PUBLIC INTERGITY AUDTING-ENABLING PUBLIC VERIFIABILITY FOR STORAGE SECURITY IN CLOUD COMPUTING

[1]K. Gowri Shankari, [2]R. Maruthi

[1,2]Ponnaiyah Ramajayam Institute of Science & Technology (PRIST) Vallam, Thanjavur-613403,Tamil Nadu, India.

OBJECTIVE:

A privacy-preserving mechanism is proposed that supports public auditing on shared data stored in the cloud.
DOMAIN: Cloud computing

1 INDEX TERMS:

1.1 Cloud Computing:

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

It is an expression used to describe a variety of different types of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet. Cloud computing is a term without a commonly accepted unequivocal scientific or technical definition. In science, cloud computing is a synonym for distributed computing over a network and means the ability to run a program on many connected computers at the same time. The phrase is also more commonly used to refer to network-based services which appear to be provided by real server hardware, which in fact are served up by virtual hardware, simulated by software running on one or more real machines. Such virtual servers do not physically exist and can therefore be moved around and scaled up (or down) on the fly without affecting the end user arguably, rather like a cloud. Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

There are many types of public cloud computing:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)
- Storage as a service (STaaS)
- Security as a service (SECaaS)
- Data as a service (DaaS)
- Test environment as a service (TEaaS)
- Desktop as a service (DaaS)
- API as a service (APIaaS)

The business model, IT as a service (ITaaS), is used by in-house, enterprise IT organizations that offer any or all of the above services.

Using software as a service, users also rent application software and databases. The cloud providers manage the infrastructure and platforms on which the applications run. End users access cloud-based applications through a web browser or a light-weight desktop or mobile app while the business software and user's data are stored on servers at a remote location. Proponents claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.

2. Cloud computing used in:

- Autonomic computing — Computer systems capable of self-management.
- Client–server model — *Client–server computing* refers broadly to any distributed application that distinguishes between service providers (servers) and service requesters (clients).
- Grid computing — "A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks."
- Mainframe computer — Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as census, industry and consumer statistics, police and secret intelligence services, enterprise resource planning, and financial transaction processing.
- Utility computing — The "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity."
- Peer-to-peer — Distributed architecture without the need for central coordination, with participants being at the same time both suppliers and consumers of resources (in contrast to the traditional client–server model).
- Cloud gaming - Also called On-demand gaming is a way of delivering to games to computers. The gaming data will be stored in the provider's server, so that gaming will be independent of client computers used to play the game.

3. Characteristics

- **Agility** improves with users' ability to re-provision technological infrastructure resources.
- **Application programming interface** (API) accessibility to software that enables machines to interact with cloud software in the same way the user interface facilitates interaction between humans and computers. Cloud computing systems typically use REST-based APIs.
- **Cost** is claimed to be reduced and in a public cloud delivery model capital expenditure is converted to operational expenditure.

This is purported to lower barriers to entry, as infrastructure is typically provided by a third-party and does not need to be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is fine-grained with usage-based options and fewer IT skills are required for implementation (in-house).The e-FISCAL project's state of the art repository contains several articles looking into cost aspects in more detail, most of them concluding that costs savings depend on the type of activities supported and the type of infrastructure available in-house.

- **Device and location independence** enable users to access systems using a web browser regardless of their location or what device they are using (e.g., PC, mobile phone). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect from anywhere.
- **Virtualization** technology allows servers and storage devices to be shared and utilization be increased. Applications can be easily migrated from one physical server to another.
- **Multitenancy** enables sharing of resources and costs across a large pool of users thus allowing for:
- **Centralization** of infrastructure in locations with lower costs (such as real estate, electricity, etc.)
- **Peak-load capacity** increases (users need not engineer for highest possible load-levels)
- **Utilization and efficiency** improvements for systems that are often only 10–20% utilized.
- **Reliability** is improved if multiple redundant sites are used, which makes well-designed cloud computing suitable for business continuity and disaster recovery.[30]
- **Scalability and elasticity** via dynamic ("on-demand") provisioning of resources on a fine-grained, self-service basis near real-time, without users having to engineer for peak loads.
- **Performance** is monitored, and consistent and loosely coupled architectures are constructed using web services as the system interface.
- **Security** could improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. Security is often as good as or better than other traditional systems, in part because providers are able to devote resources to solving security issues that many customers cannot afford. However, the complexity of security is greatly increased when data is distributed over a wider area or greater number of devices and in multi-tenant systems that are being shared by unrelated users. In addition, user access to security audit logs may be difficult or impossible. Private cloud installations are in part motivated by users' desire to retain control over the infrastructure and avoid losing control of information security.

