IWSC 2009 - 6th INTERNATIONAL WORKSHOP ON SOFTWARE CYBERNETICS
33rd Annual IEEE International Computer Software and Applications Conference
Seattle, Washington, USA, July 20 - 24th, 2009.

Risk-Based Adaptive Group Testing of Web Services

Xiaoying Bai
Dep. of Comp. Science and Technology,
Tsinghua University, China
baixy@tsinghua.edu.cn

Ron S. Kenett
KPA, Israel, Univ. of Torino, Italy and
Ctr. for Risk Engineering, NYU-Poly, NY, USA
ron@kpa.co.il

Abstract

Comprehensive testing is necessary to ensure the quality of complex web services that are loosely coupled, dynamic bound and integrated through standard protocols. Testing of such web services can be however very expensive due to the diversified user requirements and the large numbers of service combinations delivered by the open platform. Group testing was introduced in our previous research as a selective testing technique to reduce test cost and improve test efficiencies. This paper proposes a risk-based approach to group test selection. With this approach test cases are categorized and scheduled with respect to the risks of their target service features. The paper reports a first attempt for an ontology-based quantitative risk assessment of semantic web services. The failure rate and the importance of services are estimated based on a complexity and dependency analysis of the ontology specified domain knowledge model and various service model. The paper also discusses the risk-based group testing process and strategies for ranking and ruling-out services of the test groups, at each risk level. Runtime monitoring mechanism is incorporated to detect the dynamic changes in service configuration and composition so that the risks can be continuously adjusted online.

1. Introduction

Service Oriented Architecture (SOA) promotes the service delivery of proprietary in-house components by encapsulating them into standard programmable interfaces that can be accessible through Internet protocols [25]. On the Internet, there may exist multiple services that provide similar functionalities. For example, many companies like Google, Yahoo and Baidu provide the MAP services for map locating and visualization. An application builder can select from multiple candidates and compose the most suitable services in a specialized business workflow.

Testing and evaluation is necessary in the lifecycle of service-based software development in order to choose the services that best satisfy the users' needs of functionalities and quality [3][4][31][32][33]. However, testing is usually expensive and confidence in a specific service is hard to achieve, especially in the open Internet environment. The users may have multiple dimensions of expected features and functional points that result in a large number of test cases. To exercise a large set of test cases on the large number of service candidates is both time and resources consuming. To overcome these issues, the concept of group testing was introduced to services testing [3][31][32]. With this approeach, test cases are categorized into groups and exercised group-by-group. In each group-testing stage, the failed services are eliminated through a pre-defined ruling out strategies. With effective test case ranking and service ruling out strategies, a large number of services can be removed at the early stage of testing, and thus reduce the total number of executed tests.

This paper is an extension of our previous research on web services group testing. It introduces the risk-based strategy for test case ranking and selection. Software risk assessment determines the most demanding and important aspects of the software under test [14]. In general, the risk of a software feature is defined by two factors: the probability to fail, and the consequence of the failure. That is, $Risk(f) = P(f) * C(f)$, where f is a specific software feature, *Risk(f)* is its risk exposure, *P(f)* is the failure probability and *C(f)* is the cost of the failure. A software feature is risky if it has a high probability to fail or its failure will result in serious consequence. Intuitively, a feature with a high risk either has a high probability to fail, or its failure will result in serious consequences. Hence, a risky feature deserves more testing effort. Risk-based testing was proposed to select and schedule test cases based on software risk analysis [1][2][9][18][22][23][24][28]. With this approach, test cases can be prioritized and scheduled based on the risk of their target software features. The test cases for risky features deserve more effort and have higher priorities. When time and resources are limited, test engineers can select a subset of test cases with the highest priority to achieve good enough quality with

affordable testing effort. This approach has similarity with the studies of Taleb on risks of rare events in the financial industry [29][30].

