Computer Science and Information Systems 11(1):29–45 DOI: 10.2298/CSIS130129019C

SABUMO-dTest: Design and Evaluation of an Intelligent collaborative distributed testing framework

Ricardo Colomo-Palacios[1], José Luis López-Cuadrado[1],
Israel González-Carrasco[1], and Francisco José García-Peñalvo[2]

[1] Universidad Carlos III de Madrid
Av. Universidad 30
28911 Leganés, Madrid, Spain
{ricardo.colomo, joseluis.lopez.cuadrado, israel.gonzalez}@uc3m.es
[2] Universidad de Salamanca
Plaza de los Caídos s/n,
37008, Salamanca, Spain
fgarcia@usal.es

Abstract. Software development must increasingly adapt to teams whose members work together but are geographically separated leading to distributed development projects. Such projects consist of teams working together, but sited in different geographic locations. Under these conditions, Global Software Engineering is having a profound impact on the way products are conceived, designed, constructed and tested. One of the problems with this area is the lack of tools which supports the distributed process. Focusing on the testing process, this paper presents SABUMO-dTest, a framework based on Semantic technologies that allows software organizations to represent testing processes with the final aim of trading their services or modeling their testing needs in a social and competitive environment. The proposed framework benefits from a set of shared and controlled vocabularies that permit knowledge and process sharing with potential partners, experts and testing service providers. The evaluation of the system included two kinds of projects, the ones in which testing was not determined by SABUMO-dTest and the ones developed under its influence. Results show remarkable outcomes in SABUMO-dTest driven projects.

Keywords: Semantic technologies, software testing, distributed testing, knowledge management.

1. Introduction

Software industry and its practices have been profoundly changed due to the pressures of globalization. As with many other current industries, software development must increasingly adapt to teams whose members work together but are geographically distributed [1]. Organizations dedicated to software development are more and more confronted with a working philosophy shift towards the distribution of processes and development teams [2]. This fact leads to distributed development projects. Such projects consist of teams working together from different geographic locations [3].

Global Software Development (GSD) is a particular type of distributed software development in which teams are distributed beyond the limits of a nation [4]. The adoption of GSD means that software engineers should collaborate over geographic, temporal, cultural and linguistic distance; these characteristics are usually termed "global distance" [5]. Although the growth and spread of distributed work in itself is testament to its success, studies continue to show that distributed workers face many critical challenges [6]. In other words, GSD teams present a number of challenges which must be considered before using a model for the management of teams in such an environment [7]. These challenges have been reported in the literature: communication, coordination, and control issues (e.g. [8]), loss of efficiency (e.g. [9]), lack of team and interpersonal trust and socio-cultural distance (e.g. [10]) citing the most relevant and reported challenges. On the other hand, there are several benefits that the literature has extensively reported: closer proximity to market and customer (e.g. [11]), reduced development costs (e.g. [12]), access to large skilled labor pool (e.g. [13]) and shorter time-to-market cycles (e.g. [11]).

According to [4], GSD is having a profound impact on the way products are conceived, designed, constructed and tested. Focusing on this last process, this paper presents SABUMO-dTest, a framework based on Semantic technologies that allows software organizations to represent testing processes with the final aim of trading their services or modeling their testing needs in a social and competitive environment. Based on previous works [14], SABUMO-dTest, benefits from a set of shared and controlled vocabularies that permit knowledge and process sharing across with potential partners, experts and testing service providers.

The remainder of this paper is organized as follows. Section 2 reviews the state of the art related to semantic technologies along with main insights of testing in distributed environments. Section 3 describes research scenario and Section 4 defines the knowledge representation and the semantic technologies used. Section 5 describes the architecture of SABUMO-dTest. Section 6 depicts the validation of the framework. Section 7 includes the results and the discussion of the evaluation performed. Finally, section 8 presents the conclusions and the future research.

2. Related Work

Software testing consists of the dynamic verification of the program behavior on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior. Inadequate and ineffective testing is responsible for many problems regarding software reliability faced by computer users [15]. Software testing is a complicated process and the primary source of intricacy on testing of software systems is the intrinsic complexity of the software being tested [16]. As a result of this intrinsic complexity, software companies face serious challenges in testing their products, and these challenges are increasing as software grows more complex [17]. Distributed Software Test teams are a recent trend that has emerged due to quality and ease of communication across vast physical locations and the boundless efforts of all corporations to reduce costs associated with software development [18].

Although this is an established practice, there is a need to investigate tests in the context of distributed and global software engineering challenges [19].

However, the literature has produced a set of recent and relevant studies on the topic. Thus, [20] recommend a series of testing team configuration for a set of development and requirement scenarios. [21] describes four case studies in which one of them is offshore software testing; this author investigates the implications of these case studies for trust development in GSD teams. Casey (2003) also identified from a theoretical point of view the essential information and infrastructure required to support effective testing in GSD. On the other hand, there are relevant and recent efforts to integrate expert systems into software testing processes (e.g. [22], [23]).

In spite of the relevance of these studies, this paper focuses on remote testing i.e. testing of software remotely. Specifically, this study aims to assist the Partner-Supplier election for software testing, one of the hard decisions that software project managers must make [24].

