Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 925–932

ISBN 978-83-60810-22-4

Testing and Remote Maintenance of Real Future Internet Scenarios

Towards FITTEST and FastFix Advanced Software Engineering

Alessandra Bagnato
Corporate Research Divisions, TXT e-solutions
alessandra.bagnato@txt.it

Anna I Esparcia-Alcázar
S2 Grupo, 46022 Valencia
aesparcia@s2grupo.es

Tanja E.J. Vos
Centro de Métodos de Producción de Software (ProS)
Universitat Politècnica de València
tvos@pros.upv.es

Beatriz Marín
Centro de Métodos de Producción de Software (ProS)
Universitat Politècnica de València
bmarin@pros.upv.es

José Oliver Murillo
Infoport Valencia
joliver@infoportvalencia.es

Salvador I. Folgado
BULL Spain
salvador.folgado@bull.es

Auxiliadora Carlos Alberola
INDRA
acarlosa@indra.es

***Abstract*—In recent years, software testing and maintenance services are key factors of customers' perception of software quality. Nowadays, customers are more demanding about these services, while contribution of maintenance and testing services to products total cost of ownership should be reduced. Reducing these costs is even more crucial for SME's. To do this, new methods and techniques that will be aligned with the needs of companies are required. This paper presents the preliminary results of an interactive workshop celebrated by researchers and three companies. In the workshop, researchers present the advanced software engineering methods proposed by FastFix[1] and FITTEST[2] European projects. After that, discussions about their potential use in three application scenarios at Infoport Valencia, BULL Spain, and INDRA were performed and some lessons were learned.**

***Index terms*—testing; maintenance; practical experience.**

I. Introduction

IN RECENT years, software testing and maintenance services are key factors of customers' perception of software quality. Therefore, companies are more demanding about these services, while contribution of maintenance and testing services to products total cost of ownership should be reduced. Reducing these costs is even more crucial for SME's, which has limited resources to spend in testing and maintenance of their products.

The Future Internet (FI) aims at reducing developing, testing and maintenance services through a common space where different services can be combined to produce software reducing their total cost. Thus, it can be anticipated that the Future Internet will be a complex interconnection of services, applications, content, and media; all of which will increase with semantic information. Also, it can be foreseen that future web applications will offer a rich user experience, extending and improving current hyperlink-based navigation.

Thus, future internet applications are expected to be complex applications, which present the following characteristics: self-modifiability, autonomic behavior, low observability, asynchronous information, time- and load-dependent behavior; a huge feature configuration space, and ultra-large scale. Since current maintenance and testing techniques are not suitable to future internet applications, FastFix and FITTEST projects are developed to face the challenges of these applications.

The overall purpose of FastFix is to provide software applications with a maintenance environment featuring the highest time efficiency at the lowest cost and the strongest accuracy. To this effect, FastFix will develop a platform and a set of tools that will monitor on-line customer environments, collecting information on program execution and user interaction, with the objective of identifying symptoms of execution errors, performance degradation or changes in user behaviour. By use of correlation techniques, the platform will also allow failure replication in order to identify incorrect execution patterns, patch generation, and patch deployment.

The overall aim of the FITTEST project is to address the testing challenges that arise in Future Internet Systems due to the complexity of the technologies involved. To this end, FITTEST will develop an integrated environment for automated testing, which can monitor the Future Internet applications under test and adapt the testing to the dynamic

1 FastFix (Monitoring Control for Remote Software Maintenance) (FP7-25810) is an FP7 project.

2 FITTEST (Future Internet Testing) (FP7-257574) is an FP7 project.

changes observed. The environment will implement continuous post-release testing since a Future Internet application does not remain fixed after its release. Services and components could be dynamically added by customers and the intended use could change significantly. The environment will integrate, adapt and automate various techniques for continuous Future Internet testing (e.g. dynamic model inference, model-based testing, log-based diagnosis, oracle learning, combinatorial testing, concurrent testing, regression testing, etc.).

In order to make sure that both projects are aligned with what is needed in industry with regard to testing and maintenance, a joint FITTEST & FastFIX workshop was organized in June 2011 during which the following research questions were posed:

Question 1: Can Future Internet Maintenance and Testing support real scenarios?

Question 2: Are the approaches proposed by FastFix and FITTEST interesting to the real end-user needs?