Traditional software is built and maintained within a trusted organization. In contrast, service-based software is characterized by dynamic discovery and composition of loose-coupled services that are published by independent providers. A system can be constructed on-the-fly by integrating reusable services through standard protocols. In many cases, the constituent data and functional services of a composite application are out of the control of the application builder. As a consequence, service-based software has a higher probability to fail compared with software developed in-house. This is mostly due to the instability, unreliability, and unpredictability of the open environment. A failure may result from many factors such as misused data, unsuccessful service binding, and unexpected usage scenarios. In this work we propose a hierarchical approach to quantitatively analyze risks from three layers:

- Data layer: the risks of the input/output parameters
- Unit layer: the risks of each operations
- Integration layer: the risks of usage scenarios

A key feature of the proposed approach is an ontology-based risk assessment. Sir Time Berners-Lee brought the vision of the semantic Web as a new form of Web content in which the semantics of information and services are defined and understandable by the computers [6]. Semantic Web is now considered the future for the World Wide Web to evolve. Ontology techniques are widely used to provide a unified conceptual model of the semantics of Web content [12][20][27][34][35]. However, semantics introduce more risks to the distributed web services. First, following the loose-coupled service-oriented architecture, the ontolgoies are defined, used, and maintained by different parties. It is hard to ensure the completeness and consistency, and the unified quality of the ontologies. Second, ontologies introduce more complex dependencies among the data and services. For example, classes with relationships like “mutual exclusive” and “inheritance” may be misused. Spies and www.musing.eu [36] introduce a comprehensive approach of ontology engineering to financial and operational risks. In Web Services, XML-based standards are defined as ontology specification languages, which are self-descriptive, machine-interpretable, and platform-independent. Particularly, OWL (Web Ontology Language) is used for describing domain ontologies; and OWL-S defines the workflow semantics of the composite services based on OWL. OWL-S describes the intended system behavior in terms of inputs, outputs, process, pre-/post-conditions and constraints, for both functional and non-functional properties.

Among other things, the paper analyzes the risks of the web services based on the complexity and dependency of the ontology model. Service ontology models, including OWL and OWL-S, are used to assess the risks of the data, service and composite service. On one hand, the failure probability of an ontology-specified artifact is evaluated with respect to the complexity of its relationships with other ontology and property definitions. On the other hand, the importance of the artifact is measured by its dependence with others, especially data inheritance relationship and service data flow and control flow dependencies.

Dynamic composition and re-composition is a unique feature of SOA software. In accordance, risk assessment is also a continuous online practice. By runtime monitoring of the service communication and collaboration, it can identify the changes and instabilities in the SOA software and dynamic adjust the risks of each service and service workflow.

The rest of the paper is organized as follows. Section 2 introduces the background and related work from three aspects: risk-based testing, web service group testing and web service ontology model. Section 3 proposed a failure analysis techniques based on ontology complexity analysis. Section 4 discusses an importance measurement based on data and service dependence analysis. Section 5 presents the adaptive group testing approach. Section 6 concludes the paper.

2. Related Work

Testing is expensive, especially for today’s software with its growing size and complexities. Exhaustive testing is not feasible due to limitations in time and resources. An important test strategy is to improve test effectiveness by selecting and planning a subset of tests with a high probability to find defects. However, selective testing usually faces difficulties in answering questions like “what should we test”? and “when can we stop testing?”. The first question can be handled with statistical design of experiments methods (see Kenett et al [16][17][18]). The second question can be approached using sequential testing methods (see for example Kenett and Pollak [15]). In this paper, we focus on risk analysis to provide a basis for selective testing. Our approach is part of a more general convergence between quality and risk methodologies (Kenett and Tapiero [19]).

Risk-based testing was proposed to select and schedule test cases based on software risk analysis [1] [2][9][18][22][23][24][28]. Basically, test cases are

designed and prioritized according to the risks of their target software features. Risk-related coverage criteria are defined to manage the testing process. Practices and case studies show that testing can benefit from the risk-based approach by reduced resource consumption or improved quality by spending more time on critical functions. Amland [1] established a generic risk-based testing approach based on the Karolak's risk management process model [14]. Bach identified the general risks of software system development including complexity, change, dependency, distribution, third-party, etc [2].