▪ **Maintenance** of cloud computing applications is easier, because they do not need to be installed on each user's computer and can be accessed from different places.

4. Virtualization

Virtualization (or **virtualization**) is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system (OS), storage device, or network resources.

While a **physical computer** in the classical sense is clearly a complete and actual machine, both *subjectively* (from the user's point of view) and *objectively* (from the hardware system administrator's point of view), a **virtual machine** is *subjectively* a complete machine (or very close), but *objectively* merely a set of files and running programs on an actual, physical machine (which the user need not necessarily be aware of).

Virtualization can be viewed as part of an overall trend in enterprise IT that includes autonomic computing, a scenario in which the IT environment will be able to manage itself based on perceived activity, and utility computing, in which computer processing power is seen as a utility that clients can pay for only as needed.

The usual goal of virtualization is to centralize administrative tasks while improving scalability and overall hardware-resource utilization. With virtualization, several operating systems can be run in parallel on a single central processing unit (CPU). This parallelism tends to reduce overhead costs and differs from multitasking, which involves running several programs on the same OS.

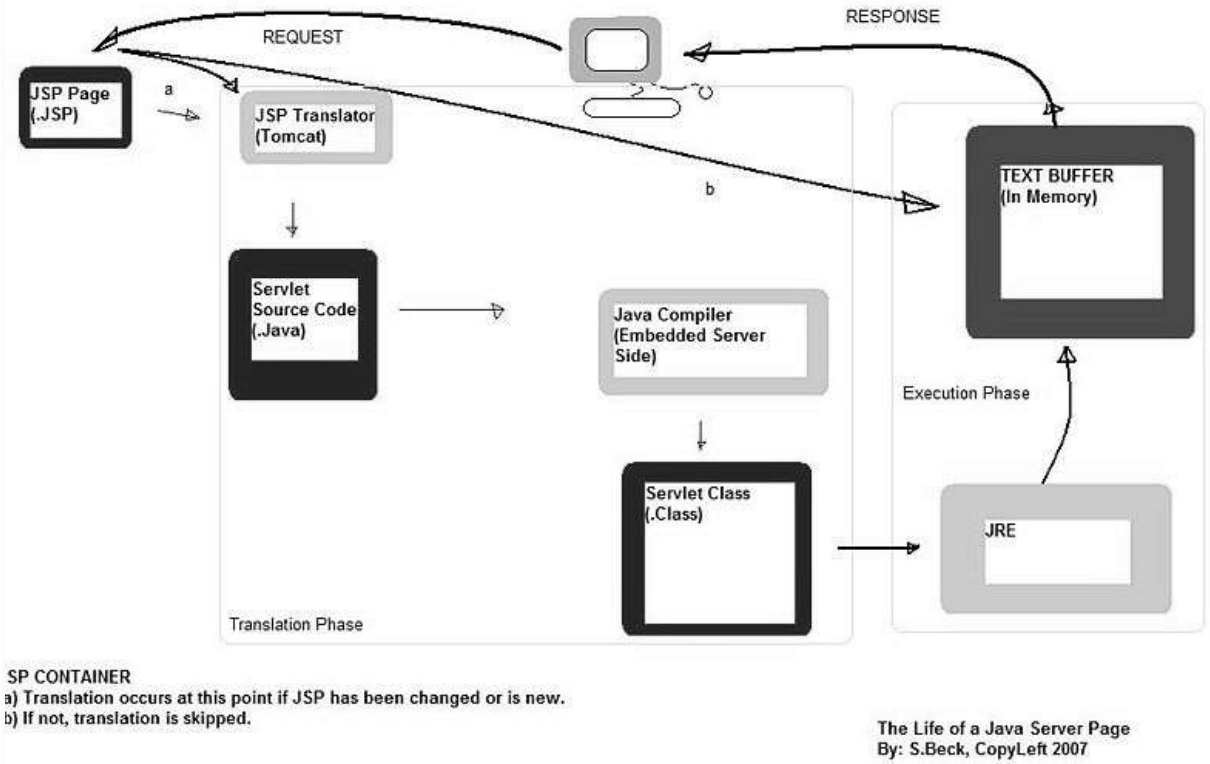
5 JSP:

JavaServer Pages (JSP) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, or it may generate byte code for the servlet directly. JSPs can also be interpreted on-the-fly reducing the time taken to reload changesJavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent.

