Risk-Aware Web Service Allocation in the Cloud Using Portfolio Theory

Faisal ALRebeish
School of Computer Science
University of Birmingham
Birmingham, UK
Fma018@ cs.bham.ac.uk

Rami Bahsoon
School of Computer Science
University of Birmingham
Birmingham, UK
R.Bahsoon@ cs.bham.ac.uk

Abstract— **In this paper, we view the cloud as market place for trading instances of web services, which can be bought or leased by web applications. Applications can buy diversity by selecting web services from multiple cloud sellers in a cloud-based market. We argue that by diversifying the selection, we can improve the dependability of the application and reduce risks associated with service level agreements violations. We propose a novel dynamic adaptive search based software engineering approach, which uses portfolio theory to construct a diversify portfolio of web service instances, traded from multiple cloud providers. The approach systematically evaluates the Quality of Service and risks of the portfolio, compare it to the optimal traded portfolio at a given time. It can then dynamically decide on a new portfolio and adapt the application accordingly. We use a hypothetical scenario to demonstrate the effective use of the portfolio-based optimization.**

Keywords—**Modern Portfolio Theory; Risk; Web Service Allocation; Cloud Based Market; Cloud Computing; Design Diversity.**

I. BACKGROUND & MOTVIATION

Web services have gained growing interest due to its importance in developing business to business (B2B) or web applications. Simultaneously, cloud computing promises the delivery of reliable, affordable and on-demand services, which could be leased or traded [1]. The popularity of the cloud and its distinctive economies of scale computation advantages, make a cloud-based market a plausible and an attractive option for publishing and trading web services. On the other hand, offering web services through a cloud-based market underlies risks associated with probable service failure caused, for instance, by undependable service provision of the cloud service provider, hardware malfunctions or unpredicted fluctuation in demands for the traded service as a shared resource and so forth. All these factors may increase the risks associated with Service Level Agreements (SLAs) violations for web applications benefiting from the cloud-based market.

We view the cloud as a marketplace for trading instances of abstract web services, which web applications can explore trade and use as substitutable and composable entities in the architecture of the application. That is, for a given abstract service A, there exist multiple concrete services Ai…An in the market offering comparable functionality but differing in their price and Quality of Service (QoS) provisions. We view the selection of concrete web service instances from cloud-based market as a "search problem", which optimize for reducing probable risks and improving SLA compliance. We look at such optimization from the buyer's (i.e. a web-based application) point of view. As a novel perspective, our risk based fitness function advocates diversifying the selection and consequently the allocation of traded web service instances. The intuition is that by diversifying through selecting multiple instances from multiple providers, we can improve dependability and reduce the probable risks of SLA violations. The challenge would be to find an efficient and effective solution for investing in such diversity. Diversity in design is a mature topic [2] and it been used as a strategy to increase the reliability and dependability of software systems, such in [3] and [4]. This can be achieved by creating two or more independent versions of the same service, where all of the independent versions tend to meet the specification. However, each independent version has its unique design decisions and is implemented in a distinctive way. In this case, if a fault occurs in one of the versions, there is great chance that the other solutions continue to be intact.

The application of search based optimization techniques to software engineering has recently witnessed intense activity right across the life-cycle from requirements engineering, project planning and cost estimation through testing, to automated maintenance, service oriented software engineering, compiler optimization and quality assessment [5]. A wide range of "classical" optimization and search techniques has been used. These are mainly local search, simulated annealing, genetic algorithms and genetic programming. As mentioned by Harman [5], the majority of search-based software engineering (SBSE) has so far been concerned with 'static' or 'offline' optimization problems. In "static" problems, the algorithm is executed off line in order to find a solution to the problem in hand. In "dynamic" optimization or "on line" Search Based Software engineering, solutions are repeatedly generated in real time and applied during the lifetime of the execution of the system to which the solution applies [5]. This is in line with the work in dynamic optimization, which is directly relevant to dynamic SBSE [6] [7]. Moreover, the runtime nature of dynamic SBSE does require lightweight, simple, scalable, and computation inexpensive optimization techniques. This is necessary as the search is continuously active and may be initiated at various time intervals of the execution not only for seeking to find an optimal solution to the said problem, but rather, it may seek to improve upon the current runtime situation. Classical optimization and search techniques may be ineffective and expensive to use in addressing the dynamic SBSE requirements for the representation of the problem and the definition of the fitness function are mere active at runtime and volatile with respect to time.