Group testing technique was originally developed at Bell Laboratories for efficiently inspecting products (Sobel [26]). The approach was further expanded to general cases (Watson [33], Dorfman[10]). It is routinely applied in testing large samples of blood samples to speed up the test and reduce the cost (Finucan [11]). In this case, a negative test result of the group under test indicates that all the people in the group do not have the disease; otherwise, at least one of them is affected. Group testing has been used in many areas such as medical, chemical and electrical testing, coding, etc., using either combinational or probabilistic mathematical models.

Tsai et al [32] introduced the ideas of progressive group testing to Web Services testing. Test cases are grouped according to their potency to detect defects, and ranked hierarchically from low potency to high. Test cases are exercised in groups following the hierarchical structure. Ruling out strategies are defined so that web services that failed at one layer can not enter the next-layer testing. The group with a high potency is exercised first with the purpose to remove as many as web services as early as possible.

Group testing is by nature a selective-testing strategy, which is beneficial with reduced number of test runs and shortened test time. To refine the model, Bai et al [3] discussed the ranking and selecting strategy based on test case dependence analysis. Two test cases tc_1 and tc_2 are dependent if a service that fails tc_1 will also fails tc_2. Hence, for any service, a test case will be exercised only if the service passes all its dependent test cases. They [31] further proposed the windowing technique that organizes web services in windows and incorporates an adaptive mechanism. This approach is following software cybernetics theories [7][8] in order to dynamically adjust the window size, determine web service ranking and derive test case ranking.

This paper is an extension of previous research and introduces a risk-based strategy for test case ranking and selection in group testing. it also investigates quantitative risk assessment technique in the context of semantic web services.

3. Failure Probability Analysis

3.1. Complexity-Based Analysis

Following the semantic web service model, the data are defined by ontologies and described using XML-encoded specifications like OWL. OWL defines three types of relationships between ontology classes based on the set theory including subClassOf, sameClassOf and disjointWith. A class may have multiple properties and OWL defines the relationships between property classes including subPropertyOf, samePropertyAs, and inverseOf. For example, the *Accommodation* ontology has two mutual exclusive sub-classes *Hotel* and *Motel*. The quality of a data is directed affected by the complexity of the domain ontology model. A complex ontology model is hard to ensure its completeness and consistency during design and maintenance, and is easy to be misused. An input/output data defined by an ontology with complex relationship is thus has a high probability to fail.

A graph is constructed for complexity analysis and measurement. Each node represents a class of ontology or property. The link represents the relationships between two classes. Different types of links are defined according the OWL definition of relationship types. They are classified into three categories: the ontology-ontology (*OO*) link, the ontology-property (*OP*) link and the property-property (*PP*) link. Figure 1 shows an example graph for the AAWS (Amazon Associates Web Services) ontologies. AAWS is an open service platform for online shopping. Its WSDL service interface provides 19 operations with complicated data structure definition. In this research, we translate the WSDL data definition to ontology definition for semantic-based service analysis. The graph contains 514 nodes and 1096 links.

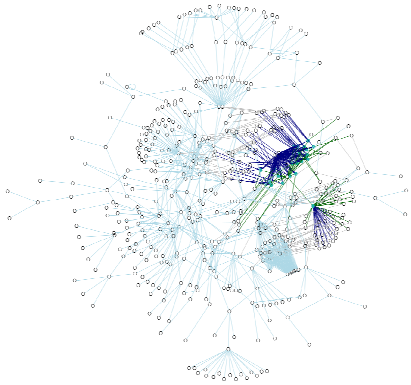


Figure 1 AAWS ontology topology

The ontology class is affected by its related classes in the following ways: 1) any failure of its parent may results in its failure; 2) any failure of its property classes will results in its failure. Hence, the failure probability of an ontology class is defined as follows:

$$P_{adj}(o) = 1-(1-P(o))\times(1-\max_{i=1..m} P(o'_i))\times\prod_{j=1..n}(1-P(p_j))$$

Where

- $P(o)$ is the estimated failure probability of the ontology *o*, and $P_{adj}(o)$ is the adjusted probability taking the relationships into considerations.
- $o'_i \in O$ where *O* is the set of its parent classes with length *m*, that is, $\forall o' \in O$, $o \subseteq o'$. $\max_{i=1..m} P(o'_i)$ is the maximum value of the failure probabilities of its parent classes.
- $p_j \in OP$ where *OP* is the set of properties of *o* with length *n*, and $P(p_j)$ is the failure probability of a property class.