Question 3: How much effort has to be put into creating new tools to support the expressed needs?

The rest of the paper is organized as follows: Section II presents the FastFix Project and Section III presents the FITTEST Project. Sections IV, V and VI describe the INDRA, BULL and Infoport Valencia Scenarios, respectively. Lessons learned are addressed in Section VII. Finally, Section VIII presents some conclusions and outlines future work.

II. FastFix Project

Maintenance and support services that are time and cost efficient is the driving goal of the FastFix project [2], which started in June 2010. This is to be achieved by monitoring software applications, replicating execution failures, and automatically generating patches.

Among the results of the project will be a platform and a set of tools that will monitor online customer environments. This will gather information on program execution and user interaction, aiming to identify symptoms of execution errors, performance degradation or changes in user behaviour. The platform will also allow the replication of failures by means of correlation techniques; the purpose of this feature is to identify incorrect execution patterns and facilitate patch generation and deployment.

In order to achieve this, mechanisms will be developed and set up to gather the required information on application execution, errors, context, and user behaviour. These mechanisms will be applicable to both new and existing applications; they should also be non-intrusive and pose a minor, acceptable burden on performance.

Information thus gathered will be sent in real time to a support centre, via the Internet. Hence, special care will be required with regard to security and privacy.

At the support centre, information will be used to replicate errors, by means of correlation techniques and error ontologies which will allow the identification of behavioral patterns and possible causes of error. At its best, the FastFix platform will be able to generate patches for the errors in an automated way. These patches will consist on application modifications, changes in the system configuration or parameterization or even a limitation in functionality in order to avoid system or application crashes. Patches will be sent back to the application's runtime environment and will be deployed automatically, resulting in a self-healing software application.

Figure 1 illustrates the components of the FastFix Architecture.

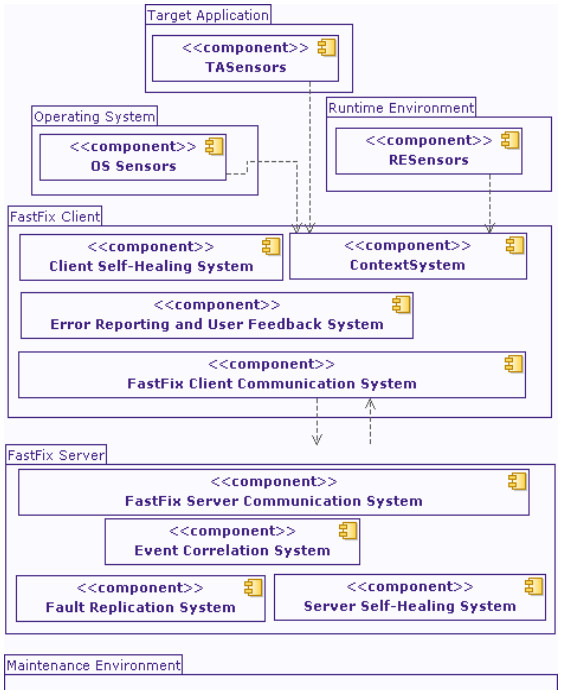

Figure 1. FastFix architecture for software maintenance

Four main lines of research converge in the FastFix project and constitute the core of innovation activities:

- **Context elicitation and user modeling:** determines which information on execution and interaction is going to be gathered independently from the application and its environment, and how this is to be done.
- **Event correlation:** allows drawing conclusions about the kind of problems the application is undergoing and their possible causes, based on the information gathered.
- **Fault replication:** provides the platform that allows replicating faults, which mimic the real circumstances as much as possible.
- **Patch generation and self-healing:** determines which patches are going to be generated and how this is to be done, plus the way they will be deployed to the application at the runtime environment. An approach based on control theory that allows for fast and reliable maintenance fixes is mentioned in [3], this approach aims to disable faulty or vulnerable system functionalities and requires to instrument the system before deployment so that it can

later be monitored and interact with a supervisor at runtime.

FastFix will provide innovation in any of its four main research area,s namely event correlation, fault replication, patch generation, and context elicitation.

Event correlation techniques will be used within FastFix in the field of software defect detection and cause identification. Developments in this area have mostly focused on system software [9]. Thanks to the event correlation FastFix will be also able to determine the type and level of monitoring that must be exercised in each execution instance.

