International Journal of Software Engineering and Knowledge Engineering

CONTROLLING THE USABILITY OF WEB SERVICES

RON S. KENETT *

KPA Ltd., Raanana, Israel
ron@kpa.co.il

AVI HAREL

Ergolight Ltd, Haifa, Israel
ergolight@gmail.com

FABRIZIO RUGGERI

CNR IMATI, via Bassini 15, I-20133, Milano, Italy
fabrizio@mi.imati.cnr.it

Service Oriented Architectures (SOA) enable dynamic integration of Web Services (WS) to accomplish a user's need. As such, they are sensitive to user errors. This article presents a framework for mitigating the risks of user errors due to changes in the service delivery context. The underlying methodology incorporates usability in the design, testing, deployment and operation of dynamic collaborative WS, so that the error-prone elements of the User Interface (UI) are identified and eliminated. The methodology incorporates Statistical Process Control (SPC) of Web Service Indices (WSI), obtained by a Decision Support system for User Interface Design (DSUID), in which the users are elements of the control loop.

Keywords: Service Oriented Architectures; Web Services; Statistical Process Control, Bayes Estimate of the Current Mean.

1. Background

This article presents a method for assuring seamless navigation of visitors of dynamic websites. Typically, website designers focus on providing 'cool and neat' layout, employing newest widgets based on state-of-the-art technology. Many designers do not pay sufficient attention to the ways site visitors may use these features. Consequently, their designs suffer from various usability limitations, such as poor readability, error-prone navigation controls, and attention distracting features. Eventually, many visitors of such websites feel inconvenient about the usability barriers, and abandon the site before achieving their goals. Skilled designers, who are aware of the user's needs and limitations, typically do their best to facilitate the site navigation, but occasionally fail to anticipate the user's goals and behavior. To understand how to enable seamless navigation, website designers need to get diagnostic information about the particular sources of the negative user experience. Common practices for getting this information are by asking for the user's feedback and by questionnaires about the user's opinions. User's feedback and opinion are of low diagnostic value, because the users can inform about their feelings, but not about the sources of these feelings. A more appropriate method for understanding the sources of negative user experience is by usability testing. These tests are conducted by usability experts. Typically, usability tests are expensive and require delay in implementing the design changes. Another limitation of traditional usability tests is that their face validity is low; a large number of users is required for getting reliable, comprehensive conclusions. Therefore, in previous work we proposed the method of **Decision Support for User Interface Design** (DSUID) by statistical analysis of the user activity (see [5]).

Modern websites are dynamic. First, the site structure and content may change frequently. This is particularly true for eCommerce sites. Second, the server resource allocation changes according to on-line demands. This is particularly valid for **Service Oriented Architectures** (SOA). Common practices of usability testing cannot handle this kind of changes. On the other hand, Web analytics tools provide valuable information

* Corresponding author.

about the site visitors and the changes in the visitors' behavior, but they do not provide sufficient indication about the need and the means to enhance the design in terms or usability. The methods presented in this article are an attempt to solve the usability problem of dynamic websites by integrating within DSUID a **Statistical Process Control** (SPC) framework. The article is focused on SOA Web Services. We begin with an introduction to SOA, Usability and DSUID, section 2 covers aspects of usability control, section 3 is a detailed description of usability tracking models, section 4 treats usability diagnostics, section 5 is a detailed case study of a DSUID. We conclude with a section summarizing our results and pointing to future work.

1.1. *Service Oriented Architectures - SOA*

SOA is a framework for dynamic integration of **Web Services** (WS) provided by different vendors, to accomplish a user's operational goal. SOA involves providers and users meeting on a service composition platform and provides an open platform for integrating legacy, internal and external software components in the form of services in a uniform and reusable approach. Software development of SOA must enable flexible and dynamic service registration, discovery, matching, composition, binding, and reconfiguration.

SOA is considered a most promising computing paradigm for large-scale reuse, business agility support, and integration across heterogeneous environment ([1]). An emerging implementation of SOA on the web is the **Web Oriented Architecture** (WOA), based on XML encoded interfacing and communication standards.

1.2. *SOA Usability Risks*

When a service is offered, its features include various characteristics such as performance, reliability, security and usability levels. Usability is a term used to denote the ease and reliability with which people can employ a system, a tool or other human-made objects in order to achieve a particular goal.

Usability failures attributed to human errors, are typically a main factor of system failure. For example, 60-80% of aircraft accidents are attributed to human errors ([2]). Usability failures are not always identified correctly. For example, many calls for technical support about TV system malfunctioning turn out to be the result of user errors. In designing and controlling software systems, both software cybernetics and operational risk management are applicable.

Operational risk management is focused on the identification, classification and evaluation of risks from operational failures affecting systems and their stakeholders. For a comprehensive treatment of operational risks combining structured, quantitative data, with unstructured data and semantic analysis see [3].

Popular SOA based implementations, such as WOA, enable the users to choose from among a list of services and integrate them in a way that suits the user's goal. It is assumed that the service seekers follow certain operational procedures to describe what service they need, and make no mistakes in the requirement specification. However, this flexibility introduces special usability concerns and provides too many opportunities for the users to fail by setting the wrong sequencing or by disregarding mode constrains. Therefore, SOA based applications are particularly error-prone.

1.3. *Usability Assurance*

Usability assurance is a continuous effort spanning through the whole system life cycle: design, testing, deployment and routine operation:

- At the **design** stage, human factors are considered in a User Centered Design paradigm, to make sure that user interface enables proper interaction with the system.
- At the **testing** stage, usability experts conduct usability testing with real users, doing typical tasks. The tests enable system designers to learn about the quality of the user interfaces and to identify usability bottlenecks.
- At the **deployment** stage, diagnostic usability reports using special statistics obtained from logs of the user activity may reveal usability barriers of users, when doing real tasks in their real operational environment ([4]).
- At the **operation** stage, special service indices may reveal changes in the system usability, due to exposure to new user profiles and to changes in the services or their availability. The original operational scenario might not be predictive of the evolving demands. After a change has been traced, diagnostic usability reports may be derived for the period following the change, enabling exploration of required changes in the service provision.

This article focuses on the operation stage, namely, on methods for tracking the changes in service indices. The framework presented in the next sections incorporates usability considerations in a system implementation, so as to ensure fluent and reliable operation of collaborative web services. The underlying methodology enables adapting the **User Interface** (UI) to changes in the service allocation, so that error-prone elements of the UI are identified and eliminated.

1.4. *Usability Diagnosis*

The term Usability Diagnosis refers to identifying usability barriers, which prevent fluent and reliable system operation. At the testing stage it is used to identify potential usability barriers, typically, by observing users executing main operational tasks. At the deployment and operation stages it may be used to examine the actual user behavior, in order to identify actual usability barriers. From this perspective, aggregate diagnostic reports provide essential feedback to service designers and managers.

We expand the approach to deal with dynamic systems and indicate how it can be applied to general Service Oriented Architectures (SOA).