3. Remote Testing Outsourcing: Scenario and Needs

There are different tools designed to help GSD embracers in their tasks. However, these tools are devoted to support software testing processes more than to provide a way to connect service providers with potential customers. Taking this into account, SABUMO-dTest bridges the gap to this problem by offering the way to bring testers and contractors into contact and support their collaboration.

The problem of distributed testing involves several practical problems to be solved. There are a number of people involved in the testing process. Simplifying the roles identified in [25], the distributed testing process involves testers and contractors. Testers have different profiles according their background and role in the project. They are also located in different places. There are a number of tests to be carried out by testers. These tests must be classified into several types and have to be carried out by different testers according to their competence, cost and availability.

All testers must have access to the pertinent software artifacts in order to perform the tests. They must also collaborate with the other members of the project along with the project manager. Secondly, testers need a way to interchange information as well as results of the tests. Finally, there is a need for counting on with an environment to represent the tests and their results as well as the organization of the testing process.

The inclusion of outsourcing remote testing partners in the problem implies the need to locate the most adequate testers for a given software component or project. Thus, the proposed framework must allow the intelligent allocation of resources based on the representation of testing processes and testers. Collaborative work problems should be solved by means of well-known state of the art tools, but the possibility of finding the most accurate testing partner has not been covered in the literature. A common platform that combines the possibility of contact with the most adequate testing providers as well as the tools to orchestrate the collaboration among testing stakeholders can ease the remote testing process as a whole.

4. Remote Testing Outsourcing: Knowledge Representation

As a basis for a knowledge-based system, defining a model which supports the representation of the knowledge relative to the domain is necessary. In the case of distributed testing, we have selected semantic technologies. Semantic technologies present a solution to knowledge codification, and they have impacted on knowledge representation and knowledge management during the last few years [14]. Semantic technologies are based on the use of ontologies. Ontologies are aimed to establish ontological agreements, which serve as the basis for communication between either human or software agents, hence, reducing language ambiguity and knowledge differences between agents, which may lead to errors, misunderstandings and inefficiencies [26]. Given that, ontology creation and management related processes are required for defining and developing semantic services [27].

Based on the premises of authors' previous research, the knowledge will be represented by means of the SABUMO ontology, concretely the concept of Process and its related concepts [14] adapted with the concepts relative to the testing domain ontologies, in order to adapt the previous framework and permit information sharing and rule-based reasoning. The main contribution of is the adaptation of the concepts of the previous framework to the new domain and the inclusion of the test concepts of the ontology.

Figure 1 depicts the main elements of the SABUMO-dTest ontology. Two main groups of concepts must be represented: on the one hand the elements to be tested with their corresponding test processes and, on the other hand, the testers who can test such elements with their competences. Testers are characterized by a number of competences that present competence levels. These competences determine the capacity of the tester in order to select the most adequate tester for a given project. The concrete concepts defined in the ontology are:

- Tester: represents testers who can participate in a testing. Contact information and location are two of their main attributes.
    - Individual: represents a person who participates in the project as an individual, with or without an entailment with an organization.
    - Company: represents a company as a generic tester that provides testing services for software projects.
- Competence: represents the competences that characterize testers. Competences are the skills, attitudes and knowledge that a tester presents at a given level. In the case of companies, this concept is referred to those that the company can provide through their personnel.
- Competence level: represents different levels for each competence.

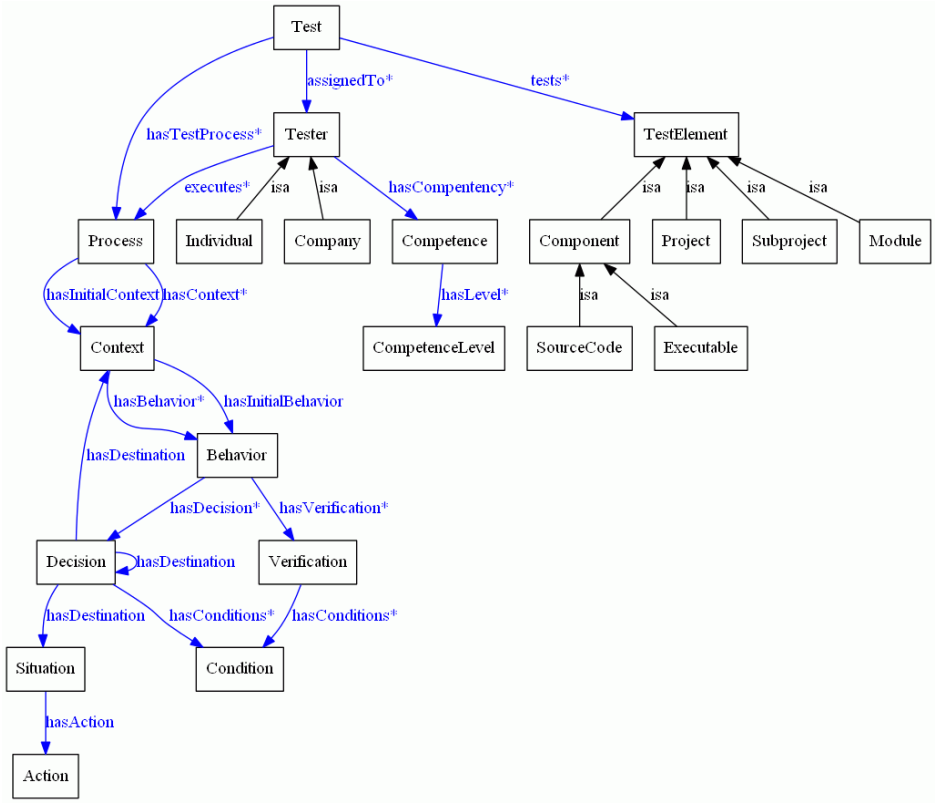

Fig. 1. . SABUMO-dTest ontology (partial view)

