International Journal of Software Engineering (IJSE)

ISSN : 2180-1320

Volume 1, Issue 5

Number of issues per year: 6

International Journal of Software Engineering (IJSE)

Volume 1, Issue 5, 2011

Edited By
Computer Science Journals
www.cscjournals.org

International Journal of Software Engineering (IJSE)

Book: 2011 Volume 1, Issue 5
Publishing Date: 08-02-2011
Proceedings
ISSN (Online): 2180-1320

IJSE Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Editorial Preface

The International Journal of Software Engineering (IJSE) provides a forum for software engineering research that publishes empirical results relevant to both researchers and practitioners. It is the fourth issue of First volume of IJSE and it is published bi-monthly, with papers being peer reviewed to high international standards.

IJSE encourage researchers, practitioners, and developers to submit research papers reporting original research results, technology trend surveys reviewing an area of research in software engineering, software science, theoretical software engineering, computational intelligence, and knowledge engineering, survey articles surveying a broad area in software engineering and knowledge engineering, tool reviews and book reviews. Some important topics covered by IJSE usually involve the study on collection and analysis of data and experience that can be used to characterize, evaluate and reveal relationships between software development deliverables, practices, and technologies. IJSE is a refereed journal that promotes the publication of industry-relevant research, to address the significant gap between research and practice.

IJSE give the opportunity to researchers and practitioners for presenting their research, technological advances, practical problems and concerns to the software engineering.. IJSE is not limited to a specific aspect of software engineering it cover all Software engineering topics. In order to position IJSE amongst the most high quality journal on computer engineering sciences, a group of highly professional scholars are serving on the editorial board. IJSE include empirical studies, requirement engineering, software architecture, software testing, formal methods, and verification.

International Editorial Board ensures that significant developments in software engineering from around the world are reflected in IJSE. The submission and publication process of manuscript done by efficient way. Readers of the IJSE will benefit from the papers presented in this issue in order to aware the recent advances in the Software engineering. International Electronic editorial and reviewer system allows for the fast publication of accepted manuscripts into issue publication of IJSE. Because we know how important it is for authors to have their work published with a minimum delay after submission of their manuscript. For that reason we continue to strive for fast decision times and minimum delays in the publication processes. Papers are indexed & abstracted with International indexers & abstractors

Editorial Board Members
International Journal of Software Engineering (IJSE)

Editorial Board

Editorial Board Members (EBMs)

Table of Content

Volume 1, Issue 5, December 2011

Pages

Evaluation of QoS Based Web- Service Selection Techniques for Service Composition

M. Sathya satsubithra@gmail.com
Department of Computer Science
School of Engineering Pondicherry University
Puducherry- 605014 India

M. Swarnamugi swathidevan@gmail.com
Department of Computer Science
School of Engineering Pondicherry University
Puducherry- 605014 India

P. Dhavachelvan dhavachelvan@gmail.com
Department of Computer Science
School of Engineering Pondicherry University
Puducherry- 605014 India

G. Sureshkumar mgsureshkumar@gmail.com
Department of Computer Science
Pondicherry University Karaikal Centre, Karaikal
Puducherry- 609605 India

Abstract

In service oriented computing, services are the basic construct that aims to facilitate building of business application in a more flexible and interoperable manner for enterprise collaboration. To satisfy the needs of clients and to adapt to changing needs, service composition is performed to compose the various capabilities of available services. With the proliferation of services offering similar functionalities around the web, the task of service selection for service composition is complicated. It is vital to provide service consumers with facilities for selecting required web services according to their non-functional characteristics or quality of service (QoS). The objective of this paper presents the exploration of various techniques of Quality of Service based Service Selection (QSS) approach in the literature. To evaluate the service selection process, a number of criteria for QSS approach have been identified and presented in this paper.

Keywords: Web Service Selection, Service Composition, Web Semantics, Quality of Service.

1. INTRODUCTION

Service-Oriented Computing (SOC) is an upcoming organizational model that allows assembling independent distributed services into complex ones. Services are autonomous, platform-independent computational entities that can be used in a platform independent and programming language independent way. The application functionality of SOC as services relies on its dynamism. That is, it has the capability to dynamically assemble complex services for developing

massively distributed, interoperable, evolvable systems. Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled. Key to this concept is the service-oriented architecture (SOA).

The Service Oriented Architecture (SOA) is a type of "software architecture that represents software functionality as services over the network" [1]. Web Services are the predominant implementation platform for SOA and it uses a set of standards, SOAP, UDDI, WSDL, which enable a flexible way for applications to interact with each other over networks. Simple Object Access Protocol (SOAP) is a standard protocol that allows network communication between services. The easiest way to publish a web service is to use a SOAP container. When a software component is published as a web service, any SOAP-enabled client that knows the network address of the web service can send a SOAP request and get a SOAP response. To get the message information, SOAP- enabled clients read a WSDL file that describes the web service. Once the Web Services Description Languages (WSDL) file is read, the client can start sending SOAP messages to the web service. WSDL describes what a web service can actually do, where it resides, and how to invoke it. Universal Description Discovery and Integration (UDDI) is a standard that allows information about businesses and services to be electronically published, queried and stored. Published information is stored into one or more UDDI registries, which can be accessed through SOAP.

All these standards are XML-based (Extensible Markup Language), which allows applications to interact with each other over networks, no matter what languages and platforms they are using. The two features, self-description and language-platform-independence, distinguish web services from other distributed computing technologies, like CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model).

Research in web services includes many challenging areas starting from service publication to service mining. The most vital among them is web service composition. Web service composition is needed when a client's complex request cannot be answered by single service, but by combining or composing various functionalities of available services or more than one services. Composition involves three different issues [2]. The first, called selection of service is concerned with selecting suitable services to composite that satisfy the user requirement. The second, called composition synthesis is concerned with synthesizing a specification of how to coordinate the component services to fulfill the client request. The third issue, called as orchestration is concerned with achieving the coordination among services by executing the specification produced by the composition synthesis.

This paper presents a study of one service selection approach called QoS based service selection for service composition. The paper is organized as follows: Section II describes the overview of service selection approaches. In Section III, the specifications of QoS based service selection and the various techniques of QSS are presented. Section IV analysis the evaluation criteria of QoS based service selection approach and compare the various techniques of QSS. Finally section V concludes with discussion and highlights new challenges need to be addressed.

2. OVERVIEW OF SERVICE SELECTION APPROACHES

The current semantic web services architecture focus on solving the issues of service discovery, service selection and service composition. Service discovery is the process of finding or locating service implementations that meet a specified condition. In the same way, service selection is a process that deals with choosing a service implementation from the located services. From this, it is clearly seen that service discovery is a prerequisite requirement for selection process, but selection is the main problem that needs to be addressed for retrieving Web services successfully. For any service selection approaches the basic requirements include: Customer

service requirement, Service offerings by the service provider and aggregating the evaluation results.

2.1 Customer-Service Requirement

The customer service requirement may be simple or complex. Simple requirement may not look for composite services to satisfy the user query. Whereas, complex requirements may have both functional and non-functional aspects which needs to be satisfied. For this kind of complex requirement, the services need to be composite. The composite service is a service formed by a composition of other available services. Google research application is accepted as a web service and integrated with other services, such as Gmail, AdWords, Picasa, Orkut, You Tube and Google Maps service, to provide an integrated environment for service consumers. The other known example for service composition is a tour booking service that can be formed as a web service and integrated with other services such as hotel booking, sight seeing, flight booking or car-rental in order to provide a collaborated environment for user. However, there may exist huge number of (tour booking) services which provide similarly functional characteristics. Service consumers not only expect the service to meet functional aspects but they also require services to meet non-functional aspects properties that is, quality of services (QoS) such as service reliability, security, trust and execution cost, etc. Thus the selection of services based on non functional qualities gain more advantages nowadays.

2.2 Provider-Service Offerings

The services offered by service provider are concerned about functional and non-functional qualities of services. The functional properties make use of domain ontology. To provide consumer the requested service with non-functional properties makes use of QOS ontology. The problem that arises here is how to map the quality preferences offered by consumer with the quality categorization in QOS ontology. This can be solved by labeling the qualities (eg. performance, security) in QoS ontology with the service Identification.

2.3 Service Selection Process

This involves matching the customer required service with the offered service. The dynamic selection of web services involves getting user requirements, the provider of service need to publish or register their services using service description language, finally the matcher will match the user requirements with the registered service description. The requirements specified by the user or customer may vary from description of service and Quality of service (QoS). To overcome this problem, domain ontology and QoS ontology may be used. The registered service descriptions by the service provider contain the semantic profile and QoS parameters. The provider of the service is also required to specify the location of a WSDL document describing a web service. A query processor may be used to analyze the requirement specified by the user with the domain ontology and QoS ontology. The semantic matcher will match the user request with service description and locate available services matching with requirements. The discovered services are then taken as input to the selection process to select the best service that satisfies the user requirement. In basic form, service selection involves mapping a set of services to a service—this can be thought of as the best service; in a more general form, service selections maps a set of services to a ranking of the services in that set [6]. Multitude of service selection techniques and algorithms are proposed in the literature such as Use of optimization algorithm [3] for service selection, integer linear programming [4], broker-based architecture [5], negotiation model for service selection etc [29] [31]. With the thorough study of service selection process in the literature, the following approaches are identified.

1. Functional based service selection approach
2. Non-Functional based service selection approach
3. User based service selection approach

The Web Service Selection process is broadly classified as Functional based approach, Non-functional based approach and User based approach. Functional based service selection approach represents the Static and Dynamic semantics. Selecting an appropriate service is

concerned with retrieving functional descriptions from service repositories and then ensuring that the described and required interfaces match with each other. Static semantics represents the properties of messages and operation semantics. The properties of messages include parameter passed (Data type, language, unit and business role) and message types (Serviceability, provider type, purpose, consumer type). Dynamic semantics represents the properties of behavior and operation logic. With dynamic semantics in the service selection process of Web service, the resultant contains more than one service provider offering similar services.

With the rapidly growing number of available services, customers are presented with a choice of functionally similar services. This choice allows customer to select services that match other criteria, often referred to as non-functional attributes. Two fundamental questions arise because of this: How can these extra attributes be described and how can one select the most appropriate service. These questions should address both the selection of isolated services as well as the selection of services within the context of other services. The non-functional based service selection represents the QoS and Context in semantic web service selection. The properties of QoS may be (security, reliability, response time, call cost etc.), the properties of Context may include context of customer (location, intention, consumer's name, application, e-mail, termination of hardware and software) and context of service (provider's details, service descriptions etc,). User based approach represents the selection of best service among numerous discovered services based on customers' feedback, trust and reputation.

Approach I. Functional Based Service Selection

Today, the advancement in Web services requires growth in the areas of service interoperation, discovery, selection, composition, choreography, orchestration and mining. A possible solution to all these problems can be provided by converting Web services to Semantic Web [23]. Semantic Web services (SWS) can provide a solution to the integration problem like composition. In general, the semantics to be added to a Web service may be called as functional semantics. In Web services, functional semantic is taken into consideration thereby avoiding unsatisfied results which are not of customer interest. Functional property is the functional semantics of a service that describes what a service actually does.

Web Service Selection is related to the process of evaluating and ranking the discovered web services to identify the ones that fulfill a set of functional and non-functional properties requested by the service customer. Most of the existing techniques rely on syntactic descriptions of service interfaces to find web services with disregard to semantic service parameters. This generates major problems in the service selection mechanism. To solve these problems, Web service descriptions are enhanced with annotations of ontological concepts, semantic matching and by considering non-functional properties.

Approach II. Non - Functional Based Service Selection

In a Web environment, multiple WSs may provide similar functionalities with different Non-functional property values (e.g., different prices). Such Web services will typically be grouped together in a single community. To differentiate the members of a community during service selection, their non-functional properties need to be considered. These properties are characterized as quality of service (QoS) and context based services. Both are highly important and are to be taken into account during the WS selection.

The W3C working group (2003) defined various QoS attributes for web services (WS) in their 25th November 2003 publication. This include: performance, reliability, scalability, capacity, robustness, exception handling, accuracy, integrity, accessibility, availability, interoperability, security, network-related QoS requirements etc. Although regular QoS attributes are listed, it remains some issues on selection of web services according to the user desired. First, there exists some web services provided with similar functional requirements which, might lead to the problem of differentiating the services with QoS. Second, the perception on QoS of web services

distinct between the customer and provider. There also exist a number of other issues which need to be considered on QoS based service selection process.

Approach III. User Based Service Selection

A User based methodology is a mechanism using consumers' feedbacks to identify good services from bad ones. It has advantages in solving the selection problem for Web services. The service consumer would like to choose a service that is trusted or a service with a high reputation. Trust and reputation play an important role in a service selection process of user based service selection. With this approach, web service selection may be customized according to users' different constrains and preferences. Most approaches proposed in the literature about personalized selection concentrate on how to rank web services according to users' preferences on various QoS metrics. A trust based methodology [7] for service selection is proposed. QoS-based semantic web service selection solution with the application of a trust and reputation management method is presented. This work is based on Virtual Internet Service Provider.

This paper focuses on one of the non – functional property known as QoS based Service Selection approach, its specification [20], techniques and criteria for evaluating techniques of QSS approach.

3. QoS BASED SERVICE SELECTION

A QoS property can be static or dynamic [24]. A static QoS property value is defined at the time it is described whereas the dynamic QoS property value requires measuring and updating its value periodically. The QoS value from the service consumer's perspective can be positive, negative, close, or exact. For example, consumers expect to buy a service with low price and expect to retrieve the service in a low response time. Whereas performance, integrity etc., have positive trend in which the consumer expects the positives values are better.

3.1 Specification of Service Selection Approaches

The specification or description for non functional based service selection approaches concentrates on many factors. These factors are separately identified and presented by analyzing various techniques of non functional based service selection approach. Table 1 depicts the QSS Specification and Description.

Spec. No.	***Specification.***	***Descriptions***
S(1)	QoS Modeling	Specify the modeling language used. Such as WSML and its variants WSML – Core, WSML – Flight, WSML – Rule, WSML – DL and WSML – Full
S (2)	QoS Categorization	Describe the Ontology of QoS categorization with its identification value.
S (3)	User Preferences	Describe the varying preferences for the non-functional criteria specified by the service consumer
S (4)	QoS Evaluation	Specify the evaluation criteria used to evaluate the non – functional properties.
S (5)	Aggregating the evaluation of	This deals with aggregating individual scores to gain a final score for the service.

	QoS	
S (6)	QoS Properties	List the number of non –functional properties considered.
S (7)	Level of Automation	States the level of automation mechanisms. A – Fully automated, SA – Semi automated, NA – Not applicable.
S (8)	Coordination Distribution	Describes how individual web service can interact in order to accomplish an application task. C – Centralized, CO – Coordination, GCO – Global coordination.
S(9)	Agent Involvement	State whether agent participation is involved in the process of service selection mechanism.
S (10)	Ranking Algorithm	A service rank is a quantitative metric that shows the "importance" of a service within the process of service selection mechanism to rank the services.

TABLE 1: QSS Specification and Description

(a) QoS Modeling

Service requestors need to distinguish services based on their non-functional criteria to make the most appropriate choice amongst a number of services with equal or similar functionality. Therefore, a QoS modeling is needed. That can be used in service descriptions as well as service requests. Due to the adaptability of non-functional properties (the new ones might be required at any time) it is unlikely that a complete standard set can be identified. The criteria differ depending on the domain. For example the E-Learning domain service should consider the accuracy, reputation, and cost. In contrast E-Publishing service should consider the security, price, quality properties. Therefore it is desirable that the model for delivering non-functional properties is designed in a simple way.

WSMO with its associated language, the WSML (Web Service Modeling Language) provides a formal syntax and semantics to describe the QoS characteristics of services. The Web Service Modeling Ontology (WSMO) defines four main elements as the main concepts of semantic Web service. This includes Ontologies, Web Services, Goals and Mediators [25]. Ontology's are formal explicit specifications of a shared conceptualization [21]. They define a common agreed terminology by providing concepts and relationships between the concepts. Goals are descriptions of web services that satisfy the user desires when confer with a service in terms of functional specification, behavior and quality of service. Web Services are description about services. The description consist of functional, non-functional and the behavioral aspects of web services. Mediators address the heterogeneity issues between different WSMO elements. The Web Service Modeling Language is a formal language for describing ontologies, goals, web services and mediators. It is based on logical formalisms of WSMO namely description logics, first – order logic, and logic programming [22]. These formalisms are the basic point to describe the variants of WSML. The variants includes, WSML – Core, WSML – Flight, WSML – Rule, WSML – DL and WSML – Full.

(b) QoS Categorization

QoS properties are designed in hierarchical way. This involves grouping properties by domains such as environment, performance or safety. Speed quality and response time on performance aspects while security, privacy and authentication are safety aspects. If such hierarchical structure exists then users should be able to express preferences at a higher level, while service provides will express their offerings in a fine way. Using WSML [26], the simplest way of modeling is done by assigning a simple value to non functional properties of WSMO elements. The data value assigned to non functional properties is used as an identifier during service publication. To specify QoS characteristics in particular it can be modeled separately with the use of building and defining QoS Ontology. Figure 1 depicts the QoS ontology with the assumed identifier value. W3C defines various QoS attributes such as performance, reliability, scalability, capacity, and so on. Here the figure 1 covers ontology of characteristics such as interoperability, capacity, integrity, environment, performance, reliability, security, business and availability. When a new service is published, the value of QoS characteristics in service description is matched with the value assigned in QoS ontology. By this way, the newly published services are aligned. Upon receiving the request from the customer, the system extract the services require and QoS characteristics specified and match with the QoS ontology to locate it.

1. QoS Characteristics
    - 1.1 Interoperability
    - 1.2 Capacity
    - 1.3 Integrity
    - 1.4 Scalability
    - 1.5 Accuracy
    - 1.6 Accessibility
    - 1.7 Environment
        - 1.7.1 Temporal
        - 1.7.2 Location
    - 1.8 Performance
        - 1.8.1 Latency
        - 1.8.2 Response Time
        - 1.8.3 Throughput
        - 1.8.4 Error Rate
    - 1.9 Reliability
        - 1.9.1 Recover
            - 1.9.1.1 Failure
            - 1.9.1.2 Disaster
        - 1.9.2 Consistency
        - 1.9.3 MTBF (Mean Time Between Failures)
    - 1.10 Security
        - 1.10.1 Encryption
            - 1.10.1.1 Data
            - 1.10.1.2 Messages
        - 1.10.2 Authentication
        - 1.10.3 Authorization
        - 1.10.4 Auditability
        - 1.10.5 Accountability
        - 1.10.6 Non – Repudiation
        - 1.10.7 Traceability
    - 1.11 Business
        - 1.11.1 Cost
        - 1.11.2 Reputation
        - 1.11.3 Monitoring
    - 1.12 Availability
        - 1.12.1 MTTR (Mean Time To Recovery)
        - 1.12.2 Load Balancing

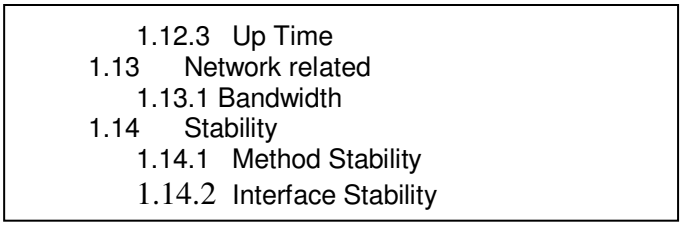

FIGURE 1: QoS Categorization

(c) User Preferences

Depending on the situation service requestors may have varying preferences for the non-functional criteria. In the same way, different requestors will have different preferences. A good mechanism should not only allow expressing values for each property, but preferably also represent the relations among the preferences. For example, a customer may consider the security property as more important than privacy when requesting a financial service. Hence, the selection approach needs to provide for mechanisms for users to specify their preferences, that is which of the non-functional properties they feel more strongly about and also relations between these properties.

(d) QoS Evaluation

It is difficult to predict how many non-functional properties will be available, and the type of these properties for a customer requested service. For example, the evaluation function to compute the speed criteria will be different from the function to calculate the location criteria. It is not easy to define a Universal evaluation function for all kinds of non-functional properties. Hence, the evaluation function for one property adapt to varying numbers of criteria, but should also automatically identify the measurement methods to be used to evaluate each criteria.

(e) Aggregating the Evaluation of QoS

After evaluation the next step is to aggregate individual scores to gain a final score for the service. In this step a suitable aggregation method needs to be selected. Global optimization or local optimization may be used [27]. Using arithmetic or geometric means to aggregate QoS properties results in complex situations.

(f) Level of Automation

Level of automation states the automation mechanisms like manual process of selection mechanism, or semi-automatic service selection mechanism or fully automated service selection mechanism involved in web service selection and composition. Most research contributions handling the service selection for service composition focus on automatic process without human intervention. For example human intervention may involve selecting QoS parameters used for selection, and changing preferences etc. Semi – automatic process involves little human intervention, the major task such as corrections and composing are done by the system [28]. Fully automated service selection approach may also use agents in the web service selection process [32].

(f.1) Agent Involvement

State whether agent participation is involved in the process of service selection mechanism. A software agent is a piece of software that acts for service consumer or provider in semantic web service to make the process of service selection automatic. Agents work cooperatively to evaluate either service providers or service consumers.

(g) Coordination – Service Composition

This describes how individual web service can interact in order to accomplish composite service selection process. The WS-Coordination defines how the coordination among the services need to take place, how the data items are to be exchanged in order to complete successful composition as part of business process defined in a Business Process Execution Language (BPEL) [30]. The composition algorithms may be centraily cooperated or globally cooperated.