Similarly, we can adjust the property failure probability based on its relationships, that is:

$$P_{adj}(p) = 1-(1-P(p))\times(1-\max_{i=1..m} P(p'_i))$$

Where

- $P(p)$ is the estimated failure probability of the property *p*, and $P_{adj}(p)$ is the adjusted probability taking the relationships into considerations.
- $p'_i \in OP$ where *OP* is the set of its parent classes with length *m*, that is, $\forall p' \in OP$, $p \subseteq p'$. $\max_{i=1..m} P(p'_i)$ is the maximum value of the failure probabilities of its parent classes.

The algorithm to calculate the adjusted failure probability based on the topology graph is described as follows:

Data Failure Probability (DFP)
Input:

1. The domain ontology model
2. The topology graph *G*
3. The estimated probabilities for ontologies *P[o]* and properties *P[p]*

Output:

1. the adjusted probabilities $P_{adj}[o]$ and $P_{adj}[p]$

1. for each ontology class, set $\lambda[o]$ =1-P[o];
2. for each element, set $P_{adj}[o]=0$ and $P_{adj}[p]=0$
3. while there is more nodes on G, do
4. get the next node *nd*;
5. if *nd* is an ontology node and $P_{adj}[nd]=0$
6. set γ =0 is the max prob. of *nd*'s parents.
7. get all the related nodes *R* of *nd;*
8. for each ontology node *rnd* in *R*, do
9. if *rnd* is a parent of *nd*
10. recursively calculate the adj. prob. of *rnd*;
11. $\gamma = \max(\gamma, P_{adj}[rnd])$;
12. end if
13. $\lambda[nd] = \lambda[nd]\times(1-\gamma)$;
14. end for
15. for each property node *pd* in R, do
16. set $\beta = 0$ is the max prob. of *pd*'s parents
17. get all the related nodes *rpd* of *pd*;
18. for each *rpd* do
19. if *rpd* is a parent of *pd*
20. recursively calculate the prob. of *rpd*;
21. $\beta = \max(\beta, P_{adj}[rpd])$;
22. end if
23. end for
24. $P_{adj}[pd] = 1-(1-P[pd])\times(1-\beta)$;
25. $\lambda[nd] = \lambda[nd]\times P_{adj}[pd]$;
26. end for
27. $P_{adj}[nd] = 1-\lambda[nd]$
28. end if
29. end while

The failure probability of a service is calculated by multiplying that of its functions and its input/output messages, as follows:

$$P(s) = 1-(1-P_f(s))\times\prod_i(1-P_{adj}(d_i))$$

Where

- $P(s)$ is the failure probability of a service.
- $P_f(s)$ is the failure probability of the functionality.
- $d_i \in D$ is the set of parameters of the service *s*.
- $P_{adj}(d_i)$ is the adjusted failure probability of each data ontology.

3.2. Workflow-Based Analysis

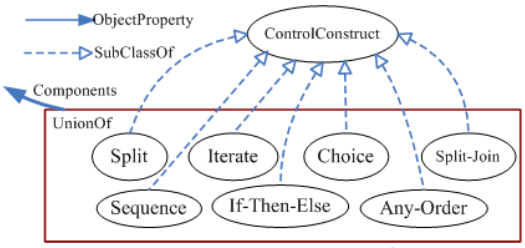
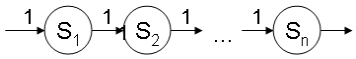
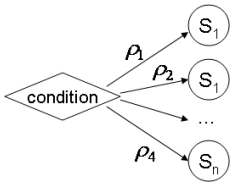


Figure 2. Basic Construct of OWL-S

A composite service defines the workflow of a set of constituent services. For example, in OWL-S, the ServiceModel is modeled as a workflow of processes including atomic, simple and composite processes. The main process for composition of Web services is composite process which is composed of some constructs as shown in Figure 2. Each composite process holds a Control Construct which is one of the following: Sequence, Split, Split-Join, Any-Order, Iterate, If-Then-Else, and Choice. The Constructs can contain each other recursively, so the composite process can model all of the possible workflows of WS.