The novelty of the approach is on two fronts main: (i) utilizing the concept of portfolio to diversify the selection of web services from multiple cloud providers (sellers) in order to reduce the probable risks of SLA violations. To the best of our knowledge, we are not aware of any economics-driven selection approach, which explicate diversity in the allocation of multiple instances of web services, as a mechanism for ensuring SLA compliance benefiting from cloud-based markets. (ii) The use of portfolio theory, as online SBSE for diversifying the allocation of Web Services in the Cloud aimed at optimizing SLA for risk compliance (i.e. expected return takes the form of improved dependability through diversity) subject to Quality of Service (QoS) and cost constraints in a given adaptation cycle.

The rest of this paper is organized as follows. First, we give a brief background of Modern Portfolio Theory and describe a portfolio-based approach in dynamic SBSE for selecting web services in Section II. Then we present hypothetical scenario to evaluate the effective of our approach in Section III. Related work is discussed in section IV. Section V represents a reference for implementing the proposed approach in real word. Section VI concludes.

II. THE APPROCH

A. *Modern Portfolio Theory*

The foundation of the Modern Portfolio Theory [8](MPT) was developed by Nobel Prize winner Markowitz in 1950. The aim of MPT is to develop a formal procedure that can support the decision making process of allocating capital to a portfolio comprising from multiple investment assets. The portfolio in this theory is weighted compositions of the asset. The weight represents how much the investor should allocate from the capital to those assets. The MPT helps the investor to decide how much of the available capital should s/he invest in each of the available assets in order to maximize the expected return and minimize the risk from the portfolio of investment. This can be achieved by calculating expected return and risks for every possible portfolio that can be constructed from the available assets. The expected return and risk will evaluate the efficiency of every portfolio and by presenting them in plot chart that have the vertical axis as the expected return and the horizontal axis as the risk (see figure 1).

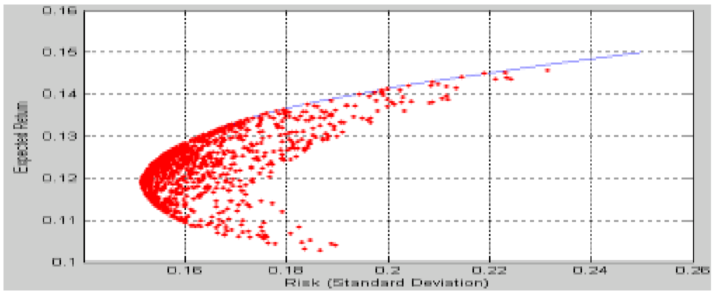

Figure 1. Efficient Frontier for Portfolio of two assets. Courtesy of Mathworks [9].

From figure 1, one can see that the uppermost point form curve in the chart represents the Efficient Frontier of the portfolio. The Efficient Frontier represents portfolios that achieve the maximum expected return from a certain level of risk. For risk adverse investor, the optimum choice will be a portfolio that represents the lowest level of risk in the Efficient Frontier.

The expected return of portfolio E_p that consists of m assets can be calculated as in equation 1 with one constrain represented in equation 2. Where w_i represents is the weight of the capital that is invested in asset i and Ea_i represents the expected return on investing on asset i.

$$\boldsymbol{E_p = \sum_{i=1}^{m} w_i \, Ea_i} \qquad (1)$$

$$\boldsymbol{\sum_{i=1}^{m} w_i = 1} \qquad (2)$$

Under the modern portfolio theory the risk of the portfolio R_p is affected by three factors: the risk associated with each asset Ra_i, the weight of the capital that is invested in each asset w_i and p_{ij}, which is the correlation between assets. The risk of the portfolio R_p is calculated as in equation 3.

$$\boldsymbol{R_p = \sqrt{\sum_{i=1}^{m} w_i^2 \, Ra_i^2 + \sum_{i=1}^{m} \sum_{j \neq i}^{m} w_i w_j \, Ra_i \, Ra_j \, p_{ij}}} \qquad (3)$$

The risk of an individual asset is estimated based on the observation of its return over time. The correlation measures the direction and strength between of the relationship between a couple of assets and it can be estimated by observing of the asset over time. Correlation is represented as a number between +1 and -1, where +1 represents a strong relationship with similar direction, -1 represents a strong relationship with opposite direction and 0 correlations indicate that there is no relation between the two assets.