1.5. *Decision Support for User Interface Design - DSUID*

The term *Decision Support for User Interface Design* (DSUID) refers to a methodology introduced in [5] for ensuring system usability. The methodology is applicable to the last two stages of the system development cycle described above: the deployment and the operation stages. It employs a method for providing feedback to service managers about usability barriers, enabling them to adjust the resources and the interaction with the users. The feedback is based on special statistical analysis of the user activity, revealing problematic patterns of the user behavior ([6]).

Over time, systems are exposed to new user profiles. Also, the services may change, according to marketing demands. Additionally, services may be added or removed, and their availability may change by time of day, etc. The original operational scenario might not be predictive of the evolving demands and there is a need to adapt the UI to these changes.

In [5] and [6] we presented a methodology for static usability diagnosis based on a seven layer model. We expand it here to tracking dynamic changes in service context. Continuous usability tracking is required in order to manage the changing service conditions typical of SOA based implementations.

The paper presents a framework for identifying user errors due to changes in the service context, and for mitigating such risks. This is an introductory article, leaving out the technical details and more elaborate DSUID examples. It lays out the foundations for integrating usability in SOA systems and Web Services.

1.6. *Related Work*

This article expands the DSUID framework presented in [5] to account for effective online tracking of changes in usability over time. we introduce here a measure of the user experience enabling controlling website usability in real time. Several measures of the user experience have been proposed in the literature (e.g., [23], [25], [26]), but these measures typically depend on results from special purpose usability tests which are typically expensive and time consuming, and therefore inadequate for real-time control.

The method proposed here is based on an analysis of user's activity obtained from log files. An initial attempt to extract usability information from logs of the user activity was presented in [11]. The approach there was that the logs of the user's activity do not include information about the user's intention, which was assumed to be essential for usability studies. However, the recent work introduced in [5] showed that sources of negative user's experience can be deduced from logs of the user's activity, enabling automation of the usability control procedures. The next section presents the framework of usability control.

2. Usability Control

2.1. *Models of usability control*

Usability control is the process of capturing instances of significant changes in the quality of service, analyzing the sources for these changes and changing the UI to compensate for degradation in the service quality.

We introduce two basic usability control activities, tracking and corrective adjustment.

- Tracking: used to find out the points in time when the usability changes are significant and corrective adjustments may be required
- Corrective adjustment: used in order to compensate for these changes, is a two step process: diagnosis, to identify the changes that caused the usability change, and intervention of service managers, and possibly also of service designers.

2.2. *Tracking the Usability Changes*

Tracking of usability is based on **Web Service Indices** (WSI) reflecting the usage characteristics of the Web Services. A WSI is a usability score of the system or any of its components that we wish to control. We apply methods from cybernetics to trace these indices and to decide when a change in an index should denote a significant deviation that requires intervention. Specifically, we apply the Statistical Process Control (SPC) paradigm to control the service usability by tracking WSIs. Figure 1 depicts the method for usability control where detection triggers intervention with effect on usability characteristics.

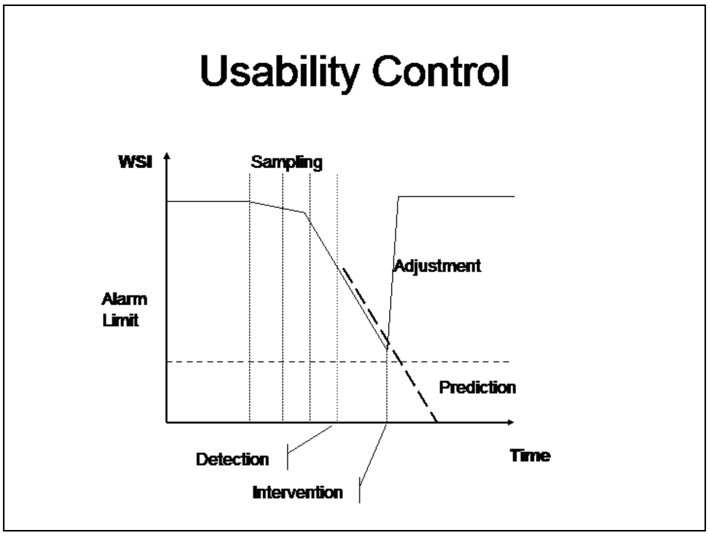

Figure 1: Usability control

SPC has been applied in the general context of software cybernetics, for example for controlling testing efforts (see [7]). In the context of tracking usability of web services, we use SPC to decide if a WSI behaves normally, by comparing actual values to reference values. Measuring the deviations from the references enables the system to identify undesired changes or drifts in the service indices. The reference may be user defined. For example, a user can define that the service reaction time should not exceed three seconds. The reference can also be set automatically according to prediction of future changes in the indices; for more on SPC see [8]. Section 3 presents an approach to usability control based on a Dynamic Linear Model (DLM) where a Bayesian Estimate of the Current Mean (BECM) is used to compute expected service indices. These tracking and change control methods are described in [8], [9] and [10].

Suppose that a service parameter needs to be changed due to usability issues. The procedure of parameter modification takes time, and process control should generate an expected event. The service control program continuously computes the predicted time for the WSI to exit the range of accepted values, and triggers the control accordingly. The predicted time may be computed by dividing the delta [Current WSI – WSI threshold] by the minimal time required for changing the parameter. The minimal time should be set by the site manager.

2.3. *Setting Web Service Indices*

A Web Service Index (WSI) is a measure of the user experience. It is closely related to the concept of 'service level' (http://en.wikipedia.org/wiki/Service_level). The main difference is in the type of service which, in case of web services, is data-related.

WSIs are defined at design time by service designers, and can be adjusted at operation time by the service managers. WSIs should reflect the risks of poor service. Therefore, they should measure the two main aspects of negative user experience: service performance and reliability. We can set distinct WSIs for service performance and service reliability, or we can set a shared index, for example, by combining separate indices. In the first option, we run two parallel usability controls: one for performance control and the other for reliability control. In the second option, we run a single usability control combining both aspects. In many cases a service level agreement is set up as a percentile, say 95%. This implies, for example, that 95% of the time the WSI should not exceed a pre specified limit.

Service reliability is a measure of user difficulties, often interpreted as user errors. Effective WSIs are calculated on the basis of **User Problem Indicators** (UPI). A UPI is a user event that suggests an instance of potential user difficulty. Typical UPIs are user actions reverting previous actions. Examples include: backward navigation, undo, cancel, selection from main menus or help requests. We can take a multivariate approach and set a distinct WSI for each UPI, and shared WSIs for service reliability control (For Multivariate Statistical Process Control techniques see [8]).

2.4. *Usability Diagnosis*

Once it is determined that a service usability level has significantly degraded, the SOA platform needs to be informed about the new risk. The alert message should include indication about possible sources for the degradation and recommendations for how to compensate for them.

