SaaS Usage Information for Requirements Maintenance

Ana Garcia[2], Ana C. R. Paiva[1,2]
[1]INESC TEC, [2]Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias, Porto, Portugal
{meg11033, apaiva}@fe.up.pt

Keywords: Requirements Engineering, Requirements evolutions, Software as a Service lifetime, Usability.

Abstract: The incorrect requirements elicitation, requirements changes and evolution during the project lifetime are the main causes pointed out for the failure of software projects. The requirements in the context of Software as a Service are in constant change and evolution which makes even more critical the attention given to Requirements Engineering (RE). The dynamic context evolution due to new stakeholders needs brings additional challenges to the RE such as the need to review the prioritization of requirements and manage their changes related to their baseline. It is important to apply methodologies and techniques for requirements change management to allow a flexible development of SaaS and to ensure their timely adaptation to change. However, the existing techniques and solutions can take a long time to be implemented so that they become ineffective. In this work, a new methodology to manage functional requirements is proposed. This new methodology is based on collecting and analysis of information about the usage of the service to extract pages visited, execution traces and functionalities more used. The analysis performed will allow review the existing requirements, propose recommendations based on quality concerns and improve service usability with the ultimate goal of increasing the software lifetime.

1 INTRODUCTION

The internet environment where Software as a Service (SaaS) are located, is more dynamic complex and unpredictable, exposing them continuously and, therefore, making them more susceptible to the high change pace. Also, the quick evolution of technology, the new politics and laws, the existing similar competitive services, quality concerns and the constant change of stakeholders needs are some factors that determine the SaaS environment changeability. To face this evolution and keep answering to the stakeholders needs, it has to adapt the service concept timely. It may require the addition of new functionalities or update existing ones. The functionalities of a service result from the implementation of functional requirements that can be defined as actions performed by a system, without considering its physical constraints (Qureshi & Perini 2010). Therefore Requirements Engineering (RE) is of utmost importance to manage and maintain requirements during SaaS lifetime. In 2009, according to The Chaos report, the number of software projects well success (projects that were timely finished and within the budget, with all the functionalities initially planned implemented) round only 16% (Dominguez 2009). The main causes pointed to the software failures are the incorrect requirements elicitation, requirements changes and their evolution during the project lifetime. The contribution of RE to overcome these problems is the specification and planning requirements. Furthermore, it allows evaluating them to identify inherent risks of their design. Uncontrolled requirement changes cause negative impacts in software development, like for example, costs over budget and a system that is not able to respond to the needs of its stakeholders (Ibrahim et al. 2009). This situation can lead to the misuse of the SaaS and consequently to the loose of the associated profits.

The requirements management allows maintaining stability and agreement among stakeholder’s requirements, through the analysis of change effect and their monitoring during software lifetime (Ibrahim et al. 2009). So that the RE supports system answers to the changes in the dynamic environment caused by changing needs of their users (Qureshi & Perini 2010) (Wang et al. 2010).

Software projects should translate the actual needs of stakeholders that vary depending on personal and cognitive factors as well as the relationship that they have with the system. This diversity of needs leads to conflicts of interest among stakeholder (Attarha & Modiri n.d.). The RE helps to mitigate those conflicts by creating a knowledge base built with a set of requirements accepted by all stakeholders. This baseline is the starting point to software implementation. However, it suffers changes along the SaaS lifetime that should be maintained and managed (Ibrahim et al. 2009).

The currently used maintenance methods for software requirements involve a lot of effort and time to be applied. That effort may not be compatible with the short time required to make adjustments to the new stakeholders needs. They are not suitable for projects with a large quantity of requirements. Also these methodologies do not offer a way of gathering and manage stakeholder's feedback objectively. In short, the existing methodologies for managing requirements in an evolutionary context are inflexible and they are not scalable solutions (Ben Charrada et al. 2012) (Aasem et al. 2010) (Babar et al. 2011).

The objective of this paper is to contribute for diminishing the problem mentioned presenting a methodology for software requirements management and maintenance which can be applied in an evolutionary context. This methodology is based on the collection and analysis of the SaaS usage information. By studying the behaviour of SaaS users, we can obtain several metrics that allow us to give recommendations about how to manage and maintain some of the functional requirements behind the SaaS under study. It will allow the software to be adapted and respond timely to the changing needs of stakeholders, in order to prolong their lifetime.