B. Portfolio Based Web Service Allocation

A cloud market place will facilitate the process of buying and selling instances of web services, where web services will be offered with different prices and QoS. Let us consider a web application that need to allocate multiple instances of web services from a cloud-based market. Previous research has used auction-based methods to dynamically allocate all the instance of web service from a single or multiple providers that have the lowest price and optimal QoS [10]. In contrast, our approach tries to secure the instances by constructing a diversify portfolio of multiple instances of a web service from multiple providers in the cloud based-market. The objective is to minimize the risk of SLA violation through diversity; the degree of SLA compliance differs among providers and henceforth the risk of likely violations.

As a motivating example, let us consider a budget flight booking website that is called Cheapflight.com, which provides online booking web service FlightBooking. We assume that FlightBooking is a web service that leases variants of web service instances from various clouds. The variants tend to provide the same core functionalities but they differ in price and the way they deal with non-functionalities. In high seasons, Cheapflight.com has decided to scale up its services to support an anticipated load in the number of users, through selecting and subsequently allocating additional 100 instances of the FlightBooking services through a cloud-based market. By using traditional auction based methods as in (e.g., [1]), the 100 instances of web services could be allocated from the sellers with lowest price without giving much consideration to the risk of SLA violation. In contrast, our portfolio-based approach is risk-aware (risk-averse constrained to cost and QoS) improve its dependability by diversifying the selection of the 100 instances of the Flightbooking web services from multiple cloud sellers. The objective is to reduce risks associated with service level agreements. Interested reader can refer to the evaluation section for more details on how portfolio thinking can solve the problem.

The main actors in cloud based market are agents acting on behalf of the buyer (cloud-based application) and sellers (cloud service providers):

1. **Seller:** a cloud provider is offering web service S_i in the market for the price C_i and promised QoS.
2. **Buyer**: is exploring the market to select and allocate multiple concrete instances of abstract web services that satisfy its required level of QoS and bounded with price limit C_{max} that it cannot exceed.
3. **Market Regulator:** independent agent that is responsible for monitoring trading in the market. It also monitors for compliance and probable risks leading to SLA violations and it is an integral part of the infrastructure.

One can see that cloud market place has similarities to the financial market. However, when portfolio theory is used to support the web service allocation process a few assumptions need to be taken into account:

- For each web service S_A which is provided by seller A and web service S_B which is provided by seller B, we assume that the performances of web service S_A (i.e. the extent to which the service can be compliant with the SLA) tends to be unique and independent from that of web service S_B . This means that the correlation P_{AB} between any two web services of different providers is equal to zero.
- The expected return E_i of investing in web service S_i is equal to aggregated QoS q_i divided by the cost allocating the web service C_i .
- The Market regulator will provide an evaluation of the risk R_i and aggregated QoS q_i for each web service S_i based in history of that web service. R_i represents the likelihood of SLAs violation for instances of web service S_i . q_i Represents the aggregated QoS of the web services S_i .
- We use the historical record of SLA compliance of S_i to predict the likely risk of SLA violation. The risk of SLA violation R_i is quantified as the percentage between the number of SLAs that have been violated to total number of SLA delivered by the service provider S_i over a given period. Our risk prediction is simplistic but fits the purpose. It is worth mentioning that there have been several sophisticated models for forecasting future risk by using historical data presented by Brooks and Persand [11] like the Short-term Moving Average, Multivariate GARCH and Short Exponentially Weighted Moving average. Our future work will report on more sophisticated variants for systematically predicting risks building on [11].

Taking into account these assumptions, we can apply the portfolio theory, where a cloud-based application is an investor buying web services. The cloud-based application needs to build a diversify portfolio of multiple instances of the web services. Multiple web services offered in the market represents the asset. For each asset it has its own risk R_i , aggregated QoS q_i and price C_i . Based on these values we can then decide on how many instances of a given web service need to be allocated in constructing a portfolio so that the global risk of the portfolio R_p is reduced.

The aggregated QoS q_i is calculated based on selected quality parameters of interest for a given web service. For the simplicity of exposition, we look at availability, execution time and security as three dimensions of QoS to demonstrate our approach:

1. Availability: availability A_i of a web service represents the probability that web service is accessible.
2. Execution Time: execution time Et_i measures the delay in seconds from the moment the web service is requested to the moment a reply is received.