2.5. *Corrective Adjustments*

Once a service barrier has been identified, special means should be taken to remove it. Most often, the diagnosis leads to further investigation. For example, if the system performance is poor, then the SOA platform needs to find out the causes for the poor performance. On the other hand, if the diagnostic reports show that a particular UI control is error-prone, a usability engineer should be asked to examine the conditions that enabled the user errors.

However, if the conditions that enable the user errors cannot be prevented by design, then a user-controlled service facilitator, such as a pop up dialog box, may be incorporated in the UI, enabling the users to select from a list of options.

3. Tracking Service Usability

3.1. *Applying Software Cybernetics*

As mentioned above, tracking service usability is based on WSIs. A WSI is a usability score of the system or any of its components that we wish to control. Cybernetics is the interdisciplinary study of the structure of complex systems, especially communication processes, control mechanisms and feedback principles. Specifically, it considers closed loops, where action by the system in an environment causes some changes in it. Such changes impact the way the system behaves, for example in order to achieve service goals ([27]).

Software cybernetics explores the interplay between software/software behavior and control ([7], [27]). Typical problems of software cybernetics include control of the software test process, software performance control and control of the software development process.

Typically, software cybernetics focuses on automatic control, disregarding the user operator. The new approach introduced here is to apply methods of software cybernetics to the process of UI adjustment.

Sequential methods may be employed to monitoring the user experience, by comparing actual results to expected results and acting on the gaps. This section demonstrates a way to control a WSI by SPC, implementing a Dynamic Linear Models, where the expected WSI is estimated by the Bayes Estimation of the Current Mean technique.

3.2. *Usability Tracking by SPC*

Statistical Process Control (SPC) is a method for identifying deviations from normal processing. Its origin is in manufacturing implementations, where statistical tools are used to observe the performance of production processes in order to identify and correct significant process deviations that may later result in rejected products ([8]). In this article we employ this approach for deciding when a deviation of a WSI from its expected value indicates a possible barrier to fluent and reliable interaction.

3.3. *A Dynamic Linear Model*

Dynamic models represent a key method in Bayesian forecasting and modeling. In particular, Dynamic Linear Models (DLM) with Gaussian distributed components have found a plethora of applications and played a central role in time series analysis. Often their use has been suggested to model the evolution of complex system over time with possible changes. Here the model is fitted to describe the evolution of various usability indexes and detect possible deviations from their behavior. In this sense, they can be used as a trigger to denote when an intervention is needed on the system to improve usability or to measure, on the positive side, when usability has improved. A typical DLM is represented by the following observation and evolution equations at time t=1, 2...

$$Y_t = F_t^{'}\vartheta_t + \varepsilon_t ,$$

$$\vartheta_t = G_t^{'}\vartheta_{t-1} + \eta_t ,$$

where Y_t is the observed quantity (e.g. a univariate or a multivariate usability index), ϑ_t is the state vector at time *t*, F_t is a known vector of regression variables and constants, ε_t is an observation noise term representing measurement error about Y_t, G_t is a known state evolution matrix modeling transition between states at times *t* and *t-1* and η_t is the evolution noise term (or innovation). Choosing Gaussian distributions for the noises ε_t and η_t, *t=1, 2 …* and a Gaussian prior distribution for ϑ_0, leads to a posterior Gaussian distribution for ϑ_s, given all the observations Y_t, *t=1… s.* Furthermore it is possible to compute the posterior predictive distribution of the next observation Y_{s+1}, given the past observations Y_t, *t=1… s.* We do not want to introduce here cumbersome mathematical notation but just illustrate a sound, widely used statistical approach. Mathematical details and precise results can be found in [9] and [10].

DLMs are essentially models which allow for the evolution of observed quantities (Y_t) in a complex manner depending on some unobserved state ϑ_t. Evolution equations are a sort of black-box with flexibility and adaptability to complex situations. The matrices F_t and G_t can be chosen after some fitting data from test cases. As usual in the Bayesian approach, priors, matrices and hyperparameters (the ones related to the distributions of the noises) are provided from past knowledge and experiences. The set up of a DLM can be approached in the following way:

1. Specify prior, matrices and hyperparameters at time *t=0*
2. Set *t=t+1* (and repeat until needed)
3. Forecast of observation Y_t at time *t*, based upon past history and prior specification
4. Observation of actual Y_t and comparison with forecasted one
5. If no significant difference occurs between observed and forecasted Y_t, then go to 2
6. Modify the model, e.g. the matrices F_t and G_t to fit the new observation, then go to 2.

Step 6 is a crucial one since it represents the point in which a change in the usability of the system is detected, and then the model is modified and used for future forecasts. If the change leads to a worse performance of the usability indexes, then the findings should be reported and actions should be taken.

3.4. *Computing the Expected WSIs*

In order to track usability changes, we need to continuously compute the service indices and to compare them to constraining values. The expected values should be estimated based on the history of the service indices. We propose a Bayesian Estimation of the Current Mean (BECM) for estimating the expected service indices.

Computing WSI is an arithmetical operation defined as:

WSI = sum over the site pages of [Weight(Page) * WPUI(Page)] divided by the number of pages where:

a) **Weight(Page)** is a score in the range (0-1) denoting the importance of successful navigation of Page. This score reflects the WS marketing needs. For example, pages in a critical path to order submission in eCommerce site should typically score high.

b) **WPUI (Page)**, is a score in the range (0-1) denoting the usability index of the Page. It can be defined as the product over several usability barriers to Page.

Observations from a continuous distribution characteristic of a process, such as a usability score, are recorded at discrete times i = 1, 2, · · ·. The observations, X_i, are determined by the value of a base average level, μ_0, and an unbiased error, ε_I. Possible system deviations at start-up are modeled by having μ_0 randomly determined according to a normal distribution with known mean μ_T (a specified target value) and a known variance σ^2. At some unknown point in time, a disturbance might change the process average level to a new value, $\mu_0 + Z$, where Z models a change from the current level.

The probability of having a disturbance between any two observations, taken at fixed operational time intervals, is considered a constant, p. Accordingly, we apply a model of at most one change point at an unknown location. The corresponding statistical model, for at most one change point, within n consecutive observations is as follows:

$$X_i = \mu_i + \varepsilon_i,\ i = 1, 2, \cdots, n,$$

where μ_i is the observation's mean and ε_i the measurement error, the at-most-one-change point model is:

$$\begin{aligned} \mu_i &= \mu_0, && i = 1, \cdots, j-1 \text{ and} \\ \mu_i &= \mu_0 + Z, && i = j, \cdots, n \end{aligned}$$

where j indicates that the disturbance occurred in the time interval between the (j−1)st and j-th observations. j = 1 indicates a disturbance before X_1, j = n + 1 indicates no disturbance among the first n observations.

The Bayesian framework is based on the following distributional assumptions:

$$\begin{aligned} &\mu_0 \sim N(\mu_T, \sigma^2) \\ &Z \sim N(\delta, \tau^2), \\ &\varepsilon_i \sim N(0, 1)\ i = 1, 2, \cdots. \end{aligned}$$

The parameters μ_T , σ^2, τ^2 are assumed to be known and N(·, ·) designates a random variable having a normal distribution. The Gaussian assumption about ε_i requires that the raw data be scaled, to have conditional variance equal to 1.