This paper is organized as follows. Section I provides some concepts and information about the developing project. The literature review is presented in section II. Section III presents the methodology developed to manage functional requirements during software lifetime. This methodology was successfully applied to two case studies presented in section IV. Finally section V describes the conclusions and future work.

2 LITERATURE REVIEW

This section describes and compares existing tools for capturing information of the usage of web systems and presents related work regarding requirements management.

There are several web analytical tools which can collect information about the usage of a system. They differ on the information gathered. There are tools that capture the so called heat maps, i.e., graphical coloured representations according the percentage of mouse clicks in each of the website area (Clickdensity 2013), (Crazyegg 2013), (Firestats 2013); page views, which represent the percentage of website visits (Firestats 2013), (Analytics 2013), (Jawstats 2013), (Piwik 2013), (Counter 2013), (Tracewatch 2013), (Web Stat) (Woopra 2013); peaks of use, i.e., time period with the higher number of accesses (Bbclone 2013), (Firestats 2013), (Analytics 2013), (Jawstats 2013), (Piwik 2013), (Woopra 2013); the origin of the accesses, i.e., the internet protocol and the URL/origin from which users get access to the service being analysed (Bbclone 2013), (Crazyegg 2013), (Crazyegg 2013), (Analytics 2013), (Jawstats 2013), (Counter 2013), (Tracewatch 2013), (Web Stat, 2013) (Woopra 2013); navigation paths, i.e., is the sequence of users' interactions with the SaaS done between an origin and a destination (Analytics 2013), (Counter 2013), (Tracewatch 2013), (Web Stat) (Woopra 2013); interaction maps, i.e., graphical representations of the visitors' mouse clicks (Analytics 2013); use percentage of functionalities i.e., the percentage of visitors that have used each one of the website functionalities (Crazyegg 2013), (Analytics 2013), (Piwik 2013), (Tracewatch 2013), (Counter 2013), (Web Stat, 2013) and (Woopra 2013); qualitative data of users' experience i.e., users' feedback about their website visit (iPerceptions 2013); visitors logins i.e. percentage of users that do website login, and how many time they spend there (Jawstats 2013), (Piwik 2013), (Tracewatch 2013).

Given the myriad of existing tools, someone needs to identify the information to be collected for the purpose and select from the existing toolset the one (or the ones) that is able to extract the maximum information required.

Besides the collected information, the choice of the tool to use can also consider other aspect such as the installation mode because it is necessary to ensure data integrity and other questions about security and data confidentiality.

Before starting the development of a software system, it is important to identify the requirements with higher priority in order to develop them first. However, requirements prioritization is also useful in software maintenance phases since it is possible to get change requests that need also to be prioritized in order to identify the ones that should be developed first. There are several approaches for prioritizing requirements in the literature: Analytical hierarchical Process (AHP), Cost-Value Approach, B Tree Prioritize, Cumulative Voting or Hundred Dollar

Test, Numerical Assisments (Grouping), Ranking, Top Ten Requirements, Planning Game, Theory W and Fuzzy Logic prioritization (Aasem et al. 2010). However, these techniques do not support the negotiation of the different stakeholders' criteria for requirements prioritization. Furthermore they are time consuming for its implementation and they can only be useful for small projects size (Babar et al. 2011). It means that these methodologies are not scalable and there are not suitable to be applied during the software lifetime (Aasem et al. 2010).

Besides prioritization, it is of utmost importance to manage change requests. There are methodologies based on the change impact analysis (Sun & Li 2011) (Hayat et al. 2010) (Ali et al. 2012). The main objectives of these methodologies are to understand which parts of the original software will be affected by the change proposed and study the ripple effect to other software components. According to Benn Charrada et al. 2012 (Ben Charrada et al. 2012) the analysis of source code modifications identifies requirements affected by the change that needs new updates. The author Gao 2011 (Gao 2011) models software requirements evolution based on the feedback collected. Inverardi et al. (Inverardi & Mori 2011) define self-adaptive systems as entities that can modify his behaviour and structure due to the software and his environment changes. Banerjee (Banerjee 2011) present a methodology to manage requirements focused on errors that occur in the introduction or updating of the software requirements. Greenwood et al. (Greenwood et al. 2011) present a tool to manage the dynamic variability of systems that suffer adaptive pressures. The authors Souza et al. (Souza et al. 2012) focus the requirement management thought the "evolutionary requirements".