3. Security: security of web service Se_i represents the level of security provision of web service.

The values of A_i , Et_i and Se_i are constants, which can be customized based on the solution domain and the acceptable minimum and maximum thresholds. Though we included three qualities in the formulation, the model can be easily extended to include more qualities. Previous literature has shown that the consideration of QoS parameters has been in the range of 1 to 9 with three being the norm [12]. For a hypothetical domain, let us assume that each of A_i , Et_i and Se_i values can vary between 30 and 100, where 100 represents the optimum value and 30 represents the minimum accepted value. The priority of each of the three quality parameters from the buyer view is represented by a priority weight: w_A , which represents the importance of the availability of the web service, w_{Et} represents the importance of the execution time and w_{Se} represents the importance of the security. The priority weight can be a number between 0 and 1. However, the sum of the three weights cannot exceed 1. The buyer will suggest weights to suit the aggregated QoS , q_i specific requirements.

The aggregated QoS q_i for service S_i is calculated in equation 4 with one constraint represented in equation 5, where w_A, w_E and w_{Se} represents the priority weights of the quality parameters of the web service: availability, execution time and security.

$$\boldsymbol{q_i = w_A A_i + w_E Et_i + w_{Se}\, Se_i} \qquad (4)$$

$$\boldsymbol{w_A + w_E + w_{Se} = 1} \qquad (5)$$

The expected return of web service portfolio E_p that is built by allocating instances of web service from m providers can be calculated as in equation 6 with one constraint represented in equation 7. w_i represents the weight of the web services that is allocated from the web service provider S_i and C_i represent the cost of web service S_i . q_i Represent the aggregated QoS of the web services S_i.

$$\boldsymbol{E_p = \sum_{i=1}^{m} w_i \frac{q_i}{C_i}} \qquad (6)$$

$$\boldsymbol{\sum_{i=1}^{m} w_i = 1} \qquad (7)$$

The risk of SLA violation R_i is quantified as the percentage between the numbers of SLAs that have been violated to total number of the total SLA delivered by the service provider S_i. The global risk of the portfolio R_p is calculated as is in equation 8, where the local risk R_i is the risk associated with the web service S_i.

$$\boldsymbol{R_p = \sqrt{\sum_{i=1}^{m} {w_i}^2\ {R_i}^2}} \qquad (8)$$

In order to optimize the global risk of the portfolio R_p and find the optimum solution, we need to find how much weight w_i should be invested in each web service S_i to construct a low risk portfolio. We can achieve that by applying a non-linear optimization method that is called Generalized Reduced Gradient (GRG2) [13] to equation 8 as the fitness function and 7 and 6 as constrains. The goal of this optimization process is to find the optimum percentage of web service instances from a specific provider so we can construct the minimum risk portfolio. Minimizing the value of R_p in equation 8 will represent the fitness function to select the optimum candidate solution.

Consider that a cloud-based application is expected to perform adaptations when there is a change in the market prices or risks, the basic steps of our portfolio based optimization can be stated as follows:

1- For the first adaptation cycle only, the buyer sets minimum accepted aggregated QOS, the required number of web services and the maximum price that she is willing to pay $\boldsymbol{C_{max}}$ as well as the weight of quality attributes w_A**,** w_E and w_{Se} of the web service.

2- The buyer agents explores the market and gets the web services offers that satisfy the SLAs and the maximum cost constraints of a cloud-based application.

3- The buyer agent access the *Knowledgebase* (see section 5) maintained by the market regulator to retrieve the likely risks R_i of SLA violation and $\mathbf{q_i}$ aggregated QoS associated with each web service provider.

4- Apply Generalized Reduced Gradient (GRG2) nonlinear optimization to equations **6, 7** and **8** to find the weight of each asset on the web service portfolio so we can construct a diversify portfolio with the minimum global risk.

5- Compare the risk of the new optimum portfolio with risk of the currently allocated portfolio(see section 5):

   a) If new optimum portfolio improves on reducing the risk, while satisfying both the QoS and cost constraints, we will perform the allocation of web services based on the recommended weights from the optimization process.

   b) Else we will keep the currently allocated portfolio.