Failure information will consider privacy concerns associated with the release of failure information providing novel data obfuscation techniques which will preserve the program re-execution's accuracy. Analysis techniques operating at source code [12][13] and binary level [10][11] are taken as a starting point for this research.

In FastFix, autonomous system principles and methods focusing on assessing the viability of auto-generating patches [14][15][16] will be applied.

As opposed to current approaches [17] [18] [19], in which collected data related to context describes only the usage of a particular tool, context elicitation in FastFix will be carried out independently from the monitored application and the usage domain,

III. FITTEST PROJECT

The FITTEST project (September 2010-2013) is being carried out by eight partners: Universitat Politècnica de València, University College London, University of Utrecht, Fondazione Bruno Kessler, Berner&Mattner System technic, Sulake, Soft-Team, and IBM Israel.

Existing literature on web testing (such as [20][21][22]) is focused on client-server applications which implement a strictly serialized model of interaction, based on <form submission, server response> sequences. Testing of Ajax and rich client web applications has been considered only more recently [23][24]. The differences between Ajax and more traditional web testing are discussed in [25]. Even though there are some recent works that consider testing of dynamic web applications, they are not addressing the testing challenges of future web applications mentioned in [5]. For this reason, we consider the development and evaluation of an integrated environment for continuous evolutionary automated testing, which can monitor the FI application and adapt itself to the dynamic changes observed.

FITTEST testing will be continuous post-release testing since the application under test does not remain fixed after its release. Services and components could be dynamically added by customers and the intended use could change. Therefore, testing has to be performed continuously after deployment to the customer.

The FITTEST testing environment will integrate, adapt and automate various techniques investigated in the project for continuous FI testing, providing a user friendly way to activate and parameterize them. See Figure 2 for a global picture of the testing environment.

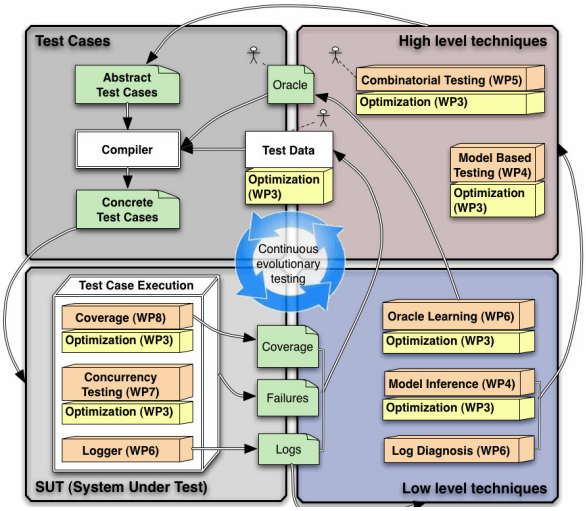

Figure 2. Global Picture of the FITTEST Testing environment

The underlying engine of the FITTEST environment, that will make it possible to automate undecidable problems and cope with the testing challenges like dynamism, self-adaptation and partial observability, will be based on search-based testing [4].

The impossibility of anticipating all possible behaviours of FI applications suggests a prominent role for evolutionary testing techniques, because it relies on very few assumptions about the underlying problem that is attempting to solve. In addition, stochastic optimisation and search techniques are adaptive and, therefore, able to modify their behaviour when faced with new unforeseen situations. These two properties – their freedom from limiting assumptions and their inherent adaptiveness – make evolutionary testing approaches ideal for handling FI applications testing, with their dynamism, self-adapting, autonomous and unpredictable behaviour. Since evolutionary testing is unfettered by human bias, misguided assumptions and misconceptions about possible ways in which the components of the system may combine, FITTEST avoids the pitfalls that are found with humans-innate inability to predict that which lies beyond their conceivable expectations and imagination. Moreover, evolutionary techniques are well understood techniques for solving general undecidable problems and will constitute a robust and stable foundation upon which to build FITTEST.

To achieve this overall aim, FITTEST will address a set of objectives that directly map to the identified challenges.

Objective 1: Search based testing approach. To cope with dynamism, self-adaptation and partial observability that characterize FI applications, we will use search-based software testing. Evolutionary algorithms themselves exhibit dynamic and adaptive behaviour and, as such, are ideally suited to the nature of the problem. Moreover, evolutionary algorithms have proved to be very efficient for solving general undecidable problems and provide a robust framework.