For n observations, let J_n denote the random variable corresponding to the index of the time interval at which a change occurred. It is assumed that the prior distribution of J_n is the truncated geometric, i.e.,

$$\begin{aligned} \Pr(J_n = j \mid p) &= p(1-p)^{j-1}, && j = 1, , n \\ &= (1-p)^n, && j = n+1 \\ & && 0 < p < 1 \end{aligned} \tag{1}$$

The assumption underlining (1) is that disturbances occur at random according to a homogeneous Poisson process. $p(1-p)^{j-1}$ is the probability that the change occurred between the j-1 and jth observation and $(1-p)^n$ is the probability that the change has not happen up to the n^{th} observation. Finally, we assume that μ_0, Z, J_n and ε_i are independent, for each n. The current mean μ_n can be written as

$$\mu_n = \mu_0 + I\{J_n < n\}Z,$$

where I{·} is an indicator variable, assuming the value 1 if the relation in brackets is true, and the value 0 otherwise.

The procedure described in [8] and [9] computes analytically two statistics of interest to SOA Web Services:

- The distribution of Ji, namely, the likelihood that a shift in distribution occurred in any time interval up to the current observation.
- The distribution of μn, for example, percentiles of the distribution of the current mean.

3.5. *An Example*

We demonstrate the procedure with a simulation of 15 normally distributed observations, with an induced change point at J = 10 with parameters $\mu_T = 0$, $\delta = 1.5$, $\sigma^2 = \tau^2 = 1$. Such simulations are available in any statistical package. The random values of μ_i (i = 1. . . 15) are:

$$\mu_i = -1.91523, \qquad i \leq 10 \text{ and}$$

$$\mu_i = 0.53421, \qquad i \geq 11$$

We therefore observe data with a shift in mean of 2.449 units, after the 10th observation. Table 1 lists the posterior distributions of J_i for i = 7,… 13, that is before and after the shift. The maximal probabilities are indicated in bold and point to an estimate of where the shift has occurred. The probability of shift in the next interval represents the probability of a future shift.

Table1. Probability of a shift at any given time interval, at the ith observation

i	7	8	9	10	11	12	13
Xi	-2.081	-2.837	-0.731	-1.683	1.289	0.5650	0.2560
1	0.0027	0.0028	0.0023	0.0025	0.	0.	0.
2	0.0057	0.0066	0.0037	0.0037	0.	0.	0.
3	0.0022	0.0023	0.0016	0.0016	0.	0.	0.
4	0.0072	0.0085	0.0029	0.0026	0.	0.	0.
5	0.0050	0.0054	0.0019	0.0017	0.	0.	0.
6	0.0047	0.0043	0.0017	0.0016	0.	0.	0.
7	0.0194	0.0039	0.0051	0.0055	0.0004	0.0001	0.
8	**0.9606**	0.0079	0.0078	0.0074	0.0011	0.0003	0.0002
9	-	**0.9583**	0.1583	0.0643	0.0964	0.0794	0.0927
10	-	-	**0.8149**	0.0139	0.0155	0.0149	0.0184
11	-	-	-	**0.8952**	**0.8833**	**0.9052**	**0.8886**
12	-	-	-	-	0.0032	0.0001	0.0001
13	-	-	-	-	-	0.	0.
14	-	-	-	-	-	-	0.

Up to the 10th observation interval, a shift is indicated as potentially occurring in the future. After the 11th observation the maximal probability of a shift is consistently pointing at the interval between observation 10 and 11.

Table 2: percentiles of the distribution of the mean at any given time interval, at the ith observation

i	7	8	9	10	11	12	13
Xi	-2.081	-2.837	-0.731	-1.683	1.289	0.5650	0.2560
$\mu_{.01}$	-2.937	-2.979	-2.746	-2.697	-1.789	-1.044	-0.911
$\mu_{.25}$	-2.328	-2.401	-2.193	-2.181	-.0587	-0.101	-0.094
$\mu_{.50}$	-2.085	-2.172	-1.947	-1.966	-0.113	+0.292	0.243
$\mu_{.75}$	-1.839	-1.941	-1.627	-1.735	+0.368	0.690	0.584
$\mu_{.99}$	-0.831	-1.233	0.872	-0.263	1.581	1.668	1.425
μ_i	-2.069	-2.161	-1.731	-1.918	-0.108	0.296	0.246

Table 2 shows the percentiles. As expected, the procedure shows a decrease in the mean at the 11th observation. If the data tracked is a WSI, and we set an upper limit of 1 to the 99th percentile of the index, we see that usability has deteriorated at observation 11th so that some reactive risk mitigation procedure needs to be activated.

4. Usability Diagnostics

Once it is determined that a service has significantly degraded, we need to inform the SOA platform about the problem and its possible causes. The resulting report should include indication about possible sources for the degradation and recommendations for how to compensate for them.

Methods for diagnosis include requesting the users for feedback and statistical analysis. However, a most effective method is by a combination of the former methods, namely, by requesting for the users' feedback at specific interaction points, which by statistical usability analysis were identified as error-prone. By prompting the users to relate to particular interaction points, the SOA platform can get important information about the users' intention. Then, it can compare the activity log to the users' intentions and figure out what were the reasons for the user error. This approach was first introduced in [11]. For more on mapping cause and effect see [12]. This section presents a method for identifying points of negative user experience by statistical analysis of logs of users' activity. This method was described in detail in [6].

4.1. *Modelling the User Activity*

The user activity can be described using a Bayesian Markov model with states given by screen displays or web pages. Two absorbing states are present (fulfillment of the task and unsuccessful exit) and our interest is in their probability. Transitions between one state and all the possible others are modeled as a multinomial distribution with probability p_{ij} of going from state i to state j. A Dirichlet distribution is a natural, conjugate choice for the vector $\mathbf{p}_i = (p_{i1} \ldots p_{ik})$, denoting the probabilities of all possible transitions from state i. The choice of the hyper parameters is a tough, although typical, problem in Bayesian analysis, where the statistician has to transform experts' opinions into numbers. Web log provides the number of transitions from state i to state j and data can be combined with the Dirichlet prior via Bayes' theorem, so that a posterior distribution, again a Dirichlet one, is obtained. Based on the posterior distribution on the transition probabilities, operational reliability can therefore be analyzed looking at the predictive probability of ending in the unsuccessful state, given a pre-specified pattern, i.e. a sequence of transitions between web pages. With such techniques, one can combine in a DSUID, expert opinions with data to model transitions between states and describe cause and effect relationships. For an interesting approach to improve usability by providing on line help with procedures based on past experience and modeling the transition between states see [13] and [14]. For an introduction to Markov chains see [15].