Each tester is able to perform different types of tests. Both tests and testers are related to their corresponding domain ontologies. In this way, all projects, contractors and service providers are characterized by means of common terms of the ontology. It will allow the selection of the best candidates for a given element to be assessed.

The entity Test represents the tests to be carried out with one or more test elements. According to the software engineering literature, the tests can be classified as Unit tests, Integration tests, System Integration tests, System tests and Deployment tests. The elements to be tested have been subdivided into four categories: whole projects, subprojects, modules and components. Components can be source code or executable code. In what follows, the main concepts are defined:

- Test: represents the tests to be carried out in a given software project.
    - Unit tests: this kind of test covers single software components. These components can be source code or executable code.
    - Integration test: it tests two or more components working together.
    - System Integration test: it tests the integration of the whole software modules of the project working together as a system.
    - System test: tests the functionality of the system as a whole, taking the user requirements as guidelines.

    - Deployment test: it tests the correct deployment of the software system in the execution environment.
- Test Element: represents the main products obtained in the software project that should be tested.
    - Component: is the minimum functional entity of the software. Usually it is related to single functions or class methods.
    - Module: represents a set of components with related functionality.
    - Project: represents the complete software project.
    - Subproject: represents a part of the project that can be considered separately with respect to the whole functionality.

Each test has related one or more test processes which guides the tester during the test phase. Of course, testers should have sufficient capacity for dealing with the testing phase. However, the definition of the testing processes will allow contractor to guide the activity of their service providers. Thus, the remainder of the elements of the ontology are based on the definition of test processes. The knowledge elements have been adapted from the previous works of the authors and summarized as follows:

- Process. This concept represents the test process as a whole. A test process is composed by a number of test cases (contexts, situations and actions) that includes information about the test to do and the results obtained. Each individual in this concept is directly related to one developer. Thus, the same process could be used for testing several elements from the same organization. Each process has a set of requirements and can be focused to specific test elements. Thus a Usability Test would be represented by a process related to one developer and an element to be tested (for example a module for managing an user profile in a web). The concept of process can be more specific, defining processes for concrete aspects of small components (e.g. testing the communications of a component), or more general as shown in the usability test example.
- Context. This represents the set of conditions which determines the testing process. This context is determined by the general characteristics related to the project as well as the specific characteristics arising during the previous steps of the testing process. Let's suppose a process for testing a module. The module can be subdivided by several sub-modules according its functionality. The specific conditions that involve the testing process of each sub-module would be represented by means of Contexts. Each context contains the specific test cases to be executed, and they will be represented by means of situations.
- Situation. This represents each step to be taken in the testing process. It is linked to an action to execute during the test process. For example, let's suppose the testing process of the module of user profile management. Testing this module would involve a number of functionalities characterized by one or more test cases, with related-information and actions to do. Each of these functionalities would be represented by situations. Each situation have an action related, as described next.
- Action. This represents an action to be executed in the testing process. . It allows the execution of the action by a tester or by the system itself. The result of this execution generates information to be processed. For example, in the

context of testing the communications of the user profile module, one situation could represent the testing of receive information from an external social network. This situation determines the preconditions and the characteristics of the process that the tester should execute. The action is executed by the tester and the information generated could be, for example, the result of the test (success or fail), the time required for the operation.

- Decision. This represents the rules related to a given situation. The decision defines the next situation in the test process. Decisions allow evaluation of the state of the test process in order to decide the next step. For example, the test described in the previous concept generates two pieces of data: the result of the test and the time required. For example, if the test successes (the information of the social network is retrieved) but the response time is greater than 2 seconds, then the process change to a new situation in which the developer should review the component and the tester should test again the component.

It is remarkable the flexibility of the representation model, allowing from high level definition of testing processes to low-detail tests in which each situation is a specific test case for a function.

5. SABUMO-dTest: Architecture and Implementation

The architecture of SABUMO-dTest (see Figure 2) is based on a well-established framework which allows the semantic definition of business processes [14]. The knowledge representation has been adapted to the distributed testing domain, taking into consideration the representation of the test elements as well as the different actors that participate in the testing process. The selection of the most adequate testers is driven by the competences defined for the processes. Collaboration between the actors in the recommendations and the definition of the test processes are worth highlighting. The main elements that are rooted in successful applications of this architectural approach (e.g. [28]; [29]; [30]; [31]; [32]; [33]) are described in what follows.