The services may be conditioned (e.g. If-Then-Else) or unconditioned executed (e.g. Sequence) based on the control constructs. Hence, the failure probability of the integrated services is affected by both the behavior and execution probability of each constituent service. In an unconditioned control construct, the failure of any service will result in the failure of the construct. In contrast, in a conditioned one, the contribution of each branch to the construct failure is proportional to the execution probabilities of each path. The higher the path is executed, the more the path's failure affects the construct. In general, the execution probability can be obtained from statistical usage model such as operation profile analysis. Table 1 lists the formula to calculate the failure probability of different control constructs.

Table 1. Failure probability of different categories of control constructs

Control Construct (*cc*)			*P(cc)*
Category	Graphic Expression	OWL-S Example	
Unconditioned		Sequence, Split, Any-Order	$P(cc) = 1 - \prod_{i=1..n}(1 - P(s_i))$
Conditioned		If-Then-Else, Choice Iterate, While-Repeat	$P(cc) = \sum_{i=1..n} \rho_i P(s_i)$ Where, ρ_i is the execution probability of a service s_i and $\sum_{i=1..n} \rho_i = 1$

The algorithm for calculating the failure probability of integrated service is listed as follows.

Composite Service Failure Probability (CSFP)
Input:
1. the OWL-S specified composite service *CS*
2. the failure prob. of each service in CS *P[s]*
3. The exe. prob. of each sub process *EP[ps]*

Output:
1. the failure probability P(CS)

1. set *P(CS)* = 0
2. While there is more process in CS do
3. Take the next process *proc*;
4. set $P_{proc}=0$;
5. if *proc* is an atomic or simple process
6. get the service *s* in *proc*
7. $P_{proc}=P[s]$;
8. else
9. get the control construct *cc*;
10. get the processes *PROC* in *cc*;
11. for each *proc'* in *PROC* do
12. recursively get the failure prob. $P_{proc'}$;
13. if *cc* is a conditioned construct
14. $P_{proc} = P_{proc} + EP[proc'] \times P_{proc'}$
15. else
16. $P_{proc} = 1 - (1 - P_{proc}) \times (1 - P_{proc'})$
17. end if
18. end for
19. end if
20. $P(CS) = P_{proc}$;
21. end while

4. Importance Analysis

4.1. Dependence-Based Measurement

At the data layer, we use the number of sub-classes to measure the importance of an ontology and property classes. Given a domain *D*, the relative importance of a class *c* is defined as the ratio between the number of its sons and the maximum sons for any class in *D*. That is:

$$C(c) = N_{son}(c) / \max_{i=1..n}(N_{son}(c_i))$$

Where

- $C(c)$ is the relative importance of a class.
- $N_{son}(c)$ is the number of sons of a class c.
- $CLASS^D = \{c_i\}$ is the set of classes in domain D.
- $\max_{i=1..n}(N_{son}(c_i))$ is the maximum number of sons for any class in D.

The importance of the ontology class is further adjusted by taking the importance of its properties into consideration. Given an ontology *o* with a set of *m* properties $\{p_i\}$, we choose the most important property as the major affecting factors and get the adjusted formula as follows:

$$C_{adj}(o) = a \cdot C(o) + b \cdot \max_{i=1..m}(C(p_i))$$

Where *a* and *b* are the pre-defined weights of each factor and *a+b=1*.