Objective 2: Continuous, automated testing approach. Since the range of behaviours is not known in advance, testing will be done continuously; feedback from post-release

executions will be used to co-evolve the test cases for the self-adaptive FI application; humans alone cannot achieve the desired levels of dependability, so automation is required.

Objective 3: Dynamic model inference. Self-adapting applications with low observability demand for dynamic analysis; models will be inferred continuously rather than being fixed upfront.

Objective 4: Model based test case derivation. Behavioural models inferred from monitored executions will be the basis for automated test case generation. Paths in the model associated with semantic interactions will be regarded as interesting execution sequences. To support continuous, extensive testing of FI applications, test case generation will proceed fully unattended, including the generation of input data and the verification of feasibility for the test adequacy criteria of choice.

Objective 5: Log-based diagnosis and oracle learning. Since correct behaviour cannot be fixed upfront, executions will be analysed to identify atypical ones, indicating likely faults or potential vulnerabilities.

Objective 6: Dynamic classification tree generation. The huge configuration space will be dealt with by testing combinatorially, using dynamically and continuously automated generated classification trees.

Objective 7: Test for concurrency bugs. Toward successful concurrency testing and debugging of FI applications, we will develop a mechanism to control and record factors like communication noise, delays, message timings, load conditions, etc, in a concurrent setup.

Objective 8: Testing the unexpected. Due to the high dynamism, it is impossible to define the expected interactions upfront; we will use genetic programming to simulate unpredicted, odd, or even malicious interactions.

Objective 9: Coverage and regression testing. Novel coverage and regression testing criteria and analytical methods will be defined for ultra-large scale FI applications, for which the standard criteria and analysis techniques are not applicable since they just do not scale.

Objective 10: General methodological evaluation framework for FI testing. Large scale case studies will be performed using realistic systems and software testing practitioners. The studies will be executed using an instantiation and/or refinement of the general methodological evaluation framework to fit specific software testing techniques and tools and evaluation situations.

IV. INDRA SCENARIO

INDRA [6] is a global company of technology, innovation, and talent, leader in solutions and services for Transport and Traffic, Energy and Industry, Public Administration, Healthcare, Finance, Insurance, Security and Defense, Telecom and Media sectors.

At INDRA Valencia, one thousand of professionals are working in different markets. Indra main activity (65%) is on health care, followed by other activities in Public administration and energy industry, representing the 21% and 6% of INDRA total work, respectively.

Health care software is then the most relevant sector at Valencia. In particular it consists of the following products:

- Abucasis, a web application that connects together the entire primary assistance system, including electronic prescriptions.
- INDRAHealth Solution, an integrated hospital and primary assistance web application.
- Medas, different modules that ingrate different specific applications as for example: blood transfusion center, or emergency management

INDRAHealth solution is a project that involves a huge amount of health professionals and developers, so it was necessary to define a clear development methodology to have success in the project.

The methodology defined at INDRA has been adapted to perform testing. In INDRA methodology, testing activity starts after the working tasks are assigned to the developers, meaning that testing is done in parallel with the development team. With this methodology, we have obtained quite an efficient distribution of work load. Thus, testing is not only performed by testers. Developers are also assigned a set of test activities that are performed before the source code is released.

INDRA´s testing methodology is organized in three different stages:

1. The first stage include test definition and planning, and is performed in parallel with code development
2. The second stage is performed after developers have finished, and include test execution and fault registration.
3. Once the faults are solved, they are tested again and, finally a regression test is performed.

The automated tests are launched when nobody is working on the application with a clean database. Finally, when the application is stable, INDRA starts the performance tests. The purpose of performance testing is to simulate the normal use of the system before it passes to production, in order to find potential problems and correct them.

In the simulation, it is possible to predict how INDRA Helath Solution will behave with a specific load. In a broad sense: the system capacity is verified to be adequate for the demands of work supports, and the potential bottlenecks and inefficiencies are identified by providing the necessary information for correct them.

Every semester INDRA calculates the following measures in order to improve their testing process:

- Test coverage
- Effectiveness of tests
- Number of faults detected per KLOCs
- Number of faults resolved per KLOCs
- Percent of faults resolved

The principal testing problems at INDRA are:

- No economic resources allocated to testing
- Little understanding of the test work from the developers' part.
- Changes in functionality are implemented at the final stages of projects.
- There is low stability in modules, but is not possible to automate everything. Thus, more automation is needed.
- Sometimes, the communication between teams is slow.