5. A Decision Support User Interface Diagnostics (DSUID) Case Study

Logs of users' activity typically provide time stamps for all user actions. In addition, when web server log files are available, they also include traces of image files and scripts used for the page display. The time stamps of the additional files enable us to estimate three important time intervals：

- The time the visitors wait until the beginning of the file download, is used as a measure of page responsiveness
- The download time, is used as a measure of page performance
- The time from download completion to the visitor's request for the next page, in which the visitor reads the page content, but also does other things, some of them unrelated to the page content.

The challenge is to decide, based on statistics of these time intervals, whether the visitors feel comfortable during the site navigation; when do they feel that they wait too much for the page download and how do they feel about what they see on screen. In this section we provide details of a DSUID system with specific examples.

5.1. *Statistical Time Analysis*

To understand the user experience we need to know the user activity, compared to the user expectation. Both are not available from the server log file, but can be estimated by appropriate processing. A Decision Support User Interface Diagnostics (DSUID) system flagging possible usability design deficiencies requires a model of statistical manipulations of server log data.

Assume now that we have a log of the user's activity. How can we tell whether the service users encounter any difficulty in exploring a particular screen display or a web page, and if so, what kind of difficulty do they experience, and what are the causes for this experience?

We assume that the site visitors are task driven, but we do not know if the visitors' goals are related to a specific service. Also, we have no way to tell if visitors know anything, a priori, about the service, and if they believe that the service is relevant to their goals, or if they have used it before. It may be that the users are simply exploring the service screens, or that they follow a procedure to accomplish a task. Yet, their behavior reflects their perceptions of the service, and estimates of their effort in subsequent investigation.

How can we conclude that a time interval is acceptable by page visitors, is too short, or is too long? For example, consider an average page download time of five seconds. Site visitors may regard it as too lengthy, if they expect the page to load fast, for example, in response to a search request. However, five seconds may be quite acceptable if the user's goal is to learn or explore specific information, if they expect it to be related to their goal. Setting up a DSUID involves integration of two types of information:

- Design deficiencies, which are common barriers to seamless navigation, based on the first part of the model described above – visitor's page evaluation
- Detectors of these design deficiencies, common indicators of possible barriers to seamless navigation, based on the second part of the model – visitor's reaction.

The way to conclude that the download time of a particular page is too long is by a measure of potentially negative user experience, namely, the site exit rate. The diagnostic-oriented time analysis is by observing the correlation between the page download time and the page exits. If the visitors are indifferent about the download time, then the page exit rate should be invariant with respect to the page download time. However, if the download time matters, then the page exit rate should depend on the page download time. When the page download time is acceptable, most visitors may stay in the site, looking for additional information. However, when the download time is too long, more visitors might abandon the site, and go to the competitors. The longer the download time, the higher is the exit rate.

5.2. *Usability Problem Indicators - UPI*

Exit rate is a preferred measure for deciding on the effect of download time, but is irrelevant to the analysis of the visitors' response time. The reason for it is that server logs do not record events of the visitor leaving the site, which means that we cannot measure the time intervals of terminal page views. Therefore, we need to use other indicators of potential negative navigation experience.

A model of user's mental activities in website navigation lists the most likely visitors' reaction to exceptional situations. Based on this model, we can list the following indicators about the visitor's tolerance to web page design deficiencies:

- The visitor returning to the previous page may indicate that the current page was perceived as less relevant to the goal, compared to the previous page .
- The visitor linking to a next website page may indicate that the link was perceived as a potential bridge to a goal page
- The visitor activating a main menu may indicate that the visitor is still looking for the information, after not finding it in the current page
- The visitor exiting the site may indicate that either the goal has been reached, or the overall site navigation experience became negative.

Accordingly, besides site exit, other indicators of potential usability problem are the rates of navigation back to a previous page and the visitor escaping from the page to a main menu.

5.3. *Page Relevance*

Once we have an estimate of the task-related mental activities we can adapt the method for deciding about problematic performance, to decide about problematic task-related mental activities. Visitors who feel that the information is irrelevant to their needs are more likely to respond quickly, and are more likely to go backward, or to select a new page from the main menu. Therefore, the average time on page over those visitors who navigated backwards or retried the main menu should be shorter than that of the average of all visitors. On the other hand, visitors who believe that the information is relevant to their needs, but do not understand the page text very easily, are likely to spend more time than the average reading the page content, and the average time on page should be longer.

5.4. *Page Readability*

Page usability is a compound property, consisting of several factors. One factor, used for demonstrating the method, is the page readability. We demonstrate here the method for computing the confidence level of the actual page readability. The model consists of one independent variable, which is the page readability, and two dependent variables: the user's reaction time and the user's next action. The model assumes that:

1. The user's reaction time is correlated with the reading difficulties
2. The rate of UPIs, such as backward navigation, is correlated with the reading difficulties.

Based on these assumptions, we expect that on average, the user's reaction time of visits followed by a UPI will be higher than that of visits followed by another user action. Therefore, we apply a t-test for comparing the user's reaction time of two samples: one is of visits followed by a UPI and the other is of visits followed by another user action. The page readability is determined by a t-test comparing the two samples.

The time that visitors spend reading a page depends on various perceptual attributes, including the page relevance to their goals, the ease of reading and comprehending the information on page, the ease of identifying desired hyperlinks, etc. Assume that for a particular page, the average time on screen before backward navigation is significantly longer than the average time over all page visits. Such case may indicate a readability problem, due to design flaws, but it can also be due to good page content, which encouraged users who spent long time reading the page to go back and reexamine the previous page.

5.5. *Validating the Model of User Mental Activities*

To illustrate the approach we use www.israeliz.co.il log data extracted with the Ergolight Webtester from 2003-02-01, 12:37:46 to 2003-02-27, 11:22:28.
(www.ergolight-sw.com/pub/Sites/Israeliz/All/First/Reports/wtFrameSet.html.

A typical scheme for page event sequences is:

User links to Page -> Page starts download -> Page download complete -> User recognizes page -> User reads page -> User activates a link, about to leave the page.

The main variables of interest in a typical eCommerce DSUID are presented in Table 3. Variables marked in bold represent key variables to be analyzed in greater depth.

The time variables with potential impact on the visit experience are:

- The page download time, measured from the first page view until the end of subsequent embedded files (images, Java scripts, etc)
- The elapsed time until the next user event, interpreted as reading time.
- The elapsed time since the previous user event, interpreted as the time until the user found the link to this page.

The page visitor experience is regarded as negative or positive according to whether or not a UPI was indicated. The expected relationships between the key variables are as follows:

- Page size should directly affect the page download time
- The amount of text on page should directly affect the page reading time
- The ratio TextSize/PageSize may be equal to 1, if the page is pure html (has no graphics, video etc.) or very small (since images consume much more space compared to text)
- Page download time should affect the visit experience. Too long download time should result in negative visit experience. Research indicates that too short download time might also result in negative visit experience; however, this might be a problem only for extremely good service (server performance).
- Page seeking time may affect the visit experience. Too much seeking time might have a 'last straw' effect, if the page is not what they expected to have.
- Page reading time may have effect on the visit experience. Users may stay on page because the page includes lot of data (in this case study, the data is textual) or because the text they read is not easy to comprehend.