At the service level, a service dependence model is constructed as a directed graph. Each node on the graph represents a service. A directed link from service s_1 to s_2 means that the inputs of s_2 depend on the outputs of s_1. For example, to book a ticket, the user needs to first find the flight number, hence, the input of the service *BookTicket()* depends on the output of *SearchFlight()*. Suppose $S=\{s_i\}$ is the set of services, the $Input(s)=\{d_i\}$ is the set of input data for a service *s* and $Output(s)=\{d_i\}$ is the set of output data for a service *s*. The dependence relationship can be defined as follows

Definition 1: $Dep(s)=\{s_i\}$ is the set of services that are dependent on the service *s* such that $\forall s_i \in Dep(s)$, $Input(s_i) \cap Output(s) \neq \Phi$

Such dependence can be obtained by data flow analysis in a composite service. We use the number of dependent services, or the length of *Dep* as $|Dep(s)|$, to measure the service importance within a composite context. Similar to the data analysis, the importance of a service is defined as a relative ratio as follows:

$$C(s) = |Dep(s)| / \max_{i=1..n}(|Dep(s_i)|)$$

Where

- $C(s)$ is the relative importance of a service *s*.
- $S^{CS} = \{s_i\}$ is the set of services in a composite service *CS*.
- $\max_{i=1..n}(|Dep(s_i)|)$ is the maximum number of dependences for all the services in the composition.

Each service may take multiple input/output messages defined by the ontology classes. Hence, the importance of a service is further adjusted with its data importance. Given a service *s* with a set of *m* input/ output data $\{d_i\}$, the adjusted importance of *s* is calculated as follows:

$$C_{adj}(s) = a \cdot C(s) + b \cdot \max_{i=1..m}(C(d_i))$$

Where *a* and *b* are the pre-defined weights of each factor and *a+b=1*.

4.2. Usage-Based Measurement

In SOA, data, services and service compositions are decoupled with each other. Hence, a data will be used by different service operations and a service will be integrated into various applications in different business context. The usage-based approach is to tack the usage of data and services. The more they are used, the more important they are.

We can use different algorithms to calculate the importance of a data or service. For example, given an element (data or service) *e*, suppose that 1) it is used by *m* contexts; 2) the times used in each context is n_i; 3) the importance of each usage context is w_i. We can use the weighted average to measure as follows:

$$C(e) = \sum_{i=1..m} w_i n_i \Big/ \sum_{i=1..m} n_i$$

Given a large number of measured elements, we can also use the statistic models to normalize the value. For example, we can use the Bayesian average as follows:

$$C_{adj}(e) = [\frac{1}{n}\sum_{i=1..k} C(e_i) N_{e_i} + N_e \times C(e)] \Big/ [\frac{1}{n}\sum_{i=1..k} N_{e_i} + N_e]$$

Where,

- $E = \{e_1, e_2, ..., e_k\}$ is the set of *k* measured elements.
- N_e is the total number of usage for an element.

5. Risk-Based Group Testing

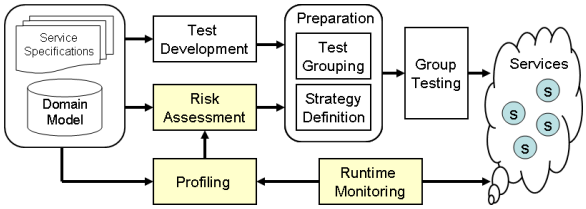


Figure 3 Risk-Based Group Testing

Figure 3 shows the overall approach of risk-based group testing. Risks assessment is a parallel process to the test development and provides the basis for preparing group testing, including categorizing and prioritizing the test cases and deciding the strategies for service ranking and ruling out.

Runtime monitoring mechanism is incorporated for tracing the dynamic behavior of services and detecting any changes in the service and service compositions [5]. For example, when a service fails or becomes unavailable due to network instability, the system may re-discover and re-bind to another candidate with similar functionalities at runtime. As a result, the dependency and usage relationships are changed and risks needs to be re-assessed with the changed model.

Profiling is the process to analyze the complexity and dependences of the service-based software based on both the models and monitoring logs.