III. Preliminary Evaluation

We take a hypothetical example to demonstrate how a cloud-based application can reduce risks associated with web service allocation by implementing our proposed portfolio based allocation. We compare the results of our method with traditional auction based mechanisms [10], [14]. These are generic resource allocation mechanisms for the cloud, where they match resource demand to provision; they do not implement diversity and tend to perform the allocation based on minimum price (price-based).We compare our results to price-based and risk-based implementation of an auction mechanism.

Recall the case of the budget flight booking website, Cheapflight.com, which provides online booking web service FlightBooking which is a web service that leases variants through various clouds. As mentioned before, the variants of the web service instances from various providers tend to provide the same core functionalities but they differ in price and the way they deal with non-functionalities. In high seasons, Cheapflight.com has decided to scale up its services to support an anticipated load in the number of users, through selecting and subsequently allocating 100 instances. Assume that the maximum price that they are willing to pay is $42 for each instance to ease the illustration.

In table 1, we can see a snapshot of the current FlightBooking web service offers, which are available in the cloud-based market, which satisfy the QoS requirements and the price constrains as well as the risk of SLA violation associated with each web service.

TABLE 1: CURRENT FLIGHTBOOKING WEB SERVICE OFFERS AVAILABLE IN THE CLOUD MARKET

Web service	**Aggregated QoS**	**Price**	**Risk of SLA violation**
FlightBooking 1	66	$30	14%
FlightBooking 2	70	$30.5	13%
FlightBooking 3	80	$35	10%
FlightBooking 4	90	$42	9%

A. *Web service allocation using auction based methods*

If Flight.com is using priced-based auction method to select the web service with the lowest price, the result will be allocating 100 web services from FlightBooking 1 with a cost of $3000 and 14 % risk of SLA violation.

On the other hand, if Flight.com is using Risk-based auction method to select the web service with lowest risk; the result will be allocating 100 web services from FlightBooking 4 by the cost of $4200 and 9% risk of likely SLA violation.

B. *Web service allocation using portfolio based optimization.*

When we use portfolio based optimization to allocate instances of web services, the first step in to find the percentage of instances that we should allocate from each of the identified candidate services in order to achieve the minimum global risk. We can find the weights by applying Generalized Reduced Gradient (GRG2) [13] to equation 8 as the fitness function and 7 and 6 as constrains. The optimum weights of the optimization process are depicted in table 2.

TABLE 2: OPTIMUM WEIGHT OF ALLOCATION FOR EACH OF TRADED WEB SERVICES

Web service	**FlightBooking 1**	**FlightBooking 2**	**FlightBooking 2**	**FlightBooking 4**
Optimum weight	11%	17%	27%	45%

These weights imply that we will be able to construct the minimum risk portfolio of a web service by allocating 11 services from FlightBooking1, 17 services from FlightBooking2, 26 services from FlightBooking3 and 45 services from FlightBooking4. Now equation 8 can be used to calculate the global risk of the portfolio. The global risk of the new portfolio is 5.56% and the cost of constructing this portfolio is $3683.5.

The results of the web service allocation process the CheapFlight.com example using the price-based auction, risk-based auction and our portfolio based allocation are shown in table 3.

TABLE 3: RESULTS OF THE WEB SERVICE ALLOCATION PROCESS THE FLIGHT.COM

Approach	**Risk of SLA Violation**	**Cost of Allocation Process**
Price-based Auction allocation	14%	$3000
Risk-based Auction allocation	9%	$4200
Portfolio based allocation	5.56%	$3683.5

One can see from the results allocation process in table 3 that portfolio based approach have achieved the minimum risk because it utilizes the concept portfolio to diversify the allocation of web services from four different providers instead of relying on one provider to allocate all needed instances of a web services.

IV. RELATED WORK

There have been a number of research on web service selection (e.g. [15], [16], [17]); however, our research explicates services, which are traded and leased in the cloud market. Though the fundamentals of selecting services in cloud and non-cloud environments could appear to be same, there are differences: Cloud-based markets tend to be volatile and dynamic, where cloud providers continuously update their provision for services, QoS and price. In such model, competition is respected, where cloud providers continuously compete for providing better services, QoS and price. Such setting provides ready means for dynamically and adaptively engineering diversity in the way we select and allocate traded web services instances to a web application.

Selection of cloud traded services is a relatively new area; however, none of the approaches have considered the problem of diversity in web service selection through trading and portfolio thinking. The current web services allocation solutions in cloud market (from the buyer's point view), for example, have used traditional auctions based methods (e.g.