(h) Ranking Algorithm

A service rank is a quantitative metric that shows the importance of a service within the process of service selection mechanism. It is known that semantic based service discovery concerns on the matchmaking process between customer's requirement and service profile or description. Its semantic matchmaking process plays a role as a ranking mechanism in service selection process. However ranking based on semantic similarity does not suit for efficient service selection. Because, from customers perspective, it is always not true that a web service with high semantic similarity is suitable than a web service with lower similarity. The other difficulty with semantic similarity is that the users find it hard to distinct which service is better suitable between a pool of similar services [17]. To achieve better ranking performance many ranking algorithms have been proposed in the literature. One such approach is to integrate more information besides semantic information. The information may range from time, place, location [18], customer and providers situation [19] etc. The limitation with this approach is that the system becomes more complicated when new constraints are added. To overcome this, the authors [33] have proposed a method a social collaborative filtering method for ranking. This method makes use of learning other user's previous experiences. This method is used most successfully in all kinds of recommendation systems but the limitations with this method are information distortion and independence of service selection.

3.2 QSS Service Selection Techniques

The various techniques of QoS based service selection identified from the literature are discussed in this section. Figure 2 portraits the various QSS techniques identified.

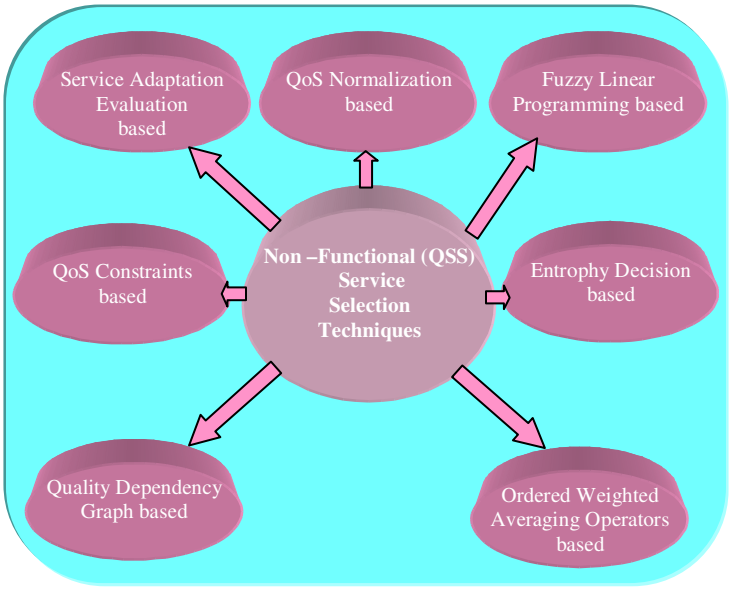

FIGURE 2: QSS Service Selection Techniques

3.2.1 Service Adaptation Evaluation Based QSS Technique

Baopeng et al [8] proposed a QoS model and used hierarchy policy approach to capture goals of users, applications, environment and resources to form rational service composition and adaptation action. The authors have proposed a Service adaptation evaluation (SAE) algorithm to handle service adaptation problem and service composition decision problem in pervasive computing environment. The system model consists of property primitives for policy hierarchy such as Control Construct, InterpretedAs, BelongTo, ExistIn, and Commit. The proposed multidimensional QoS model is focused not only on the traditional QoS properties but also Requirements of functional QoS parameters and Environmental QoS. The QoS model is

designed with four layers namely Resource layer, Environment layer, Application layer and User layer. Each layer describes its own QoS properties. The use of policies at different layers triggers the service adaptation and provides better service composition performance. A policy driven service selection algorithm is proposed by the authors to make selection mechanism semantic-aware and QoS-aware. It is said semantic-aware because, the algorithm performs well even if the composite service semantic logic changes to form new semantic logic. QoS-aware represents the input, output QoS parameter consistency, and end-to-end QoS properties such as delay etc. A policy description language defined in this technique consists of three symbols namely primitive symbols, action symbols and function symbols.

3.2.2 QoS Normalization Based QSS Technique

In order to enable quality-driven web service selection, Yutu Liu et al [9] proposed a dynamic and secure framework to evaluate the QoS of a number of web services. The three key aspects that are developed in this technique include Extensible QoS model, Preference-oriented service ranking, fair and open QoS computation. The QoS model in this technique is designed to evaluate the QoS of web services without changing the computational model. In service ranking, this technique concentrates on representing QoS from the service requestor's preference perspective. The QoS computation aspects ensure that the information is collected in a fair manner. For QoS based service selection modeling, three quality parameter or properties is measured for generic quality services namely execution price, duration and reputation. It considers transaction, compensation rate, penalty rate for business related quality criteria. In order to rank the web services, this technique prefer normalization. The purposes of normalization are: one to allow for a uniform measurement of service qualities independent of units. Two, to provide a uniform index to represent service qualities for each provider. Three, to allow setting a threshold regarding the qualities. The number of normalizations performed depends on how the quality criteria are grouped. The authors have proposed a prototype model to implement the QoS registry with hypothetical phone service. They have analyzed collecting service quality information, collecting quality information from active execution monitoring and collecting quality information from user feedback. In their proposed framework, the authors have defined deterministic and non-deterministic criterion to indicate the value of QoS quality and when a service is invoked. The non-deterministic indicate for QoS quality that is uncertain when web service is invoked. The advantage of this technique is, it lessens the overhead of QoS registry, and it dose not need expensive middleware to select the service provider.

3.2.3 Fuzzy Linear Programming Based QSS Technique

Ping et al [10] proposed a QoS-aware service selection model based on fuzzy linear programming (FLP) technologies, to identify their dissimilarity on service alternatives and assist service consumers in selecting most suitable services with needs and preferences of customers. The proposed model has key aspects such as vague reference, weighting of QoS attributes, and service ranking. In the process of selecting web services, the vague preference of QoS by service consumer is handled by the proposed model. Weighting of QoS attributes is designed to explore the optimal solution. Service ranking deals with ranking on web services. A fuzzy group consensus aware service selection algorithm is proposed based on LINMAP (Linear Programming techniques for Multidimensional Analysis of Preferences) model to find the optimal QoS weighting attribute for web services. For the proposed service selection algorithm, the authors have represented arithmetic operations on fuzzy numbers. This includes representation for Triangular Fuzzy number, Fuzzy arithmetic operations for addition, subtraction, multiplication and division. The normalized Euclidean distance between two triangular fuzzy numbers, and the weighted square distance from positive ideal solution. Further, the authors have addressed the consistence and inconsistence measurement of service customers by aggregating difference between fuzzy performance rating and FIPS. The square distance defined is used for accessing QoS attributes weights.

3.2.4 QoS Constraints Based QSS Technique

Tao yu et al [11] proposed the service selection problem in two models the combinatorial model and the graph model. A QoS service broker acts as an external, independent broker entity that

can help users construct composite services. To conduct service selection for the general flow structure the combinatorial model is used. The combinatorial model reflects the service selection problem as multidimensional multichoice 0-1 knapsack problem. The graph model sees the selection problem as a multiconstraint optimal path problem. To provide end – to – end QoS constraint for distributed services, the authors have proposed a broker based architecture. This architecture includes, service discovery, planning, selection and adaptation as its main function. The service selection algorithm proposed in this technique is designed with different composition structure. An efficient algorithm designed for quality driven web service composition ensures that the services selected satisfies the QoS requirements of users. Four different algorithms have been proposed by the authors and the algorithms does the task of service selection, algorithm for designing QoS constraints, heuristic algorithm to find the near optimal solutions, and algorithm to handle composition structures namely sequential, parallel, conditional, and loops. The QoS service broker called QBroker help customers to select the best service for the process of composite service before invocation. The authors have proposed different stages of process for service composition namely: Process plan, Function graph, and Service candidate graph. This technique supports constructs for composition model such as Sequential, AND split, XOR split, Loop, AND join, and XOR join. The QoS service selection problem as MMKP is designed in such a way that it ensures to select one service candidate from each service class to build composite service that meets the QoS constraints. To find optimal solution, BBLP (branch and bound) algorithm is used with MMKP. WS_HEU algorithm is used in this technique to find feasible solution in polynomial time. It has three main important steps namely: To find an initial feasible solution, Improve the solution by feasible upgrades, and to improve the solution by infeasible upgrades.

3.2.5 Entrophy Decision Model Based QSS Technique

A fuzzy entropy decision model, called Linguistic Entropy Method is proposed [12] to assign linguistic weights of QoS attributes and prioritizes the ranking order of service alternatives. To overcome the issue of measuring the QoS criteria in web service selection process, the authors have evaluated fuzzy weights of QoS attributes and rank the web services. The proposed technique is composed of enhanced version of Linguistic Entropy Method (LEM) and Fuzzy Synthetic Evaluation Method (FSEM). The Shannon entropy method uses probability function estimate uncertainty of object based on information theory. The weights for linguistic terms are evaluated with the use of triangular fuzzy number. The ratings to linguistic terms is provided by decision maker and designed by triangular fuzzy number. The algorithm Linguistic Entropy Method has accomplished a set of procedures to assign weights to QoS attributes in the web service selection process. First step is to organize the evaluation framework. That is, the QoS attributes are classified and taxonomy of QoS attributes is prepared. The next procedure is weighting the QoS attributes. This is performed by the decision maker. The third procedure is to select QoS attributes using fuzzy entropy weights assigned. Next procedure is to evaluate the score for each QoS attributes. The next procedure deals with constructing the fuzzy decision matrix by applying fuzzy weighting rules. Final procedure is about ranking the attributes and the services are selected.

3.2.6 Quality Dependency Graph Based QSS Technique

Chao Lv et al [13] proposed a technique for service selection mechanism to utilize “serve, be served” relationship and to evaluate the quality of services in business environment to select the enterprise to collaborate with. Quality Dependency Graph (QDG) method is used to model the relationship among enterprises. An Analytic Hierarchy Process (AHP) model is used to calculate the evaluation result of each candidate organization. The authors have presented Quality Dependency graph based on the characteristics of enterprise collaboration technique namely Dependency and Diversity. This QDC is used to evaluate the candidate enterprises in the service selection process. And an AHP model is used to weights the QoS attributes. The authors have proposed two algorithms to do the service selection task. First algorithm to create QDC from business specification. The second algorithm is used to get the service guideline for business role.

3.2.7 Ordered Weighted Averaging Operator Based QSS Technique

Hong Qing et al [14] proposed a novel non functional property-based service selection method by modifying the Logic Scoring Preference (LSP) method with Ordered Weighted Averaging (OWA) Operators to automate the service selection process. The authors have focused on two main issues of service selection process. They are Service automation and Dynamic aggregation function. Service automation deals with the automated ranking of QoS attributes. In order to make the selection process automatic, the ranking problem is transformed into OWA problem to automatically calculate the LSP orness degree. To evaluate the aggregation function, a method is used which combines LSP metrics with OWA operators. An algorithm is proposed to show the modified LSP method. Two new operators called Conjunction and Disjunction is introduced by the authors in the new LSP algorithm to represent relation between criteria such as replaceability, simultaneity etc. This LSP algorithm evaluates quantitative features for the different entities. The four main steps or procedure of this algorithm includes, specifying the evaluation variables, defining the elementary criteria, analyzing the degree decision and analyzing the preference. To overcome the change of criteria and preferences in the dynamic environment of service selection, a type based evaluation matrix is proposed and defined three types of criteria. They are Numerical type, Boolean type and Set overlap type. The advantage of this technique is that, this addresses both the issues of service selection process by assigning a proper quantitative aggregation metrics. And provided an automatic mechanism to facilitate the dynamic metric invocation and aggregation.

3.2.8 Summary of QSS Based Service Selection Process

QoS based service selection plays an important role in the process of service composition. Table 2 shows the comparative study of QSS techniques with the specification discussed. QoS aware service selection for compositing the services overcomes the problem faced in functional based service selection in which they provide only similar functional semantic properties, which might lead to the problem of differentiating available services. The techniques discussed above have advanced the process of QoS-aware service selection. However, the issues that need to address includes:

- Representation of QoS characteristics and QoS modeling.
- Assigning the QoS weightings.
- The fuzzy view on the QoS parameters between service consumers and service providers.
- The universal metric for evaluating the QoS parameters.

T	***QSST (1)***	***QSST (2)***	***QSST (3)***	***QSST (4)***	***QSST (5)***	***QSST (6)***	***QSST (7)***
S (1)	Yes. Multidimensional QoS model	Yes. Extensible QoS model	Yes. QoS model based on LINMAP	Yes	Not applicable	Yes	No
S (2)	Average	Yes	Yes	Yes	Yes	Yes	Not applicable
S (3)							

	Yes	Yes	Yes	No	Not applicable	No	Not applicable
S (4)	No	Yes	Yes	Yes	Yes	Not applicable	Yes
S (5)	No	Yes	Yes	Yes	Yes	Not applicable	Not applicable
S (6)	Availability of resources	Execution price, duration, reputation	Not given specifically	Not given specifically	Runtime, Transaction, Cost, Security, Network	Not applicable	Yes
S (7)	Semi - automated	Automated	Automated	Automated	Automated	Semi automated	Semi automated
S (8)	Centralized	Centralized	Centralized	Centralized	Peer - Peer	Not applicable	Peer - peer
S (9)	No	No	No	No	No	No	No
S (10	Yes. Policy driven ranking algorithm	Yes. Using normalization matrix	Yes. Consensual ranking	BBLP algorithm and WS_HEU algorithm	No	Yes(QDG, AHP)	Yes(Logic scoring preferences)

TABLE 2: Comparison of QSS Techniques.

4. ANALYSIS OF QSS APPROACH

The techniques of QSS approach have their advantages and disadvantages when compared with each other. There are many issues related to QSS approach that need to be addressed. Researchers all over the world are currently working on various aspects of QSS issues such as achieving consensus achievement, QoS modeling, etc. The analysis of QSS approach evaluates each technique based on evaluation criteria to identify which technique suits well for certain kind of application development. This section describes the various possible evaluation criteria for QSS approach.

To say whether a technique is good or bad for certain application development, they need to be evaluated based on some parameters. This process is like testing a program or software. The general parameters that are to be addressed for Non - functional based service selection approach include, Accuracy of the technique, Performance of the technique, Service availability, Complexity of Time, Complexity of cost, Scalability, Supportability, Failure rate, Threats to validity, Selection rate, Effectiveness, Information Retrieval metrics like precision and recall, Efficiency, F – measure, Mean average precision, Geometric mean average precision, Interpolated precision, Interpolated recall. The following are the evaluation metrics used for information retrieval system [15] [16]. The same set of metrics can be applied for evaluating QSS techniques.

Notations	Relevant	Non relevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

TABLE 3: Notations for True Positive and True Negatives

Precision (P): It is the fraction of retrieved documents that are relevant to the user's need.

$$P = \frac{number\ of\ relevant\ items\ retrived}{total\ number\ of\ reterived\ items} = \frac{tp}{tp + fp}$$

Where tp and fp are specified in Table 3.

Recall (R): It is the fraction of relevant documents that are retrieved to the user's need.

$$R = \frac{number\ of\ relevant\ items\ retrived}{total\ number\ of\ relevant\ items} = \frac{tp}{tp + fn}$$

Where tp and fn are specified in Table 3.

Accuracy (A): It specifies the fraction of classifications that are correct.

$$A = \frac{tp + tn}{tp + fp + fn + tn}$$

Where tp, fp, tn and fn are specified in Table 3.

F-measure: A measure that trades off precision versus recall is the F-measure. It is the harmonic mean of precision and recall.

$$F - measure = 2 * \frac{precision * recall}{precision + recall}$$

A new evaluation criteria is applied to [8] evaluate the adaptation of service selection assessment. Considering the user and environment requirements the criteria is proposed. [9] Conducted a series of experiment to investigate the relationship between QoS value and business criteria, study of effectiveness of price and the sensitivity factors in QoS computation. In [10], the approach not only deals with the decision maker's imprecise perceptions under incomplete information, but also objectively determines the importance weights of QoS criteria. The computational time is evaluated for this. The performances of algorithms for sequential and general flow structure are evaluated in [11]. This study includes two parts: the comparison of optimal and heuristic algorithms where runtime, approximation ratio, memory usage as metrics are used and the comparison of combinatorial and graph models where the provisioning success rate as a metric is used. The performance rating of each service alternative and the score of each alternative service is evaluated in [12]. The evaluation parameters for evaluating the QSS techniques are depicted in Table 4.

Spec. No.	***Specification.***	***Descriptions***
E(1)	**Accuracy**	Accurate gives many results in many senses. In service selection, accuracy defined as how relevant services are acquired that satisfy the user requirement
E (2)	**Service availability**	Service availability defines the existence of services in the registry.
E (3)	**Computational Time**	Time to retrieve the related or best relevant services that satisfy the customer need.
E (4)	**Computational Cost**	The total amount of cost required to get or select the services from the register which is been already registered by the service provider.
E (5)	**Scalability**	The possibility to register or select more services in the future.
E (6)	**Information Retrieval metric**	The kind of metrics used to measure the retrieved services.
E (7)q	**Supportability**	Support to modify or replace the services in the registry by the service provider.
E (8)	**Security**	States the security measure defined in the technique proposed.
E(9)	**Usability**	States how usable and efficient the retrieved services are.

TABLE 4: QSS Evaluation Metrics for Web Service Selection.

5. DISCUSSION AND CONCLUSION

With the increasing availability of Web services as a solution to enterprise application integration, the QoS parameters offered by Web services are becoming the chief priority for service providers and their service consumers. This paper have outlined the approach of non – functional (QoS) Web service selection based on requirements and specification identified from the thorough study from the literature. This paper reviewed a number of techniques in the context of the QoS based approach and have presented a summary of QoS parameters involved in the techniques identified and also the evaluation metrics that can be applied to obtain and test how the techniques perform against the specification criteria.

Due to the agile and dynamic nature of the web, providing the suitable QoS for enterprise business application is really a challenging task. In addition to this, modelling the QoS parameters also relies on the consensus between service consumer and service provider. To achieve the consensus among the service holders, their fuzzy view on QoS parameters have to be modelled and weighted in universal manner. This may cause service providers and consumers to better understand about QoS characteristics. The measurement process for each QoS parameters is very complex since it should consider what and how to measure, who does the measuring and where the measurements are taken. This raises the issue of conflicts on QoS characteristics metrics between service consumer and provider.

It can be concluded that most approaches contribute specific aspects to the overall picture of service selection, which requires methods for expressing user requirements, expressing service offerings and also the actual service selection method. Approaches tend to concentrate on specific of these areas and employ a variety of techniques to do that. It is more appropriate to make some suggestions for future developments in the area of selection approaches.

Important aspects that need addressing are powerful mechanisms to capture user requirements that are both user friendly and also expressive enough to capture large numbers of preferences and the logical relations between preferences. One aspect that falls into this area is the measuring of weights. Also, in the process of capturing the needs of users, their preference of data, research has to show interest and capability to automatically capture this, to reduce the burden on the user part, and to react to changes in circumstances automatically.

6. REFERENCES

[1] Matthias Klusch, Patrick Kapahnke. “Semantic Web Service Selection with SAWSDL-MX”. German Research Center for Artificial Intelligence, 2008.

[2] Roy Grønmo, Michael C. Jaeger. “Model-Driven Methodology for Building QoS-Optimised Web Service Compositions”. In Proceedings of the fifth IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS’05), pp. 68–82, Athens, Greece, May 2005. Springer Press.

[3] Matteo Baldon, Cristina Baroglio, Alberto Martelli, Viviana Patti. “Reasoning about interaction protocols for customizing web service selection and composition”. The Journal of Logic and Algebraic Programming, Elsevier, No. 70, pp. 53 – 73, 2007.

[4] Hassina Nacer Talantikite, Djamil Aissani, Nacer Boudjlida.“ Semantic annotations for web services discovery and composition”. Journal of Computer Standards & Interfaces, Elsevier, pp. 1 – 10, 2008.

[5] Dong-Hoon Shin, Kyong-Ho Lee, Tatsuya Suda. “Automated generation of composite web services based on functional semantics”. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, vol:7, pp. 332 – 343, Science Direct 2009.

[6] Matthias Klusch, Patrick Kapahnke. “Semantic Web Service Selection with SAWSDL-MX”. German Research Center for Artificial Intelligence, 2008.

[7] Galizia, S. Gugliotta ,A. Domingue, J. Milton Keynes. “A Trust Based Methodology for Web Service Selection”. In Proceedings of the IEEE International Conference on Semantic Computing, pp. 193 – 200, CA, 2007.

[8] Baopeng Zhang, Yuanchun Shi and Xin Xiao. “A Policy-Driven Service Composition Method for Adaptation in Pervasive Computing Environment”. First International Symposium on Pervasive Computing and Applications, pp. 619 – 624, 2006.

[9] Yutu Liu, Anne H. Ngu, Liang Z. Zeng. “QoS computation and policing in dynamic web service selection”. In Proceedings of the thirteenth international World Wide Web conference on Alternate track papers & posters , pp. 66 - 73 , 2004.

[10] Ping Wang, Kuo-Ming Chao, Chi-Chun Lo, Chun-Lung Huang, Yinsheng Li. "A Fuzzy Model for Selection of QoS-Aware Web Services”. In Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06), ICEBE, pp.585-593, 2006.

[11] Tao Yu and Kwei-Jay Lin. “Service Selection Algorithms forWeb Services with End - to- End QoS Constraints”. In Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service , pp. 129–136, Hong Kong, China, March 2005.

[12] Ping Wang. “An Entropy Decision Model for Selection of QoS-Aware Services Provisioning”. Department of MIS, Kun Shan University, Taiwan, 2006.

[13] Chao Lv, Wanchun Dou, Jinjun Chen. “QoS-Aware Service Selection Using QDG for B2B Collaboration”. In Proceedings of the fourteenth IEEE International Conference on Parallel and Distributed Systems, pp. 336 – 343, 2008.

[14] Hong Qing Yu, Stephan Reiff-Marganiec. “A Method for Automated Web Service Selection”. Interaction and Context Based Technologies for Collaborative Teams, project. IST-2006-034718, 2007.

[15] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. “An Introduction to Information Retrieval”. Cambridge University Press Cambridge, England, 2009.

[16] “Common Evaluation Measures”. In Proceedings of the Text Retrieval Conference (TREC) Appendices, 2006.

[17] Wenge Rong, Kecheng Liu and Lin Liang. “Personalized Web Service Ranking via User Group combining Association Rule”. In Proceedings of the IEEE International Conference on Web services, pp. 445 – 452, 2009.