5.1. Collaboration Layer

The collaboration layer is one of the main points for the SABUMO-dTest implementation. The new approach introduces more aspects in the collaboration between users. In fact, the interaction between users is twofold: on the one hand, companies and service providers must establish collaboration relationships; on the other hand, contractors and testing service providers collaborate on the definition and execution of testing processes. For this reason, the collaboration layer provides, firstly, the way to characterize the different users who participate and, secondly, the tools for representing and executing the testing processes.

Finding the appropriate tester or set of testers for a given test element and test process is, in fact, the main problem to be solved. The collaboration layer permits characterization of each player in the process as well as the tests to be defined. The

characterization of actors and test elements are based on the competences defined in [25]. Thus, each tester defines within their own profile the competences with their corresponding level. Contractors must also define the competences and competence-levels required for the tests they need. In this way, it is possible to find the best candidates for a given test, according to the requirements of the process.

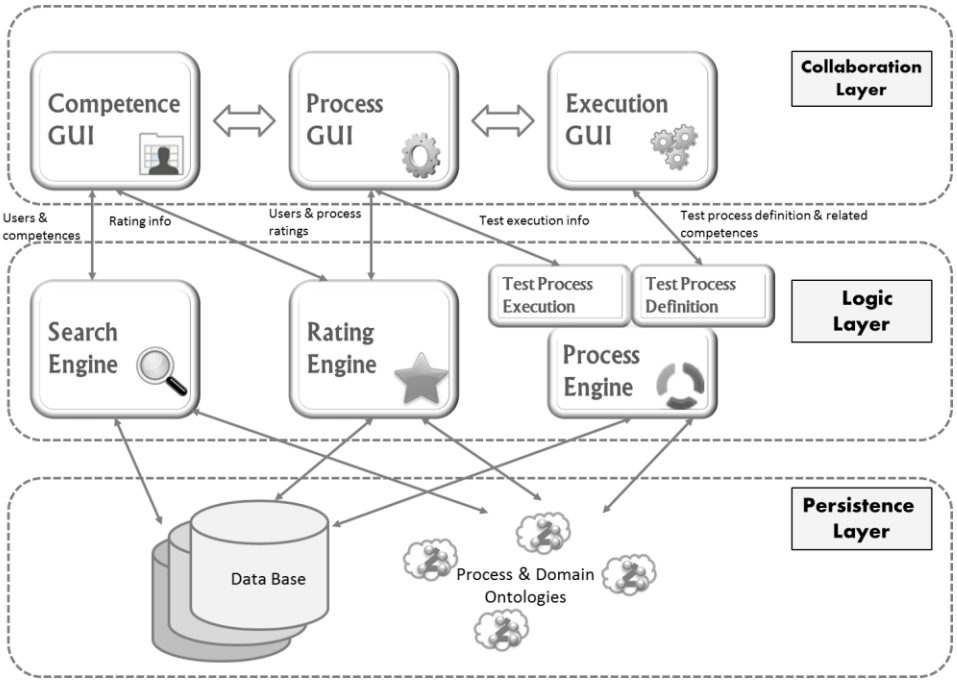

Fig. 2. . SABUMO-dTest architecture

The testing processes are based on the actions to be executed by means of web interactions and characterized by the competences required. Thus, when a process is defined it can be published in the tool. Moreover, the web environment is accessible for all users. Each of them can execute processes and rate them. In this way, contractors define testing processes and label these processes with required competences, and testers execute the testing process, providing a rating for both process and actions. This mechanism provides necessary feedback for the improvement of the processes.

So far two different user profiles have been mentioned: testers and contractors. Each of them has different roles in the process, but they have in common the testing process itself.

Given this setup, the collaboration layer is formed by:

- Competences GUI. This element permits interaction between contractors and testers, in order to find testers who test a piece of software according to a number of competences and competence levels required. This component allows communication between users as well as establishing relationships between them according their competences. Each tester and contractor has a profile characterized according to competences and competence levels. In this

way, each tester is related to a number of concepts according to their expertise and interests that help find the best processes and testers. Competence GUI also allows contractors to contact the testers related to the topics of a given project in order to carry out a testing process. Once an agreement between testers and contractors has been reached, the test process is assigned to the tester. Finally, this GUI includes a module relative to the assessment of testing processes, contractors and testers, allowing for the sharing of opinions and valuations in order to ease the search for the most adequate testing partner.

- Process Definition GUI. This element allows the definition of the workflow of the testing processes based on the ontology described earlier. Contractors define the testing processes according to software requirements and the competences and competence levels required for execution. Each testing process is characterized according to such competences in order to ease their location and the assignment of testers.
- Testing GUI. This element allows testers to follow the workflow of the testing processes defined by the developer. Testers connect to the system and execute the testing processes assigned to them by contractors.

The annotation is based on OWL ontologies and the process of annotation is embedded in the interface because it is present in Competences GUI (characterization of testers and contractors) and Process and Testing GUI (characterization of processes).

Once the testing processes have been defined and labeled, they are published and assigned to one or several testers. The testing environment is based on [14]. After the execution of the testing processes, the tester rates the processes and the contractor in order to establish a public valuation of each. Moreover, after the execution of the testing process, contractors rate testers in order to establish a public valuation of this stakeholder. The rating system will help users to find the best candidates for a given task as well as to improve the testing processes based on the feedback of the users involved in the project.