[18], [14]) and continuous double auction as in [10]. None of the approaches have considered reducing the risk of SLA violations by diversifying allocation of web services instances from multiple providers.

The work of [19] , [20] and [21] have investigated auction based methods to solve the resources allocation problem in the grid; however, these solutions have looked at classical infrastructural resource allocations problems (e.g, job scheduling and allocation etc.). These approaches have not dealt with the problem of web services allocation.

There have been several research attempts in dealing with dynamic adaptive SBSE as mention by Harman et al [22] with work on architectures to support adaptive middleware [23], Artificial Immune Systems (AIS) for intrusion detection [24] and fault tolerance [25]. Furthermore, Portfolio theory has found its way in numerous applications. This has included optimizing selecting of software project [26], informing the selection of algorithms for numerical optimization problems [27] and an energy-aware resource management in a cloud computing from the provider perspective [28].

To the best of our knowledge, neither Portfolio theory nor dynamic adaptive SBSE have addressed the problem that we explicate in our work. Combining portfolio and dynamic adaptive SBSE is a promising approach for dynamically and adaptively improving QoS, SLA compliance and reducing risks associated with SLA violations through diversity. The approach dynamically links technical decisions of diversity and dependability to value and risks.

V. REALIZATION OF THE MECHANISM

In the evaluation, we have demonstrated an improvement in risk management and also some savings compared to buying resources from a single provider with the best reliability. In this section, we will present a strategy for realizing and implementing the approach in highly dynamic market settings, where multiple sellers and buyers can continuously trade web services with dynamic prices and QoS. The system shall be self-adaptive and self-optimizing to changes that may happen in the market. Therefore, the system has to construct an optimum portfolio of web services and modify it in response to their perception of the market in a timely manner.

According to De Lemos et al [29], adaptation control can be achieved by a sequence of four components: monitor, analyze, plan, and execute (MAPE). These components, when put together, form the building blocks for feedback control systems as explained in control theory. Because scalability is a key concern for our system, we will use a decentralized pattern, where each buyer agent will have its local M, A, P and E components and will interact with other peer agents directly as represented in figure 2.

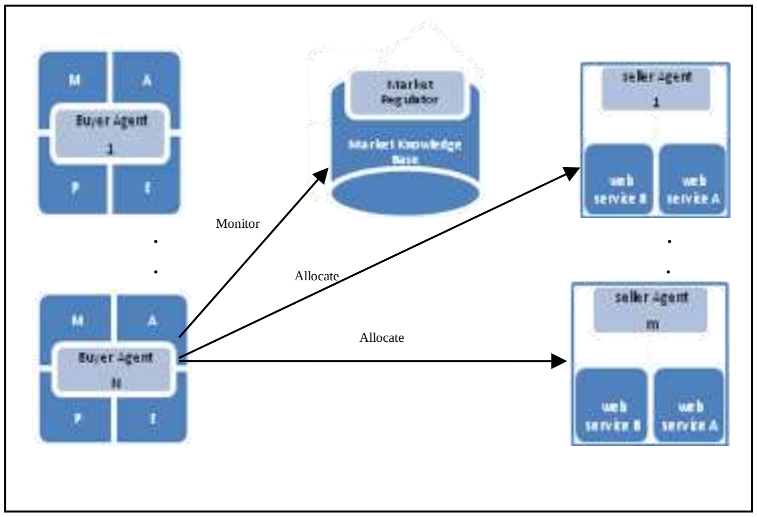

Figure 2. Overview of self-adaptive system implementation

In order to achieve a global consistent view on the market status and limit the interaction among agents, we use a KnowledgeBase which is going to be implemented using a publish/subscribe architectural style. The KnowledgeBase coordinates knowledge of traded web services in relation to QoS, price, risks, compliance and availabilities. Such architecture has the potential of scalability, where the KnowledgeBase will eliminate the overheads of potential buyers interacting with the sellers in each trading cycle. The Market KnowledgeBase represents a reference point in the system. By taking a decentralized architecture, it will be up to the buyer agent to make allocation decisions. For each abstract web service offered in the market, the knowledge base will provide the following information:

- List of the different offers in the market with the prices, QoS and number of available services.
- Historical data related to the availability, execution time, security and risk of SLA violation of traded web services.