[18] J. Kuck, and M. Gnasa. “Context-Sensitive Service Discovery meets Information Retrieval”. In Proceedings of fifth IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 601-605, 2007.

[19] Z. Maamar, S. Mostéfaoui, and Q. Mahmoud. “Context for Personalized Web Services”. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Hawaii, USA, 2005.

[20] Swarnamugi .M, Sathya .M. “Specification Criteria for Web Service Selection Approaches”. International Journal on Computer Engineering and Information Technology, pp. 29 – 38, vol(23), Issue No: 01, ISSN 0974-2034, May 2010.

[21] T. R. Gruber. “A translation approach to portable ontology specifications”. International Journal on Knowledge Acquisition, vol: 5(2), pp. 199–220, 1993.

[22] Ioan Toma, Douglas Foxvog, Michael C. Jaeger. “Modeling QoS characteristics in WSMO”. In Proceedings of the first workshop on Middleware for Service Oriented Computing, pp. 42 – 47, 2006.

[23] Gennady Agre, Tatiana Atanasova, Joachim Nern. “Migrating from Web Services To Semantic Web Services: InfraWebs Approach”. EU Project INFRAWEBS - IST FP62003/IST/2.3.2.3 Research Project No. 511723.

[24] Vuong Xuan Tran, Hidekazu Tsuji ,Ryosuke Masuda. “A new QoS ontology and its QoS-based ranking algorithm for Web services”. Journal on Simulation Modelling Practice and Theory, ScienceDirect, vol(17), Issue No: 8, pp. 1378-1398. September 2009.

[25] Web Service Modeling Ontology (WSMO) - http://www.wsmo.org/TR/d2/v1.3/20061021/

[26] The Web Service Modeling Language WSML - http://www.wsmo.org/TR/d16/d16.1/v0.21/20051005/

[27] Mohammad Alrifai, Thomas Risse. “Combining global optimization with local selection for efficient QoS-aware service composition”. In Proceedings of the 18th international conference on World Wide Web, ACM, ISBN: 978-1-60558-487-4, pp. 881- 890, 2009.

[28] Miguel Angel Corella, Pablo Castells. “Semi-automatic Semantic-Based Web Service Classification”. In Proceedings of Business Process Management Workshops, LNCS, pp. 459 – 470, 2006.

[29] Sathya M, Dhavachelvan P, Swarnamugi M, Sureshkumar G. “A Negotiation Model for Context-Aware Web Service Selection”. In Proceedings of the International Conference on Advanced Computing and Communication, pp. 38 – 44, Kerala, India, May 2010.

[30] IBM Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/specification/ws-bpel/

[31] Swarnamugi M, Sathya M, Dhavachelvan P. “A Negotiation Model for Web Service Selection”. In Proceedings of International Conference on Recent Trends in Soft Computing and Information Technology, pp. 251 – 256, Bhopal, India, Jan 2010.

[32] Yijun Chen, Abdolreza Abhari. “An Agent-based Framework for Dynamic Web Service Selection”. In Proceedings of the 2008 Spring simulation multiconference, ACM, pp. 13 – 16, 2008.

[33] U.S. Manikrao, and T.V. Prabhakar. “Dynamic Selection of Web Services with Recommendation System”. In Proceedings of 2005 International Conference on Next Generation Web Services Practices, pp. 117-121, 2005.

ANP-GP Approach for Selection of Software Architecture Styles

K. Delhi Babu kdb_babu@yahoo.com
Department of Computer Science and Engineering
Sri Vidyanikethan Engineering College, Tirupati,
Andhra Pradesh, India -517102

P. Govindarajulu pgovindarajulu@yahoo.com
Department Computer Science
S.V. University, Tirupati,
Andhra Pradesh, India -517502

A. Ramamohana Reddy aramamohanareddy@yahoo.com
Department Computer Science
S.V. University, Tirupati,
Andhra Pradesh, India -517502

A.N. Aruna Kumari kolla_aruna@yahoo.com
Department of Computer Science and Engineering
Sri Vidyanikethan Engineering College
Tirupati, Andhra Pradesh, India -517102

Abstract

Selection of Software Architecture for any system is a difficult task as many different stake holders are involved in the selection process. Stakeholders view on quality requirements is different and at times they may also be conflicting in nature. Also selecting appropriate styles for the software architecture is important as styles impact characteristics of software (e.g. reliability, performance). Moreover, styles influence how software is built as they determine architectural elements (e.g. components, connectors) and rules on how to integrate these elements in the architecture. Selecting the best style is difficult because there are multiple factors such as project risk, corporate goals, limited availability of resources, etc. Therefore this study presents a method, called SSAS, for the selection of software architecture styles. Moreover, this selection is a multi-criteria decision-making problem in which different goals and objectives must be taken into consideration. In this paper, we suggest an improved selection methodology, which reflects interdependencies among evaluation criteria and alternatives using analytic network process (ANP) within a zero-one goal programming (ZOGP) model.

Keywords: Software Architecture; Selection of Software Architecture Styles; Multi-Criteria Decision Making; Interdependence; Analytic Network Process (ANP); Zero-One Goal Programming (ZOGP)

1. INTRODUCTION

Software architectures significantly impact software project success [1]. However, creating architectures is one of the most complex activities during software development [2]. When creating architectures, architecture styles narrow the solution space: First, styles define what elements can exist in architecture (e.g. components, connectors). Second, they define rules on how to integrate these elements in the architecture. Moreover, styles address non-functional issues (e.g. performance) [3]. Selecting the best style is difficult because there are multiple criteria and factors such as project risk, budget, limited availability of resources, etc. Moreover,

this selection is a multi-criteria decision-making problem in which different goals and objectives must be taken into consideration. When we evaluate, we need to collect group opinion in order to know the interdependence relationship among criteria.

The contributions of this paper are as follows:
1. This paper presents a method called SSAS (Selection of Software Architecture Styles).
2. It uses analytic network process (ANP) to determine the degree of interdependence relationship among the alternatives and criteria.
3. It provides a way of collecting expert group opinion along with stakeholders interests (e.g. reliability, performance)
4. It uses a systematic procedure to determine the following factors in constructing the GP model through a group discussion: (i) objectives, (ii) desired level of attainment for each objective, (iii) a degree of interdependence relationship, and (iv) penalty weights for over or under achievement of each goal [4]

Therefore, the information obtained from ANP is then used to formulate zero-one goal programming (ZOGP) model [5]. The objective of this paper is to describe an integrated approach of style selection using ANP and GP. Thus, in this paper, we suggest an improved selection methodology, which reflects interdependencies among evaluation criteria using analytic network process within a zero-one goal programming model. Thus a systematic approach is adopted to set priorities among multi-criteria and also among alternatives.

2. ANP-GP APPROACH FOR SSAS

2.1. Analytic Network Process

The initial study identified the multi-criteria decision technique known as the Analytic Hierarchy Process (AHP) to be the most appropriate for solving complicated problems. Many decision problems cannot be structured hierarchically because they involve the interaction and dependence of higher-level elements on a lower-level element [6]. Also he suggested the use of AHP to solve the problem of independence on alternatives or criteria and the use of ANP to solve the problem of dependence among alternatives or criteria [7].
The ANP addresses how to determine the relative importance of a set of activities in a multi-criteria decision problem. The process utilizes pairwise comparisons of the style alternatives as well as pairwise comparisons of the multiple criteria [8]. Figure 1 is a standard form of a 'supermatrix' introduced by Saaty to deal with the interdependence characteristics among elements and components. He suggested Supermatrix for solving network structure [7]. The supermatrix is column stochastic as all its columns sum to unity [9]. This matrix means that any column of the limiting power $\lim_{k\to\infty} A^{2k+1}$ gives the outcome of the cyclic interaction of the alternatives and the criteria.

$$
\mathbf{W} = \begin{array}{c}
\\
C_1 \\
\\
\\
C_2 \\
\\
\\
C_n \\
\\
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} C_1 & & & & C_2 & & & \cdots & C_n & \end{array} \\
\begin{array}{cccccccccc} e_{11} & e_{12} & \cdots & e_{1m_1} & e_{21} & e_{22} & \cdots & e_{2m_2} & \cdots & e_{nm_n} \end{array} \\
\begin{bmatrix}
\mathbf{W_{11}} & & & \mathbf{W_{12}} & & & \cdots & \mathbf{W_{1n}} \\
\\
\\
\mathbf{W_{21}} & & & \mathbf{W_{22}} & & & \cdots & \mathbf{W_{2n}} \\
\vdots & & & \vdots & & & \vdots & \vdots \\
\mathbf{W_{n1}} & & & \mathbf{W_{n2}} & & & \cdots & \mathbf{W_{nn}}
\end{bmatrix}
\end{array}
\begin{array}{l}
e_{11} \\
e_{12} \\
\vdots \\
e_{1m_1} \\
e_{21} \\
e_{22} \\
\vdots \\
e_{2m_2} \\
\vdots \\
e_{n1} \\
e_{n2} \\
\vdots \\
e_{nm_n}
\end{array}
$$

Figure 1. Supermatrix

Figure 2 depicts the difference of structures and corresponding supermatrix between a hierarchy and a network. A node represents a component with elements inside it; a straight line/or an arc denotes the interactions between two components; and a loop indicates the inner dependence of elements within a component. When the elements of a component *Node*1 depend on another component *Node2*, we represent this relation with an arrow from component *Node*1 to *Node*2. The corresponding supermatrix of the hierarchy with three levels of clusters is also shown: where ***w***21 is a vector that represents the impact of the *Node*1 on the *Node*2; **W**32 is a matrix that represents the impact of the *Node*2 on each element of the *Node*3; and **I** is the identity matrix. It is observed that a hierarchy is a simple and special case of a network.

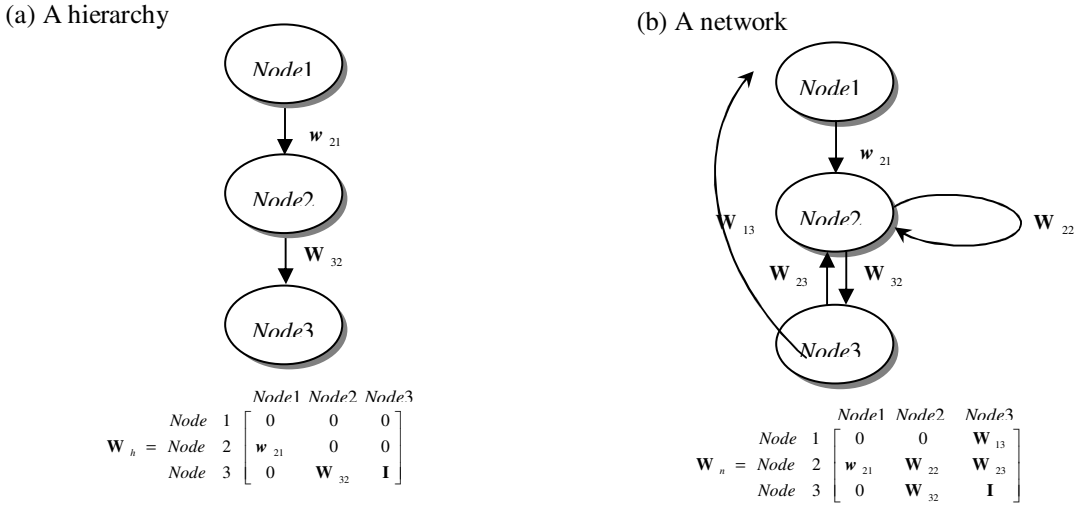

FIGURE 2: (a) Linear hierarchy and (b) Nonlinear network

The process of solving interdependence problem is summarized as follows: In order to consider interdependence, the first step is to identify the multiple criteria of merit consideration and then draw a relationship between the criteria that show the degree of interdependence among the criteria. Next step is determining the degree of impact or influence between the criteria or alternatives. When comparing the alternatives for each criterion, the decision maker will respond to questions such as: "In comparing style 1 and style2, on the basis of performance, which style is preferred?" When there is interdependence, the same decision maker answers the following kind of question (pairwise comparisons): "Given an alternative and an attribute, which of the two alternatives influences the given alternatives more with respect to the attribute? and how much more than the other alternative?" The responses are presented numerically, scaled on the basis of Saaty's proposed 1-9 scale with reciprocals, in a style comparison matrix. The final step is to determine the overall prioritization.

2.2. Goal programming

The information obtained from the ANP is then used to formulate a zero-one goal programming (ZOGP) model as a weight. The solution to ZOGP will provide a pattern by which weights will be allocated among architecture styles [10, 11].

The ZOGP model for architecture style selection can be stated as follows:

Minimize $$Z = P_K(w_j d_i^+, w_j d_i^-) \quad (1)$$

Subject to $$a_{ij} x_j + d_i^- - d_i^+ \le b_i$$

for i = 1, 2, …, m, j = 1, 2, …, n (2)

$$x_j + d_i^- = 1$$

for i = m+1, m+2, …m+n, j = 1, 2, …, n (3)

x_j = 0 or 1

for $\forall j$ (4)

where m = the number of goals to be considered in the model, n = the pool of architecture styles from which the optimal set will be selected, w_j = the ANP mathematical weight on the j =1, 2,…, n architecture style, P_K = some k priority preemptive priority $(P_1 > P_2 > \cdots > P_k)$, for i =1, 2,…, m goals, d_i^+, d_i^- = the ith positive and negative deviation variables for i = 1, 2,…, m goals, x_j = a zero-one variable, where j = 1, 2,…, n possible projects to choose from and where x_j = 1, then select the jth architecture style or when x_j =0, then do not select the jth architecture style, a_{ij} = the jth parameter of the ith resources, and b_i = the ith available resource or limitation factors that must be considered in the selection decision.

The ZOGP model selects the best architectural style x_j for which the weight w_j is derived from ANP which has maximum value and minimum deviation d_j.

3. A CASE-STUDY FOR SELECTION OF SOFTWARE ARCHITECTURE STYLE

A case study to illustrate the advantages of the integrated ANP and ZOGP based on the expert opinion of an organization is taken [10, 11]. The problem consisted of prioritizing three architectures styles [1] on the basis of seven criteria deemed to be important for an organization. The criteria used are (1) Efficiency (E), (2) Scalability (S), (3) Evolvability (Ev), (4) Portability (P), (5) Reliability (R), (6) Performance (Pe) and (7) Configurability (C). It should be noted that, the traditional AHP is applied to the problem without considering interdependence property among the criteria.

However, we are of the opinion that there is an existence of interdependence relationship among these seven criteria. The attribute of criteria P influence criteria C, the attribute of criteria Ev influence criteria R, E, S, Pe, C and P, and criteria R influence criteria C, Pe, Ev, E and S and so on. In order to check network structure or relationship of criteria or alternative, we need to have group discussion because the type of network or relationship depends on the stakeholders' judgment. The relationship having interdependence among the criteria is shown in Figure 3.

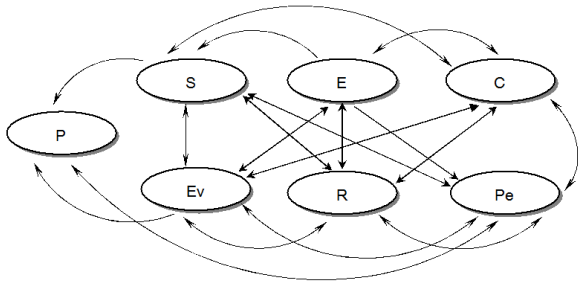

FIGURE 3: Interdependent relationship among the criteria.

In order to find the weight of the degree of influence among the criteria, we will show the procedure using the matrix manipulation based on Saaty's supermatrix. The procedure is shown as follows:

Step 1: Compare the criteria, through the question: "Which criteria should be emphasized, and how much more?". Then by pairwise comparison of all pairs with respect to the three architecture styles (LS, PF, BB) [16], we will get the following data via AHP method (E, S, Ev, P, R, Pe, C) = (0.383, 0.163, 0.098, 0.022, 0.223, 0.072, 0.040). Assume that there is no interdependence among criteria and architecture styles [15]. The weight matrix criteria W_1 = (E, S, Ev, P, R, Pe, C) = (0.383, 0.163, 0.098, 0.022, 0.223, 0.072, 0.040).

Step 2: Again assume that there is no interdependence among the three architecture styles with respect to each criterion yielding the each column normalized to one, as shown in Table 1.

Table 1. Data of three architecture styles to seven criteria (E, S, Ev, P, R, Pe, C)

W_2	*E*	*S*	*Ev*	*P*	*R*	*Pe*	*C*
LS	7	7	7	5	9	7	9
PF	7	9	5	7	7	9	7
BB	5	7	9	3	5	7	5
LS	0.368	0.304	0.333	0.333	0.429	0.304	0.429
PF	0.368	0.391	0.238	0.467	0.333	0.391	0.333
BB	0.263	0.304	0.429	0.200	0.238	0.304	0.238
	W_{21}	W_{22}	W_{23}	W_{24}	W_{25}	W_{26}	W_{27}

The second row of data in Table 1 gives the degree of relative importance for each criterion, and the data of third row sum is normalized to one, for each criteria. We defined the weight matrix of three styles for criteria E as

$$w_{21} = \begin{bmatrix} 0.368 \\ 0.368 \\ 0.263 \end{bmatrix}$$

Step 3: Next, we considered the interdependence among the criteria. When we select the architecture style, we cannot concentrate only on one criterion, but we must consider the other criteria also. Therefore, we need to examine the impact of one criterion on all other criteria by using pairwise comparisons and so on [12]. In Table 2, we obtain the seven sets of weights through expert opinion. The data of Table 2 shows seven criteria's degree of relative impact for each seven criteria. For example, the E's degree of relative impact for Ev is 0.291, the Ev's degree of relative impact for C is 0.059, and the R's degree of relative impact for Pe is 0.168.

Table 2. Data among seven criteria

W_3	E	S	Ev	P	R	Pe	C
E	0.564	0.093	0.291	0	0.093	0.256	0.022
S	0	0.422	0.085	0.118	0.268	0.053	0.156
Ev	0.055	0.047	0.402	0.263	0.025	0.090	0.059
P	0	0	0	0.564	0	0	0.270
R	0.118	0.244	0.049	0	0.398	0.168	0.037
Pe	0.263	0.169	0.146	0	0.047	0.402	0.088
C	0	0.025	0.027	0.055	0.169	0.033	0.369

We defined the interdependence weight matrix of criteria as

$$W_3 = \begin{bmatrix} 0.564 & 0.093 & 0.291 & 0 & 0.093 & 0.256 & 0.022 \\ 0 & 0.422 & 0.085 & 0.118 & 0.268 & 0.053 & 0.156 \\ 0.055 & 0.047 & 0.402 & 0.263 & 0.025 & 0.090 & 0.059 \\ 0 & 0 & 0 & 0.564 & 0 & 0 & 0.270 \\ 0.118 & 0.244 & 0.049 & 0 & 0.398 & 0.168 & 0.037 \\ 0.263 & 0.169 & 0.146 & 0 & 0.047 & 0.402 & 0.088 \\ 0 & 0.025 & 0.027 & 0.055 & 0.169 & 0.033 & 0.369 \end{bmatrix}$$

Table 3 to Table 9 shows the data interdependence among criteria's degree of relative impact for each criteria individually.

Table 3. Data among four interdependent criteria's degree of relative impact for criteria 1 (E)

W_{31}	*E*	*P*	*R*	*Pe*
E	1	7	5	3
P	1/7	1	1/3	1/5
R	1/5	3	1	1/3
Pe	1/3	5	3	1

The interdependence weight of the criteria W_{31} = (0.564, 0.055, 0.118, 0.263).

Table 4. Data among six interdependent criteria's degree of relative impact for criteria 2 (S)

W_{32}	*E*	*S*	*Ev*	*R*	*Pe*	*C*
E	1	1/5	3	1/3	1/3	5
S	5	1	7	3	3	9
Ev	1/3	1/7	1	1/5	1/5	3
R	3	1/3	5	1	3	7
Pe	3	1/3	5	1/3	1	7
C	1/5	1/9	1/3	1/7	1/7	1

The interdependence weight of the criteria W_{32} = (0.093, 0.422, 0.047, 0.244, 0.169, 0.025).

Table 5. Data among six interdependent criteria's degree of relative impact for criteria 3 (Ev)

W_{33}	*E*	*S*	*Ev*	*R*	*Pe*	*C*
E	1	5	1/3	7	3	9
S	1/5	1	1/5	3	1/3	5
Ev	3	5	1	7	3	7
R	1/7	1/3	1/7	1	1/3	3
Pe	1/3	3	1/3	3	1	5
C	1/9	1/5	1/7	1/3	1/5	1

The interdependence weight of the criteria W_{33} = (0.291, 0.085, 0.402, 0.049, 0.146, 0.027).

Table 6. Data among four interdependent criteria's degree of relative impact for criteria 4 (P)

W_{34}	*S*	*Ev*	*P*	*C*
S	1	1/3	1/5	3
Ev	3	1	1/3	5
P	5	3	1	7
C	1/3	1/5	1/7	1

The interdependence weight of the criteria W_{34} = (0.118, 0.263, 0.564, 0.055).

Table 7. Data among six interdependent criteria's degree of relative impact for criteria 5 (R)

W_{35}	*E*	*S*	*Ev*	*R*	*Pe*	*C*
E	1	1/3	5	1/5	3	1/3
S	3	1	9	1/3	7	3
Ev	1/5	1/9	1	1/9	1/3	1/7
R	5	3	9	1	5	3
Pe	1/3	1/7	3	1/5	1	1/5
C	3	1/3	7	1/3	5	1

The interdependence weight of the criteria W_{35} = (0.093, 0.268, 0.025, 0.398, 0.047, 0.169).

Table 8. Data among six interdependent criteria's degree of relative impact for criteria 6 (Pe)

W_{36}	*E*	*S*	*Ev*	*R*	*Pe*	*C*
E	1	5	3	3	1/3	7
S	1/5	1	1/3	1/5	1/5	3
Ev	1/3	3	1	1/3	1/5	3
R	1/3	5	3	1	1/3	5
Pe	3	5	5	3	1	7
C	1/7	1/3	1/3	1/5	1/7	1

The interdependence weight of the criteria W_{36} = (0.256, 0.053, 0.090, 0.168, 0.402, 0.033).