Java is a set of several computer software products and specifications from Oracle Corporation that provides a system for developing application and deploying it in a cross-platform computing environment. Java is used in a wide variety of platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common, Java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web on desktop computers.

6. Architecture OF JSP:



7. The Advantages of JSP:

- Active Server Pages (ASP). ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- Pure Servlets. JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.
- Server-Side Includes (SSI). SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- JavaScript. JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like.
- Static HTML. Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

8. SERVLETS

The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to non-

Java dynamic Web content technologies such as PHP, CGI and ASP.NET. Servlets can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

The Servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of a Web container and a servlet. A Web container is essentially the component of a Web server that interacts with the servlets. The Web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

A Servlet is an object that receives a request and generates a response based on that request. The basic servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the Web server and a client. Servlets may be packaged in a WAR file as a Web application. Servlets can be generated automatically by JavaServer Pages (JSP), or alternately by template engines such as Web Macro. Often servlets are used in conjunction with JSPs in a pattern called "Model 2", which is a flavor of the model-view-controller pattern. Servlets are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Building Web pages on the fly is useful (and commonly done) for a number of reasons:

- The Web page is based on data submitted by the user. For example the results pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well.
- The data changes frequently. For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date.
- The Web page uses information from corporate databases or other such sources. For example, you would use this for making a Web page at an on-line store that lists current prices and number of items in stock.

9. The Servlet Run-time Environment:

A servlet is a Java class and therefore needs to be executed in a Java VM by a service we call a servlet engine.

The servlet engine loads the servlet class the first time the servlet is requested, or optionally already when the servlet engine is started. The servlet then stays loaded to handle multiple requests until it is explicitly unloaded or the servlet engine is shut down.

Some Web servers, such as Sun's Java Web Server (JWS), W3C's Jigsaw and Gefion Software's LiteWebServer (LWS) are implemented in Java and have a built-in servlet engine. Other Web servers, such as Netscape's Enterprise Server, Microsoft's Internet Information Server (IIS) and the Apache Group's Apache, require a servlet engine add-on module. The add-on intercepts all requests for servlets, executes them and returns the response through the Web server to the client. Examples of servlet engine add-ons are Gefion Software's WAICoolRunner, IBM's WebSphere, Live Software's JRun and New Atlanta's ServletExec.

All Servlet API classes and a simple servlet-enabled Web server are combined into the Java Servlet Development Kit (JSDK), available for download at Sun's official Servlet site .To get started with servlets I recommend that you download the JSDK and play around with the sample servlets.

10 Life Cycle of Servlet

The Servlet lifecycle consists of the following steps:

1. The Servlet class is loaded by the container during start-up.
2. The container calls the init() method. This method initializes the servlet and must be called before the servlet can service any requests. In the entire life of a servlet, the init() method is called only once.
3. After initialization, the servlet can service client-requests. Each request is serviced in its own separate thread. The container calls the service() method of the servlet for every request. The service() method determines the kind of request

being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an implementation for these methods. If a request for a method that is not implemented by the servlet is made, the method of the parent class is called, typically resulting in an error being returned to the requester.

4. Finally, the container calls the destroy () method which takes the servlet out of service. The destroy () method like init () is called only once in the lifecycle of a Servlet.

11 FEASIBILITY STUDY

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly.Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

Technical Feasibility:

The technology used can be developed with the current equipments and has the technical capacity to hold the data required by the new system.

- This technology supports the modern trends of technology.
- Easily accessible, more secure technologies.

Technical feasibility on the existing system and to what extend it can support the proposed addition. New modules easily without affecting the Core Program can be added. Most of parts are running in the server using the concept of stored procedures.

Operational Feasibility:

This proposed system can easily implemented, as this is based on JSP coding (JAVA) & HTML .The database created is with MySql server which is more secure and easy to handle. The resources that are required to implement/install these are available. The personal of the organization already has enough exposure to computers. So the project is operationally feasible.