Table 3: Variables used in DSUID analysis

ID	***Variable***	***Definition***
1	ActionCount	"Count of page visits as recorded in the log file, including actual visits and faked visits (refreshing and form submission)"
2	NuOfRealVisits	"The actual visits, after removing the faked visits."
3	**TextSize**	Size of html page
4	**PageSize**	TextSize + size of subsequent images recorded individually
5	NuOfEntries	Count of site entries through this page
6	NuOfPreExits	Count of site exit through next page
7	NuOfExits	Count of site exit through this page
8	NuOfPreBacks	Count of backward navigations from this page
9	NuOfBacks	Count of backward navigations to this page
10	NuOfDownloads	Count of page visits with positive download time
11	NuOfGetToPost	"Count of transitions from Get (typically, viewing) to Post (typically, form submission)"
12	TotalDownloadFrequency	Sum of (1 / DownloadTime) over all real page visits
13	TotalReadingFrequency	Sum of (1 / ReadingTime) over all real page visits
14	TotalSeekingFrequency	Sum of (1 / SeekingTime) over all real page visits
15	TotalProcessingFrequency	Sum of (1 / ProcessingTime) over all real page visits
16	TotalGetToPostFrequency	Sum of (1 / FormFillingTime) over all real page visits
17	AvgDownloadFrequency	Average = Total / Count
18	AvgReadingFrequency	Average = Total / Count
19	AvgSeekingFrequency	Average = Total / Count
20	AvgGetToPostFrequency	Average = Total / Count
21	AvgPostToPostFrequency	Average = Total / Count
22	**AvgDownloadTime**	Harmonic average of page download time over all real page visits
23	**AvgReadingTime**	Harmonic average of page reading time over all real page visits
24	**AvgSeekingTime**	Harmonic average of page seeking time over all real page visits
25	AvgProcessingTime	Harmonic average of page processing time over all real page visits
26	AvgResponseTime	Harmonic average of page response time over all real page visits
27	PercentExitsSlowDownload	"Number of page visits characterized as slow downloads, followed by site exits / NuOfRealVisits * 100"
28	PercentPbackSlowDownload	"Number of page visits characterized as slow downloads, followed by backward navigation / NuOfRealVisits * 100"
29	PercentExitsSlowSeeking	"Number of page visits characterized as slow seeking, followed by site exits / NuOfRealVisits * 100"
30	PercentExitsFastSeeking	"Number of page visits characterized as fast seeking, followed by site exits / NuOfRealVisits * 100"
31	PercentPbackSlowSeeking	"Number of page visits characterized as slow seeking, followed by backward navigation / NuOfRealVisits * 100"
32	PercentPbackFastSeeking	"Number of page visits characterized as fast seeking, followed by backward navigation / NuOfRealVisits * 100"
33	PercentPexitSlowReading	"Number of page visits characterized as slow reading, followed by site exit from next page / NuOfRealVisits * 100"
34	PercentPexitFastReading	"Number of page visits characterized as fast reading, followed by site exit from next page / NuOfRealVisits * 100"
35	PercentPbackSlowReading	"Number of page visits characterized as slow reading, followed by backward navigation / NuOfRealVisits * 100"
36	PercentPbackFastReading	"Number of page visits characterized as fast reading, followed by backward navigation / NuOfRealVisits * 100"
37	**UsabilityScore**	"Normalized harmonic average of: download time, seeking time, reading time"
38	**UsabilityAlert**	Sum of all UPIs (6-11 above)
39	**AccessibilityScore**	a measure of the speed of finding the links to the current page
40	**PerformanceScore**	a measure of the download speed
41	**ReadabilityScore**	a measure of the reading speed (characters per second).

Simple histograms of key variables tracking 122 page visits are presented in Figure 2. The data represents averages per visit. Some distributions are flat, some are skewed to the right and some have obvious outliers.

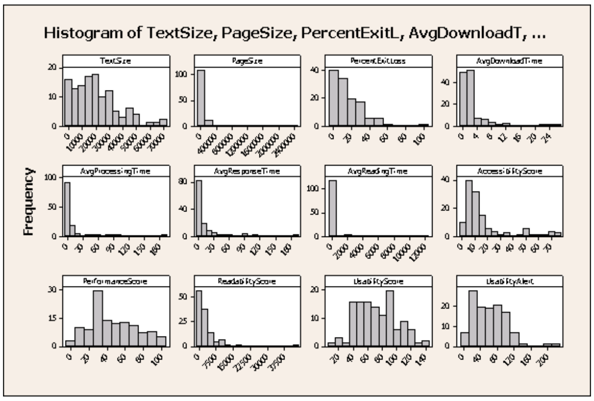

Figure 2: Histogram of DSUID key variables.

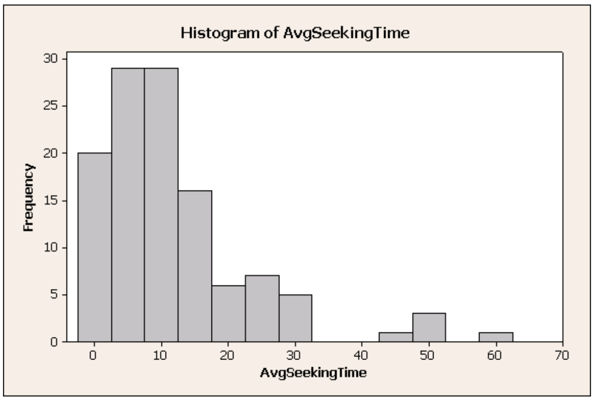

Figure 3: Distribution of average seeking time.

Figure 3 focuses on the average seeking time distribution, without displaying an outlier of 5624 seconds and two relatively high observations of 80 and 74 seconds. By running a basic factor analysis of the data we can see that average seeking time and average reading time carry very similar loadings and are therefore positively correlated. They are, however, negatively correlated to text size and accessibility scores. So are the Usability and Performance scores that bring out an additional dimension reflected by the relatively high position on the second factor (see Figure 4).

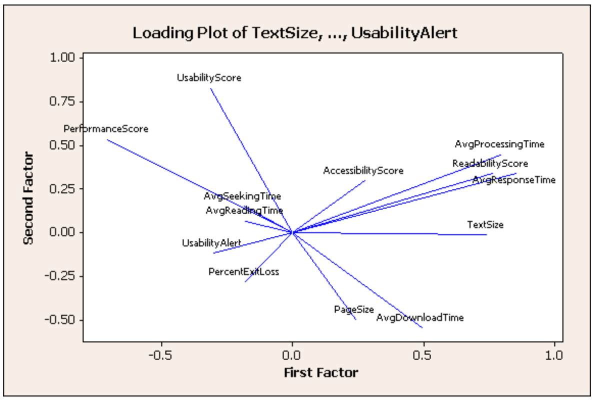

Figure 4: Loadings of DSUID key variables.