Table 9. Data among seven interdependent criteria's degree of relative impact for criteria 7 (C)

W_{37}	*E*	*S*	*Ev*	*P*	*R*	*Pe*	*C*
E	1	1/7	1/3	1/9	1/3	1/5	1/9
S	7	1	3	1/3	5	3	1/3
Ev	3	1/3	1	1/5	3	1/3	1/5
P	9	3	5	1	7	5	1/3
R	3	1/5	1/3	1/7	1	1/3	1/7
Pe	5	1/3	3	1/5	3	1	1/5
C	9	3	5	3	7	5	1

The interdependence weight of the criteria W_{37} = (0.022, 0.156, 0.059, 0.270, 0.037, 0.088, 0.369).

Step 4: Next, we dealt with the interdependence among the architecture styles with respect to each criterion [14]. To satisfy the criteria, "which style contributes more and how much more?" The stake holder response for each criterion is tabulated as shown from Table 10 to Table 16.

Table 10. Data among three architecture styles for criteria 1 (E)

W_{41}	*LS*	*PF*	*BB*
LS	1	1/3	5
PF	3	1	5
BB	1/5	1/5	1
LS	0.238	0.217	0.455
PF	0.714	0.652	0.455
BB	0.048	0.130	0.091

In Table 10, the data of second row is obtained from stake holders (Saaty's nine scale), which shows the degree of interdependence among the alternatives with respect to each style and the column sum is normalized to one. The project interdependence weight matrix for criteria E is W_{41}.

Table 11. Data among three architecture styles for criteria 2 (S)

W_{42}	*LS*	*PF*	*BB*
LS	1	1/5	1/3
PF	5	1	3
BB	3	1/3	1
LS	0.111	0.130	0.077
PF	0.556	0.652	0.692
BB	0.333	0.217	0.231

Table 12. Data among three architecture styles for criteria 3 (Ev)

W_{43}	*LS*	*PF*	*BB*
LS	1	7	3
PF	1/7	1	1/5
BB	1/3	5	1
LS	0.678	0.538	0.714
PF	0.097	0.077	0.048
BB	0.226	0.385	0.238

Table 13. Data among three architecture styles for criteria 4 (P)

W_{44}	*LS*	*PF*	*BB*
LS	1	1/3	5
PF	3	1	5
BB	1/5	1/5	1
LS	0.238	0.217	0.455
PF	0.714	0.652	0.455
BB	0.048	0.130	0.091

Table 14. Data among three architecture styles for criteria 5 (R)

W_{45}	*LS*	*PF*	*BB*
LS	1	3	5
PF	1/3	1	3
BB	1/5	1/3	1
LS	0.652	0.692	0.556
PF	0.217	0.231	0.333
BB	0.130	0.077	0.111

Table 15. Data among three architecture styles for criteria 6 (Pe)

W_{46}	*LS*	*PF*	*BB*
LS	1	1/7	1/7
PF	7	1	3
BB	7	1/3	1
LS	0.067	0.097	0.035
PF	0.467	0.678	0.724
BB	0.467	0.226	0.241

Table 16. Data among three architecture styles for criteria 7 (C)

W_{47}	*LS*	*PF*	*BB*
LS	1	3	5
PF	1/3	1	5
BB	1/5	1/5	1
LS	0.652	0.714	0.455
PF	0.217	0.238	0.455
BB	0.130	0.048	0.091

Step 5: The interdependence priorities of the criteria by synthesizing the results from Step 1 to Step 3 as:

$$W_c = W_3 \times W_1 = \begin{bmatrix} 0.564 & 0.093 & 0.291 & 0 & 0.093 & 0.256 & 0.022 \\ 0 & 0.422 & 0.085 & 0.118 & 0.268 & 0.053 & 0.156 \\ 0.055 & 0.047 & 0.402 & 0.263 & 0.025 & 0.090 & 0.059 \\ 0 & 0 & 0 & 0.564 & 0 & 0 & 0.270 \\ 0.118 & 0.244 & 0.049 & 0 & 0.398 & 0.168 & 0.037 \\ 0.263 & 0.169 & 0.146 & 0 & 0.047 & 0.402 & 0.088 \\ 0 & 0.025 & 0.027 & 0.055 & 0.169 & 0.033 & 0.369 \end{bmatrix} \times \begin{bmatrix} 0.383 \\ 0.163 \\ 0.098 \\ 0.022 \\ 0.223 \\ 0.072 \\ 0.040 \end{bmatrix} = \begin{bmatrix} 0.300 \\ 0.150 \\ 0.088 \\ 0.023 \\ 0.192 \\ 0.186 \\ 0.063 \end{bmatrix}$$

W_c = (E, S, Ev, P, R, Pe, C) = (0.300, 0.150, 0.088, 0.023, 0.192, 0.186, 0.063).

Step 6: The priorities of the architecture styles W_p with respect to each of the seven criteria are given by synthesizing the results from Step 2 to Step 4 as follows:

$$W_{P1} = W_{41} \times W_{21} = \begin{bmatrix} 0.238 & 0.217 & 0.455 \\ 0.714 & 0.652 & 0.455 \\ 0.048 & 0.130 & 0.091 \end{bmatrix} \times \begin{bmatrix} 0.368 \\ 0.368 \\ 0.263 \end{bmatrix} = \begin{bmatrix} 0.287 \\ 0.622 \\ 0.089 \end{bmatrix}$$

$$W_{P2} = W_{42} \times W_{22} = \begin{bmatrix} 0.111 & 0.130 & 0.077 \\ 0.556 & 0.652 & 0.692 \\ 0.333 & 0.217 & 0.231 \end{bmatrix} \times \begin{bmatrix} 0.304 \\ 0.391 \\ 0.304 \end{bmatrix} = \begin{bmatrix} 0.108 \\ 0.634 \\ 0.256 \end{bmatrix}$$

$$W_{P3} = W_{43} \times W_{23} = \begin{bmatrix} 0.678 & 0.538 & 0.714 \\ 0.097 & 0.077 & 0.048 \\ 0.226 & 0.385 & 0.238 \end{bmatrix} \times \begin{bmatrix} 0.333 \\ 0.238 \\ 0.429 \end{bmatrix} = \begin{bmatrix} 0.660 \\ 0.071 \\ 0.269 \end{bmatrix}$$

$$W_{P4} = W_{44} \times W_{24} = \begin{bmatrix} 0.238 & 0.217 & 0.455 \\ 0.714 & 0.652 & 0.455 \\ 0.048 & 0.130 & 0.091 \end{bmatrix} \times \begin{bmatrix} 0.333 \\ 0.467 \\ 0.200 \end{bmatrix} = \begin{bmatrix} 0.272 \\ 0.633 \\ 0.095 \end{bmatrix}$$

$$W_{P5} = W_{45} \times W_{25} = \begin{bmatrix} 0.652 & 0.692 & 0.556 \\ 0.217 & 0.231 & 0.333 \\ 0.130 & 0.077 & 0.111 \end{bmatrix} \times \begin{bmatrix} 0.429 \\ 0.333 \\ 0.238 \end{bmatrix} = \begin{bmatrix} 0.642 \\ 0.249 \\ 0.108 \end{bmatrix}$$

$$W_{P6} = W_{46} \times W_{26} = \begin{bmatrix} 0.067 & 0.097 & 0.035 \\ 0.467 & 0.678 & 0.724 \\ 0.467 & 0.226 & 0.241 \end{bmatrix} \times \begin{bmatrix} 0.304 \\ 0.391 \\ 0.304 \end{bmatrix} = \begin{bmatrix} 0.069 \\ 0.627 \\ 0.304 \end{bmatrix}$$

$$W_{P7} = W_{47} \times W_{27} = \begin{bmatrix} 0.652 & 0.714 & 0.455 \\ 0.217 & 0.238 & 0.445 \\ 0.130 & 0.048 & 0.091 \end{bmatrix} \times \begin{bmatrix} 0.429 \\ 0.333 \\ 0.238 \end{bmatrix} = \begin{bmatrix} 0.626 \\ 0.281 \\ 0.093 \end{bmatrix}$$

The matrix W_p by grouping all the seven columns:

$$W_p = (W_{p1}, W_{p2}, W_{p3}, W_{p4}, W_{p5}, W_{p6}, W_{p7}) .$$

$$W_p = \begin{bmatrix} 0.287 & 0.108 & 0.660 & 0.272 & 0.642 & 0.069 & 0.626 \\ 0.622 & 0.634 & 0.071 & 0.633 & 0.249 & 0.627 & 0.281 \\ 0.089 & 0.256 & 0.269 & 0.095 & 0.108 & 0.304 & 0.093 \end{bmatrix}$$

Step 7: Finally, the overall priorities for the architecture styles W_A are calculated by multiplying W_p by W_c.

$$W_A = W_p \times W_c = \begin{bmatrix} 0.287 & 0.108 & 0.660 & 0.272 & 0.642 & 0.069 & 0.626 \\ 0.622 & 0.634 & 0.071 & 0.633 & 0.249 & 0.627 & 0.281 \\ 0.089 & 0.256 & 0.269 & 0.095 & 0.108 & 0.304 & 0.093 \end{bmatrix} \times \begin{bmatrix} 0.300 \\ 0.150 \\ 0.088 \\ 0.023 \\ 0.192 \\ 0.186 \\ 0.063 \end{bmatrix} = \begin{bmatrix} 0.342 \\ 0.484 \\ 0.174 \end{bmatrix}$$

The final results in the ANP Phase are (LS, PF, BB) = (0.342, 0.484, 0.174). These weights are used as priorities in goal programming formulation. That is (LS, PF, BB) = (w_1, w_2, w_3) = (0.342, 0.484, 0.174), w_j are the values of the three architecture styles.

The weight vector obtained from the above ANP model is used to optimize the solution further by zero-one goal programming as follows: There exist several obligatory and flexible goals that must be considered in the selection from the available pool of three architecture styles. There are three obligatory goals: (1) a maximum time of 24 working days is required to select the best architecture style, (2) a maximum duration of 35 months is required to complete the software project and (3) a maximum budget of $ 30,000 is allocated to develop the project.

In addition to the obligatory goals of selecting the best architecture style, there are two other flexible goals, stated in order of importance: (1) allocation of budget is set at $30,000 and (2) allocation of miscellaneous fees is set at $4200, deviation from this allocation is not allowed. In Table 17, the cost and resource usage information for each of the three styles is presented.

Table 17.Cost and resources usage information

	Project resource usage (a_{ij})			
	x_1	x_2	x_3	b_i
Planning and design days	10	24	18	24 days
Construction months	32	34	30	35 months
Budgeted cost (00)	$150	$300	$280	$300
Misc cost (00)	$18	$24	$15	$42

Based on the weight vector computed using ANP, we can formulate the goal constraints in Table 18. This ZOGP model is solved using LINDO Ver 6.1. The results are summarized as follows:

Table 18. ZOGP model formulation

ZOGP model formulation	Goals
Minimize Z =	
$pl_1(d_1^+ + d_2^+ + d_3^+)$	Satisfy all obligatory goals
$pl_2(0.342d_5^- + 0.484d_6^- + 0.174d_7^-)$	Select highest ANP weighted architecture styles
$pl_3(d_8^- + d_8^+)$	Use $30,000 for all architecture styles selected
$pl_4(d_4^- + d_4^+)$	Use $4200for all architecture styles selected
Subject to	
$10X_1 + 24X_2 + 18X_3 + d_1^- - d_1^+ = 24$	Avoid over utilizing max. planning and design days
$32X_1 + 34X_2 + 30X_3 + d_2^- - d_2^+ = 35$	Avoid over utilizing max. construction months
$150X_1 + 300X_2 + 280X_3 + d_3^- - d_3^+ = 300$	Avoid over utilizing max. budgeted dollars
$X_1 + d_5^- = 1$	Select Layered Style (LS)
$X_2 + d_6^- = 1$	Select Pipe & Filter (PF)
$X_3 + d_7^- = 1$	Select Blackboard Style (BB)
$18X_1 + 24X_2 + 15X_3 + d_4^- - d_4^+ = 42$	Avoid over or under utilizing misc cost
$150X_1 + 300X_2 + 280X_3 + d_8^- - d_8^+ = 300$	Avoid over or under utilizing expected budget
$Xj = 0$ or $\forall j = 1,2,3$	

$$x_1 = 0\ x_2 = 1,\quad x_3 = 0$$

$$d_1^- = 0,\ d_1^+ = 0,\ d_2^- = 1,\ d_2^+ = 0,\ d_3^- = 0,\ d_3^+ = 0,\ d_4^- = 18,\ d_4^+ = 0, d_5^- = 1,\ d_6^- = 0,\ d_7^- = 1,\ d_8^- = 0,\ d_8^+ = 0.$$

Architecture Style 2 is chosen as it is consumes the total budgeted cost of $30,000 and use 14 days of time for decision. Also, the selected style will save one month construction time (total time is 35 months) as $d_2^- = 1$.

4. DISCUSSION

Several methods have been proposed to help organizations for solving problems related to interdependence among criteria. The existing methodologies range from single-criteria cost/benefit analysis to multiple criteria scoring models, ranking methods and AHP. However they did not consider interdependence property. But they have addressed consider independence property among alternatives or criteria. Also Ranking, Scoring, AHP methods are not applicable to problems having resource feasibility, optimization requirements. In spite of this limitation, the ranking and scoring method and AHP method have been used with real problems because they are simple and easy to understand. In order to solve optimization problems, researchers have used mathematical methods such as goal programming, dynamic programming, etc. [25, 30]. Many real-world problems are related to interdependence among alternatives and/or criteria (multiple criteria) and these problems are need to apply resource feasibility, optimization and so on. Table 15 shows the list of methods for various problem characteristics.

Table 19. List of methods for various problem characteristics

Method	Multiple Criteria	Resource Feasibility	Interdependence	Optimization required
Ranking [16]	Yes	No	No	No
Scoring [17]	Yes	No	No	No
AHP [18]	Yes	No	No	No
Goal Programming [20]	Yes	Yes	No	Yes
Dynamic Programming[19]	No	Yes	Yes	Yes
AHP-GP [13]	Yes	Yes	No	Yes
ANP-GP (This paper)	Yes	Yes	Yes	Yes

According to experts, in selecting a style there is no single decision involved but in the decisions consideration may be better or worse but still significant. For example, a style with a low weight might be selected over a style with a high weight if developers are more familiar with the style which has a lower score. The weight vector obtained using AHP for the above example is (0.371, 0.474, 0.154) [18]. AHP and ANP approaches have no much difference in solving the example given, but there are some differences with respect to decision variables. It is evident that resource feasibility, optimization requirements cannot be fulfilled with AHP method. But it is simple and easy to understand and so the method more frequently used [21, 22, 24]. Table 18 shows the comparison among the AHP and ANP approaches.

Table 20. Comparison of AHP and ANP approaches

Method	Resources Used			
	Planning and design days	Construction months	Budgeted cost (00)	Misc cost (00)
AHP	24	35	300	42
ANP	24	34*	300	18**

* We will save one month construction time (total time is 35 months) as $d_2^- = 1$

** We will use only Misc cost \$1800 (<\$4200) more than the initial Budgeted cost as $d_4^- = 18$.

The proposed model, ANP is to demonstrate the procedure of finding weight that considers interdependence among criteria or alternatives [23] which has highest weight wj. The ZOGP model selects the best architectural style for which the weight wj is derived from ANP which has maximum value and minimum deviation dj. Finally, architecture Style 2 is chosen which is optimum as it is consumes the total budget cost of \$30,000 and use exactly 24 days of time for decision. The selected style will save one month construction time (total time is 35 months) as $d_2^- = 1$.

In literature, all techniques mainly focused on problems related to independence among criteria. Also recent survey indicates that the use of mathematical models is becoming prevalent for solving this kind of problems [25, 26]. This paper shows an example solving interdependence problem using the integrated approach ANP and ZOGP by using group expert interview. Using this approach we conclude that we can select suitable architecture style having multiple criteria, interdependence and resource feasibility.

5. CONCLUSIONS

There are mainly two inadequacies in the traditional approaches for selection of architecture styles. First, they focused on relative importance among criteria to minimize the cost. However, the interests of stakeholders and experts opinion were neglected. Second they considered only quantitative factors.

To overcome the above drawbacks, this paper presented a method for a selecting the best architecture style. In this method, ANP is used to determine the interdependency among the alternatives and criteria. The priority vector obtained from **Analytic Network Process** is used to formulate **Zero-One Goal Programming** model. For some scenarios, it might be obvious if all architecture element types and all architecture properties are taken into consideration. So in this paper three architecture styles and seven criteria are used in the case study. The major advantage of this integrated approach is both the interests of stakeholder and expert opinion are focused. Qualitative factors are also considered. Therefore, it is believed that this approach is much more practical and the results obtained in this approach are better than earlier approaches like Fuzzy Logic, AHP, ANP for selecting the best architecture style.

6. REFERENCES

[1] Shaw M., Clements P.: “The golden age of software architecture”, IEEE Softw., 2006

[2] Garlan D.: “Software architecture: a roadmap’’, in Finkelstein A. (ED.): “The future of software engineering” (ACM Press, 2000)

[3] Dobrica L., Niemela E.: “A survey on software architecture analysis methods”, IEEE Trans. Softw. Eng., 2002

[4] Soung Hie K., Jin Woo L.: "An Optimization usability of information system project resources using a QFD and ZOGP for reflection customer wants", Korea Advanced Inst of Science & Tech.

[5] Jin Woo Lee, Soung Hie Kim: “Using analytic network process and goal programming for interdependent information system project selection”, Computers & Operation Research Volume 27, Issue 4, April 2000

[6] Saaty, T. L., “A Scaling Method for Priorities in Hierarchical structures”, Journal of Mathematical Psychology, 1984

[7] Saaty and Takizawa: "The Theory of Ratio Scale estimation", Management Science Vol 33, Nov 1987

[8] Gloria E., Eugene D.and Guisseppi A.,:"A multiple-criteria framework for evaluation of decision support systems", Omega Volume 32, Issue 4, August 2004

[9] Thomas L. Saaty, Luis Gonzalez Vargas: "Decision making with the analytic network process", Management Science and Operation Research, 2006

[10] Reza, K., Hossein, A., Yvon, G.,: "An integrated approach to project evaluation and selection", IEEE Transactions on Engineering Management 1988

[11] Elim Liu, Shih-Wen Hsiao: “ANP-GP approach of Product Variety Design”, Int J Adv Manuf Technol 2006.

[12] Selcuk Percin : "Using the ANP approach in selecting and benchmarking ERP systems" Emerald Group Publishing Limited, 1994

[13] K. Delhi Babu, P. Govindarajulu, A. Ramamohana Reddy, A.N. Aruna Kumari : “An Integrated Approach of AHP-GP and Visualization for Selection of Software Architecture: A Framework”, International Conference on Advances in Computer Engineering, 2010.

[14] Ingu Kim, Shangmun Shin: "Development of a Project Selection Method on Information System Using ANP and Fuzzy Logic", World Academy of Science, Engg. and Tech. 2009

[15] Galster M.,Eberlein A. and Moussavi A.:"Systematic selection of software architecture styles", Published in IET Software 2009.

[16] Martin D.J. Buss : "How to Rank Computer Projects", Harvard Business Review Article, Jan, 1983.

[17] HC Lucas, JR Moor : "A multiple-criterion scoring approach to information system project selection", INFOR 1976

[18] K Muralidhar, R Santhanam : "Using the analytic hierarchy process for information system project selection" Information & Management, Volume 18, Issue 2, February 1990

[19] G. L. Nemhauser, Z. UllmannDiscrete : "Dynamic Programming and Capital Allocation", Management Science, Vol. 15, No. 9, Theory Series May, 1969

[20] R Santhanama, K Muralidhara, M Schniederjans : "A zero-one goal programming approach for information system project selection", Omega Volume 17, Issue 6, 1989

[21] Saaty T.L.: “Decision making for leaders: the analytic hierarchy process for decisions in a complex world”, RWS Publications, 2001

[22] Fleiss J.L., Levin B., Paik M.C.: “Statistical methods for rates and proportions”, John Wiley & Sons, 2003

[23] Yubo Gao An AHP-GP Model for Determining Weights, 3" World Congress on Intelligent Control and Automation, 2000

[24] K. Delhi Babu, P. Govindarajulu, A. Ramamohana Reddy, A.N. Aruna Kumari : "Selection of Architecture Styles using Analytic Network Process for the Optimization of Software Architecture", International Journal of Computer Science and Information Security, IJCSIS, Vol. 8 No. 1, April 2010

[25] P. Shoval, Y. Lugasi : "Models for computer system evaluation and selection", Information and Management archive, Volume 12 Issue 3, March 1987

[26] Weber, R.,Werners, B.,Zimmermann, H. J.,Zimmermann, H. J., : "Planning models for research and development" European Journal of Operational Research, 1990

Defect Management Practices and Problems in Free/Open Source Software Projects

Dr. Anu Gupta anugupta@pu.ac.in
Assistant Professor
Department of Computer Science and Applications
Panjab University, Chandigarh, 160014, India.

Dr. R.K. Singla rksingla@pu.ac.in
Professor
Department of Computer Science and Applications
Panjab University,Chandigarh, 160014, India.

Abstract

With the advent of Free/Open Source Software (F/OSS) paradigm, a large number of projects are evolving that make use of F/OSS infrastructure and development practices. Defect Management System is an important component in F/OSS infrastructure which maintains defect records as well as tracks their status. The defect data comprising more than 60,000 defect reports from 20 F/OSS Projects is analyzed from various perspectives, with special focus on evaluating the efficiency and effectiveness in resolving defects and determining responsiveness towards users. Major problems and inefficiencies encountered in Defect Management among F/OSS Projects have been identified. A process is proposed to distribute roles and responsibilities among F/OSS participants which can help F/OSS Projects to improve the effectiveness and efficiency of Defect Management and hence assure better quality of F/OSS Projects.