5.2. Logic Layer

As mentioned before, the annotation is based on domain ontologies. Thanks to the interaction of these ontologies and the competences defined, all elements are inter-related and it allows intelligent searching.

The logic layer contains the engines that ensure the operation of the system. First of all this layer is in charge of performing the intelligent searching of the best testers for each test process. Secondly, this layer allows the execution of the processes defined by the contractors, tracing and storing the results. Finally, based on the evaluation of testers and contractors, this layer allows the calculation of the valuation of each tester and process in order to improve the quality of the recommendations.

This layer consists of three components detailed below:

- Process Engine. The process engine is in charge of the execution of the workflow of the testing processes defined by the developers. The process is defined in order to guide the tester in the testing process. Then, the testing process shows the steps to be taken by testers and will register the evolution of

its execution with the feedback provided by testers. Despite the fact that some testing tasks can be automatized, according to the characteristics of the ontology [14], full automation will be addressed in a future study. Thus, testers should execute assigned tasks and provide feedback on the results.

- Search Engine. The search engine allows searching users and processes based on the competences and competence-levels required as well as the functional environment related to the domain ontologies. Thus, the search process is based on competences, annotations and ratings. The framework includes the option of establishing a threshold in the rating in order to provide more accurate results. For instance, contractors can search for testers rated with more than 5 points. Thanks to this, contractors can identify the most competent and best rated testers in their areas of interest according to the requirements of the process. Likewise, the best testing processes can be identified by means of the ratings obtained in order to improve the results of future projects.
- Rating Engine. The rating engine allows the rating of testers, contractors and testing processes. This engine automatically updates the rating of each element according the valuation of the users involved.

5.3. Persistence Layer

Finally the persistence layer provides the logic layer with permanent storage of data for the process definition, annotations, ratings and the execution of the testing processes. Processes and domain ontologies are represented and stored in OWL format, while annotations and ratings mix the OWL storage and a conventional database system in order to improve the system performance. This hybrid approach is based on the fact that the results of the process execution are stored in a database system. Hence, the database of annotations and ratings links the OWL storage with the database system.

6. Evaluation

6.1. Design

Given that SABUMO-dTest is aimed to be tested in a distributed environment, the evaluation proposed has been carried out by comparing a set of similar projects in which testing was performed outsourced. In order to do so, a collection of direct measures was taken from the set of projects described in the Sample section. These direct measures included conventional productivity metrics for software projects, specifically, Function points; number of defects discovered before, during and after testing process; defect detection efficiency (number of defects detected / total number of defects). Later, two combined metrics were calculated: defects per function point and

defects detected in testing per function point. This latter will shed some light about the efficiency of the approach presented in the paper. Finally, a set of data was harvested regarding collaboration history between contractor and supplier, namely, number of projects working together and satisfaction (before the current project).

Given that a set of comparable projects were considered (some of them using SABUMO-dTest pilot and others not), this setup permits comparison between the two scenarios. Table 1 shows the detailed information relative to the set of projects.

Table 1. Results of the application of SABUMO-dTest

	Without framework				SABUMO-dTest			
	P1	P2	P3	P4	P5	P6	P7	P8
Function points	512	643	361	445	498	598	365	438
Defects discovered before testing	701	892	543	523	725	798	539	592
Defects discovered during testing	1065	1115	654	891	922	1187	721	921
Defects discovered after testing	72	105	94	73	19	47	65	47
Total defects discovered	1838	2112	1291	1487	1666	2032	1325	1560
Defect detection efficiency	96.08 %	95.03 %	92.72 %	95.09 %	98.86 %	97.69 %	95.09 %	96.99 %
Defects per function point	3.59	3.28	3.58	3.34	3.35	3.40	3.63	3.56
Defects detected in testing per function point	2.08	1.73	1.81	2.00	1.85	1.98	1.98	2.10
Collaboration: number of projects	3	1	2	3	4	5	1	0
Collaboration: satisfaction	5	4	3	4	5	5	5	

6.2. Sample Description

The sample was composed of a set of eight different projects developed by two different companies based in Spain. This set of projects was taken from a collection of potential participants who responded positively to a personal invitation sent by the authors to professional contacts. Projects were analyzed to form a coherent sample in terms of project size, type and complexity. Regarding testing partners, the sample included four different testing partners. In order to isolate the three distances (temporal, cultural and geographical), all testing partners were Spaniards. Every partner participated in two different projects, in one the partner was assigned by means of SABUMO-dTest and in the other not.

6.3. Data Collection

Data collection was conducted through a questionnaire. Most of the data to be coded was available in post-mortem documents related to the projects. As stated before, data included adjusted function points (using IFPUG Value Adjustment Equation), number of defects discovered before, during and after the testing process and finally, defect detection efficiency. Two measures were also calculated from this data: Defects per function point and Defects detected in testing per function point. Regarding collaboration history, number of projects in the collaboration track and overall satisfaction of previous projects were coded. Regarding the latter, responses were coded using a 1-6 Likert-type scale (1= Extremely dissatisfied; 2= Very dissatisfied; 3= Somewhat dissatisfied; 4= Somewhat satisfied; 5= Very satisfied; 6= Extremely satisfied).