5.1. Risk-Based Ruling-Out Strategies

Test cases are grouped and ranked based on the risks of their target features. Different risk levels are defined and the test cases of the same risk level are organized in one group. The groups are scheduled for exercise based on their risk levels. At each risk level, the ruling out strategy is defined so that services with failure rates exceed the threshold will be eliminated. The general process is as follows:

1. Evaluate the risks of the service features;
2. Identify the risks of the test cases based on their target features;
3. Define the risk levels and group the test cases into different risk levels;
4. Define the rule-out strategy for each risk level;
5. Select the test group with the highest risk level and exercise the test cases on the services;
6. Rank the services based on their failure rates on the test cases;
7. Compare the service ranks with the rule out threshold. Services that do not meet the threshold are ruled out. The remaining services enter the next-level group testing.
8. Repeat steps 5~7 until all the groups are exercised.
9. The remaining services are selected, which are of the satisfied quality.

The rank of the services is defined as weighted average of failures, that is,

$$R_l(s) = \sum_{i=1..|T^l|} r_i f_i$$

Where

- $R_l(s)$ is the rank of a service s for a test group of risk level l.
- $T^l = \{t_1^l, t_2^l, ..., t_n^l\}$ is the set of n test cases in the group of risk level l.
- r_i is the risk of a test case $t_1^l \in T^l$.
- f_i identifies whether the service s fails or passes the test case $t_1^l \in T^l$, and

$$f_i = \begin{cases} 1, & if\ s\ passes\ t_i^l \\ -1, & if\ s\ fails\ t_i^l \end{cases}$$

Ruling-out threshold is direct proportion to the risk level. The higher the risk level, the more trick the rule out strategy and the higher the threshold. For example, the strategy may be defined as follows:

$$Out_l(s) = \begin{cases} \sum_{i=1..|T^l|} r_i, & for\ any\ level\ that\ R_l > K \\ \frac{1}{2} \sum_{i=1..|T^l|} r_i, & for\ any\ level\ that\ R_l \le K \end{cases}$$

6. Conclusion

Semantic Web, ontology based group testing is a breakthrough approach that can prove critical to Service Oriented Architecture systems. Without assuring performance and reliability, such systems will not deliver their promise.

In setting up progressive group testing strategies several challenges need to be addressed:

1. Estimation of probabilities
2. Specifying dependencies
3. Updating estimates dynamically
4. Evaluating the sensitivity of the rules parameters.

Statistical methodology can help meet these challenges. In particular, Bayesian techniques such as Bayesian inference and Bayesian Networks can provide complementary methodologies to the ones presented in this paper. For examples of such procedures in the context of Web Usability assessment see Harel et al [].

7. Acknowledgement

This research is supported by National Science Foundation China (No. 60603035) and National High Technology Program 863 (No. 2006AA01Z157). The second author was partially supported by the FP6 MUSING project (IST- FP6 27097).

References

[1] S. Amland, "Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study", *The Journal of Systems and Software*, 53(3), 2000, p. 287-295.

[2] J. Bach, "Heuristic Risk-Based Testing", *STQE Magazine* 1(6), Nov. 1999.

[3] X. Bai, Z. Cao and Y. Chen, "Design of a Trustworthy Service Broker and Dependence-Based Progressive Group Testing", *Int. J. Simulation and Process Modeling*, vol. 3, Nos. 1/2, p. 66-79, 2007.

[4] X. Bai, Y. Chen and Z. Shao, "Adaptive Web Services Testing", *IWSC*, p. 233-236, 2007.

[5] X. Bai, S. Lee, R. Liu, W.T. Tsai and Y. Chen, "Collaborative Web Services Monitoring with Active Service Broker", COMPSAC 2008, p. 84-91.

[6] T. Berners-Lee, J. Handler and O. Lassila, The semantic Web, *Scientific American Magazine*, May, 2001. (Revised 2008)

[7] Kai-Yuan Cai, "Optimal Software Testing and Adaptive Software Testing in the Context of Software Cybernetics", *Information and Software Technology*, pp. 44:841-844, 2002.

[8] J.W. Cangussu, S. D. Miller, K. Y. Cai, A. P. Mathur, "Software Cybernetics", in *Encyclopedia of Computer Science and Engineering*, to be published by John Wiley & Sons.

[9] Y. Chen, X and R.L. Probert, "A Risk-based regression test selection strategy", 14th ISSRE, 2003.