Most of these methodologies do not have the flexibility needed to manage large quantities of dynamic and evolutionary requirements (Aasem et al. 2010) (Babar et al. 2011). They imply a lot of time in their application which means that they do not allow the project team to have timely information about requirement changes (Ben Charrada et al. 2012). Moreover these methodologies do not offer a way of collecting and manage stakeholders' feedback timely and objectively. This information could be helpful to analyse changes in requirements evolution.

The methodology developed in this paper, collects usage information about SaaS in order to get users preferences about the functionalities and navigation paths. Thereby the collection of usage information is a way of getting some users feedback objectively. From the analysis of such information, it is possible to provide recommendations for reviewing the priority of the functional requirements that may be performed in whole or selected SaaS modules (group of related web pages of the SaaS).

3 METHODOLOGY

This section describes the proposed methodology for managing software requirements. From the automatic collection of information about the usage of a SaaS, the methodology proceeds with the analysis of such information. From this analysis, it is possible to propose updates to the software requirements and, in addition, other proposals may emerge for improving the usability of the service under study. Figure 1 shows the methodology developed.

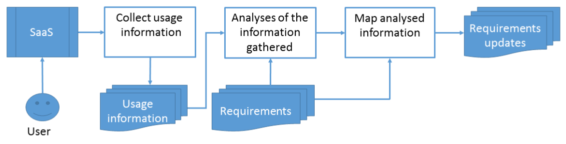

Figure 1: Requirement Management Methodology

3.1 Collect usage information

The web analytic tools presented in the literature review can be used for collecting data. In this particular context, it is useful to collect information about page views, accessed functionalities, navigation paths, heat maps, qualitative information about users' experience and the origin and destination of the visits. Regarding navigation paths, they can be obtained directly through some tools, such as Google Analytics, but when these tools are not available, navigation paths may be calculated during the following phase (Analyses of information gathered). In this case, origin and destination URLs must be provided. This information is needed to calculate other metrics from which information for helping requirements management will come up.

3.2 Analysis of the information gathered

After collecting the usage information of the service, the methodology proceeds with the analysis of such information in order to identify different users (different roles with different access modes); the most and least accessed pages; shortest, longest and most used navigation paths; and to determine which parts of the website are the most and least accessed, using heat maps.

It is important to identify different website visitors according to their different roles and permissions because they have access to different subsets of the overall functionality of the service. One typical example is the administrator that usually has access to configuration pages not accessible to other kinds of users. In addition, this information is useful for calculating the most and least visited pages of the SaaS when the usage information collection is done without the web analytic tools presented in the literature review. These pages are identified taking into account the total number of users that can access them and deserve special attention. In particular, it is possible to propose updates to the service in order to highlight the most visited pages, for instance, providing links for them on the entry page or improving the existing navigation paths from the entry page of the service in order to facilitate reaching them. Considering the related pages with fewer visits they will be evaluated to analyse the possibility of aggregate them with others in order to simplify the existing navigation within the SaaS.

The shortest, longest, most accessed navigation paths to specific functionalities of the service deserve also special attention. In particular, it may justify propose updates to highlight the shortest (or the most used). The analysis of these particular paths can be complemented with usability tests to get users feedback about the navigation experience.

From the analysis of the information in Heat Maps it is possible to identify the most accessed website areas. The areas surrounding the most used, may be useful for highlight SaaS content, i.e., to locate new functionalities, place marketing information or to make more visible a specific navigation path or functionality.

The percentage of page views in conjunction with Heat Maps and navigation paths is useful to analyse the usage of each functionality within the SaaS. The functionalities usage percentage reflects the importance that the corresponding features have to the users. Therefore, it can be used to review the baseline requirements priority for managing the subsequent responses to change requests. If a specific functionality has a large number of accesses, the priority of the respective requirement may be increased. On the contrary, if the functionality has a small number of accesses, meaning that it is not so important for users, its priority may be decreased. In the context of SaaS maintenance, change requests related to functionality with higher priority will be implemented firstly than those that are classified with a lower priority.

The following section presents how the functional requirements can be mapped with the respective SaaS functionality.

3.3 Map analysed information

After the analysis phase, the most and least used functionalities of the SaaS are identified. Now, it is time to analyse the related requirements and update them as needed.