Economical Feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible which assess the brain capacity with quick & online test. So it is economically a good project

12. Java (programming language)

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is considered by many as one of the most influential programming languages of the 20th century, and widely used from application software to web application.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of their Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

12.1 J2EE Application:

A J2EE application or a Java 2 Platform Enterprise Edition application is any deployable unit of J2EE functionality. This can be a single J2EE module or a group of modules packaged into an EAR file along with a J2EE application deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers.

Enterprise applications can consist of the following:

- EJB modules (packaged in JAR files);
- Web modules (packaged in WAR files);
- connector modules or resource adapters (packaged in RAR files);
- Session Initiation Protocol (SIP) modules (packaged in SAR files);
- application client modules;
- Additional JAR files containing dependent classes or other components required by the application;
- Any combination of the above.

13. Servlet

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are server-side Java EE components that generate responses (typically HTML pages) to requests (typically HTTP requests) from clients. A servlet can almost be thought of as an applet that runs on the server side—without a face.

```
// Hello.java
import java.io.*;
import javax.servlet.*;

public class Hello extends GenericServlet {
   public void service(ServletRequest request, ServletResponse response)
         throws ServletException, IOException {
      response.setContentType ("text/html");
      final PrintWriter pw = response.getWriter();
      pw.println("Hello, world!");
      pw.close();
  }}
```

The import statements direct the Java compiler to include all of the public classes and interfaces from the java.io and javax.servlet packages in the compilation.

The Hello class overrides the service (ServletRequest, ServletResponse) method defined by the Servlet interface to provide the code for the service request handler. The service() method is passed a Servlet Request object that contains the request from the client and a Servlet Response object used to create the response returned to the client. The service() method declares that it throws the exceptions Servlet Exception and IO Exception if a problem prevents it from responding to the request.

The set Content Type (String) method in the response object is called to set the MIME content type of the returned data to "text/html". The getWriter() method in the response returns a Print Writer object that is used to write the data that is sent to the client. The print ln (String) method is called to write the "Hello, world!" string to the response and then the close () method is called to close the print writer, which causes the data that has been written to the stream to be returned to the client.

14. TESTING:

The various levels of testing are:

1. White Box Testing
2. Black Box Testing
3. Unit Testing
4. Functional Testing
5. Performance Testing
6. Integration Testing
7. Objective
8. Integration Testing
9. Validation Testing
10. System Testing
11. Structure Testing
12. Output Testing
13. User Acceptance Testing

White Box Testing

- Execution of every path in the program.

Black Box Testing

- Exhaustive input testing is required to find all errors.

Unit Testing

- Unit testing, also known as Module Testing, focuses verification efforts on the module. The module is tested separately and this is carried out at the programming stage itself.
- Unit Test comprises of the set of tests performed by an individual programmer before integration of the unit into the system.
- Unit test focuses on the smallest unit of software design- the software component or module.
- Using component level design, important control paths are tested to uncover errors within the boundary of the module.
- Unit test is white box oriented and the step can be conducted in parallel for multiple components.

Functional Testing:

- Functional test cases involve exercising the code with normal input values for which the expected results are known, as well as the boundary values

Objective:

- The objective is to take unit-tested modules and build a program structure that has been dictated by design.

Performance Testing:

- Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization of the program unit. It occurs throughout all steps in the testing process.

Integration Testing:

- It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with in the interface.
- It takes the unit tested modules and builds a program structure.
- All the modules are combined and tested as a whole.
- Integration of all the components to form the entire system and a overall testing is executed.

Validation Testing:

- Validation test succeeds when the software functions in a manner that can be reasonably expected by the client.
- Software validation is achieved through a series of black box testing which confirms to the requirements.

- Black box testing is conducted at the software interface.
- The test is designed to uncover interface errors, is also used to demonstrate that software functions are operational, input is properly accepted, output are produced and that the integrity of external information is maintained.

System Testing:

- Tests to find the discrepancies between the system and its original objective, current specifications and system documentation.

Structure Testing:

- It is concerned with exercising the internal logic of a program and traversing particular execution paths.

Output Testing:

- Output of test cases compared with the expected results created during design of test cases.
- Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
- Here, the output format is considered into two was, one is on screen and another one is printed format.
- The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
- The output comes out as the specified requirements as the user's hard copy.