5.6. *Cause and Effect Analysis*

To validate the model of the user's mental activity we use **Bayesian Networks**. A Bayesian Network is a graphical model representing cause and effect relationships between variables. Each node in the graph represents a random variable, whereas edges denote probabilistic relations between the random variables (for a general introduction see [12] and [16]). Bayesian networks are used for dealing with complex problems since they provide a model of cause and effect relationships at a *local* level, involving just few random variables, identifying *parents* nodes which are affecting *children* ones, as specified by conditional probabilities. A key characteristic of Bayesian Networks is that they can be continuously updated turning prior information into posterior distributions that reflect the propagation of variability through the network structure (see [16]). This property is coherent with a Bayesian approach in which experts' knowledge and experimental evidence are combined to make inferences on quantities of interest. For example, for computing the page readability confidence level we assumed that rate of UPIs is correlated with the user's reaction time. We validate of this assumption using Bayesian Networks.

As an example, we show in Figures 5 and 6 a Bayesian Network derived from analysis of our case study weblogs. The arrows in the network indicate cause and effect relationships between variables that are derived from an analysis of conditional probabilities. In general, such relationships can be determined by combining data driven self learning with expert knowledge. The variables have been discretized into five categories with uniform frequencies. The network can be then be used to represent posterior probabilities, reflecting the network conditioning structure, of the 1-5 categorized variables. We can use such network for predicting posterior probabilities after conditioning on variables affecting others or, in a diagnostic capacity, by conditioning on end result variables. In our example, we condition the network to low and high seek time, Figures 5 and 6 respectively.

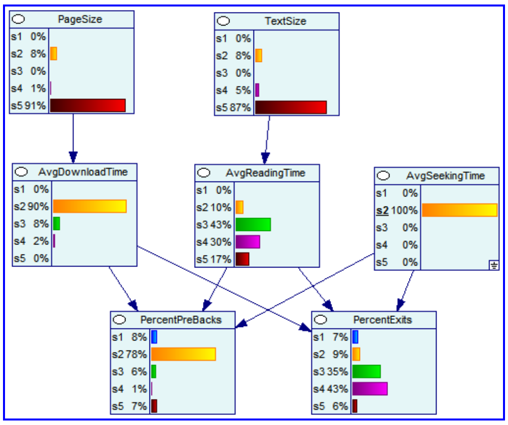

Figure 5: A Bayesian network of web log data, conditioned on low seek time

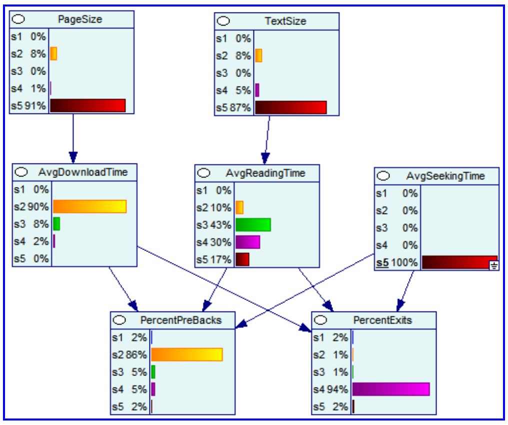

Figure 6: A Bayesian network of web log data, conditioned on high seek time

With low seeking time we experience 49% of high and very high exit rates. With high seeking time these numbers jump to 96%. Clearly seeking time has affected the behavior of the users. We can even quantify this phenomenon and link, quantitatively, the impact of a second of seek time on exit rate percentages.

5.7. *Statistical Decisions*

The way to conclude that the download time of a particular page is too long is by a measure of potentially negative user experience, namely, the site exit rate. To enable statistical analysis, we need to change our perspective on these variables, and consider how the download time depends on the exit behavior. We compare the download time of those visits that were successful, with that of those visits that ended up in the site exit. If download time matters, we should expect that the average download time of those visitors who abandoned the site will be longer than the average of those who continued with the site navigation. Otherwise, if the site visitors are indifferent about the download time, we should expect that the download time of the two groups would not be significantly different.

To decide about the significance of the usability barrier we compare the download time of two samples: one of page visits that ended up in site exit and the other of all the other page visits. The null hypothesis is that the two samples are of the same population. If the null hypothesis is rejected, we may conclude that the page visitors' behavior depends on the download time: if the download time of the first sample exceeds that of the second sample, and the difference is statistically significant, then we may conclude that the download time of the particular page is significantly too long. A simple two-tailed t-test is sufficient to decide whether the visitors' behavior is sensitive to the page download time.

5.8. *A DSUID Implementation Framework: the Seven Layers Model for Usability Diagnosis*

In setting up a Decision Support system for User Interface Design we identify seven layers of data collection and data analytics:

5.8.1. *The lowest layer – user activity*

This layer records correspond to significant user actions (involving screen changes or server-side processing).

5.8.2. *The second layer – page hit attributes*

This layer consists of download time, processing time and user response time.

5.8.3. *The third layer – transition analysis*

The third layer statistics are about transitions and repeated form submission (indicative of visitors' difficulties in form filling)

5.8.4. *The fourth layer – User Problem Indicator identification*

Indicators of possible navigational difficulty, including:

- Estimates for site exit by the time elapsed until the next user action (as no exit indication is recorded on the server log file)
- Backward navigation
- Transitions to main pages, interpreted as escaping current sub task.

5.8.5. *The fifth layer – usage statistics*

The fifth layer consists of usage statistics, such as:

- Average introduction time
- Average download time
- Average time between repeated form submission
- Average time on screen (indicating content related behavior)

- Average time on a previous screen (indicating ease of link finding).

5.8.6. *The sixth layer – statistical decision*

For each of the page attributes, DSUID compares the statistics over the exceptional page views to those over all the page views. The null hypothesis is that (for each attribute) the statistics of both samples are the same. A simple two-tailed t test can be used to reject it, and therefore to conclude that certain page attributes are potentially problematic. A typical error level is set to 5%.

5.8.7. *The seventh layer – interpretation*

For each of the page attributes DSUID provides a list of possible reasons for the difference between the statistics over the exceptional navigation patterns and that over all the page hits. However, it is the role of the usability analyst to decide which of the potential source of visitors' difficulties is applicable to the particular deficiency.

6. Summary and Future Work

Currently, website owners and designers still do not have proper means to learn the user’s experience in real time. Web Analytics tools, such as Google Analytics, provide valuable information to site owners, but these tools do not provide data to conclude about the need for intervention.