Thus, in the future, INDRA would like to show the managers that test is a necessary part of the software process development. Also INDRA has to make developers understand that test can be a useful tool to help them in the software development. As web applications may present large changes in a short time, INDRA needs an easy and fast way to test, which means less manual test.

V. BULL SCENARIO

As the only European player that covers the whole of the IT value chain, BULL [7] is ideally placed to help its customers build value-creating information systems. In an open and fast-changing world, Bull helps companies and administrations to liberate themselves from technological shackles, enabling them to radically gear up their innovative capabilities. In its relentless quest for safe, cost-effective, sustainable, open-critical infrastructures, BULL cultivates cutting-edge know-how, teams up with top-tier partners, and develops a broad, powerful, and modular range of products and services.

By consistently offering customers the fruit of innovative, purpose-designed, high-performance ideas, BULL has earned worldwide recognition and enjoys a leading position in Europe.

At the Software Quality department at BULL Spain S.A. (BULL QA), assessing the quality of Service-Oriented Architectures (SOA) through a specialization of the testing environment is particularly interesting.

SOA, by its nature has the following characteristics: self-contained; highly modular and independently deployed; consists of distributed components that are available over the network; has a published interface; only needs to see the interface; stresses interoperability; different implementation languages and platforms are involved; is discoverable; needs a directory of services that are registered and located; is dynamically bound; and can locate services and bind them at runtime.

An Enterprise Service Bus (ESB) is a logical architectural component that provides an integration infrastructure consistent with the principles of SOA. The most effective way to test SOA environments is through a systematic approach following the V testing model. All components that are part of the architecture first need to be tested independently and then in integration with other components and systems involved.

SOA test design should follow a top down approach, and the test execution should follow a bottom up approach, starting at individual service (or component) level. The following characteristics must be considered when testing:

Functionality. Since there are no user interfaces, a "formal contract" to reach adequate testing quality must be used. Key business stakeholders and users should be more actively involved in all project lifecycle. Regression testing must be made more efficient and should be automated.

Interoperability. It is necessary to focus on interfaces, and assure that interface behavior and information sharing between services are working as specified. Integration testing should include communication, network protocols, transformations, etc.

Compliance. It is necessary to satisfy standard (law) more formal and more accurately.

Backward Compatibility. BULL needs to assure that SOA architecture continuously work successfully even when any modification is done in any component/service, BULL needs to assure "Loose Coupling".

Security. SOA provide access and potential modification of services (data) from different physical locations, over the WAN. Thus, How safe is the data? Business requirements must include security requirements. BULL must perform a security risk analysis during design and will need formal reviews to assure that the organization security standard is satisfied. Penetration security test must be planned and executed.

Performance. Is the most degraded quality characteristic the designer must take care of when doing SOA design. The following specific characteristics that would impact the performance at SOA must be considered in order to perform a correct performance testing:

- *Distributed computing*. Services are normally located in different containers, most often in different machines.
- *Heterogeneous message and protocols*. Transformation must be managed and transform inside ESB.
- *Different platforms*. Involves different technologies, different behaviors, BULL must be care with the "The Weakest Link" effect.
- *XML intensive*. XML message can be 10 or 20 times larger than equivalent binary representation, so transmission over a network takes longer. XML uses text format, so it must be processes before any operation is performed (Parsing, Validation and Transformation) are CPU and MEMORY intensive.
- *Scalability (vertical, horizontal).* Because service users know only about service interface and not its implementation, providing scalable solutions requires little overhead.

- *Latency*. Distributions means interconnections, between networks and its service quality, conventional TCP is a guaranty delivered protocol. TCP has a direct inverse relationship between latency and throughput, then, massive message at ESB implementation could increase latency.

- *Not data oriented*. ESB is not a final data oriented solution, however governance produces a significant data quantity to be managed, as accounting, auditing, service location, etc.