Keywords: Free Software, Open Source, Defect Management, Quality, Metrics

1. INTRODUCTION

Free/Open Source Software (F/OSS) is an evolving paradigm of software development which allows the entire Internet community to use its combined programming knowledge, creativity and expertise to develop software solutions, which could render benefits to whole community without involving cost and proprietary issues [1]. F/OSS participants rely on extensive peer collaboration through the Internet using Version Control System, Mailing List, Defect Management System, Internet Relay Chat, Discussion Forum etc. [2]. These tools enable participants to collaborate in the F/OSS development process as well as act as repositories to store the communication activities of the participants, manage the progress and evolution of F/OSS Projects. These repositories contain explicit and implicit knowledgebase about F/OSS projects that can be mined to help developers in improving the product as well as to facilitate the users in evaluating the product.

Even though there are number of qualitative and quantitative studies about F/OSS, little attention has been paid to the rich information stored in Defect Management Systems of F/OSS projects [3-8]. Defect Management System provides an effective mechanism for recording and tracking of defects as well as promotes user involvement and peer review process. All the users may not have knowledge to participate in the development or code review of an F/OSS Project but such users may report bugs or request new features. They may also comment on existing defect reports or help in their removal, for example by reproducing them or supplying more information. A large amount of defect related data flows back and forth between the developers and the users of the F/OSS projects. Hence in most of the F/OSS projects, substantial amount of defect data gets accumulated in the Defect Management Systems over the period. This valuable defect data can be used to analyze the past experience, degree of improvement or deterioration in resolving defects and determine responsiveness towards users. As the potential F/OSS users need to evaluate the extensibility and maintainability before

taking any decision to adopt a particular F/OSS product, so the defect related analysis can greatly help them to evaluate how efficiently and effectively the requests for fixing bugs, enhancing features, translation requests, support requests etc. are being managed. Moreover the availability of huge amount of information with a great variety in size, programming languages, tools, methods etc. offers the possibility of creating a comparison framework among F/OSS projects from which knowledge and experience can be gained.

In the current study, the defect data of various F/OSS projects is analyzed from various perspectives, with special focus on evaluating the efficiency and effectiveness in resolving defects and determining responsiveness towards users. Based on the findings, effective ways and means are suggested to improve defect management and thus enhance the quality of F/OSS projects.

2. F/OSS PROJECT SELECTION AND DATA COLLECTION

F/OSS projects are selected from SourceForge, a centralized place for F/OSS developers to host their projects [9]. It is the world's largest F/OSS projects repository with more than 230,000 F/OSS projects and over 2 million registered users. It provides some of the best empirical data on F/OSS research. A single source is chosen to select projects in order to control for differences in available tools and project visibility. In spite of large number of projects hosted, only a small proportion of these projects are actually active. Also many of the F/OSS projects either do not use or do not allow public access to Defect Management System. Hence those projects are considered for which defect related data is publicly accessible and is being maintained completely at SourceForge. Another criterion used for selection of projects is the project development stage (1-6 where 1 is the planning and 6 is a mature stage). A cut-off of 5 is chosen which indicates that the selected projects are at similar stage of development and are not in the early stage of development lifecycle. A total of 20 projects are selected which constitute a diverse mix of project size, team size, nature of application and targeted end user type. Selection of limited number of projects has helped to carry out in-depth study. For all the selected F/OSS projects, detailed defect data is downloaded from SourceForge Research Data Archive (SRDA) for the period starting from their respective Registration Date to October 2008 [10]. The defect data is downloaded on the basis of unique Project ID assigned to each project at SourceForge and is stored in the local repository (mySQL) aggregating more than 60,000 defect records. Further the Defect Analysis and Reporting Tool (DART) is used to carry out exhaustive analysis of defect data and generate variety of textual/graphical reports. For selected F/OSS projects, various parameters used for analyzing defect data and their quantitative results are discussed in subsequent sections.

3. Quality Metrics used for evaluating Defect Management

In order to evaluate Defect Management among F/OSS projects, various metrics used are mentioned in Table 1.

Sr. #	Metric Name	Formula	Objective
1.	Defect Resolution	Cumulative Defect Arrival Pattern and Defect Closure pattern over time interval (in months)	To check consistency and efficiency in defect resolution over the period
2.	Pending Defects	Frequency as well volume of increase/decrease in pending defects over period (in months)	To check the trend of pending defects over the period
3.	Defect Removal Rate	Proportion of defects resolved out of defects submitted for a particular period	To observe the rate at which defects are resolved over the period
4.	Backlog Management	Ratio of number of defects closed to number of defects arrived during the period	To measure the capability to handle the pending defects
5.	Software Release and Backlog Management	Tracing the shapes of BMI curves with release history of the project	To observe the relationship of software releases with defect handling over the period
6.	Defect Resolution Age	Number of days elapsed since a defect arrived till the time defect is resolved/closed.	To measure the resolution efficiency
7.	Fix/Non-Fix Defect Resolution	Defect Resolution Age for Fix versus Non-Fix Defects	To compare the efficiency in handling defects requiring code change with defects requiring no code change
8.	Defect Pending Age	Number of days elapsed since a defect arrived and still remained pending at the end of the month	To measure the age of pending defects at any point of time
9.	Defect Resolution (Defect Type Wise)	Defect Resolution Age for Bugs versus Feature Requests versus others	To compare the efficiency in handling defects belonging to various defect types

TABLE 1: Quality Metrics used for Evaluating Defect Management

4. QUANTITATIVE RESULTS

The detailed results obtained are being presented with the help of statistics and various graphs in the following subsections.

4.1. Defect Resolution

Defect arrival curves and defect closure curves have been drawn for all the selected F/OSS projects on the basis of live defect data consolidated on monthly basis. Defect arrival curve is related to the defects reported by F/OSS community, represented as Cumulative Defects arrived over the period. Defect closure curve is related to the resolution and closing of defects by F/OSS community, represented by Cumulative Defects closed over the period. The distance between these two curves at a given point in time represents the number of defects pending at that time. An ideal defect resolution process needs to be

- **Continuous:** when cumulative closed curve is quite smooth without having any peaks or steps.
- **Efficient:** when cumulative closed curve stays near to the cumulative open curve without raising overall number of pending defects.

The graphs for all the selected F/OSS projects have been drawn which show varying patterns. Those patterns can be classified among the following four categories [11]:

- Continuous and Efficient
- Discontinuous and Efficient
- Continuous and Inefficient
- Discontinuous and Inefficient

The patterns for all the selected F/OSS projects are identifiable in one or the other category and helpful in determining the quality of defect resolution process. The example graphs for each of the above categories are shown in Figure 1 to 4.

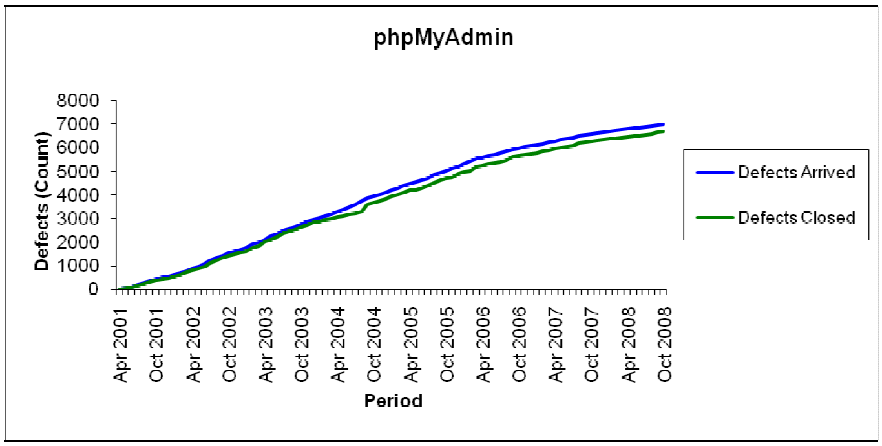

FIGURE 1: Continuous and Efficient Defect Resolution

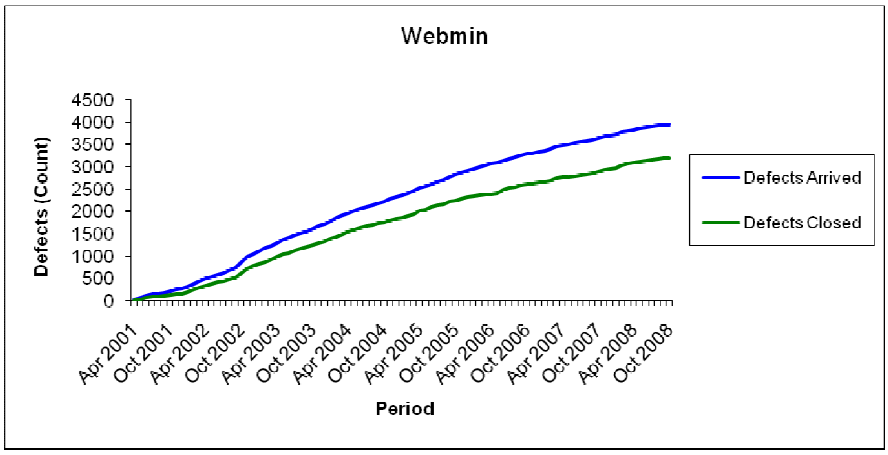

FIGURE 2: Continuous and Inefficient Defect Resolution

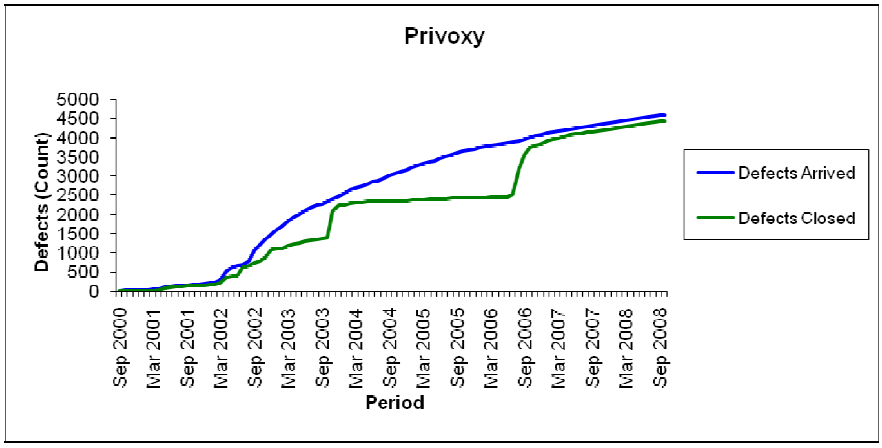

FIGURE 3: Discontinuous and Efficient Defect Resolution

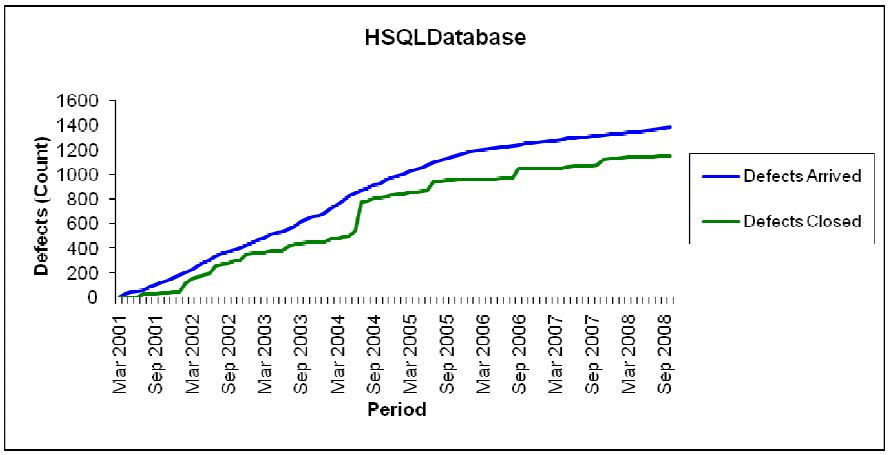

FIGURE 4: Discontinuous and Inefficient Defect Resolution

4.2. Pending Defects

Pending defects refers to all those defects which still need to be addressed. Ideally pending defects should decrease with the passage of time or at least it should remain constant. Large number of pending defects may discourage participating users from providing further feedback and many opportunities of improvement in the software may be lost. Figure 5 shows that number of monthly pending defects for all the 20 projects taken together keeps on increasing. To confirm the same statistically, a paired two-sided *t*-test is applied between number of pending defects in the beginning and at the end of the observation period for each of the 20 projects. It is clearly seen that there is significant difference ($t(19)=3.93634888$, $p<0.05$; *t* critical =2.09302405).

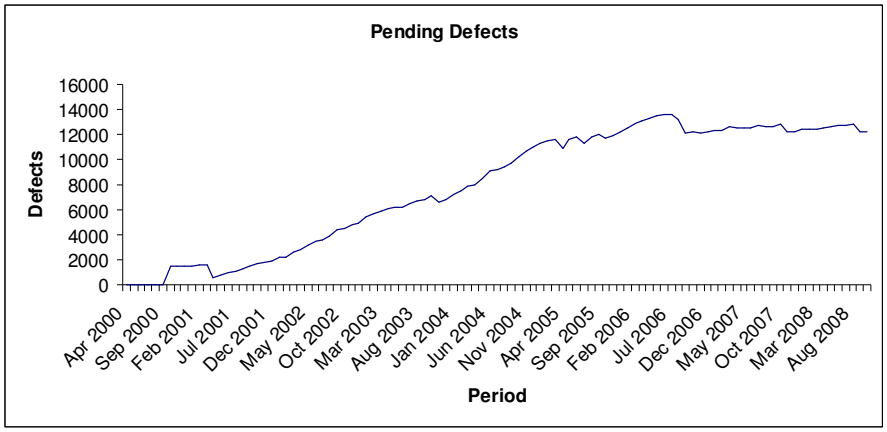

FIGURE 5: Aggregate Pending Defects for 20 F/OSS projects Together

The closer examination of pending defects over the period January 2006- November 2007 (Figure 6) shows that there are usually gradual increases and steep decreases in the number of pending defects. This suggests that defects slowly accumulate over the period and are removed in bursty manner. To test the hypothesis statistically, changes in pending defects from one month to the next month are recorded in form of upward change (for an increase) and downward change (for a decrease) frequencies for each of the 20 projects. Paired two-sided *t*-test shows that the difference between upward and downward changes in the number of pending defects is significant ($t(19)=11.9702$; $p<0.05$; *t* critical = 1.7291). There are overall 2.91 times more upward changes than downward changes. On an average basis, whenever there is an increase in pending defects, the upward change is 16.63 defects per month. On the other hand, if pending defects decrease, the downward change is 30.36 defects per month on an average. The reason for bursty nature of defect resolution is further discussed in subsection 4.5.

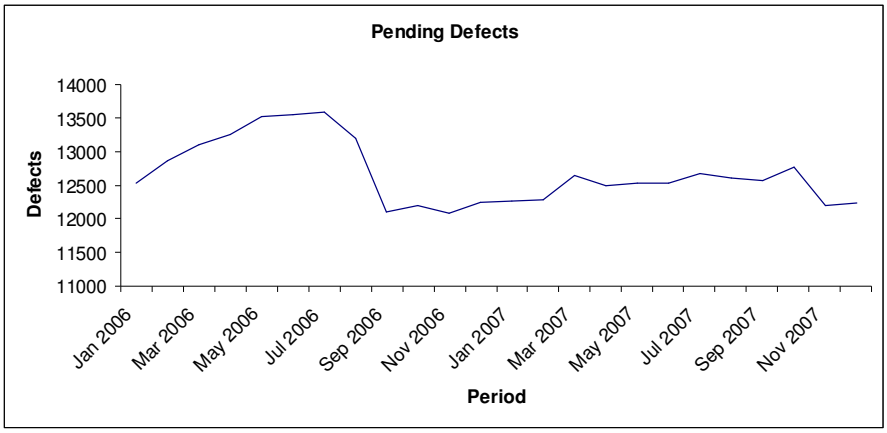

FIGURE 6: Gradual Increases and Steep Decreases in Pending Defects

4.3. Defect Removal Rate

Defect removal rate refers the proportion of defects resolved out of defects submitted for a particular period. The ever increasing number of pending defects indicates that the defect removal rate is decreasing. The size of core team has remained roughly same among the selected F/OSS projects. The hypothesis is that certain percentage of defects does not get resolved over the period as defect reports are submitted, thus number of pending defects accumulate.

In order to investigate this hypothesis statistically, a period of five years from 2003 to 2007 is considered. For each selected project, all the defects reports submitted during a particular year have been considered and then the status of each defect report exactly after 1 year of defect submission is observed whether the defect is resolved/closed or not [12]. The application of ANOVA reveals that the period in which a defect is submitted has significant influence on the defect removal rate ($F(4,94)=6.058928$; $p<0.05$; F critical=2.468533). The defects that have been reported during the year 2003, 81% of them have been resolved after 1 year (Table 2). The defect removal rate reduces to 71% in year 2005 and further to 65% in year 2007. This clearly shows that the defect removal rate is declining which results in ever increasing number of pending defects.

Period	Average Removal Rate	Standard Deviation
Year 2003	0.81	0.11
Year 2004	0.74	0.20
Year 2005	0.71	0.15
Year 2006	0.66	0.23
Year 2007	0.65	0.26

TABLE 2: Defect Removal Rate Over Five Years

4.4. Backlog Management

Backlog management refers to the capability of F/OSS developers to handle the pending defects, measured using Backlog Management Index (BMI). BMI is a ratio of number of defects closed to number of defects arrived during the period.

$$BMI = \frac{Number\ of\ defects\ closed\ during\ the\ period}{Number\ of\ defects\ arrived\ during\ the\ period} \times 100$$

If BMI is larger than 100, it means that the backlog is reduced as defects are being closed at the same or higher rate at which the defects are arriving. If BMI is less than 100, the backlog is increased. Of course, the goal is always to strive for a BMI larger than 100. With enough data points, the technique of control charting can help to calculate the overall backlog management capability of the software process [13]. A control chart is a graph or chart with limit lines, called control lines. In fact BMI chart is a pseudo-control chart because BMI data are auto correlated and assumption of independence for

control charts is violated. As the BMI values are in wide range, c control chart is more suitable [13]. In this case, three kinds of control lines are calculated as follows:

- Central Line (CL) equal to Mean BMI
- Lower Control Limit $LCL = (CL - 3 \times \sqrt{\text{CL}})$
- Upper Control Limit $UCL = (CL + 3 \times \sqrt{\text{CL}})$

If a process is mature and under statistical process control, all values should lie within the LCL and UCL. If any value falls out of the control limits, the process is said to be out of statistical process control. Figure 7 shows a project having very good backlog management. Most of the times the BMI curves are able to maintain themselves above the LCL. In case of Figure 8, the project was not having good backlog management initially but later on it improved. Figure 9 shows poor backlog management throughout the period.

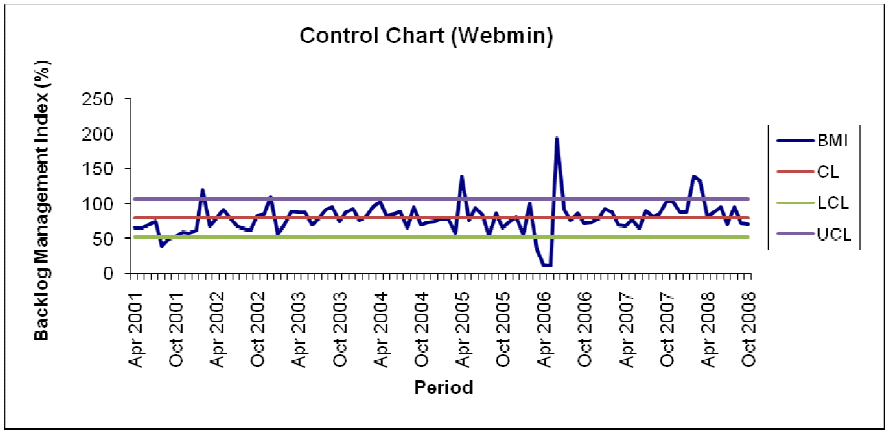

FIGURE 7: Backlog Management of Defects (Good)

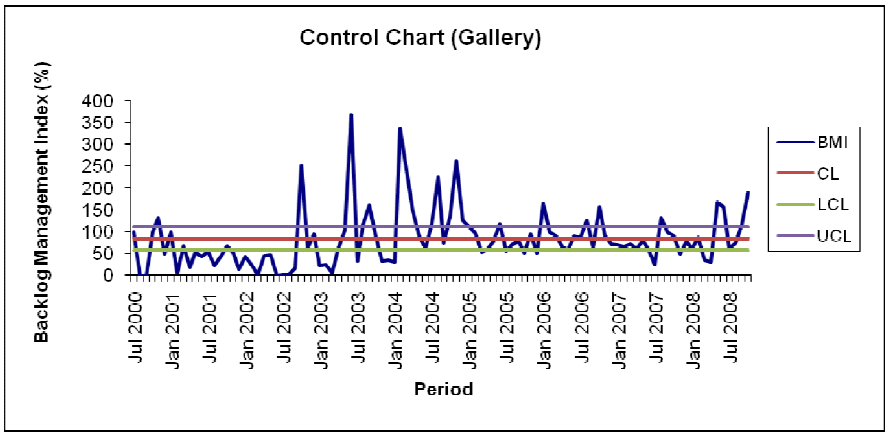

FIGURE 8: Backlog Management of Defects (Improved Later)

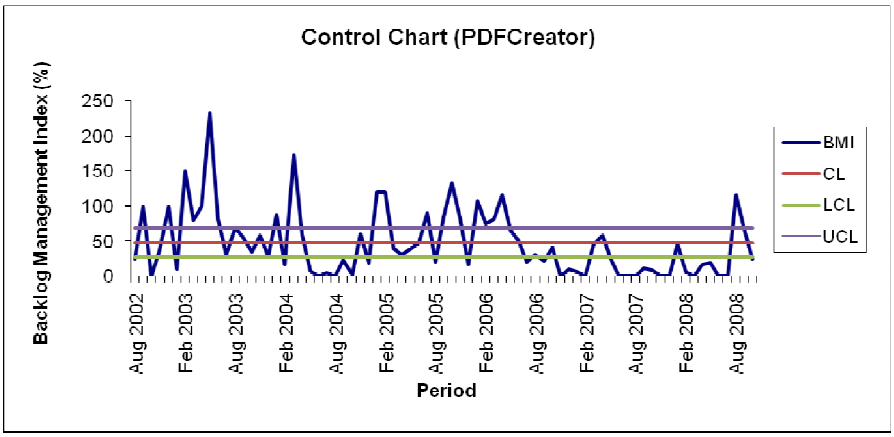

FIGURE 9: Backlog Management of Defects (Poor)

4.5. Software Release and Backlog Management

In the subsection 4.4, it is observed that BMI curves for most of the F/OSS projects are very fluctuating in spite of the fact that BMI values remain greater than 100 or lesser. To find out the reasons for such behavior, a detailed analysis of release data with BMI curves was carried out. Detailed inspection of release data revealed that the F/OSS projects are releasing their minor/major versions very frequently confirming the premise "*Release Early, Release Often*" [1]. In the Figure 10 and 11, efforts are made to trace back the shapes of BMI curves with release history of the projects. The dotted red colored vertical lines are drawn corresponding to major/minor releases in each of the following graphs.

