session.

Data owner (User)

In this module, the receiver can receive the data file from the service provider via Base station. The receivers receive the file by without changing the File Contents. Users may receive particular data files within the network only.

2.4 USER CONSTRAINTS

User Constraints for project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

Economical Constraints
Technical Constraints
Social Constraints

ECONOMICAL CONSTRAINTS

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL CONSTRAINTS

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL CONSTRAINTS

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

2.5 HARDWARE REQUIREMENTS

• Processor	:	I3 or higher
• Speed	:	2.9GHz
• RAM	:	4 GB (min)
• Hard Disk	:	160 GB

2.6 SOFTWARE REQUIREMENTS

• **Operating system**	**:**	Windows 7 Ultimate
• **Coding Language**	**:**	Python
• **Back-End**	**:**	Django-ORM
• **Designing**	**:**	Html, css, javascript
• **Data Base**	**:**	MySQL (WAMP Server)

Functional requirements :

Functional requirements describe what the system should do. The functional requirements can be further categorized as follows:

- What inputs the system should accept?
- What outputs the system should produce?
- What data the system must store?
- What are the computations to be done?

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and the steps are necessary to put transaction data in to a usable form for processing that can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

2.7 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the constraints that must be adhered during development. They limit what resources can be used and set bounds on aspects of the software’s quality.

User Interfaces

The User Interface is a GUI developed using Python.

Software Interfaces

The main processing is done in Java and console application.

Manpower Requirements

5 members can complete the project in 2 – 4 months if they work fulltime of.

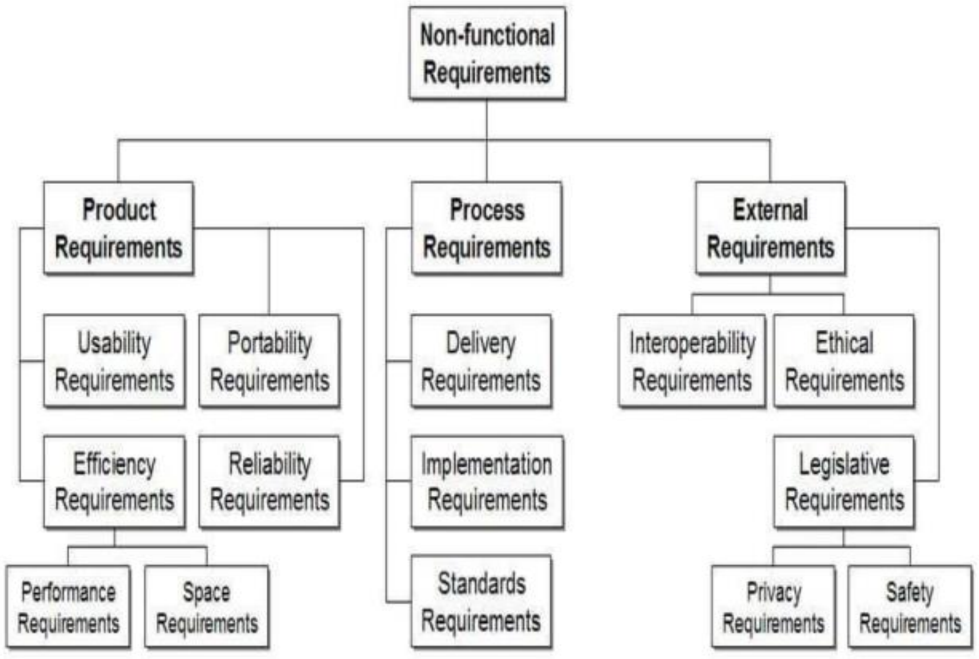

Fig2.7.1. Non functional requirements

Reliability

The packages will pick-up current transactions online. Regarding the old transactions, user will enter them into the system.

Security

The web server and database server should be protected from hacking, virus etc.

Portability

The software will work both on Windows and Linux OS. Hence portability problems will not arise.

Maintainability

The database will be running at the server. Users access these forms by using the user-ids and the passwords.

Robustness

Time to restart after failure percentage

e of events causing failure probability of data corruption on failure.

Usability

The system is used by the four persons namely Administrator, Project Manager, Developer andthe customer. Each person is having their own roles and separated by the security issues.

Performance

System is highly functional and good in performance. The system must use the minimal set of variables and minimal usage of the control structures will dynamically increase the performance of the system.