In this context, for SOA testing at BULL the following conclusions and challenges are listed:

- SOA testing is different from other applications testing.
- SOA implementations are a combination of any kind of components.
- SOA testing is more complex than traditional testing (granularity, systems, messages, etc.)
- Functionality has less risk than others quality characteristics.
- Early testing (design, unit testing) is highly recommended to assure a proper SOA architecture.
- Testing approach strategy depend on SOA implementation (test design top-down approach, test execution bottom-up approach)
- The SOA design and implementation will provide the critical main quality characteristics to satisfy
- Performance characteristics could be impacted in a SOA Approach.

VI. INFOPORTVALENCIA SCENARIO

Infoport Valencia [8] is an information technology and communications service provider. The company has been working on development and software maintenance projects since its creation, more than 10 years ago.

One of the projects Infoport is working on is a set of IT operations and testing service for Valencia Port Community System (PCS), which is an information system that offers services for managing port, sea, and land logistic operations, tracking, and other operations with more than 400 organizations as users, including private enterprises and public institutions as Valencia Port Authority.

PCS platform allows electronic data interchange over the Internet between organizations involved in port logistics activity. The system is fully developed with Microsoft .NET technology. More than 30 IT professionals belonging 4 different companies work on development and operations tasks of the platform. Technical infrastructure includes more than 20 servers between physical and virtual ones.

PCS project started more than 7 years ago and goes on launching new services and functionalities every year. This means a continuous flow of software development and maintenance packages that require a high level coordination and well-defined processes to guarantee an agile evolution of the system.

The software development, testing and deployment processes for new functionalities and maintenance packages make use of four different environments (development, testing, pre-production, and production). Obviously, a package that does not pass the tests and validation process in one environment will not be deployed in the next one.

Software maintenance requires an environment maintenance and synchronization too, at data and code levels, and a common planning including all software packages dependencies, timing, and resources. A common problem is when a package does not pass the acceptance (or any other) tests and other packages planned to be deployed after that package with dependencies on that one need to be replanned.

Once a package is deployed in production environment two kinds of maintenance operations are needed: one adaptive (or preventive) for dealing with changes in the software environment or user requirements, and a corrective maintenance to deal with incidents found and fix them.

Infoport has defined an incident lifecycle with a 3-tier support: first, a user reports an incident to the user's support center, where operators may attend and solve functional requests. If an incident requires a technical analysis, it is assigned to the second support level, where a technician may decide to fix an incident in production environment with a hotfix, depending on its impact and priority. If the incident is not critical or requires a major modification to be fixed, then, it is assigned to the third support level, where developers will fix and prepare a new package to be delivered.

The testing process at Infoport requires a planning prior to deliver a release. At this stage, information about dependencies with other packages, priority, and resources are defined. Next step is to deliver the required documentation to testing and operations teams (as functional and technical design and requirements). These documents are reviewed in order to check its completeness and used them as an input to prepare the test cases (scenarios, data set and expected results). Once the release is delivered, it is built, and software verification and validation is done. If the package is accepted, it is deployed in the testing environment, and test cases are executed. A report containing the results of the executed test cases is created. If faults have been found, these are sent to development teams to fix them in a new package. This process is repeated as many times as needed until the package passes all test cases. Sometimes, this means several iterations and delays in planning.

In order to improve this, Infoport is evaluating now some changes in the process to anticipate functional testing to an early stage and reduce the number of iterations that a package requires to pass tests, and therefore, be deployed in production environment.

A common problem in the testing process is the time required to prepare and execute test cases. Data sets need to be prepared with so many combinations as exist. If this is a

manual process, it is highly time-costly and usually data sets do not cover all possible cases. If testing tools are used, time may be reduced in some parts of the process but on the other hand it is increased in others, as testing tool needs to be prepared too. Infoport combines the use of manual and automated testing. Usually, Infoport testing process is a manual process, but in some cases software simulators are developed to test specific parts.

In this context, Infoport's main challenges to improve their testing and maintenance processes are:

- **Automation**. Vulnerability to failure because most of the testing process is manual. Currently Infoport is evaluating the use of testing tools as VS2010 and Lab Management 2010 to automate part of the process.

- **Number of test-error-fix iterations**. Impact of replanning testing tasks and delays in production environment deployments. To mitigate this, Infoport has created testing teams to execute functional tests in early stages of the process, before delivery to the operations team.

- **Environments synchronization**

- **Requirements detail level**. Sometimes requirements definition and functional design is not enough for testing teams to create the test cases. Infoport is evaluating the use of Microsoft Team Foundation Server for quality assurance and requirements management.