Printed questionnaires were designed to be completed by software project managers, assisted on site by a researcher who gave the respondents all the instructions they need to fill out the questionnaire. Subsequently, responses were codified using a statistical analysis software tool.

6.4. Threats to Validity

Regarding internal validity, it is concerned with correctly concluding that an independent variable is, in fact, responsible for variation in the dependent variable, in this case, defect detection efficiency. In this particular case, results provide an acceptable level of internal validity as the independent variables included in the questionnaire are based on a literature review on the topic along with authors' previous studies on software productivity and estimation (e.g. [34], [35]).

External validity is concerned with the generalizability of research findings to and across populations of participants and settings. The authors face two possible threats. The first is the limited number of projects analyzed, which complicates generalization of the results. The second is subject representativeness. Although both threats are notable, due to the nature of the study, the authors consider the design of the study acceptable.

7. Results and Discussion

Table 1 shows results from evaluation. Projects are divided into two groups: the ones in which testing was not determined by SABUMO-dTest and the ones developed under its influence. Regarding the size of the projects, the mean is 482 function points with a standard deviation of 101. However, if standard deviation is calculated only taking into account projects developed in the same company (P1, P2, P5 & P6 on the one hand and P3, P4, P7 & P8 on the other hand) this is 69.40 and 45.44 points respectively. Defect detection efficiency ranges from 92.72% (P3) to 98.86% (P5). In general all figures related to defect detection efficiency are higher for SABUMO-dTest environments.

Regarding collaboration history, the number of projects range from zero in the case of P8 to 5 (P6). It is important to note that SABUMO-dTest suggested a partner with no collaboration history with the contractor. This means that, in spite of not having a common background, the system suggested a partner with a competent profile in the testing needed for P8; defect testing efficiency reaches the third best score with a remarkable 96.99% of defects found.

Although the sample is quite small, some interesting conclusions can be drawn from data available. The first conclusion is that defect detection efficiency is better in SABUMO-dTest than in the absence of it, the average being 97.16% over 95.20%. With the aim of verifying whether SABUMO-dTest users obtained results significantly better than non-users, the statistical method Student's t-test (comparison of two means) was used to carry out one-way between-groups analysis of variance. The level of statistical significance was set at 0.05. The results indicate that SABUMO-dTest users do not present statistically significant differences with non-users ($t(6)=-2.283$, $p>.05$). In any case, the average of defect detection efficiency metric is higher in SABUMO-dTest projects.

The second finding is the irregular distribution of the calculated metrics “Defects per function point” and “Defects detected in testing per function point”. For the first metric, figures range from 3.28 to 3.63. In this case, defects introduced appear randomly in projects no matter if they are using the tool or not. Given that SABUMO-dTest is focused on the testing stage, it does not affect the introduction of defects. Therefore, for “Defects detected in testing per function point”, the conclusion is the same, given that this metric depends on the number of defects introduced. However, a new metric can be calculated counting “Defects discovered after testing” and Function points. Thus, these measures are P1:0.080; P2:0.163; P3:0.260; P4:0.157; P5:0.038; P6:0.079; P7:0.178; P8:0.107. A quick look at results suggests that measures are again comparable, but slightly better in the case of SABUMO-dTest users, with P5 showing the minimum figure with 0.038 defects not detected in testing per function point. However, this difference cannot be considered statistically significant given Student's t-test results ($t(6)=-1.367$, $p>.05$).

Finally, it is important to note that SABUMO-dTest provides recommendations for testing partners that are normally highly ranked in satisfaction (P5=5, P6=5, P7=5), but also takes into account that high competence with no previous interaction could be attractive for testing contractors.

8. Conclusions and Future work

This research is focused on remote testing, which is related to testing of software remotely. More precisely, this study aims to assist the Partner-Supplier election for software testing, one of the hard decisions that software project managers must make. In this sense, the authors present SABUMO-dTest, a framework based on Semantic technologies that allows software organizations to represent testing processes with the final aim of trading their services or modeling their testing needs in a social and competitive environment. SABUMO-dTest, benefits from a set of shared and

controlled vocabularies that permit knowledge and processes sharing between potential partners, experts and testing service providers.

As has been explained, the architecture of SABUMO-dTest is based on a well-established framework which allows the semantic definition of business processes [14]. The knowledge representation has been adapted to the distributed testing domain, taking into consideration the representation of the test elements as well as the different actors who participate in the testing process. The selection of the most adequate testers is driven by the competences defined for the processes. Collaboration between the actors in the recommendations and the definition of the test processes is worth mentioning. .

Finally, the evaluation proposed has been carried out by comparing a set of similar projects in which testing was performed outsourced, i.e. in a distributed environment. Projects were divided into two groups: the ones in which testing were not determined by SABUMO-dTest and the ones developed under its influence. The main conclusion obtained is that defect detection efficiency is slightly better in SABUMO-dTest than in its absence, the average being 97.16% over 95.20%. This small improvement can be improved in less mature environments in terms of defect introduction. Furthermore, the results suggest that the others measures included in the evaluation are comparable, but slightly better in the case of SABUMO-dTest users.