Availability

The system is implemented based on the web browser and server. Using this web browser, the user can access the data and store the data in the server.

Supportability

The system is designed to be the cross platform supportable. The System is supported on a wide range of hardware and any software platform.

Testability

The system software can be tested. Operability is the better it works, the more efficiency it can be tested. It operates clearly. Observability is what you see is what you test. The results of each test case are readily observer.

SOFTWARE DEVELOPMENT LIFE CYCLE(SDLC)

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software. The methodology within the SDLC process can vary across industries and organizations, but Standards such as

ISO/IEC 12207 represent processes that establish a lifecycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

SDLC consists of following activities

1. **Planning:** The most important parts of software development, requirement gathering or requirement analysis are usually done by the most skilled and experienced software engineers in the organization. After the requirements are gathered from the client, a scope document is created in which the scope of the project is determined and documented.
2. **Implementation:** The software engineers start writing the code according to the client's requirements.
3. **Testing:** This is the process of finding defects or bugs in the created software.
4. **Documentation:** Every step in the project is documented for future reference and for the improvement of the software in the development process. The design documentation may include writing the application programming interface (API).
5. **Deployment:** The software is deployed after it has been approved for release.
6. **Maintaining:** Software maintenance is done for future reference. Software improvement and new requirements (change requests) can take longer than the time.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
- Evaluating the first prototype in terms of its strengths, weakness, and risks.
- Defining the requirements of the second prototype.
- Planning an designing the second prototype.
- Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system I sthroughly evaluated and tested Routine maintenance is carried on a containing basis to prevent scale failures and to minimize down time.

The following diagram shows how a spiral model acts like

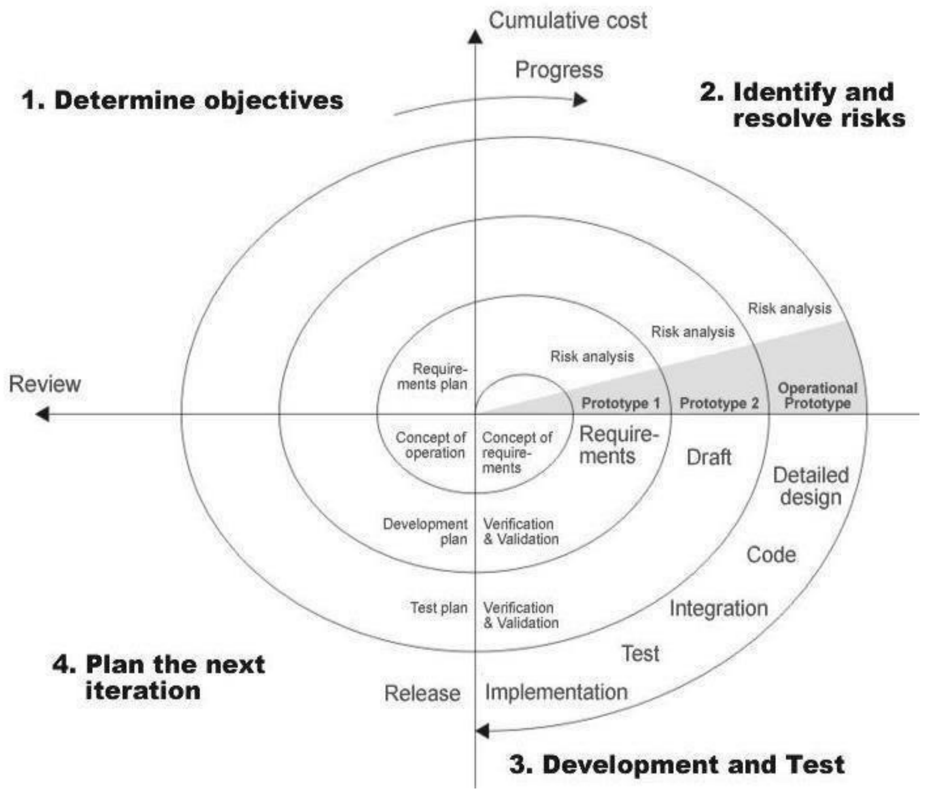

Fig: 2.7.2. Spiral Model

Advantages

- High amount of risk analysis
- Good for large and mission-critical projects.
- Software is produced early in the software life cycle.