There are two important questions in realizing the adaptation mechanism. The questions are: what are the dynamics, which triggers observations? And consequently what trigger adaptation?

Monitoring and Analyzing phase: In this phase we are trying to answer what triggers observation in our system: we will use an events-based observation and that will help our system to gain a lightweight interaction between the agents and market KnowledgeBase. First, the buyer agent subscribes to the abstract web service, which she is interested in (e.g. FlightBooking web services, PhotoStorage web services). Then a change in the prices or risk of these abstract web services will trigger an observation and activate the control loop.

In the motoring component, a buyer agent retrieves all the key information about the abstract web services from the Market Knowledge Base. Then the buyer agent analyzes the information to get the customized aggregated QoS that match his preference by using equation 4 and 5 and that will make the agent ready to start the next Planning and execution phase.

Planning and execution phases: Here we are trying to find the answer to the second question, what triggers adaptation in our system: The first step is to evaluate the currently allocated portfolio risk $R_{pcurrent}$ and the optimum potential portfolio risk $R_{poptimum}$ that we could allocate based in the new market state by using equation 8.

I_c represents the level of improvement that the system cloud gain by allocating the new optimum portfolio $R_{poptimum}$. In other words, I_c represent the potential improvement in risk between the current portfolio risk $R_{pcurrent}$ and the new optimum portfolio $R_{poptimum}$ and I_c is calculated as in equation 9. A Positive number will represent an improvement in the portfolio risk.

$$I_c = R_{pcurrent} - R_{poptimum} \qquad (9)$$

Furthermore, with the reallocation of web services there will be an extra cost as penalties for breaking the contract. So, the cost of the adaptation process will be the cost of allocating the new web services plus the cost of penalties for breaking any contract.

The buyer will set I_m that represents the minimum accepted improvement level in the new portfolio to trigger the adaptation and replace the current portfolio with a new optimum one. This will make the adaptation of a new optimum portfolio subject to the satisfaction of two conditions.

1. The cost of constructing the new optimum portfolio plus the penalties is less than the price limit C_{max}.
2. The level of improvement I_c in a new optimum portfolio is greater than or equal to the minimum level of accepted improvement I_m.

We will build prototype of such system using open source tools like open nebula [30]. This will evaluate the approach in real cloud setting.

VI. CONCLUSION

In this paper, we have introduced a novel dynamic and adaptive search based optimization approach for web services selection and allocation in the cloud using Portfolio thinking. We viewed the cloud as a marketplace for trading instances of abstract web services, which web applications can explore, trade and use as substitutable and composable entities. In particular, we have used a portfolio based optimization to improve SLA compliance by diversifying the selection and consequently the allocation of traded instances of web services from multiple providers. Our approach represents an efficient and effective solution for investing in diversity, while reducing risks as demonstrated in our hypothetical example. We have described an implementation strategy of the solution, which dynamically evaluate the efficiency of the constructed portfolio and self-maintain it. In the future, we will investigate several sophisticated models of forecasting future risk of SLA violation using historical data.

REFERENCES

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, "Cloud Computing and Emerging IT Platforms:Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, pp. Volume 25, Issue 6, June 2009, Pages 599–616, 2009.

[2] Bev Littlewood, Peter Popov, and and Lorenzo Strigini, "Modeling software design diversity: a review," *Computing Surveys* , vol. 33, no. 2, pp. 177-208, June 2001.

[3] I. Gashi and P. T. Popov, "Rephrasing rules for off-the-shelf SQL database servers," in *Sixth European Dependable Computing Conference (EDCC)*, Coimbra, Portugal, 2006.

[4] Hassan B. Diab, Albert Y. Zomaya, and Lorenzo Strigini, "Fault Tolerance Against Design Faults," in *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*. Hoboken, NJ-USA: Wiley , 2005.

[5] Mark Harman, "The Current State and Future of Search Based Software Engineering," in *FOSE '07 Future of Software Engineering*, 2007.

[6] S. Yang, Y.S. Ong, and Y. Jin, "Evolutionary Computation in Dynamic and Uncertain Environments," *in the book series Studies in Computational Intelligence*, vol. 51, no. Springer-Verlag Berlin Heidelberg, 2007.

[7] Y. Jin and J. Branke, "Special Issue on Evolutionary Optimization in the Presence of Uncertainties," *IEEE Transactions on Evolutionary Computation*, vol. 10, 2006.