If the map between functionalities and requirements is not documented, one should do it now to build a traceability matrix of each SaaS web pages and functional requirements from the baseline. Table 1 presents the template of the traceability matrix used in this phase.

Table 1: Map between functional requirements and SaaS web pages

SaaS Web Page	**Functional Requirement**
URL_1 of a SaaS web page	Requirement_A … Requirement_N
…	Requirement_B … Requirement_M
URL_z of a SaaS web page	Requirement_C … Requirement_S

4 CASE STUDIES

This section presents the results of application of the methodology introduced in the previous section in two real SaaS: SIGARRA, which is service provided for the Engineering Faculty of Porto (FEUP) community; Health Insight is a company that provides a service for the community interested in nativity which includes a portal where visitors can search articles and a social network called *Rede Mãe*. For confidentiality reasons, the information is displayed in a generic way without referring to concrete data.

4.1 SIGARRA (FEUP)

The main objectives established for this case study are:

- Identify the most and least accessed internal pages and the functionalities of the service;

- Extract the navigation paths for specific functionalities;
- Review the prioritization of the baseline functional requirements from the gathered information;

The service usage information has been collected by FEUP and logged into a data base file with records with the form:

$$<C, C_0, I, Io, U, U_0, T,>$$

where C is the date of the interaction, C_0 keeps the time in which the interaction occurred, I is a code which allows to distinguish users, I_0 is the internet protocol used, U is the URL origin, U_0 is the URL destination, T is the code of the user's session.

Since SIGARRA is a complex and large service, the percentage of views per page is low. This happens because the accesses spread along the several pages of the service. So, we decided to group the pages with related functionalities on module in order to simplify the statistical information.

From this database file it is possible to identify the most and least accessed web pages/functionalities and calculate the navigation paths. This is an example of a situation in which navigation paths must be calculated because the approach followed to extract information about the usage of the service does not provide them.

Given an URL as origin and an URL as destination, it is possible to calculate the navigation paths, i.e., the different existing sequences of visited pages from the origin until reaching the URL destination. This was calculated through MYSQL queries. In addition associated with each calculated path, it was also calculated the number of times such path was used. Figure 2 and Figure 3 show the navigation path between Page A (origin) and Page Z (destination). In these two figures, the boxes represent different URL pages (e.g., Pages A, B, C, D, E, E1, E2, E3 and Z). Pages E1, E2 and E3 belong to module E. The arrows represent the sequence of users' interactions. Figure 2 shows the shortest navigation path to from Page A to Page Z going through pages B and C. Figure 3 shows the most used navigation paths to Page Z departing from Page A.

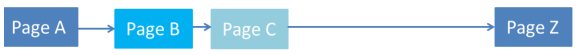

Figure 2: Shorter navigation path to Page Z

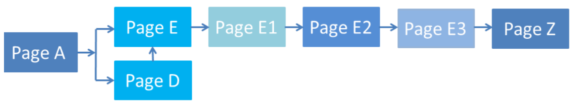

Figure 3: Most used navigation path to Page z

Table 2 shows part of the map built between functional requirements and the SaaS web pages. It only presents the functional requirements studied in this case study.

Table 2: Map between SaaS web pages and functional requirements

SaaS Web Page	Functional Requirement
Page E1	FR100
Page E2	FR102
Page E3	FR103
	FR105
	FR106
Page X	RF104

4.1.2 Results

Regarding the identification of the most and least accessed web pages and navigation paths, some improvements were proposed which are described in the sequel.

Considering the existing different navigation paths to functionality Z (page Z), we proposed a set of improvements to each of the intermediate pages of such paths. Regarding Page C, we concluded that few of its accesses intend to achieve Page Z. So that we propose to remove the access to Page Z from Page C or to aggregate it with another related page.

Regarding the most used path, we noticed that Page E3 has a high number of accesses so it can be highlighted to give it more visibility (requirement RF103 (Page E3)). In order to do that it is proposed to redesign Page E3.

In Pages E1 and E2, we identified related content so we propose to aggregate them (requirements FR100 and RF101).

Page Z, allows accessing to Page X, however there are no users that perform this specific navigation path in order to access functionality described by requirement RF104. In consequence we suggest giving access to this functionality directly from Page A. The information obtained from Heat Maps can be helpful to determine exactly the website area which has more attention from the SaaS users. For that reason heat maps support the decision of where to place it.