In future research, the inclusion of the remaining phases of GSD in the SABUMO architecture could improve the overall process, allowing identification of the best partners and the communication between the distributed members of the project, as well as the control by defining the workflow of each project phase using adaptations of the proposed ontology. It is also intended to calculate correlations among several factors of the model (e.g. defect detection efficiency and collaboration satisfaction before and after SABUMO-dTest project). Finally, the inclusion of the Linked Data paradigm in SABUMO-dTest would open up the possibilities of finding possible testers and topics related to new projects in larger datasets.

References

1. Prikladnicki, R., Audy, J.L.N., Shull, F.: Patterns in Effective Distributed Software Development. *IEEE Software*, vol. 27, no. 2, 12 –15, (2010).
2. Palacio, R.R., Vizcaíno, A., Morán, A.L., González, V.M.,: Tool to facilitate appropriate interaction in global software development," *IET Software*, vol. 5, no. 2, 157–171, (2011).
3. Holmström, H., Fitzgerald, B., Ågerfalk, P.J., Conchúir, E.O.: Agile Practices Reduce Distance in Global Software Development, *Information Systems Management*, vol. 23, no. 3, 7–18, (2006).
4. Herbsleb J. D., Moitra, D.: Global software development, *IEEE Software*, vol. 18, no. 2, 16 –20, (2001).
5. Noll, J., Beecham, S., Richardson, I.: Global software development and collaboration: barriers and solutions, *ACM Inroads*, vol. 1, no. 3, 66–78, (2011).
6. Johri, A.: Sociomaterial bricolage: The creation of location-spanning work practices by global software developers, *Information and Software Technology*, vol. 53, no. 9, 955–968, (2011).

7. Misra, S., Colomo-Palacios, R., Pusatli, T., Soto-Acosta, P.: A discussion on the role of people in global software development, *Tehnički vjesnik*, vol. 20, no. 3, 525, 525–531, 531, (2013).
8. Avritzer, A., Paulish, D., Cai, Y., Sethi, K.: Coordination implications of software architecture in a global software development project, *Journal of Systems and Software*, vol. 83, no. 10, 1881–1895, (2010).
9. Milewski, A.E. Tremaine, M., Köbler, F., Egan, R., Zhang, S., O'Sullivan, P.: Guidelines for effective eridging in global software engineering, *Software Process: Improvement and Practice*, vol. 13, no. 6, 477–492, (2008).
10. Casado-Lumbreras, C., Colomo-Palacios, R., Soto-Acosta, P., Misra, S.: Culture dimensions in software development industry: The effects of mentoring. *Scientific Research and Essays*, vol. 6, no. 11, 2403–2412, (2011).
11. Conchúir, E. O., Holmström Olsson, H., Ågerfalk, P. J., Fitzgerald, B.: Benefits of global software development: exploring the unexplored. *Software Process: Improvement and Practice*, vol. 14, no. 4, 201–212, (2009).
12. Patil, S., Kobsa, A., John, A., Seligmann, D.: Methodological reflections on a field study of a globally distributed software project. *Information and Software Technology*, vol. 53, no. 9, 969–980, (2011).
13. Khan, S.U., Niazi, M., Ahmad, R.: Factors influencing clients in the selection of offshore software outsourcing vendors: An exploratory study using a systematic literature review. *Journal of Systems and Software*, vol. 84, no. 4, 686–699, (2011).
14. López-Cuadrado, J.L., Colomo-Palacios, R., González-Carrasco, I., García-Crespo, A., Ruiz-Mezcua, B.: SABUMO: Towards a collaborative and semantic framework for knowledge sharing. *Expert Systems with Applications*, vol. 39, no. 10, 8671 – 8680, (2012).
15. Catelani, M., Ciani, L., Scarano, V.L., Bacioccola, A.: Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use. *Computer Standards & Interfaces*, vol. 33, no. 2, 152–158, (2011).
16. Dhavachelvan, P., Uma, G.V. Venkatachalapathy, V.S.K.: A new approach in development of distributed framework for automated software testing using agents. *Knowledge-Based Systems*, vol. 19, no. 4, 235–247, (2006).
17. Whittaker, J.A.: What is software testing? And why is it so hard?. *IEEE Software*, vol. 17, no. 1, 70–79, (2000).
18. Hossain, L., Zhu, D.: Social networks and coordination performance of distributed software development teams. *The Journal of High Technology Management Research*, vol. 20, no. 1, 52–61, (2009).
19. Sangwan R.S., Laplante, P.A.: Test-Driven Development in Large Projects. *IT Professional*, vol. 8, no. 5, 25–29, (2006).
20. Yu L., Mishra, A.: Risk Analysis of Global Software Development and Proposed Solutions. *AUTOMATIKA: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije*, vol. 51, no. 1, 89–98, (2010).
21. Casey, V.: Global Software Development Testing Infrastructure. *The proceedings of the 9th European Software Engineering Conference*, Helsinki, Finland, (2003).
22. Banzi, A.S., Nobre, T., Pinheiro, G.B., Árias, J.C.G., Pozo, A., Vergilio, S.R.,: Selecting mutation operators with a multiobjective approach. *Expert Systems with Applications*, vol. 39, no. 15, 12131–12142, (2012).
23. Ding, S., Yang, S.L., Fu, C.,: A novel evidential reasoning based method for software trustworthiness evaluation under the uncertain and unreliable environment," *Expert Systems with Applications*, vol. 39, no. 3, 2700–2709, (2012).
24. Garcia-Crespo, A., Colomo-Palacios, R., Soto-Acosta, P., Ruano-Mayoral, M.: A qualitative study of hard decision making in managing global software development teams. *Information Systems Management*, vol. 27, no. 3, 247–252, (2010).