User acceptance Testing:

- Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
- The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.
- It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.

Two set of acceptance test to be run:

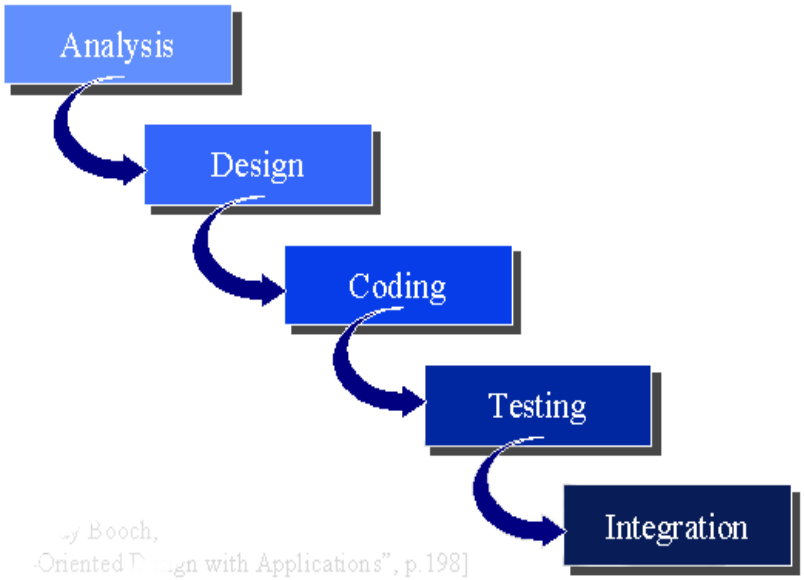
1. Those developed by quality assurance group.
2. Those developed by customer.
3. Methodology

15.Waterfall Approach

While the Waterfall Model presents a straightforward view of the software life cycle, this view is only appropriate for certain classes of software development. Specifically, the Waterfall Model works well when the software requirements are well understood (e.g., software such as compilers or operating systems) and the nature of the software development involves contractual agreements. The Waterfall Model is a natural fit for contract-based software development since this model is document driven; that is, many of the products such as the requirements specification and the design are documents. These documents then become the basis for the software development contract.

There have been many waterfall variations since the initial model was introduced by Winston Royce in 1970 in a paper entitled: "managing the development of large software systems: concepts and techniques". Barry Boehm, developer of the spiral model (see below) modified the waterfall model in his book *Software Engineering Economics* (Prentice-Hall, 1987). The basic difference in the various models is in the naming and/or order of the phases.

The basic waterfall approach looks like the illustration below. Each phase is done in a specific order with its own entry and exit criteria and provides the maximum in separation of skills, an important factor in government contracting.



Example of a typical waterfall approach

While some variations on the waterfall theme allow for iterations back to the previous phase, "In practice most waterfall projects are managed with the assumption that once the phase is completed, the result of that activity is cast in concrete. For example, at the end of the design phase, a design document is delivered. It is expected that this document will not be updated throughout the rest of the development. You cannot climb up a waterfall." (Murray Cantor, Object-oriented project management with UML, John Wiley, 1998)