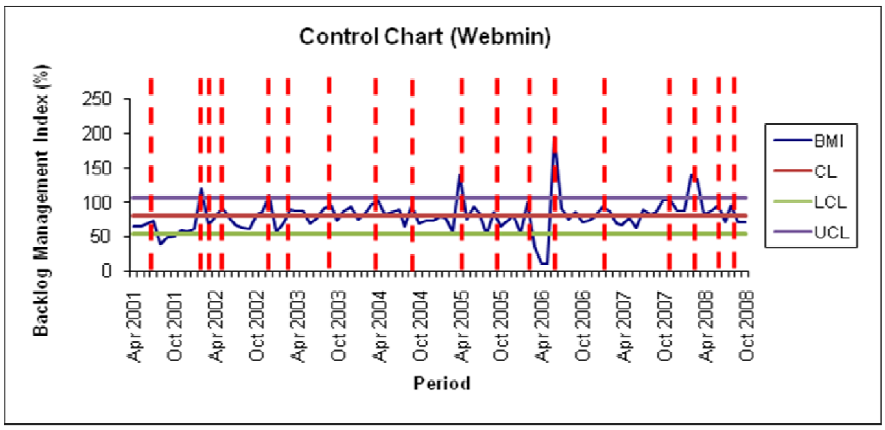

FIGURE 10: Software Release and Backlog Management of Defects

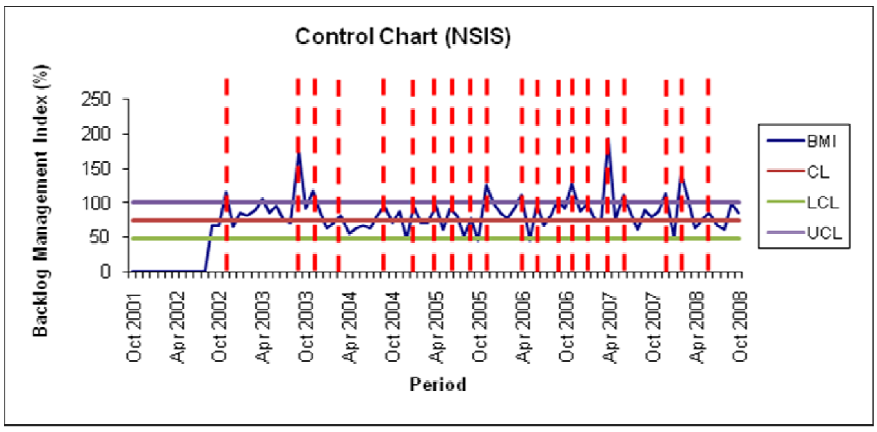

FIGURE 11: Software Release and Backlog Management of Defects

It is found that more than 90% of spikes in BMI curves are matching with the version releases. This phenomenon refers that generally F/OSS developer community do not resolve the defects on regular basis, instead put additional efforts to resolve defects near to each release.

4.6. Defect Resolution Age

Defect Resolution Age (DRA) refers to the number of days elapsed since a defect arrived till the time defect is resolved/closed. The average defect resolution age should be short as well as quite consistent to have efficiency in defect resolution. The monthly average of defect resolution age (MADRA) is computed using the following formula:

$$DRA(d_i) = \text{Defect Closing Date}(d_i) - \text{Defect Opening Date}(d_i)$$

$$MADRA = \frac{\sum_{i=1}^{N} DRA(di)}{N}$$ Where d_i refers to a defect closed

The graphs are drawn to show curves for average defect resolution age over the period for the F/OSS projects. Corresponding linear trend lines are also plotted. The projects should have preferably decreasing or at least constant trend of average defect resolution age to bring efficiency in defect resolution. For the F/OSS projects under study, it is observed that none of the projects has decreasing trend, very few projects are having near to constant trend lines (Figure 12) and most of the projects are showing upward trends in average defect resolution age over the period (Figure 13).

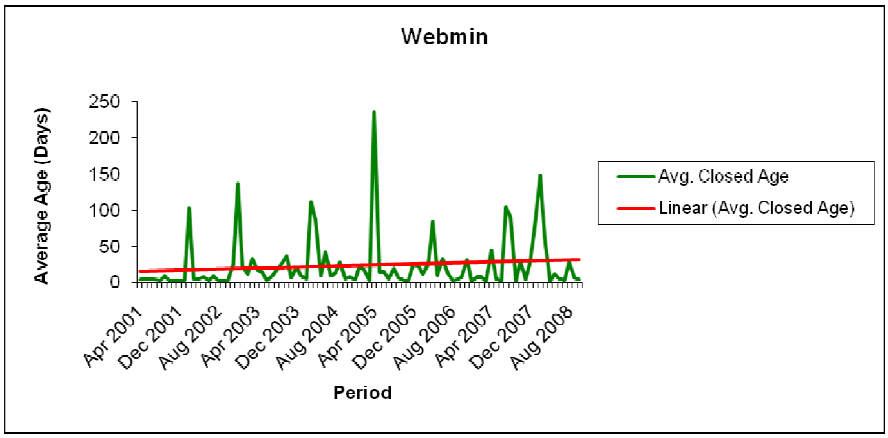

FIGURE 12: Defect Resolution Age (Near to Constant Trend)

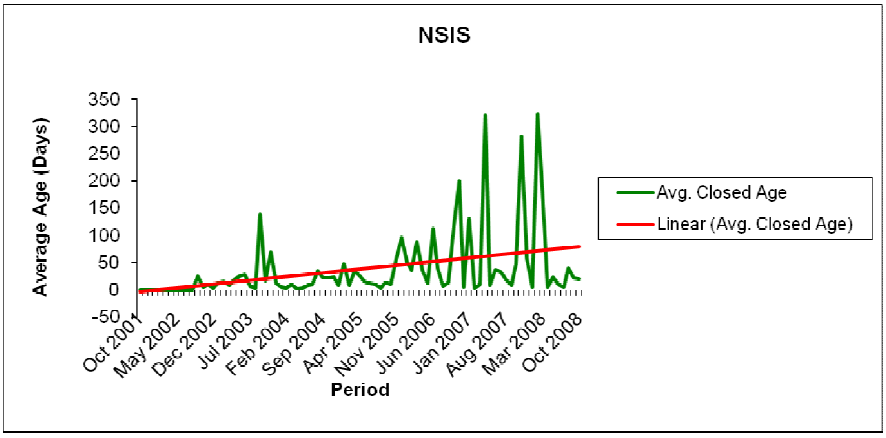

FIGURE 13: Defect Resolution Age (Increasing Trend)

Project	Period	Average Resolution Age (Days)	Standard Deviation	ANOVA Statistics
Webmin	Jan.1, 2002 to Dec. 31, 2003	21.56	32.46	$F(2,69)= 0.220411$; $p<0.05$; F critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	28.90	51.62	
	Jan.1, 2006 to Dec. 31, 2007	23.89	29.62	
NSIS	Jan.1, 2002 to Dec. 31, 2003	16.48	30.25	$F(2,69)=7.176098$; $p<0.05$; F critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	19.98	21.48	
	Jan.1, 2006 to Dec. 31, 2007	69.61	86.51	

TABLE 3: One Way ANOVA Statistics on Defect Resolution Age

To confirm the observation about trends in Defect Resolution Age, a standard analysis of variance (ANOVA) is carried out on monthly average resolution age over the period for all the selected F/OSS projects. Statistics about two projects are shown in Table 3. It is clearly seen that there is no significant difference in the average resolution over the period in case of Webmin, while it differs significantly for NSIS.

To analyze the overall defect resolution age for all the selected projects together during the investigation period, average resolution age for each of the 20 projects for various years is taken into consideration and standard analysis of variance (ANOVA) is applied which shows that there is significant change in defect resolution age over the period (F (4,94) =4.29461975;$p<0.05$;F critical =2.468533). The Table 4 also shows a continuous increasing trend in average defect resolution age (days) for various years for all the 20 projects taken together.

Period	Average Defect Resolution Age (Days)	Standard Deviation
2004	61.77	58.34
2005	76.35	41.62
2006	98.73	71.95
2007	113.07	89.99
2008	149.53	104.62

TABLE 4: Average Defect Resolution Age for 20 F/OSS projects Together

Figure 14 is a scatter plot for one of the F/OSS projects where each point represents resolution age for each defect closed. While Figure 15 shows the number of defects resolved with same resolution age value. The quality of the defect resolution process can be quantified by considering two statistical indices of the resolution age distribution i.e. skewness and kurtosis [51]. Skewness measures the asymmetry of the distribution and high values indicate that there are certain defects which have resolution age much higher than the average one. While Kurtosis measures the peaked ness of the distribution and high values mean that the variance of the resolution age is caused by very few defects with extremely long closing time (Table 5).

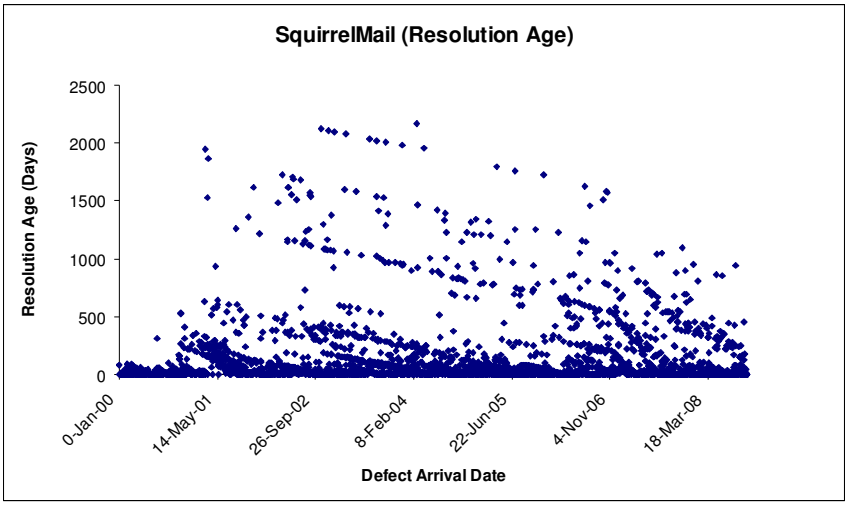

FIGURE 14: Scatter Plot of Resolution Age

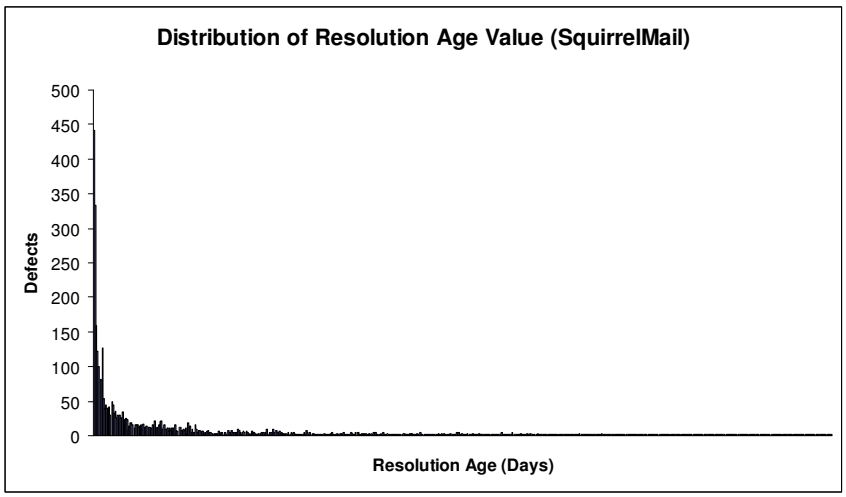

FIGURE 15: Distribution of Resolution Age

	SquirrelMail	NSIS	Webmin
Mean	6.20	7.02	13.09
Standard Deviation	25.29	37.83	100.33
Kurtosis	188.75	131.18	127.30
Skewness	12.58	10.97	11.04
Sum of Resolved Defects	3880	1362	3194

TABLE 5: Descriptive Statistics on Distribution of Resolution Age

It is clearly indicated that in most of the selected F/OSS projects, larger number of defects are resolved in shorter period while smaller number of defects are resolved in longer period which leads to an increase in overall mean resolution age.

4.7. Fix/Non-Fix Defect Resolution

It is observed that there is generally an increasing trend in defect resolution age and some of the defects are even resolved after 365 days. Many defects are resolved by making change/fix in the

source code whereas others may be resolved with non-fix status such as Invalid, Won't fix, Out of date, Duplicate, Works for me, Rejected etc.

Hence further analysis is carried out by comparing the resolution age in fix and non-fix categories. Figure 16 and 17 show graphs for two of the selected projects where comparison is made between fix and non-fix resolutions by distributing the resolved defects on the basis of defect resolution age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days). It is found that even the defects with non-fix resolution are closed in higher ranges of resolution age i.e. 91 to 365 days or More than 365 days. It is also observed that the proportion of non-fix resolved defects remain more or less same across all the resolution age categories.

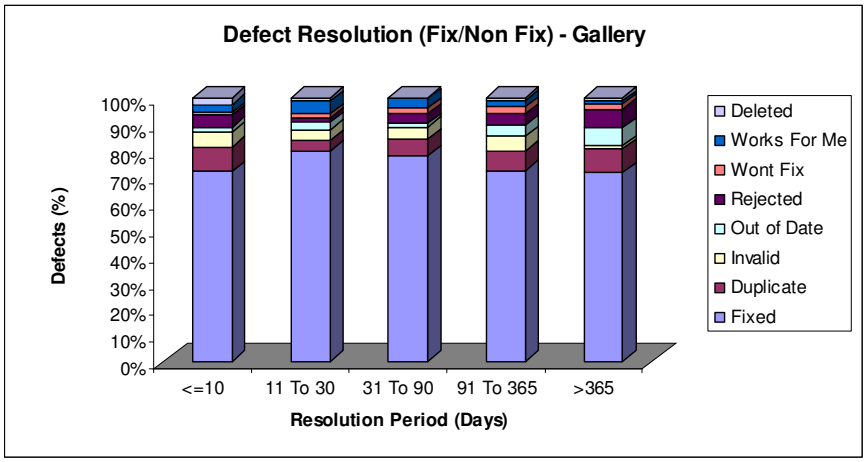

FIGURE 16: Defect Resolution Fix/Non-Fix (Gallery)

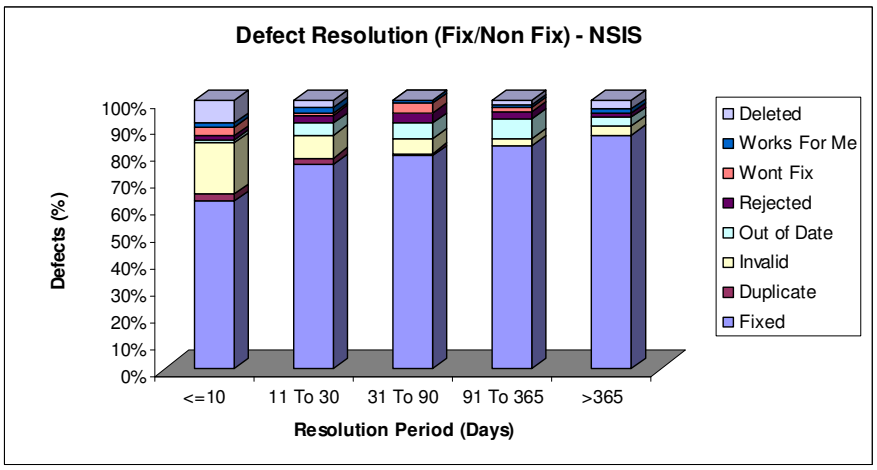

FIGURE 17: Defect Resolution Fix/Non-Fix (NSIS)

An unpaired two-sided *t*-test is conducted between defects with fix and non-fix resolution using their monthly average resolution age over all the months. The *t*-values in the last column of Table 6 for various F/OSS projects are below the critical values which clearly show that there is no significant difference in the resolution age of fix and non-fix resolved defects. An unpaired two-sided *t*-test is also applied to overall average age of defects with fix and non-fix resolution for all the 20 F/OSS projects. The test statistics ($t(38)=0.984940769$; $p<0.05$; t Critical=1.685954461) shows that there is no difference in efficiency for defects with fix and non-fix resolution as a whole also.

Project	Resolution Type	Average Resolution Age(Days)	Standard Deviation	*t* value*
Squirrelmail	Fix	132.58	165.75	0.820547
	Non-Fix	106.78	260.96	
Gallery	Fix	104.66	110.02	0.65281
	Non-Fix	117.43	167.31	
Webmin	Fix	22.47	37.78	0.21268
	Non-Fix	24.05	75.40	
NSIS	Fix	43.12	85.56	2.123759
	Non-Fix	21.41	47.08	
Netwide Assembler	Fix	132.13	308.57	0.645107
	Non-Fix	101.68	278.20	
aMSN	Fix	62.91	69.01	0.572375
	Non-Fix	55.69	81.39	

*p<0.05

TABLE 6: *t*-test statistics on Defect Resolution (Fix/Non-Fix)

4.8. Defect Pending Age

Defect Pending Age (DPA) refers to the number of days elapsed since a defect arrived and still remained pending at the end of the month. For all the selected F/OSS projects, monthly average of defect pending age (MADPA) is computed using the following formula:

$$DPA(d_i) = \text{Current Date} - \text{Defect Opening Date}(d_i)$$

$$MADPA = \frac{\sum_{i=1}^{N} DPA(di)}{N}$$

Where d_i refers to a pending defect

The graphs are plotted to show the curves for monthly averages of defect pending age for all the projects. It is observed that all the projects are showing increasing trend of monthly average defect pending age. Further detailed analysis of defects pending age is carried out by distributing the pending defects according to their pending age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days). Figure 18 and 19 also show curves for the overall monthly average pending age of all the pending defects as well as monthly average pending age for defects falling in each of the categories. By observing the pattern of defect pending age over the period, it is found that in almost all the projects the average pending age is increasing. But this increase in defect pending age trend is attributed mainly by those defects whose average pending age is 90 days or more. While in other lower age categories, trend remains either constant or slightly downward/upward.

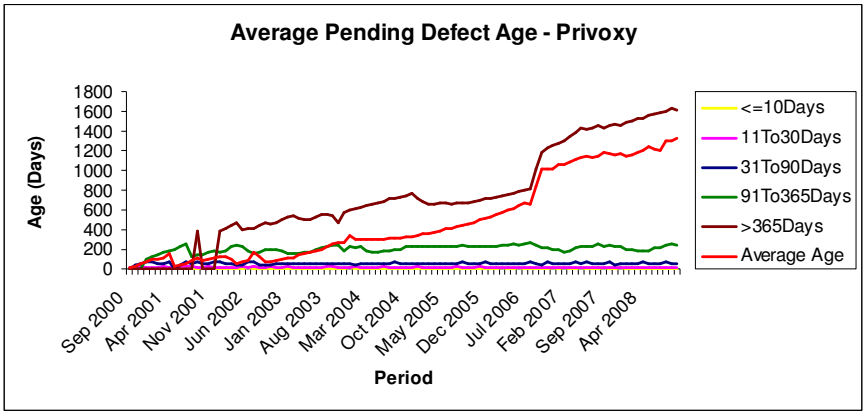

FIGURE 18: Defect Pending Age - Pending Age Wise (Privoxy)

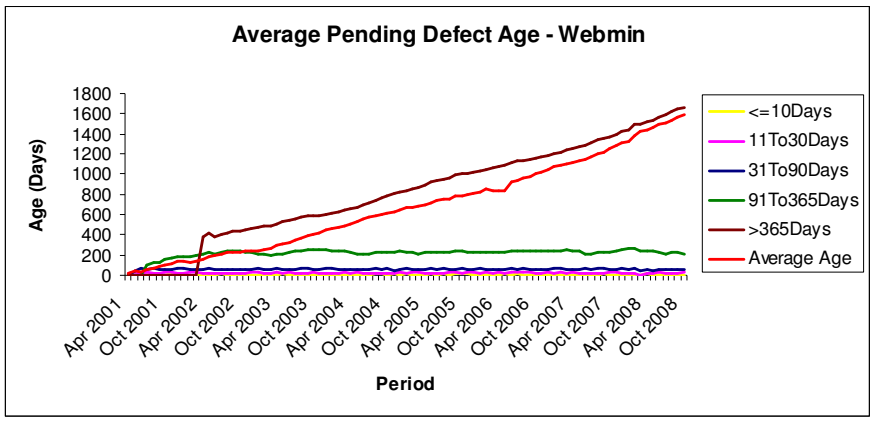

FIGURE 19: Defect Pending Age - Pending Age Wise (Webmin)

To observe the difference in pending age over the period for each of the 20 projects, ANOVA is applied. The statistics for two projects are highlighted in Table 7. Since the test statistic for both the projects is larger than the critical value, it is concluded that there is a (statistically) significant difference in average pending age over the periods. To analyze the overall defect pending age for all the selected projects together during the investigation period, average pending age for each of the 20 projects for various years is taken into consideration and standard analysis of variance (ANOVA) is applied which shows that there is significant change in defect pending age over the period ($F(4,95)=15.2694$; $p<0.05$; F critical=2.467494). The Table 8 also shows a continuous increasing trend in average defect pending age (days) for various years for all the 20 projects taken together.