Since Page E3 is one of the most accessed, we reviewed the priorities of hers requirements. Functional requirements within such page that have low usage are not very important to their users, so that, we recommend assigning a lower priority. The most used requirements have a special importance to their users, which means that they may get a higher priority. Table 3 presents the priority revision done in this particular case.

Summarizing, considering this case study, the recommendations given by us are: remove the access to Page Z from Page C or aggregate Page C with another related page; highlight Page E3 (requirement RF103) to give it more visibility; aggregate contents of page E1 and E2 (requirements FR100 and RF101); give access to functionality within Page X directly through Page A; increase priority of requirement FR105; decrease priority of FR106.

Table 3: Revision of requirements priorities

Requirement	Baseline priority	Revised priority
FR105	2	5
FR106	4	1

4.2 Health Insight

The main objectives established for the case study are:

- Identify the most used features of the service;
- Analyse heat maps to identify the most accessed contents of the website;
- Review and propose changes in functional requirements based on the information gathered.

Health Insight company collected usage information of the service through Google Analytics and User Report. Its operation is based on the emission of cookies to track user visits and record data to explore their navigation sessions (Analytics 2013). The data gathered includes page views, navigation paths and interaction maps which can be configured with a colour graduation. Although Google Analytics allows to gather the majority of metrics needed for the goals mentioned, it has some constrains which are presented in the following section.

4.2.2 Results

Google Analytics saves sequences of users interactions. Within each user interaction, the tool groups, in the same module, the pages with the same visit flow. The visit flow is defined as the percentage of users that accesses a page and the percentage of users that finished his navigation in this specific page. Thus, unrelated pages could be presented at the same module which makes path analysis difficult.

In addition, the interaction map is constructed based on page views. It does not reflect the number of mouse clicks. Furthermore, Google Analytics does not allow monitoring AJAX pages. Considering all these aspects, we can say that Google Analytics has somehow restricted the analysis to perform but there was not possible to use another exploration tool inside this company.

Besides the restrictions imposed by the information gathered with Google Analytics, it was possible to calculate the percentage of page views and identify the most and least accessed functionalities. We concluded that *Rede Mãe* (social network for people interested in nativity issues) has a small number of accesses. In fact, the majority of users access this service to search papers in *Bebepédia* (portal of nativity contents) so, in order to increase the accesses to the least used functionality (*Rede Mãe*) we propose to evaluate the possibility of increasing the integrity between the *Rede Mãe* service and *Bebepédia*. For example, create a bottom inside *Rede Mãe* main page to share *Bebepédia* contents and highlight the new contents in a dashboard.

The test of nativity is the most used functionality of the social network *Rede Mãe* and because of that deserve higher priority regarding following change requests to improve them.

The identification of most used pages can also be helpful to increase the profits of the company by using the advertising located near the most accessed pages.

Summarizing, considering this case study, the recommendations given by us are: increase the integrity between the *Rede Mãe* service and *Bebepédia* by creating a bottom inside *Rede Mãe* to share *Bebepédia* contents and highlight the new contents in a dashboard; increase the priority of the test of nativity; use advertising located near the most accessed pages.

5 CONCLUSIONS AND FUTURE WORK

This paper describes a methodology to manage/maintain software requirements during the lifetime of a SaaS. This methodology was successfully validated in two services provided by two different companies. It allows reviewing requirements, proposing improvements in the software usability and consequently, extending the SaaS lifetime.

One advantage of this methodology is the fact that it is based on real/concrete usage of SaaS overlapping the problem related to commonly used approaches based on the diversity and subjectivity of stakeholders needs.

Considering the prioritization revision criteria, it is not necessary to compare all the requirements of the SaaS in study to review the priority of a specific SaaS module requirement. Thus, it is a scalable methodology that allows managing a higher number of dynamic requirements with less effort.

In order to diminish the time needed to apply this methodology, some steps could be automatized. Therefore it would be helpful to develop a tool for collecting information about the usage of the SaaS and represent it statistically. Also, the analysis of the collected information could be a computerized task based on a traceability matrix, which represents the relation between requirements and functionalities implemented within web pages.

As future work we intend to complement this methodology with usability tests and heuristic evaluation. With these tests, we expect to identify the main users' difficulties when it is noticed that to achieve a specific task that majority of the users opt for a navigation path that does not correspond to the shortest one.