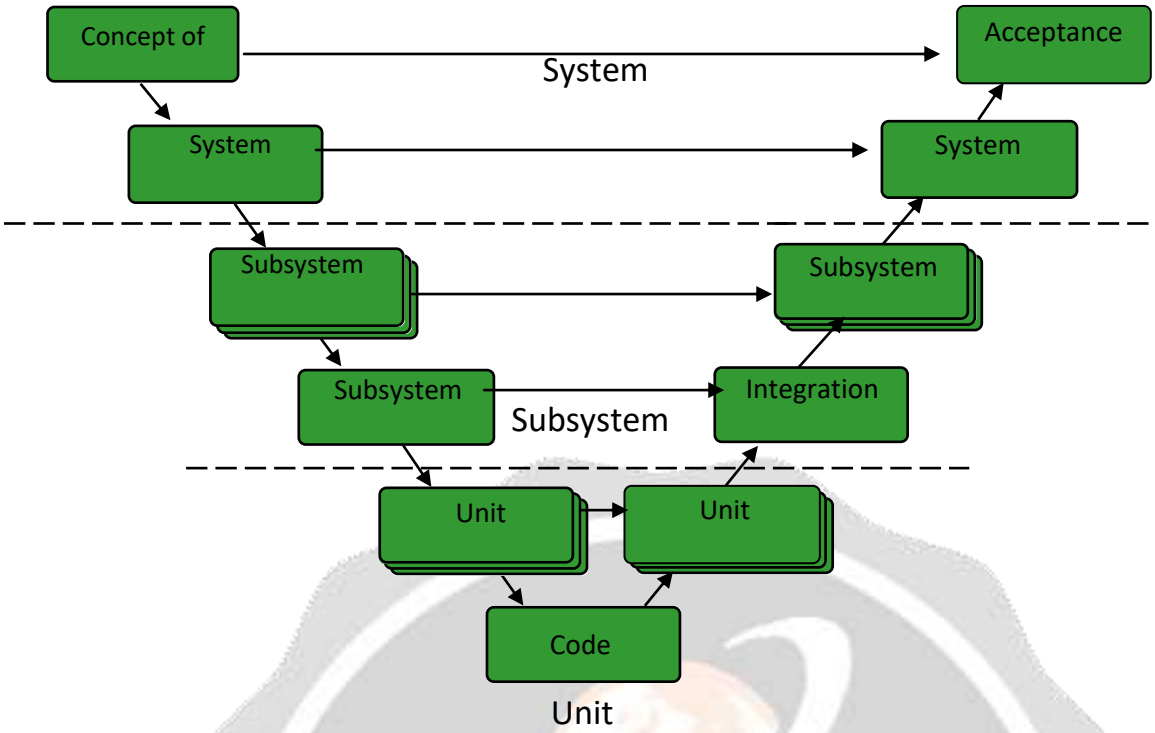
The waterfall is the easiest of the approaches for a business analyst to understand and work with and it is still, in its various forms, the operational SLC in the majority of US IT shops. The business analyst is directly involved in the requirements definition and/or analysis phases and peripherally involved in the succeeding phases until the end of the testing phase. The business analyst is heavily involved in the last stages of testing when the product is determined to solve the business problem. The solution is defined by the business analyst in the business case and requirements documents. The business analyst is also involved in the integration or transition phase assisting the business community to accept and incorporate the new system and processes.

a. V Model

The "V" model (sometimes known as the "U" model) reflects the approach to systems development where in the definition side of the model is linked directly to the confirmation side. It specifies early testing and preparation of testing scenarios and cases before the build stage to simultaneously validate the definitions and prepare for the test stages. It is the standard for German federal government projects and is considered as much a project management method as a software development approach.

"The V Model, while admittedly obscure, gives equal weight to testing rather than treating it as an afterthought. Initially defined by the late Paul Rook in the late 1980s, the V was included in the U.K.'s National Computing Centre publications in the 1990s with the aim of improving the efficiency and effectiveness of software development. It's accepted in Europe and the U.K. as a superior alternative to the waterfall model; yet in the U.S., the V Model is often mistaken for the waterfall…

"In fact, the V Model emerged in reaction to some waterfall models that showed testing as a single phase following the traditional development phases of requirements analysis, high-level design, detailed design and coding. The waterfall model did considerable damage by supporting the common impression that testing is merely a brief detour after most of the mileage has been gained by mainline development activities. Many managers still believe this, even though testing usually takes up half of the project time." (Goldsmith and Graham, "The Forgotten Phase", Software development, July 2002) As shown below, the model is the shape of the development cycle (a waterfall wrapped around) and the concept of flow down and across the phases. The V shows the typical sequence of development activities on the left-hand (downhill) side and the corresponding sequence of test execution activities on the right-hand (uphill) side.



Example of a typical V Model (IEEE)

The primary contribution the V Model makes is this alignment of testing and specification. This is also an advantage to the business analyst who can use the model and approach to enforce early consideration of later testing. The V Model emphasizes that testing is done throughout the SDLC rather than just at the end of the cycle and reminds the business analyst to prepare the test cases and scenarios in advance while the solution is being defined.