VII. Lessons Learned

Question 1, "Future Internet Maintenance and Testing can support real scenarios?" and Question 2 "Are the approaches proposed by FastFix and FITTEST interesting to the real end-user needs?" have been positively answered by the companies during the workshop. As researchers and experts agreed, it has been acknowledged that the relative cost for software testing, maintenance, and management of its evolution represents around the 70% of the total cost.

FastFix could help significantly reduce time used in failure cause identification, patch generation, and deployment meeting end-user needs and expectances.

Maintenance encompasses all the costs incurred to fix faults ("corrective maintenance"), maintain the engineering integrity of the application ("adaptive maintenance"), change the structure of the application to meet changing business needs ("perfective maintenance"), and stop predictable faults in the future ("preventive maintenance"). FastFix could help in corrective maintenance by identifying the failure and its context, and partially in adaptive and preventive maintenance by identifying failures and patching the system, at least, providing temporary patches.

Even in cases where FastFix will not be able to automatically identify causes or generate patches, it has been considered as very valuable the fact that FastFix could provide valuable context information, both on execution environment and in user interaction, which will facilitate the task for a software engineer.

The outcomes of the **FITTEST** project will support companies with their test automation needs. Moreover, the FITTEST continuous testing approach will help them to cope with changing requirements and dynamic nature of the Internet applications.

In particular:

- **INDRA** will see how advances in FITTEST and FastFix projects can help in the automation of test cases and in the improvement of the testing work in place.
- The objectives of the FITTEST and FastFix projects were considered as extremely important for **Infoport**, as their lines of research and results may help Infoport to incorporate some improvements to their testing and maintenances processes to make them more efficient, particularly to automate part of the process and reduce the number of test-error-fix iterations.
- During the workshop it became clear that the FITTEST project's objectives tackle many of **BULL**'s SOA testing challenges. BULL will see how the advances in the FITTEST and FastFix projects could help in providing an integrated framework capable of analyzing and evaluating the quality of SOA.

Question 3 "How much effort has to be put into creating new tools to support the expressed needs?" had to be postponed to the next conference event that will be organized in a year time by FITTEST and FastFix projects. The two projects are in the early phases of their research efforts, and currently they are working in building their first prototypes and could not provide a precise estimation.

A very positive aspect that was remarked by the involved researchers was how close to the business needs of the involved SMEs the two projects are. This is an incentive to pursue further the collaboration among FITTEST and FastFix projects, and to the organization of the second "FITTEST & FastFIX joint workshop - Conference Testing and Remote Maintenance of the Future Internet workshops" 2012.

VIII. Conclusion and Future Work

In this article, we have summarized the needs and the feedbacks gained by introducing the FastFix and FITTEST advanced research ideas to three different industrial environment and scenarios context. The results of the potential of the tools developed within the two projects are promising and it was acknowledge that their adoption would allow improving the current practice in real industrial scenarios. Using FastFix and FITTEST tools, software can be tested and maintained with improved quality and in a faster way.

It has been considered as very important for the acceptance of the produced tools the fact that testers and maintenance engineers could be able to easily use the tools in the testing phase and in the maintenance phase of the software development lifecycle.

It was evident from the discussions carried out that SMEs needs to take full advantage of an adequate on-site customer support for maintenance as the one proposed within FastFix. Software vendors need a system to remotely provide a high quality support service to their customers, improve user experience and facilitate corrective, adaptive and preventive maintenance – of both new and existing software products.

It was also evident from the discussions carried out that SMEs need testing techniques and tools that perform automatically the testing process of their applications. This is precisely the main result of FITTEST project, which consist in a set of testing techniques and tools that allows the automatic generation and execution of test cases for future internet applications. At this stage, the difficulties related to the selection, usage and adaption of different available testing tools in the different contexts remain an issue to be treated case by case.

As future work, the two projects (i.e., FastFix and FITTEST) plan to host another conference in a year time when more results will be available.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community`s Seventh Framework Programme (FP7/2007-2013) under the grant agreement FP7-258109 FastFix and FP7-257574 FITTEST.