Usability is gaining attention. New books are being published ([23], [24]) and recent workshops by UPA and others are now accessible on the web ([25], [26]). For a general presentation on challenges in software cybernetics and dynamic feedback control see [27]. In this work we describe a way to integrate usability quantification in SOA and WS. It is related to the concept of trust presented in [17], needs to be adaptive, context sensitive and situation aware like other SOA application characteristics (see [18] and [19]). Our work was focused on expanding this effort to cover wider and deeper aspects of usability. Specifically we show how an SPC model can be used to monitor, collect and interpret usability data. We discuss several analytical models, including Markov chains, Bayesian Networks, Dynamic Linear Model and the BECM procedure of [9] to analyze such data in a Bayesian framework.

Standard usability analysis is based on eliciting the opinion of experts, for example on possible user’s reactions when navigating through web sites (see [20]). We introduce here a quantitative approach that allows combining expert opinion's prior assessment of the probability of failure/success when some patterns are followed by the users with dynamically collected data. For a similar approach to set up recommendation algorithms in web services see [21] and [22]. The Bayesian framework we introduce here provides for efficient and practical tracking and correction of usability aspects in Service Oriented Architecture systems in general. This approach obviously needs to be expanded and further studied in specific software cybernetic applications. In particular, the WSI defined here should be calculated for sequences of log files, to learn how this index changes in real dynamic situations, such as in peak hours and during update of a website database.

Our intention was to focus on methodological aspects. Obviously tools are needed to support such implementations. In [6] we mention tools such as Webtester from Ergolight and GeNie for running Bayesian Networks. Providing more information on such tools has been beyond the scope of this paper. It is reasonable to propose that new tools should be designed, enabling real time validation and control of the user’s experience. Such tools may use a WSI for the process control, in which the WSI is obtained from usability factors that may be calculated in future Web Analytics tools.

In future studies, it may be possible to install a JavaScript in web pages, similar to those installed today in many web analytic tools. Such JavaScript will enable better distinction between users, and will enable enhancing the indication of site exits in order to achieve improved seamless navigation in dynamic websites.

7. References

[1] X. Bai, S. Lee, W. Tsai and Y. Chen Y., Collaborative Web Services Monitoring with Active Service Broker, in *Proc. Computer Software and Applications Conference (COMPSAC'08)*, 2008, pp. 84—91.

[2] S. Shappell and D. Wiegmann, U.S. Naval aviation mishaps 1977-1992, *Aviation, Space, and Environmental Medicine* **67**, 1996, pp. 65—69.

[3] MUSING, (MUlti-industry, Semantic-based next generation business INtelliGence) http://www.musing.eu, 2006.

[4] Nielsen, J. *Usability Engineering*, Academic Press Inc, 1994, p 165.

[5] A. Harel, R.S. Kenett and F. Ruggeri, Decision support for user interface design: usability diagnosis by time analysis of the user activity, in *Proc. Computer Software and Applications Conference (COMPSAC'08)*, 2008, pp.836—840.

[6] A. Harel, R.S. Kenett and F. Ruggeri, Modeling Web Usability Diagnostics on the basis of Usage Statistics, in *Statistical Methods in eCommerce Research*, ed. W. Jank and G. Shmueli, Wiley, 2008, pp. 131—172.

[7] J.W. Cangussu, S.D. Miller, S. D, K. Cai, and A.P. Mathur, Software Cybernetics, Encyclopedia of Computer Science and Engineering, John Wiley & Sons, Vol. 4, pages 2575 - 2583, January 2009

[8] 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 edition 2003 - Chinese edition, 2004.

[9] S. Zacks, and R.S. Kenett, Process tracking of time series with change points, in *Recent Advances in Statistics and Probability (Proc. 4th IMSIBAC)*, ed. J.P. Vilaplana and M.L. Puri, VSP International Science Publishers, Utrecht, 1994, pp. 155—171.

[10] M. West and P.J. Harrison, *Bayesian Forecasting and Dynamic Models*, Springer-Verlag, New York, 1989.

[11] A. Harel, Automatic Operation Logging and Usability Validation, in *Proc. HCI International '99 (the 8th International Conference on Human-Computer Interaction)*, Munich, Germany, 1999.

[12] R.S. Kenett, Cause and Effect Diagrams, in *Encyclopedia of Statistics in Quality and Reliability*, ed. F. Ruggeri, F., R.S. Kenett and F. Faltin, Wiley, 2007.

[13] C. Robert, *The Bayesian Choice: from Decision-Theoretic Motivations to Computational Implementation*, Springer-Verlag, New York, 2001.

[14] D. Rios Insua and F. Ruggeri, ed., *Robust Bayesian Analysis*, Springer-Verlag, New York, 2000.

[15] S. Karlin and H.M. Taylor, *A First Course in Stochastic Processes (2nd Ed.)*, Academic Press, New York, 1975.

[16] I. Ben Gal, Bayesian Networks, in *Encyclopedia of Statistics in Quality and Reliability*, ed. F. Ruggeri, F., R.S. Kenett and F. Faltin, Wiley, 2007.

[17] L. Tan, C. Chi and J. Deng, Quantifying Trust Based on Service Level Agreement for Software as a Service, in *Proc. Computer Software and Applications Conference (COMPSAC'08)*, 2008, pp.116—119.

[18] K. Oyama, C. Chang, H. Jaygarl, A. Takeuchi, J. Xia, and H. Fujimoto, Requirements Analysis Using Feedback from Context Awareness Systems, in *Proc. Computer Software and Applications Conference (COMPSAC'08)*, 2008, pp.625—630.

[19] S.S. Yau and J. Liu, A Situation-aware Access Control based Privacy-Preserving Service Matchmaking Approach for Service-Oriented Architecture, in *Proc. International Conference on Web Services (ICWS)*, 2007, pp.1056—1063.

[20] International Standards Organization, ISO 9241-11, Guidance on Usability, 1998.

[21] P. Geczy, N. Izumi, S. Akaho and K. Hasida, Behaviorally Founded Recommendation Algorithm for Browsing Assistance Systems, *Annals of Information Systems*, 2008 (in print).

[22] N. Izumi, P. Geczy, S. Akaho and K. Hasida, Browsing Assistance Service for Intranet Information systems, in *Advances in Data Mining: Medial Applications, E-Commerce, Marketing, and Theoretical Aspects, ICDM 2008*, ed. P. Perner, Leipzig, Germany (Lecture Notes in Computer Science, Springer Verlag, Volume 5077), 2008.

[23] T. Tullis and B. Albert, *Measuring the User Experience: Collecting Analyzing, and Presenting Usability Metrics*, Morgan Kaufmann, 2008

[24] B. Albert, T. Tullis and D. Tedesco, *Online Usability Testing: Improving the User Experience through Automated Studies.* Elsevier/ Morgan Kaufmann (2010)

[25] J. Sauro, www.measuringusability.com

[26] UPA 2009, Portland OR, http://www.usabilityprofessionals.org/conference/2009/

[27] A.P. Mathur, "Software Cybernetics: Progress and Challenges" presentations given on December 20, 2004 at the University of Paderborn, Paderborn, Germany. http://www.cs.purdue.edu/homes/apm/talks/Paderborn-talk.ppt