[8] H.M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*. New York: John Wiley & Sons, 1959.

[9] mathworks.co.uk, "Plotting an Efficient Frontier,".

[10] Vivek Nallur and Rami Bahsoon, "Design of a market-based mechanism for quality attribute tradeoff of services in the cloud," *In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10).*, pp. 367-371, 2010.

[11] Chris Brooks and Gita Persand, "Volatility forecasting for risk management," *Journal of Forecasting*, vol. 22, no. 1, pp. 1-22, 2003.

[12] Vivek Nallur, "A Decentralized Self-Adaption Mechanism For Service -Based Application In The Cloud," The University of Birmingham, 2012.

[13] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," *Journal ACM Transactions on Mathematical Software (TOMS)*, vol. Volume 4 Issue 1, no. ACM, pp. 34-50 , March 1978.

[14] Rodrigo Calheiros, Rajiv Ranjan, César DeRose, and Rajkumar Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," Melbourne,Australia, 2009.

[15] Tao Yu and Kwei-Jay Lin, "Service selection algorithms for Web services with end-to-end QoS constraints," *Information Systems and E-Business Management*, vol. 3, no. 2, pp. 103-126, 2005.

[16] M.Adel Serhani, Rachida Dssouli, Abdelhakim Hafid, and Houari Sahraoui, "A QoS broker based architecture for

efficient web services selection," in *IEEE International Conference on Web Services*, 2005.

[17] Michael Maximilien and Munindar P. Singh, "A framework and ontology for dynamic web services selection," *IEEE Internet Computing Magazine* , pp. 85-92, 2004.

[18] Rajkumar Buyya, Suraj Pandey, and Christian Vecchiola, "Cloudbus Toolkit for Market-Oriented Cloud Computing," in *Cloud Computing*. Berlin: Springer, 2009, pp. 24-44.

[19] Nikolay Borissov and Niklas Wirstrom, "Q-Strategy: A Bidding Strategy for Market-Based Allocation of Grid Services," *On the Move to Meaningful Internet Systems*, pp. 744-761, 2008.

[20] D. Grosu and A. Das, "Auctioning resources in Grids: model and protocols," *Concurrency and Computation: Practice and Experience*, pp. 1909–1927, 2006.

[21] Rich Wolsk, James S. Plank, John Brevik, and Todd Bryan, "Analyzing Market-Based Resource Allocation Strategies for the Computational Grid," *International Journal of High Performance Computing Applications*, vol. 15 , no. 3, pp. 258-281, Fall 2001.

[22] Mark Harman, Edmund Burke, John Clark, and Xin Yao, "Keynote : Dynamic Adaptive Search Based Software Engineering," in *6th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Lund, Sweden, 2012.

[23] P. Oreizy et al., "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems and their Applications* , vol. 14, no. 3, pp. 54 - 62 , 1999.

[24] J. Kim et al., "Immune system approaches to intrusion detection," *Natural Computing: an international journal*, vol. 6, no. 4, pp. 413 - 466 , 2007.

[25] S. Xanthakis, C. Karapoulios, R. Pajot, and A. Rozz, "Immune system and fault-tolerant computing," *Artificial Evolution -Lecture Notes in Computer Science-*, vol. 1063, pp. 181-197, 1996.

[26] Hélio R Costa, Márcio de Oliveira Barros, and Ana Regina Rocha, "Software Project Portfolio Selection A Modern Portfolio Theory Based Technique," in *f the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010)*, Redwood City, San Francisco Bay, CA, USA, 2010.

[27] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao, "Population-Based Algorithm Portfolios for Numerical Optimization," in *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* , Oct. 2010.

[28] Inkwon Hwang and Massoud Pedram, "Portfolio Theory-Based Resource Assignment in a Cloud Computing System," in *IEEE 5th International Cloud Computing (CLOUD)*, 2012.

[29] de Lemos; H. Giese; H. Müller; M. Shaw; J. Andersson;M. Litoiu; B. Schmerl; G. Tamura; N. Villegas; T. Vogel;D. Weyns; L. Baresi; B. Becker , "Software Engineering for Self-Adaptive Systems:A Second Research Roadmap," in *Software Engineering for Self-Adaptive Systems*. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[30] "The OpenNebula Project," retrieved at 16/13/2012.