25. Saldaña-Ramos, J., Sanz-Esteban, A., García-Guzmán, J., Amescua, A.: Design of a competence model for testing teams. *IET Software*, vol. 6, no. 5, 405–415, (2012).
26. Blanco, C., Lasheras, J., Fernández-Medina, E., Valencia-García, R., Toval, A.: Basis for an integrated security ontology according to a systematic review of existing proposals. *Computer Standards & Interfaces*, vol. 33, no. 4, 372–388, (2011).
27. García-Peñalvo, F.J. Colomo-Palacios, R. García, J., Therón, R.: Towards an ontology modeling tool. A validation in software engineering scenarios. *Expert Systems with Applications*, vol. 39, no. 13, 11468–11478. (2012).
28. Colomo-Palacios, R., García-Crespo, A., Soto-Acosta, P., Ruano-Mayoral, M., Jiménez-López, D.: A case analysis of semantic technologies for R&D intermediation information management. *International Journal of Information Management*, vol. 30, no. 5, 465–469, (2010).
29. García-Crespo, A., Colomo-Palacios, R., Gómez-Berbís, J.M., Ruiz-Mezcua, B.: SEMO: a framework for customer social networks analysis based on semantics," *Journal of Information Technology*, vol. 25, no. 2, 178–188, (2010).
30. García-Crespo, A. López-Cuadrado, J.L. González-Carrasco, I. Colomo-Palacios, R., Ruiz-Mezcua, B.: Sem-Fit: A semantic based expert system to provide recommendations in the tourism domain. *Expert systems with applications*, vol. 38, no. 10, 13310–13319, (2011).
31. García-Crespo, A. López-Cuadrado, J.L. González-Carrasco, I. Colomo-Palacios, R., Ruiz-Mezcua, B.: SINVLIO: Using semantics and fuzzy logic to provide individual investment portfolio recommendations. *Knowledge-Based Systems*, vol. 27, no. 1, 103–118, (2012).
32. García-Peñalvo, F. J., Colomo-Palacios, R., Soto-Acosta, P. Martínez-Conesa, I., Serradell-López, E.: SemSEDoc: Utilización de tecnologías semánticas en el aprovechamiento de los repositorios documentales de los proyectos de desarrollo de software. *Information Research*, vol. 16, no. 4, paper 504, (2011).
33. González-Carrasco, I. Colomo-Palacios, R., López-Cuadrado, J.L. García Crespo, A., Ruiz-Mezcua, B.: PB-ADVISOR: A private banking multi-investment portfolio advisor. *Information Sciences*, vol. 206, 63–82, (2012).
34. Colomo-Palacios, R. Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F.J., Tovar, E.: Project managers in global software development teams: a study of the effects on productivity and performance. *Software Quality Journal*, 1–17.
35. González-Carrasco, I. Colomo-Palacios, R., López-Cuadrado, J.L. García-Peñalvo, F.J.: SEffEst: Effort estimation in software projects using fuzzy logic and neural networks. *International Journal of Computational Intelligence Systems*, vol. 5, no. 4, 679–699. (2012).

Ricardo Colomo-Palacios is an Associate Professor at the Computer Science Department of the Universidad Carlos III de Madrid. His research interests include applied research in Information Systems, software project management, people in software projects and future web. He received his PhD in Computer Science from the Universidad Politécnica of Madrid (2005). He also holds a MBA from the Instituto de Empresa (2002). He has been working as Software Engineer, Project Manager and Software Engineering Consultant in several companies including Spanish IT leader INDRA. He is also an Editorial Board Member and Associate Editor for several international journals and conferences and Editor in Chief of International Journal of Human Capital and Information Technology Professionals.

Jose Luis Lopez-Cuadrado is assistant professor in the Computer Science Department at the Universidad Carlos III of Madrid. He holds his PhD degree in Computer Science by this University. His research is focused on web based expert systems applications, neural networks, software engineering and processes improvement. Also he is co-author of several contributions published in international congresses and journals.

Israel Gonzalez-Carrasco is an assistant professor in the Computer Science Department of Universidad Carlos III of Madrid. He holds his PhD degree in Computer Science by this University. He is co-author of several papers in international journals and conferences and his main lines of research are Neural Networks, Expert Systems and Software Engineering. He is involved in international projects and he is also an Editorial Board and Review Board Member for several international journals.

Francisco J. García-Peñalvo received a PhD in Computer Science (2000) from the University of Salamanca, Spain. He works as a Teacher in the Computer Science Department of the University of Salamanca. He is the Director of the research GRoup in InterAction and e-Learning (GRIAL). His main research interests are e-Learning systems, web engineering, semantic web, human–computer interaction and software reuse.

Received: January 29, 2013; Accepted: October 25, 2013