Project	Period	Average Pending Age	Standard Deviation	ANOVA Results
Webmin	Jan.1, 2002 to Dec. 31, 2003	264.58	94.25	$F(2,69)=252.4181$; $p<0.05$; F critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	647.36	108.64	
	Jan.1, 2006 to Dec. 31, 2007	1047.70	151.84	
Privoxy	Jan.1, 2002 to Dec. 31, 2003	151.14	74.96	$F(2,69)=163.3788$; $p<0.05$; F critical= 3.129644
	Jan.1, 2004 to Dec. 31, 2005	369.15	70.36	
	Jan.1, 2006 to Dec. 31, 2007	927.15	244.99	

TABLE 7: One Way ANOVA Statistics on Defect Pending Age

Period	Average Defect Pending Age (Days)	Standard Deviation
2004	286.51	174.13
2005	421.28	227.67
2006	593.92	288.80
2007	802.74	355.54
2008	897.11	364.85

TABLE 8: Average Defect Pending Age for 20 F/OSS Projects Together

4.9. Defect Resolution (Defect Type Wise)

An F/OSS user can submit defects in the form of bug reports, feature requests, patches or miscellaneous (translation, support requests, plug-ins, package requests or any other project specific category). As it is observed that there is generally an increasing trend in defect resolution age, hence further analysis is carried out to observe the resolution age of each of the defect type. Figure 20 and

21 show graphs for two of the selected projects where comparison is made between various defect types by distributing the resolved defects on the basis of defect resolution age (Less than 10 days, 11 to 30 days, 31 to 90 days, 91 to 365 days and More than 365 days).

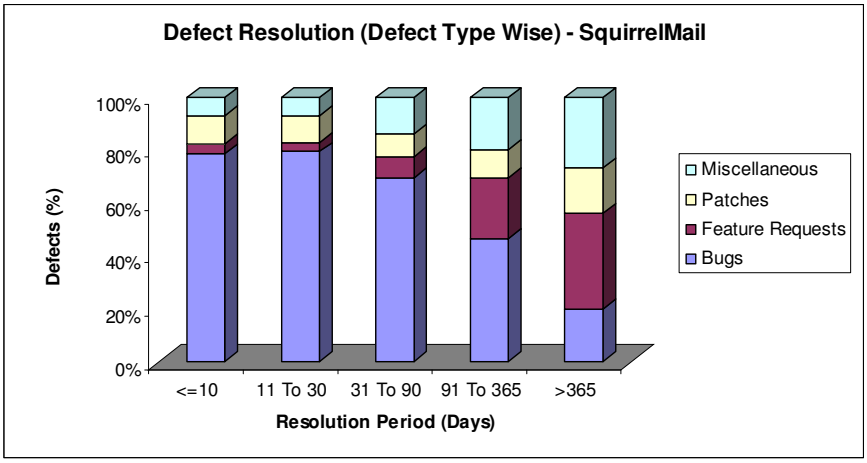

FIGURE 20: Defect Resolution - Defect Type Wise (SquirrelMail)

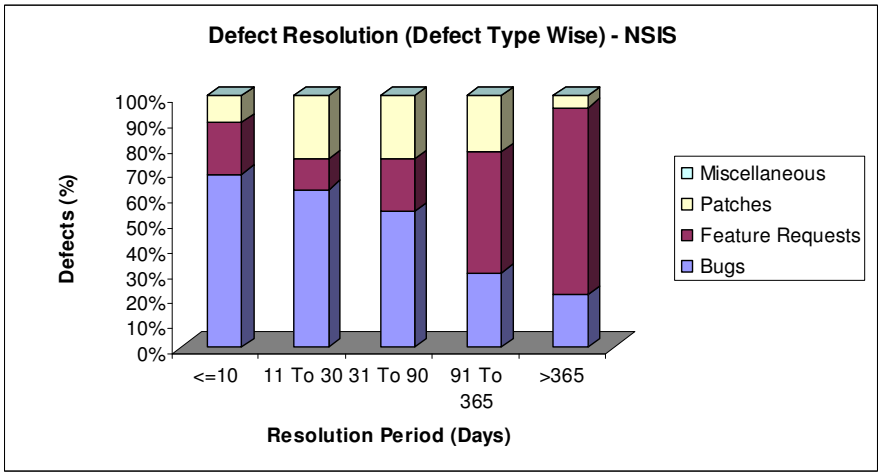

FIGURE 21: Defect Resolution - Defect Type Wise (NSIS)

It is found that all the defect types are dispersed among all the resolution age categories. It is also observed that proportion of bugs decrease with increasing resolution age while others (Feature Requests, Patches, Miscellaneous) increase with increasing resolution age. Further analysis of monthly average pending age is carried out in each of the defect type over the period (Figure 22 and 23). It is observed that each defect type is showing increasing trend in all the selected projects.

To analyze the defect pending age for each defect type taking all the selected projects together, average pending age in each defect type for each of the 20 projects for various years is taken into consideration and two way ANOVA is applied. The null hypothesis is that the differences between the defect types are consistent for various years. A significant year effect ($F(4)=23.36133; p<0.05; F$ critical=2.395431) implies that there is a difference in the effect of different years on the defect pending age regardless of the type of defect. A significant defect type effect ($F(3)=14.83437; p<0.05; F$ critical=2.628397) implies that there is a difference in the effect of different defect types on the defect pending age regardless of the level of year. While the interaction of year and Defect type ($F(12)=0.748815; p>0.05$; F critical=1.777693) implies that differences between the defect type are consistent for various years.

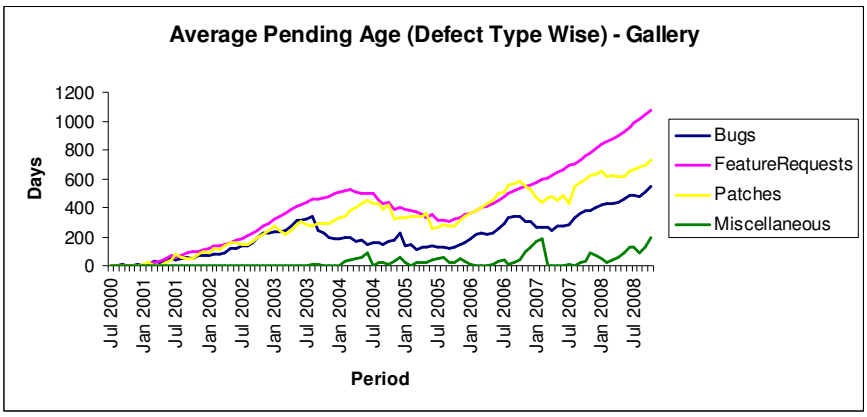

FIGURE 22: Average Pending Age - Defect Type Wise (Gallery)

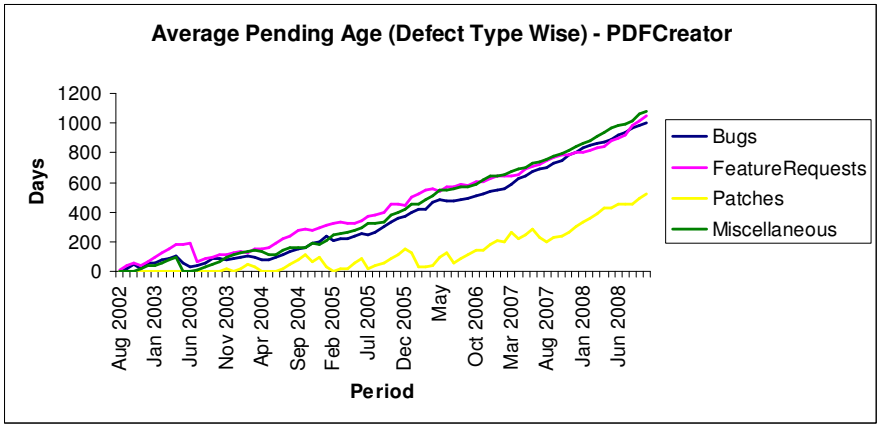

FIGURE 23: Average Pending Age - Defect Type Wise (PDFCreator)

5. PROBLEMS IN DEFECT MANAGEMENT

During the current study, various problems that have been identified in Defect Management are discussed as follows. Also an attempt is made to address these problems.

- It is observed that many F/OSS projects do not carry out defect resolution consistently and efficiently. The defect resolution is not able to keep pace with defect arrival thus accumulating pending defects. It is also found that backlog of pending defects accumulate gradually while their resolutions are carried out in bursty manner near the forthcoming releases. All these factors cause an increasing trend of overall resolution age as well as pending age. The detailed analysis shows that most of the defects are closed in reasonable time period while few defects take quite longer resolution time and aggravate the overall scenario. F/OSS development team should periodically review such long pending defects and prioritize them for resolution.

- It is also observed that there is no significant difference in resolution age of defects resolved with code fix or without any code fix (such as Duplicate, Out of Date, Won't Fix, Works for Me, Invalid etc.). It is not justified that a defect is closed after 100 days or longer with the status information as Duplicate, Out of Date, Won't Fix, Works for Me etc. Such behavior may cause loss of interest among participating users for further involvement. A process need to defined so that as soon as a defect is reported, members of development team should review it and if defect does not require any code change, it should be closed immediately with appropriate resolution status. By reducing Non-fix defect resolution age, overall resolution efficiency can be improved.

- It is found that all the defect types (Bugs, Feature Requests, Patches, Miscellaneous) are dispersed among all the resolution age categories although proportion of bugs decrease with

increasing resolution age while others (Feature Requests, Patches, Miscellaneous) increase with increasing resolution age. It is also observed that each defect type is showing an overall increasing trend of pending age in all the selected projects. Ideally bugs should be resolved within shorter period depending upon the criticality of the bugs; while feature requests, patch submissions may be delayed till forthcoming releases/patches. Under miscellaneous category, the resolution should be carried out based upon the type of request. Due to volunteer nature of F/OSS participants, nobody can ensure that they will have enough time to respond to a defect quickly. So spreading the load across several development team members may lead to more reliability and to a shorter defect removal time.

- It has been found that in few F/OSS projects, the defect resolution status remains default (None) rather than being updated with relevant resolution status (Fixed, Duplicate, Out of Date, Won't Fix, Works for Me, Invalid etc.) even after the defect is closed. Although such defects are closed but the F/OSS users are not able to know exactly what actions have been taken on their reported defects. Defect Management System should have the functionality which enforces the development team to update the resolution status correctly while closing the defect.

- It has been found that in most of F/OSS projects, the F/OSS development team is not defining the priority of each defect being reported, although Defect Management System has the functionality to assign priority to reported defects. When a defect is reported, the priority is always set to default value 5 i.e. Normal (1-Highest, 9-Lowest) which is generally not updated by Development Team. Due to lack of prioritization of reported defects, the resolution of many critical defects may be delayed. F/OSS project development team should clearly define the criterion to identify the priority of each reported defect and make some of the team members responsible to assign the priority as per the criteria.

6. PROPOSED PROCESS FOR DEFECT MANAGEMENT

Based on the suggestions mentioned in the previous section, a process is proposed as shown in Figure 24, which can help to improve the effectiveness as well as efficiency of Defect Management.

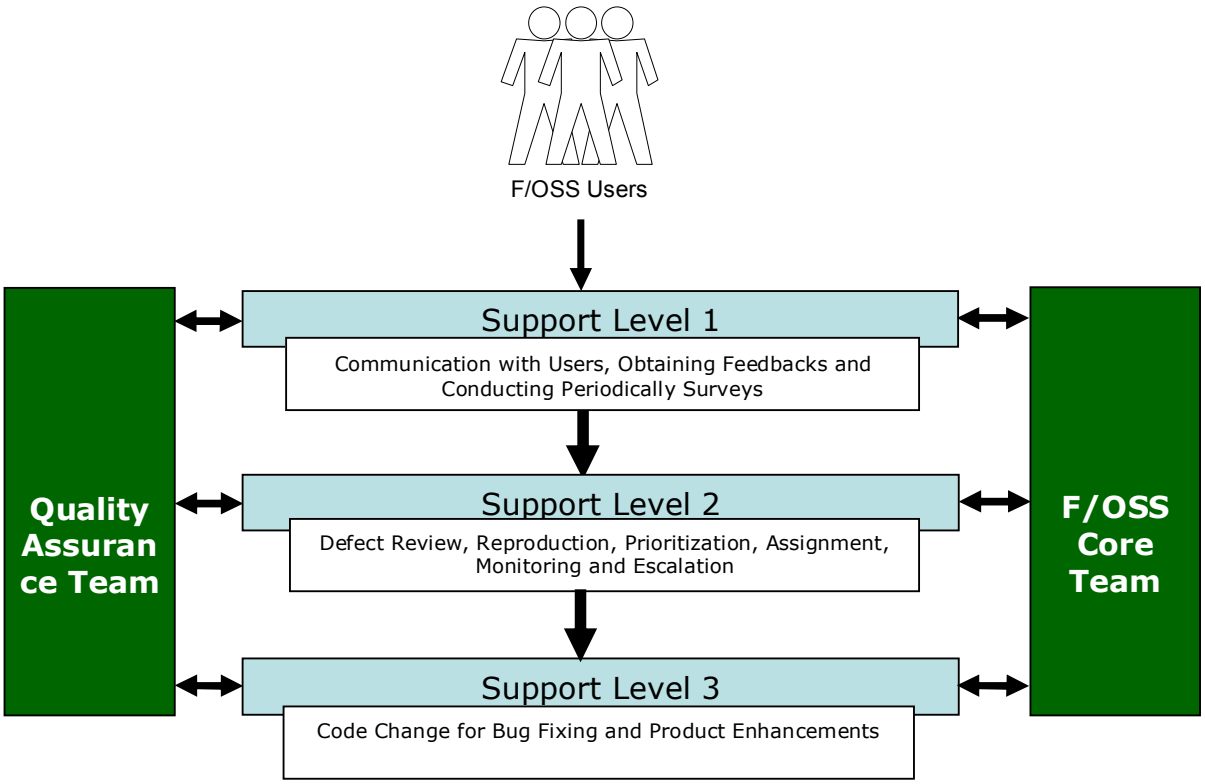

FIGURE 24: Proposed Process Diagram

It is proposed that support and maintenance activities should be distributed among various levels in order to improve the effectiveness and efficiency in Defect Management. The roles and responsibilities at these levels can be distributed as follows:

- **Support Level 1:** This level may comprise volunteer F/OSS users who may not have sufficient technical skill set to help development team but are ready to participate in F/OSS development process. This team should have responsibility to communicate with F/OSS users, obtain their feedback and conduct surveys periodically to know the level of satisfaction regarding usage of F/OSS Product and any issues that need to be addressed by development team.

- **Support Level 2:** This level may comprise volunteer F/OSS users cum developers who have sufficient technical skill set to help development team. They should be assigned the responsibility to review all the reported defects within stipulated period, make efforts to reproduce, collect additional information if required, set priority based upon prior defined criterion and assign them to team at level 3. They should keep on monitoring that no defect should remain pending for a long period without any appropriate reason. If there is any long pending defect without any justified reason, it should be escalated to Core Team for corrective measures. The members at this level should also resolve the defects which does not require any code change and set their appropriate status in the Defect Management System. They should also build knowledgebase comprising frequently occurring defects related to installation, configuration etc. and enabling F/OSS users to browse through easily.

- **Support Level 3:** This level may comprise volunteer F/OSS developers who have good technical skill set and knowledge of source code of F/OSS project. This team will have the responsibility to carry out necessary code changes to fix the defects as well to incorporate required feature enhancements. Whenever a defect is assigned, they should resolve the defect with in reasonable time frame. If some additional information is required about the defects, it should be obtained through level 2 team. Many times some of the defects can not be resolved due to constraints like software design, technology, resources, irreproducible etc. In all such cases, relevant information should be communicated to users timely.

- **Quality Assurance Team:** This team should comprise F/OSS volunteers preferably having some knowledge or experience in software quality assurance. They should have responsibility to monitor the activities carried out at all levels e.g. responsiveness towards users, defect resolution period, backlog of defects, code review etc. and should assure that quality is maintained at all the levels. They should generate and analyze the statistics periodically and should escalate serious concerns (if any) to core team.

- **F/OSS Core Team**: This team comprises the initiators and project leaders who have the overall responsibility. They should control the overall direction of project, take corrective measures for serious concerns and decide future strategy for forthcoming releases.

7. CONCLUSION

Defect Management Systems have been used to record and track defects for many years, but there is little analysis of the recorded defect data. Analyzing the defect data is of substantial value since it reveals how various variables connected to the defects change over time such as defect arrival rate, defect removal rate, defect resolution period, handling of pending defects etc. An analysis of more than 60,000 defect reports associated with 20 F/OSS projects reveals that many important insights can be gained through the analysis of defect data that has been recorded over the years. The quality of an F/OSS project can be improved a lot if defects are identified, reported and resolved in efficient manner. Generally an F/OSS project is developed by a small team of core developers which is surrounded by a community consisting of large number of globally distributed users. Not every F/OSS user has the technical skills to take part in code review or to carry out development. However, these users can contribute to the project by reporting bugs or by suggesting new features.

For effective Defect Management, the defect reports should be updated correctly and regularly. Also for efficient defect management, the defects should be resolved and closed at the earliest and consistently. During the analyses, it has been found that generally defect resolution is not performed consistently. This results in declining defect removal rate and an ever increasing average age of defect resolution. This problem needs to be addressed timely otherwise important user feedback is

not incorporated into the software and many opportunities of improving the software are lost. It is also observed that defects get accumulated gradually and then additional efforts are put to resolve them near the forthcoming software releases. An observation of the BMI reports also confirms that backlog is increasing gradually but decreasing steeply. It is also found that a few defects remain pending for fairly long period of time in the Defect Management System. They are neither resolved nor their status is updated, if resolved. Such ignored defects keep on accumulating and result in increasing trend in overall defect pending age. The inefficient defect resolution has serious effects in the long term if effective countermeasures are not found. Moreover, as defects become older, reproducing them becomes increasingly more complex because the software continuously changes. Finally, users will perceive that their feedback does not have any impact and will stop providing valuable input. This minimizes the benefits that F/OSS projects can draw from peer review and user involvement, which is an important characteristic of F/OSS projects. A layered process is proposed where roles and responsibilities are clearly defined and distributed among F/OSS participants. F/OSS projects may use the proposed process which can help to improve the effectiveness and efficiency in Defect Management and thus assure better quality of F/OSS Products.

ACKNOWLEDGMENTS

We are thankful to the University of Notre Dame for providing access to Sourceforge Research Data Archive (SRDA) for retrieving data on F/OSS projects.

REFERENCES

1. Eric S. Raymond, "The Cathedral and the Bazaar", First Monday, 3(3), 1998.

2. Walt Scacchi, “Software Development Practices in Open Software Development Communities: A Comparative Case Study”, Proceedings of 1st Workshop on Open Source Software Engineering, May 2001, Toronto, Ontario, Canada.

3. Audris Mockus, Roy Fielding and James D. Herbsleb, “Two Case Studies of Open Source Software Development: Apache and Mozilla” ACM Transactions on Software Engineering and Methodology, 11(3): 309–346.

4. Dawid Weiss, “A Large Crawl and Quantitative Analysis Of Open Source Projects Hosted On Sourceforge”, Research Report ra-001/05(2005), Institute of Computing Science, Pozna University of Technology, Poland.

5. A. G. Koru and J. Tian, “Defect Handling in Medium and Large Open Source Projects”, IEEE Software, 21(4):54-61, July 2004.

6. Daniel German and Audris Mockus, “ Automating the Measurement of Open Source Projects”, Proceedings of the 3rd Workshop on Open Source Software Engineering, International Conference on Software Engineering, May 2003, Portland, Oregon, USA.

7. Stefan Koch, “Effort Modeling and Programmer Participation in Open Source Software Projects ", Information Economics and Policy, 20 (4): 345-355, 2008.

8. Ionic Stamelos, Lefteris Angelis, Apostolos Oikonomou and Georgios L. Bleris, “Code Quality Analysis in Open Source Software Development”, Information Systems Journal, 12(1): 43:60, 2002.

9. “SourceForge”, http://sourceforge.net/

10. Yongqin Gao, Matthew Van Antwerp, Scott Christley and Greg Madey, "A Research Collaboratory for Open Source Software Research", Proceedings of 29th International Conference on Software Engineering + Workshops (ICSE-ICSE Workshops 2007), International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007), May 2007, Minneapolis, Minnesota, USA.

11. Chiara Francalanci and Francesco Merlo, “Empirical Analysis of the Bug Fixing Process in Open Source Projects”, Open Source Development, Communities and Quality, Springer Boston, 275 :187-196, 2008.

12. Martin Michlmayr and Anthony Senyard, “A Statistical Analysis of Defects in Debian and Strategies for Improving Quality in Free Software Projects”, The Economics of Open Source Software Development, Elsevier B.V., 2006, pp 131–148.

13. Stephen H. Kan, “Metrics and Models in Software Quality Engineering”, Second Edition, 2003, Pearson Education.

A Simplified Model for Evaluating Software Reliability at the Developmental Stage

Shelbi Joseph achayanshelbil@gmail.com
Division of Information Technology
School of Engineering
Cochin University of Science and Technology
Cochin, India

Shouri P.V pvshouri@gmail.com
Department of Mechanical Engineering
Model Engineering College
Cochin, India

Jagathy Raj V. P jagathy@cusat.ac.in
School of Management Studies,
Cochin University of Science and Technology
Cochin, India

Abstract

The use of open source software is becoming more and more predominant and it is important that the reliability of this software are evaluated. Even though a lot of researchers have tried to establish the failure pattern of different packages a deterministic model for evaluating reliability is not yet developed. The present work details a simplified model for evaluating the reliability of the open source software based on the available failure data. The methodology involves identifying a fixed number of packages at the start of the time and defining the failure rate based on the failure data for these preset number of packages. The defined function of the failure rate is used to arrive at the reliability model. The reliability values obtained using the developed model are also compared with the exact reliability values.