REFERENCES

[1] FITTEST Project Consortium. Web Site. http://www.facebook.com/FITTESTproject

[2] FastFix Project Consortium. Web Site http://www.fastfix-project.eu/

[3] Gaudin B., Bagnato A. (2011). Software Maintenance through Supervisory Control. SEW-34, the 34th Annual Ieee Software Engineering Workshop Proceedings Limerick, Ireland, 21 June 2011

[4] Mark Harman. The current state and future of search based software engineering. In 2007 Future of Software Engineering, FOSE '07, pages 342–357. IEEE Computer Society, 2007.

[5] Tanja E. J. Vos, Paolo Tonella, Joachim Wegener, Mark Harman, Wishnu Prasetya, Elisa Puoskari, and Yarden Nir Buchbinder. Future internet testing with fittest. Software Maintenance and Reengineering, European Conference on, 0:355–358, 2011.

[6] INDRA Web site http://www.indracompany.com/

[7] BULL Web site http://www.bull.es

[8] InfoPort Valencia Web site http://www.infoportvalencia.es

[9] Viliam Holub, LERO et alter. "Run-Time Correlation Engine for System Monitoring and Testing". ICAC'09. June 2009

[10] Newsome, J., Brumley, D., Franklin, J., and Song, D. 2006. Replayer: automatic protocol replay by binary analysis. In Proceedings of the 13th ACM Conference on Computer and Communications Security (USA, October 30 - November 03, 2006). CCS '06. ACM, New York, NY, 311-321.

[11] David Brumley, Juan Caballero, Zhenkai Liang, James Newsome, and Dawn Song. Towards Automatic Discovery of Deviations in Binary Implementations with Applications to Error Detection and Fingerprint Generation. Proceedings of USENIX Security Symposium, Aug 2007.

[12] S. Elbaum, H. N. Chin, M. B. Dwyer, and J. Dokulil. Carving differential unit test cases from system test cases. In Symp. Foundations of Software Engineering, pages 253–264, 2006.

[13] Xu, G., Rountev, A., Tang, Y., and Qin, F. 2007. Efficient checkpointing of java software using context-sensitive capture and replay. In Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering ESEC-FSE '07. ACM, New York, NY, 85-94.

[14] G. Williamson, D. Cellai, S. Dobson, and P. Nixon, "Self-management of routing on human proximity networks," in In Proceedings of International Workshop on Self-Organising Systems, Springer Verlag Lecture Notes in Computer, 2009.

[15] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," ACM Transactions on Autonomous and Adaptive Systems, vol. 1, pp. 223–259, December 2006.

[16] B. Gaudin, P. Nixon, K. Bines, F. Busacca, and Casey, "Model bootstrapping for auto-diagnosis of enterprise systems," in Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE), p. to appear, IEEE Press, December 2009.

[17] W. Maalej, Task-First or Context-First? Tool Integration Revisited, In 24th ACM/IEEE International Conference On Automated Software Engineering, 2009

[18] W. Maalej and H.J. Happel, From Work to Words: How do Software Developers Describe Their Work, in Proceedings of the 6th IEEE Conference On Mining Software Repositories, IEEE CS, 2009

[19] W. Maalej, H.J. Happel, A. Rashid, When Users Become Collaborators: Towards Continuous and Context-Aware User Input, In companion of ACM OOPSLA 2009.

[20] F. Ricca, P. Tonella, Analysis and Testing of Web Applications, in: International Conference on Software Engineering (ICSE), 2001, pp. 25-34.

[21] S.G. Elbaum, G. Rothermel, S. Karre, M. Fisher, Leveraging User-Session Data to Support Web Application Testing, IEEE Trans. Software Eng. , vol. 31 nº 3 (2005), pp. 187-202.

[22] S. Sampath, S. Sprenkle, E. Gibson, L.L. Pollock, A.S. Greenwald, Applying Concept Analysis to User-Session-Based Testing of Web Applications, IEEE Trans. Softw Eng., vol. 33 nº 10 (2007), pp. 643-658.

[23] A. Mesbah, A. van Deursen, Invariant-based automatic testing of AJAX user interfaces, in: International Conference on Software Engineering, 2009, pp. 210-220.

[24] A. Marchetto, P. Tonella, F. Ricca, State-Based Testing of Ajax Web Applications, in: ICST, 2008, pp. 121-130.

[25] A. Marchetto, F. Ricca, P. Tonella, A case study-based comparison of web testing techniques applied to AJAX web applications, STTT, vol. 10 nº 6 (2008), pp. 477-492.