REFERENCES

Aasem, M. et al., 2010. Analysis and optimization of software requirements prioritization techniques.

Ali, H.O., Rozan, M.Z.A. & Sharif, A.M., 2012. Identifying challenges of change impact analysis for software projects. 2012 International Conference on Innovation Management and Technology Research, pp.407–411. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6236428.

Analytics, G.,Visited at May 2013. Google Analytics. Available at: http://www.google.com/intl/pt-PT_ALL/analytics/index.html.

Attarha, M. & Modiri, N., Focusing on the Importance and the Role of Requirement Engineering. , pp.181–184.

Babar, M.I., Rarnzan, M. & Ghayyur, S.A.K., 2011. Challenges and Future Trends in Software Requirements Prioritization. , pp.319–324.

Banerjee, A., 2011. Requirement Evolution Management: A Systematic Approach. 2011 IEEE Computer Society Annual Symposium on VLSI, pp.150–155. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5992497 [Accessed September 5, 2013].

Bbclone, Visited at May 2013. Bbclone. Available at: http://help.bbclone.de/index.php?n=Main.HomePage.

Ben Charrada, E., Koziolek, A. & Glinz, M., 2012. Identifying outdated requirements based on source code changes. 2012 20th IEEE International Requirements Engineering Conference (RE), pp.61–70. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6345840.

Clickdensity, Visited at May 2013. Clickdensity Benefits. Available at: http://www.clickdensity.com/.

Counter, S., Visited at May 2013. Stat Counter Features. Available at: http://statcounter.com/?PHPSESSID=4a7ck7tttvei7ivkd7fidkkm63.

Crazyegg, Visited at May 2013. Here Are The Features That Make Your Websites Convert More Visitors Into Revenue. Available at: http://www.crazyegg.com/#what-you-get.

Dominguez, J., 2009. Chaos report-2009 on it project failure. Available at: http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure.

Firestats, Visited at 2013. Firestats. Available at: http://firestats.cc/.

Gao, T., 2011. A Process Model of Software Evolution Requirement Based on Feedback. , pp.172–175.

Greenwood, P. et al., 2011. Modelling adaptability and variability in requirements. 2011 IEEE 19th International Requirements Engineering Conference, pp.343–344. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6051667.

Hayat, F. et al., 2010. A methodology to manage the changing requirements of a software project. 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp.319–322. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5643642.

Ibrahim, N., Kadir, W.M.N.W. & Deris, S., 2009. Propagating Requirement Change into Software High Level Designs towards Resilient Software Evolution. 2009 16th Asia-Pacific Software Engineering Conference, pp.347–354. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5358735 [Accessed July 2, 2013].

Inverardi, P. & Mori, M., 2011. Requirements Models at Run-time to Support Consistent System Evolutions. , pp.1–8.

iPerceptions, Visited at May 2013. iPerceptions. Available at: http://www.iperceptions.com/.

Jawstats, Visited at May 2013. Jawstats. Available at: http://www.jawstats.com.

Piwik, Visited at May 2013. Piwik Features. Available at: http://piwik.org/.

Qureshi, N. a. & Perini, A., 2010a. Requirements Engineering for Adaptive Service Based Applications. 2010 18th IEEE International Requirements Engineering Conference, pp.108–111. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5636635 [Accessed March 3, 2013].

Qureshi, N. a. & Perini, A., 2010b. Requirements Engineering for Adaptive Service Based Applications. 2010 18th IEEE International Requirements Engineering Conference, pp.108–111. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5636635 [Accessed July 2, 2013].

Souza, V.E.S., Lapouchnian, A. & Mylopoulos, J., 2012. (Requirement) evolution requirements for adaptive systems. 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp.155–164. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6224402.

Stat, W., Visited at May 2013. Web Stat. Available at: http://www.webstat.com/.

Sun, X. & Li, B., 2011. Using Formal Concept Analysis to support change analysis. 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp.641–645. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6100146.

Tracewatch, Visited at May 2013. Tracewatch. Available at: http://www.tracewatch.com/.

Wang, H. et al., 2010. Quantitative Analysis of Requirements Evolution across Multiple Versions of an Industrial Software Product. 2010 Asia Pacific Software Engineering Conference, pp.43–49. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5693179 [Accessed July 2, 2013].

Woopra, Visited at May 2013. Woopra. Available at: http://www.woopra.com/.