The business analyst's role in the V Model is essentially the same as the waterfall. The business analyst is involved full time in the specification of the business problem and the confirmation and validation that the business problem has been solved which is done at acceptance test. The business analyst is also involved in the requirements phases and advises the system test stage which is typically performed by independent testers – the quality assurance group or someone other than the development team. The primary business analyst involvement in the system test stage is keeping the requirements updated as changes occur and providing "voice of the customer" to the testers and development team. The rest of the test stages on the right side of the model are done by the development team to ensure they have developed the product correctly. It is the business analyst's job to ensure they have developed the correct product.

CONCLUSION

Oruta, the first privacy-preserving public auditing mechanism for shared data in the cloud is proposed. With Oruta, the public verifier is able to efficiently audit the integrity of shared data, yet cannot distinguish who is the signer on each block, which can preserve identity privacy for users. An interesting problem in our future work is how to efficiently audit the integrity of shared data with dynamic groups while still preserving the identity of the signer on each block from the third party auditor.

REFERENCE

[1]. J. Larimer. (Oct. 28, 2014). Pushdo SSL DDoS Attacks. [Online].Available::http://www.iss.net/threats/pushdoSSLDDoS.html

[2] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," Comput. Netw., vol. 44, no. 5, pp. 643–666, 2004.

[3] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in Proc. Netw. Distrib. Syst. Secur. Symp., 1999, pp. 151–165.

[4] T. J. McNevin, J.-M. Park, and R. Marchany, "pTCP: A client puzzle protocol for defending against resource exhaustion denial of service attacks," Virginia Tech Univ., Dept. Elect. Comput. Eng., Blacksburg, VA, USA, Tech. Rep. TR-ECE-04-10, Oct. 2004.

[5] R. Shankesi, O. Fatemieh, and C. A. Gunter, "Resource inflation threats to denial of service countermeasures," Dept. Comput. Sci., UIUC, Champaign, IL, USA, Tech. Rep., Oct. 2010. [Online]. Available: http://hdl.handle.net/2142/17372

[6] J. Green, J. Juen, O. Fatemieh, R. Shankesi, D. Jin, and C. A. Gunter, "Reconstructing Hash Reversal based Proof of Work Schemes," in Proc. 4th USENIX Workshop Large-Scale Exploits Emergent Threats, 2011.

[7] Y. I. Jerschow and M. Mauve, "Non-parallelizable and non-interactive client puzzles from modular square roots," in Proc. Int. Conf. Availability, Rel. Secur., Aug. 2011, pp. 135–142.

[8] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Dept. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TR-684, Feb. 1996. [Online]. Available:

[9] W.-C. Feng and E. Kaiser, "The case for public work," in Proc. IEEE Global Internet Symp., May 2007, pp. 43–48.

[10] D. Keppel, S. J. Eggers, and R. R. Henry, "A case for runtime code generation," Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, USA, Tech. Rep. CSE-91-11-04, 1991.

[11] E. Kaiser and W.-C. Feng, "mod_kaPoW: Mitigating DoS with transparent proof-of-work," in Proc. ACM CoNEXT Conf., 2007, p. 74.

[12] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in Proc. Netw. Distrib. Syst. Secur. Symp., 1999, pp. 151–165.

[13] J. Green, J. Juen, O. Fatemieh, R. Shankesi, D. Jin, and C. A. Gunter "Reconstructing Hash Reversal based Proof of Work Schemes," in Proc. 4th USENIXWorkshop Large-Scale Exploits Emergent Threats, 2011.

[14] R. L. Rivest, A. Shamir, and D. A. Wagner "Time-lock puzzles and timed-release crypto,", Dept. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA,USA, Tech. Rep. MIT/LCS/TR-684, Feb. 1996.

[15] E. Kaiser and W.-C. Feng "mod_kaPoW: Mitigating DoS with transparent proof-of-work,", in Proc. ACM CoNEXT Conf., 2007.

[16] M. Jakobsson and A. Juels "Proofs of work and bread pudding protocols," in Proc. IFIP TC6/TC11 Joint Working Conf. Secure Inf. N. [6] Rashmi Nayakawadi, Md Abdul Waheed, Rekha Patil "Avoiding Permanent disabling of Wireless Sensor Network from the attack of malicious Vampire Nodes", in International Journal of Advanced Scientific and Technical Research Issue 4 volume 3, May-June 2014 on http://www.rspublication.com/ijst/index.html.

[17] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Dept. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TR-684, Feb. 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1. 1.110.5709