[10] R. Dorfman, "The detection of defective members of large population", *Annals of Mathematical Statistics*, 14, pp. 436-440, 1964.

[11] F.M. Finucan, "The blood testing problem", *Applied Statistics*, 13, pp. 43-50, 1964.

[12] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering*, Springer press, 2005.

[13] A. Harel, R.S. Kenett and F. Ruggeri, "Decision support for user interface design: Usability diagnosis by time analysis of the user activity", *COMPSAC*, 2008.

[14] D.V. Karolak, *Software Engineering Risk Management*, IEEE Computer Society Press, Silver Spring, MD, 1996.

[15] R.S. Kenett and M. Pollak, "A Semi-Parametric Approach to Testing for Reliability Growth with an Application to Software Systems", *IEEE Transactions on Reliability*, Vol. R-35, 3, pp. 304-311, 1986.

[16] R.S. Kenett and S. Zacks, *Modern Industrial Statistics: Design and Control of Quality and Reliability*, Duxbury Press, San Francisco, 1998, Spanish edition 2000, 2nd paperback edition 2002, Chinese edition 2004.

[17] R.S. Kenett and E. Baker, *Software Process Quality: Management and Control*, Marcel Dekker Inc., New York, 1999, Taylor and Francis, Kindle Edition, 2007.

[18] R.S. Kenett and D. Steinberg, "New Frontiers in Design of Experiments", *Quality Progress*, pp. 61-65, August 2006.

[19] R.S. Kenett and C. Tapiero, "Quality and Risk: Convergence and Perspectives", *QPRC*, 2009.

[20] D. Martin et al, "Bringing Semantics to Web Services: The OWL-S Approach", *SWSWPC*, 2004.

[21] G. Myers, *The Art of Software Testing*, Wiley & Johns, 1979.

[22] I. Ottevanger, "A risk-based test strategy", *STARWest*, 1999.

[23] F. Redmill, Exploring risk-based testing and its implications, *Software Testing, Verification and Reliability*, no. 14, p.3-15, 2004.

[24] L.H. Rosenberg, R. Stapko, A. Gallo, "Risk-based object oriented testing", 24th *SWE*, 1999.

[25] M. P. Singh, M. N. Huhns, *Service-Oriented Computing*, John Wiley & Sons, 2005.

[26] M. Sobel and P.A. Groll, "Group testing to eliminate all defectives in a binomial sample", *Bell System Technical Journal*, v. 38, no. 5, pp 1179-1252, 1959.

[27] M. Spies, "An ontology modeling perspective on business reporting", *Information Systems*, in press, 2009.

[28] H. Stallbaum, A. Metzger, K. Pohl, "An automated technique for risk-based test case generation and prioritization", *ICSE*, p. 67-70, 2008.

[29] N.N. Taleb, *The Black Swan: The impact of the highly improbable*, Random House, NY, 2007.

[30] N.N. Taleb, "The Fourth Quadrant: A Map of the Limits of Statistics", *Edge*, http://www.edge.org/ 3rd_culture/taleb08/taleb08_index.html 2008.

[31] W.T. Tsai, X. Bai, Y. Chen, and X. Zhou, "Web Services Group Testing with Windowing Mechanisms", *SOSE*, p. 213-218, 2005.

[32] W.T. Tsai, Y. Chen, Z. Cao, X. Bai, H. Huang, R. Paul, "Testing Web Services Using Progressive Group Testing", *Advanced Workshop on Content Computing*, pp.314 – 322, 2004.

[33] G.S. Watson, "A Study of the Group Screening Method". *Technometrics* 3(3):371–388, 1961.

[34] W3C, "OWL Web Ontology Language Overview", available at: http://www.w3.org/TR/ 2004/REC-owl-features-20040210/.

[35] W3C, "OWL-S: Semantic Markup for Web Services", available at: http://www.w3.org/ Submission/ 004/ SUBM-OWL-S-20041122/.

[36] MUSING: Next Generation Business Intelligence Semantic Technologies, available at: http://www.musing .eu.