Key words: Bugs, Failure Density, Failure Rate, Open Source Software, Reliability

1. INTRODUCTION

Open Source Software (OSS) has attracted significant attention in recent years [1]. It is being accepted as a viable alternative to commercial software [2]. OSS in general refers to any software whose source code is freely available for distribution [3]. However the OSS development approach is still not fully understood [4]. Reliability estimation plays a vital role during the developmental phase of the open source software. In fact, once the package has stabilized (or developed) then chances of further failure are relatively low and package will be more or less reliable. However, during the developmental stage failures or bug arrival are more frequent and it is important that a model has to be developed to evaluate the reliability during this period. The bug arrivals usually peak at the code inspection phase and get rather stabilized in the system test phase [5]. Software reliability evaluation is an increasingly important aspect of software development process [6].

Reliability can be defined as the probability of failure free operation of a computer program in a specified environment for a specified period of time [4,5]. It is evident from the definition that there are four key elements associated with the reliability namely element of probability, function of the product, environmental conditions, and time.

Reliability is nothing but the probability of success. As success and failure are complementary, a measure of the failure is essential to arrive at the reliability. That is,

$$Reliability = Probability\ of\ success = 1 - Probability\ of\ failure \quad (1)$$

From equation (1), it is evident that the first step in reliability analysis is failure data analysis. This involves fixing up a time interval and noting down the failures at different time intervals. The number of packages at the start of the analysis is defined as the initial population and the survivors at any point of time is the difference of initial population and the failures that have occurred till this point. Failure rate associated with a time interval can be defined as the ratio of number of bugs reported during the

NOMENCLATURE

$f_d(t)$	failure density
N	initial population
R(t)	reliability
t	time
Z(t)	failure rate
λ	constant failure rate

given time interval to the average population associated with the time interval. Once the variation of failure rate with respect to time can be established an equation can be used to fit the variation which will be the failure model for reliability estimation. Typical reliability models include Jelinski-Moranda [6], Littlewood [7] , Goel-Okumoto [8], Nelson model [9], Mills model [10],Basin model [10],Halstead model [11] and Musa-Okumoto [4]. For software projects that have not been in operation long enough, the failure data collected may not be sufficient to provide a decent picture of software quality, which may lead to anomalous reliability estimates [12, 13] Weibull function is also used for reliability analysis and the function has been particularly valuable for situations for which the data samples are relatively small [14].

Concern about software reliability has been around for a long time [15,16] and as open source is a relatively novel software development approach differing significantly from proprietary software waterfall model, we do not yet have any mature or stable technique to assess open source software reliability [17].

It is clear from the above discussions that even though a variety of models are available for reliability prediction, a deterministic model is presently not available. Or in other words, none of these models quantifies reliability. The present work focuses on development of an algorithm and there by a simplified method of quantifying reliability of a software.

2. MODEL DEVELOPMENT AND ALGORITHM

An open source program typically consists of multiple modules [18]. Attributes of the reliability models have been usually defined with respect to time with four general ways to characterize [19, 20] reliability, time of failure, time interval between failures, cumulative number of faults upto a period of time and failure found in a time interval. The present methodology involves defining an equation for the pattern of failure based on the available bug arrival rate and developing a generalized model for the reliability of the software. The following are the assumptions involved in the analysis.

1. The software analyzed is an open source.
2. As the open source software is made up of a very large community the environmental changes are not considered.
3. The total number of packages at the beginning of the analysis is assumed to remain constant and is taken as the initial population.
4. The failures of various packages are assumed to be independent of each other.
5. The model is developed for evaluation of the software reliability at the developmental stage and the packages that fail during this period are not further considered. It is further assumed that by the end of

developmental stage the bug associated with the failed packages would be eliminated and will be stable further.

6. The reliability of the software is inversely proportional to the number of bugs reported at any point of time.
7. The beginning of the time period after which the bug arrival or failure rate remains constant marks the culmination of the developmental stage and the software will be stabilize.

Based on the above assumptions a 6- step algorithm is developed for the analysis as detailed below.

1. Indentify the total initial population. This corresponds to the total number of packages existing at the beginning of the time period. That is, at the start of analysis.
2. Define a time period and find out the bugs reported during this time interval. As the failure would have occurred anywhere between the time interval, the reported failures are indicated in between the time interval.
3. Calculate the cumulative failures and thereby the survivors and different points in time.
4. Estimate the failure rate associated with the time intervals by dividing the number of failures associated with the given unit time interval by average population associated with the time interval. Average population associated with a given time interval is the average of survivors at the beginning and end of the time period.
5. Plot the graphs defining the relation between failure rate and time and obtain the equation defining the relation between failure rate and time.
6. Obtain the expression for reliability of the software by substituting the equation of failure rate in equation(1) given as

$$R(t) = e^{-\int_0^t Z(t)dt} \quad (1)$$

3. RESULTS AND DISCUSSION

A total of 1880 packages were available at the start of the analysis as per the details available from the official website of Debian [21]. This is taken as the initial population. A time interval of 1 month is fixed and the bug arrival rate during this interval is noted. The reported errors at different time intervals are given in the Table 1. The observations are taken for 1 year after which the bug arrival is negligible indicating that the software has more or less stabilized.

Time	No. of Failures	Cumulative Failures	Survivors	Failure Rate
Feb-08		0	1880	
	25			0.013386881
Mar-08		25	1855	
	61			0.033433817
April-08		86	1794	
	340			0.209359606
May-08		426	1454	
	49			0.034277719
Jun-08		475	1405	
	55			0.039927405
Jul-08		530	1350	
	214			0.172164119
Aug-08		744	1136	
	136			0.127340824
Sept-08		880	1000	
	37			0.037697402
Oct-08		917	963	
	40			0.042417815
Nov-08		957	923	
	48			0.048929664
Dec-08		1005	875	
			Average	0.075893525

TABLE 1: Failure Data Analysis

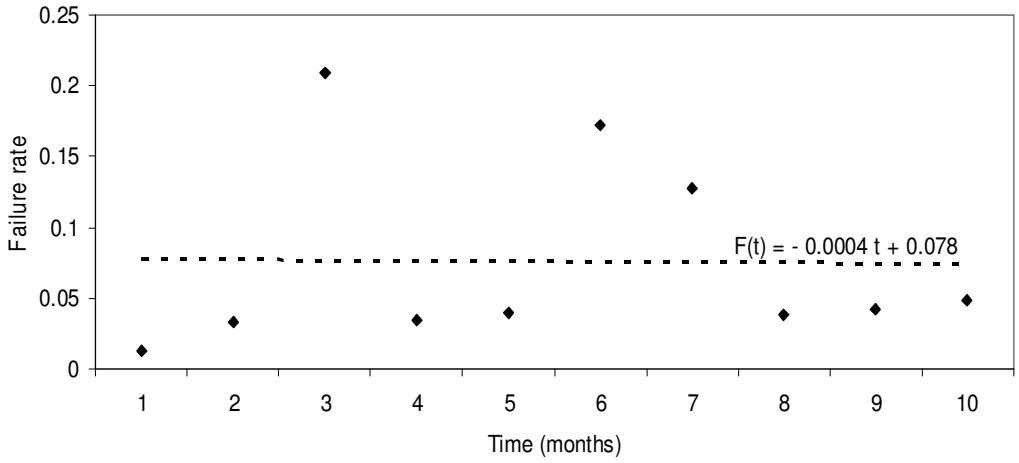

FIGURE 1. Variation of failure rate with time

The variation of failure rate with respect to time is shown in Fig. 1. It can be seen that after the 8^{th} month onwards the software has somewhat stabilized indicating the completion of developmental phase. The failure model corresponding to the failure rate can be expressed by the equation (2)

$$Z(t) = -0.0004t + 0.078 \tag{2}$$

The corresponding reliability can be expressed by the equation (3) as

$$R(t) = e^{-\int_0^t (-0.0004t + 0.078)\,dt}$$

$$\text{That is, } R(t) = e^{\frac{0.0004t^2}{2} - 0.078t} \tag{3}$$

Failure density associated with the time intervals is the ratio of number of failures associated with the given unit time interval to the initial population. Failure density can be related with Reliability and failure rate using the equation (4) as

$$f_d(t) = R(t) \times Z(t) \tag{4}$$

Therefore, based on the developed model failure density can be expressed as

$$f_d(t) = e^{\frac{0.0004t^2}{2} - 0.078t} \times (-0.0004t + 0.078) \tag{5}$$

The reliability of the software at different points in time is calculated using the equation (3). The actual values of reliability obtained by dividing the survivors at the given point in time by the initial population are also calculated. The Musa model assumes a constant value for the failure rate and by considering this as the average value of failure rates the reliability values are calculated using the equation

$$R(t) = e^{-\lambda t} \tag{6}$$

The reliability values calculated using the three different methods and the failure density values are shown in table 2.

Time	Failure Density	Reliability (Actual)	Reliability (Musa)	Reliability (Simplified Model))
Feb-08		1	1	1
	0.013297872			
Mar-08		0.98670	0.92691	0.92496
	0.032446809			
April-08		0.95426	0.85917	0.85556
	0.180851064			
May-08		0.77340	0.79638	0.79136
	0.02606383			
Jun-08		0.74734	0.73818	0.73198
	0.029255319			
Jul-08		0.71809	0.68423	0.67706
	0.113829787			
Aug-08		0.60427	0.63422	0.62625
	0.072340426			
Sept-08		0.53191	0.58787	0.57926
	0.019680851			
Oct-08		0.51223	0.54490	0.5358
	0.021276596			
Nov-08		0.49096	0.50508	0.49559
	0.025531915			
Dec - 08		0.46543	0.46816	0.45841

TABLE2. Reliability and failure density

Fig. 2 shows a comparison of reliability obtained using the developed simplified model and Musa model with the actual reliability values. It can be seen that the simplified model and the Musa model nearly provides the same results. Further, these two models very closely approximate the real situation. The variation of failure density with time is also shown in Fig. 3

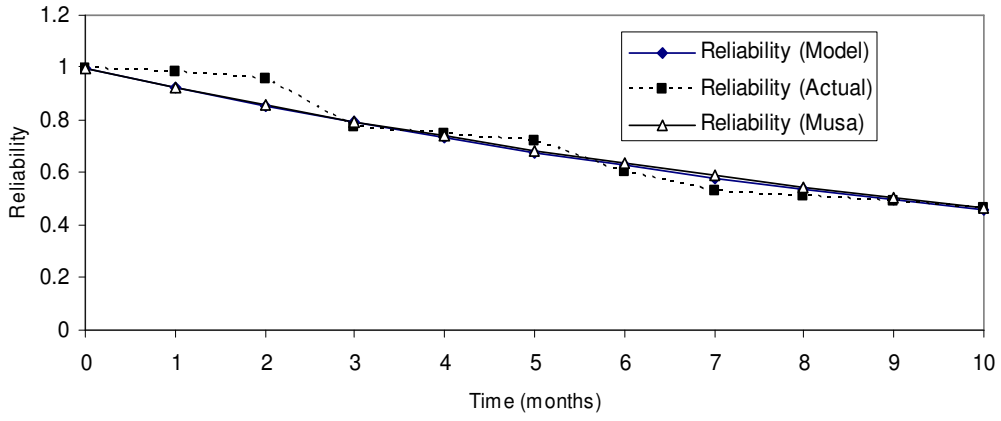

FIGURE2. Comparison of reliability obtained using different models

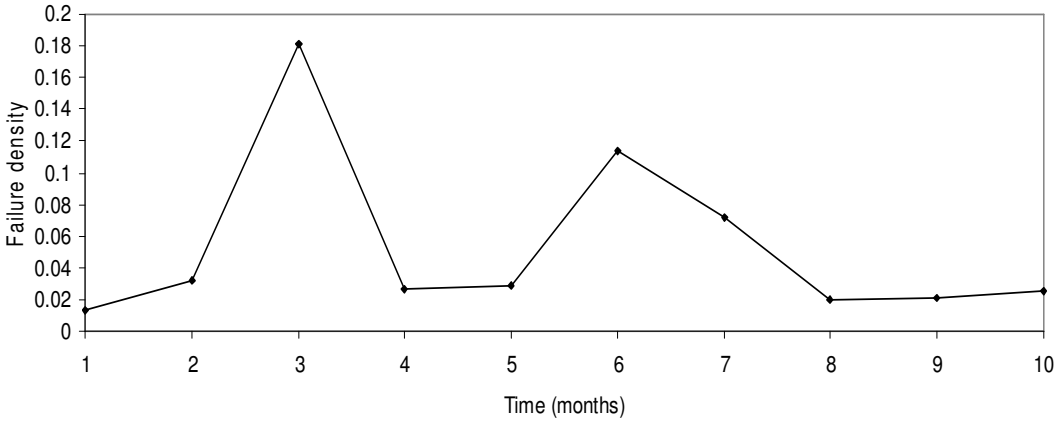

FIGURE 3. Variation of Failure density with time

Fig 4 compares the reliability value obtained using the model with the theoretical value. It can be seen that the percentage error is always within 10% of the actual value which is a reasonably good result for all engineering problems.

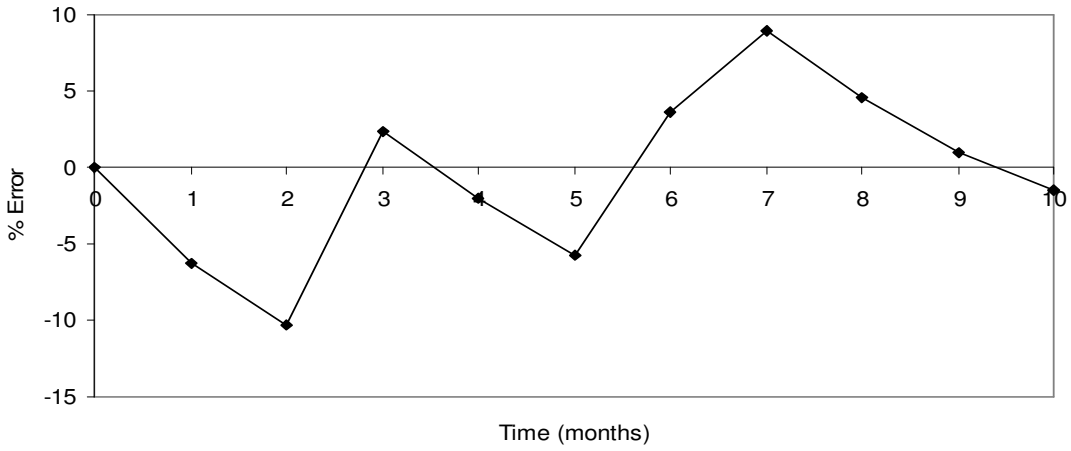

FIGURE 4. Error Analysis

4. CONCLUSION

A simplified model for evaluation of software reliability was presented. This is a relatively simplified and a totally new method of analysis of software reliability as the initial population is assumed to remain constant. The method provides fairly good results and the related errors are negligible. It is hoped that this model will prove to be a powerful tool for software reliability analysis.

5. REFERENCES

1. Ying ZHOU,Joseph Davis. Open Source Software reliability model: an empirical approach, ACM 2005

2. Sharifah Mashita Syed-Mohamad,Tom McBride. A comparison of the Reliability Growth of Open Source and In-House Software. 2008 IEEE 15th Asia-Pacific Software Engineering Conference

3. Cobra Rahmani,Harvey Siy, Azad Azadmanesh. An Experimental Analysis of Open Source Software Reliability. Department of Defense/Air Force Office of Scientific Research

4. Lars M.Karg, Michael Grottke, Arne Beckhausa . Conformance Quality and Failure Costs in the Software Industry: An Empirical Analysis of Open Source Software. 2009 IEEE

5. Kan H.S. Metrics and models in software quality engineering, 2nd edition, Addison-Wesley(2003)

6. S.P. Leblanc, P.A.Roman Reliability Estimation of Hierarchical Software Systems. 2002 Proceedings annual reliability and maintainability symposium

7. J D Musa and K. Okumoto. A logarithamic Poisson execution time model for software reliability measurement 7th international conference on Software Engineering(ICSE),1984, pp. 230-238

8. H. Pham, Software Reliability Spriner-Veriag , 2000

9. E.C. Nelson, A Stastistical Basis for Software Reliability Assessment, TRW-SS-73-03. 1973.

10. ShaoPing Wang, Software Engineering. BEIJING BUAA PRESS.

11. M.H.Halstead, Elements of Software Science, North Holand, 1977.

12. Z. Jelinski and P.B. Moranda, Software Reliability research, in Stastistical Computer Performance Evaluation, W. Freiberger, Ed. New York: Academic Press, 1972, pp.465-484

13. B. Littlewood and J.L. Verrall, A Bayesian reliability growth model for computer software, Applied Stastistics, Vol 22, 1973, pp 332-346

14. A. L. Goel and K. Okumoto, A time-dependent error-detection rate model for software reliability and other performance measure, IEEE Transactions on Reliability, vol. R-28, 1979, pp. 206-211.

15. Adalberto Nobiato Crespo, Alberto Pasquini ,"Applying Code Coverage Approach to an Infinite Failure Software Reliabilit Model" , 2009 XXIII Brazilian Symposium on Software Engineering, 2009 IEEE.

16. Hudson, A, "Program Error as a British and Death Process", Technical Report SP- 3011, Santa Monica, Cal,:Systems development Corporation, 1967.

17. Fenghong Zou , Joseph Davis "Analysing and Modeling Open Source Software Bug Report Data', 19th Australian Confeence on Software Engineering.,2008 IEEE.

18. Fenghong Zou , Joseph Davis " A Model of Bug Dynamics for Open Source ", The second International Conference on Secure System Integration and Reliability Improvement., 2008 IEEE.

19. Sharifah Mashita Syed-Mohamad, Tom McBride, 'Reliability Growth of Open Source Software using Defect Analysis', 2008 International conference on Computer Science and Software Engineering, 2008 IEEE.

20. Musa, J.D., Iannino, A. and Okumoto, K. (1987),"Software Reliability: Measurement, Prediction, Application', pp. 621.

21. http://www.debian.orgs

CALL FOR PAPERS

Journal: International Journal of Software Engineering (IJSE)
Volume: 2 **Issue:** 1
ISSN:2180-1320
URL: http://www.cscjournals.org/csc/description.php?JCode=IJSE

About IJSE

The **International Journal of Software Engineering (IJSE)** provides a forum for software engineering research that publish empirical results relevant to both researchers and practitioners. IJSE encourage researchers, practitioners, and developers to submit research papers reporting original research results, technology trend surveys reviewing an area of research in software engineering and knowledge engineering, survey articles surveying a broad area in software engineering and knowledge engineering, tool reviews and book reviews. The general topics covered by IJSE usually involve the study on collection and analysis of data and experience that can be used to characterize, evaluate and reveal relationships between software development deliverables, practices, and technologies. IJSE is a refereed journal that promotes the publication of industry-relevant research, to address the significant gap between research and practice.

IJSE List of Topics

The realm of International Journal of Computer Networks (IJSE) extends, but not limited, to the following:

- Ambiguity in Software Development
- Architecting an OO System for Size Clarity Reuse E
- Computer-Based Engineering Techniques
- History of Software Engineering
- Impact of CASE on Software Development Life Cycle
- Iterative Model
- Licensing
- Object-Oriented Systems
- Quality Management
- SDLC
- Software Deployment
- Software Engineering
- Application of Object-Oriented Technology to Engin
- Composition and Extension
- Data Modeling Techniques
- IDEF
- Intellectual Property
- Knowledge Engineering Methods and Practices
- Modeling Languages
- Project Management
- Rational Unified Processing
- Software Components
- Software Design and applications in Various Domain
- Software Engineering

Demographics
- Software Engineering Methods and Practices
- Software Ergonomics
- Structured Analysis
- Systems Engineering
- UML
- Economics
- Software Engineering Professionalism
- Software Maintenance and Evaluation
- Structuring (Large) OO Systems
- Test Driven Development

Important Dates

Volume: 2
Issue: 1
Paper Submission: January 31, 2011
Author Notification: March 01, 2011
Issue Publication: March / April 2011

CALL FOR EDITORS/REVIEWERS

CSC Journals is in process of appointing Editorial Board Members for ***International Journal of Software Engineering (IJSE)***. CSC Journals would like to invite interested candidates to join **IJSE** network of professionals/researchers for the positions of Editor-in-Chief, Associate Editor-in-Chief, Editorial Board Members and Reviewers.

The invitation encourages interested professionals to contribute into CSC research network by joining as a part of editorial board members and reviewers for scientific peer-reviewed journals. All journals use an online, electronic submission process. The Editor is responsible for the timely and substantive output of the journal, including the solicitation of manuscripts, supervision of the peer review process and the final selection of articles for publication. Responsibilities also include implementing the journal's editorial policies, maintaining high professional standards for published content, ensuring the integrity of the journal, guiding manuscripts through the review process, overseeing revisions, and planning special issues along with the editorial team.

A complete list of journals can be found at http://www.cscjournals.org/csc/byjournal.php. Interested candidates may apply for the following positions through http://www.cscjournals.org/csc/login.php.

Please remember that it is through the effort of volunteers such as yourself that CSC Journals continues to grow and flourish. Your help with reviewing the issues written by prospective authors would be very much appreciated.

Feel free to contact us at coordinator@cscjournals.org if you have any queries.

Contact Information

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: +603 6207 1607
+603 2782 6991
Fax: +603 6207 1697

BRANCH OFFICE 1
Suite 5.04 Level 5, 365 Little Collins Street,
MELBOURNE 3000, Victoria, AUSTRALIA

Fax: +613 8677 1132

BRANCH OFFICE 2
Office no. 8, Saad Arcad, DHA Main Bulevard
Lahore, PAKISTAN

EMAIL SUPPORT
Head CSC Press: coordinator@cscjournals.org
CSC Press: cscpress@cscjournals.org
Info: info@cscjournals.org

COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA