International Journal of Computer Science and Software Engineering (IJCSSE), Volume 5, Issue 12, December 2016
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 291-296

Contextualization of Defect Management in Data Integration

Hassane TAHIR[1] and Patrick BREZILLON[2]

[1, 2] LIP6, University Pierre & Marie Curie (UPMC), 4 Place Jussieu, 75005, Paris, France

[1]hassanetahir@hotmail.com, [2]patrick.Brezillon@lip6.fr

ABSTRACT

The management of defect tracking is mostly useful for any software of an organization. In data integration project, as development of ETL (Extraction, Transformation and Loading) processes is completed, the testing phase will be started. If bugs are found, test engineers can log such bugs in a bug tracking system to the different actors: Developers, DBA (A database Administrator), etc. For instance, in the case of database bugs, after the first cycle of bug tracking is completed, the system will notify the DBA. The DBA can log in to system and get the bug list with priority. He can then solve the bug and change the status of that bug in the system. The problem is that most of the time, only technical factors are taken into account. Contextual elements about Human and social factors are not considered (i.e. experience of actors). Therefore it is important to add the context in which bug tracking tasks should be performed. This paper proposes to use "Contextual Graphs" formalism to improve existing procedures for bug tracking in a data migration project.

Keywords: *Context, Contextual Graphs, Data Integration, Defect tracking, Procedures.*

1. INTRODUCTION

Developers, Test Engineers, database and system administrators use Bug Tracking Systems to record and track the progress of bugs (defects). Specific features should be understood to evaluate a bug tacking system. In data integration project, as development of ETL (Extraction, Transformation and Loading) processes is completed, the testing phase will be started. If bugs are found, test engineers can load log such bugs in a bug tracking system to the different actors: Developers, DBA (A database Administrator), etc. For instance, in the case of database bugs, after the first cycle of bug tracking is completed, the system will notify the DBA. The DBA can log in to system and get the bug list with priority. He can then solve the bug and change the status of that bug in the system. The problem is that most of the time, only technical factors are taken into account. Contextual elements about Human and social factors are not considered.

Contextual elements are relevant at a given time (e.g. memory size, hard drives), and the values taken by these contextual elements at that moment: (memory size: 70%, full, hard drives: HP-1, IBM-23). The DBA often developed practices to manage these contextual elements in order to solve the problem at hand. Practices encompass what the users do with procedures. We can point two categories of problems: technical and social. Technical problems can impact the performance of the entire information system of the company. This includes problems due to the database, the server, the network and/or the application. For instance, one of the most important database problems is when users are unable to connect to the database because of a locked account, slow time response or bad performance, and sometimes because the database is down. Social problems are mainly due to bad communications and collaborations with other users. Another example that we can give concerns some collaboration problems due to the bad collaboration between DBA and other actors. In some cases, developers do not cooperate with a DBA to solve database errors due to a bad application coding. The reason for this is that developers may not feel comfortable while their code is being reviewed if their managers are invited.

Defect tracking and processing must be integrated in the data migration project life cycle and the testing process. A defect management process is used to decide what is to become of software defects, or bugs, found in the development cycle for migrating data into the target system. Fig. 1 illustrates the process of defect tracking. In this example, the prevalent defect, or bug process can have three basic states:

1. Submitted/Opened
2. Resolved
3. Closed

This work relies on the Contextual-Graphs formalism [4] for improving bug-tracking process in a data migration process. The main advantage of Contextual Graphs is the possibility to enrich incrementally the system with new knowledge and practice learning capability when needed. Moreover, a contextual graph is a good communication tool for helping the different actors of the project to exchange their experiences and viewpoints when solving defects. The paper begins by the description of a case study about data migration project: actors involved in the defect tracking, contextual elements. After, we present related works in the literature. Then we present the main features of the used approach followed by a presentation of contextual graph platform. Finally we conclude and evaluate our work.

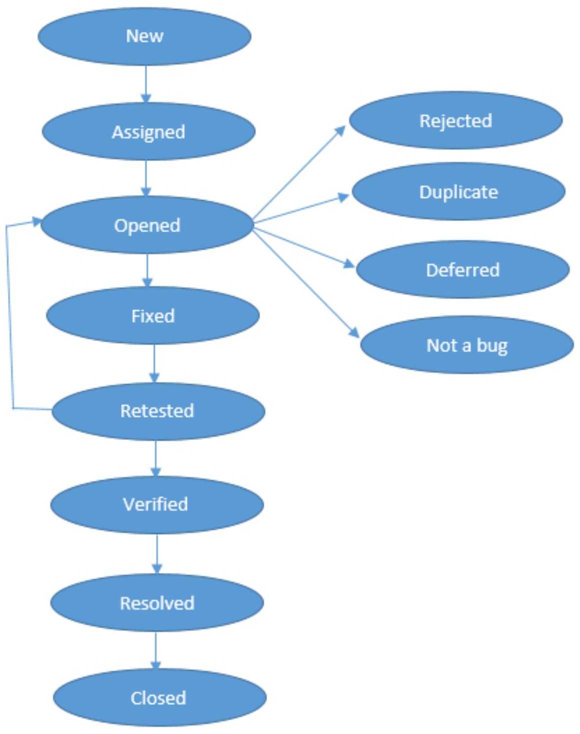


Fig. 1. Example of process of defect tracking.

2. A CASE STUDY

This case study presents data migration process where defects are processed. Data has been processed using ETL (Extraction, Transformation and Loading). The ETL functionality includes the following steps (a) Identifying relevant data in the source systems, (b) Extracting the required relevant data, (c) Customizing and integrating data coming from multiple sources into a common format, (d) Cleaning the resulting data set according to the database and business rules, and (e) Propagating and loading of the data into a target system.
The ETL process can involve a great complexity, and critical operational problems that can appear with bad and improperly design. Each ETL system depends on a Database Management System (DBMS), which is composed of a set of subsystems executing specific tasks and compete for system resources allocated by the DBMS (Fig. 2). Actors in the defect tracking of data migration are: Project Manager of Data Migration (Main Actor), Developers, DBAs, Business Analysts; In this case study, we focus on DBA problems. Many questions may be asked by actors involved in the ETL process. They concern some of the different contextual elements that intervene in the different phases of the ETL process (with their known values).

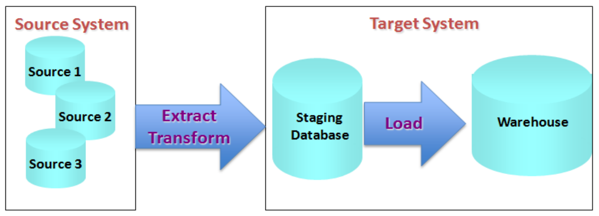


Fig. 2. ETL process.

When any actor (i.e Developer or a DBA) presents a bug report, most probably, he is asked many questions. Some of them are what is the name of the product? What is the defect? In which component is the defect? In which module is the defect? In which method the defect is? in which environment the defect arises? In which platform the application is created? In which OS the application runs? The information given by the DBA who reports defect might be incomplete initially.
The following are examples of defects contextual elements:

- What is the response time? (i.e. excellent, good or bad)
- Are performance problems identified?
- Should indexes be dropped and recreated, respectively before and after each data loading and how?

Other contextual elements are shown in Table 1.

Table 1: Contextual elements

Contextual element	Contextual element Values
Missing Data	Bad data from the source database Invalid joins
Truncation of Data	Invalid field lengths on target database
Data Type Mismatch	Source data field not configured correctly
Null Translation	Absence of the null translation in the transformation
Wrong Translation	Incorrectly translated the source field
Misplaced Data	Wrong mapping between the source and target data
Extra Records	Developer did not include filter in their code
Not Enough Records	Developer had a filter in their code
Undocumented Requirements	Undocumented requirements not understood by actors.
Duplicate Records	No appropriate code to filter out duplicate records
Numeric Field Precision	Developer rounded the numbers to the wrong precision

The following section discusses some of the commonly used approaches to intelligent assistance for database management.

3. RELATED WORK

Defect tracking is a very important activity in software development. Its helps to reduce the cost, resources and time required for rework. Terefore defect detection and prevention are two stages of defect management which helps in improving the quality of software. This section is concerned with some of the related work in software engineering and database management. Many solutions and have been proposed to deal with database administration and incident management as discussed in [5], [7]. Gopalakrishnanm has made analysis of defect detection and prevention techniques which are employed in Agile development [8]. For this analysis data has been gathered from five projects of leading software development companies. The result of the research is that on an average 13 % to 15% of inspection and 25% - 30% of testing out of entire project effort time is required for 99% - 99.75% of defect elimination. (O. Don, 2003) presented a study of how agile development environment involves defect detection and its prevention once a defect is detected [6]. He has discussed two wide categories of defect management: requirements defects and implementation defects. He concluded that Agile practices lack effective defect management but actually agile developments reduce defects in first place. These categories include finding defects in all types of requirements and technical implementation of a project. However context about social factors are not taken into account. (R. JÖRG, 2005) discussed the repeated and sustainable discovery process, handling, and treatment of quality defects in software systems [9]. Information about quality defects found in source code has been stored using an automation language. Automation language also represents the defect and treatment history of small parts of the software products. (S. Abhiraja et al, 2012) discussed in his paper [1] that quality defects have been detected using test case and preventive actions to improve the quality of software process. If the software process is not working correctly then defect is found. Some preventive actions have been employed to avoid the defects like defects classification and discovering the root causes of the defects. (V. Suma, 2011) presented a paper about Defect Management Strategies in Software Development [14]. He described in his research that inspection is significant to discover the static defect close to the origin. (Rajni et al, 2013) presented a study to use defect tracking and defect prevention for the improvement of the quality [12], Testing is performed when the software is developed and defects found are removed using defect prevention. According to Rajni [12], Defect Tracking System still needs improvement and a lot of research is required to mature the Defect Tracking Systems. (Sydney et al., 2009) has used a different technique that is The Defect Management Meeting [15]. In this meeting team members communicate face to face. The meeting is time-boxed to review and prioritize all new defects found. Time-boxing is particularly very helpful when request for change arises late in project and risk of defects increases due to chaning requirement. Main goal of this meeting is to review existing defects.
(K. Ansar, 2013) proposes defect detection and analysis to discover the root causes of potential defects and prevention technique to remove defects [2]. He proposed a defect management process model to produce quality products. This model has been proved very valuable to handle harmful defects. (K. Macros, T. Guilherme, 2008) presented a unique concept of Defect causal analysis (DCA) to recover software development process and to reduce amount of potential defects [10].
The above solutions cannot always successfully handle all defect tracking tasks in multitude of specific new situations and contexts that differ from the set procedures.
• Only physical parameters and sensors are considered;
• Not Human-Centered Context (i.e. Social Context: DBA Profile, experience, Knowledge, Conflict with DEVELOPERS, degree of collaboration between DBA and other shareholders…).
• Bad Context Sharing (i.e. Context is implicit). Defects are not always understood by all actors because contextual elements are not explicit.
For these reasons, we are interested to take context into consideration and incorporate it in the defect tracking procedures. The following section presents how the Contextual Graphs Formalism can help in the contextualization of defect tracking especially when many actors are interacting with each other (i.e. sharing context)

4. CONTEXTUAL-GRAPHS FOR DEFECT TRACKING

4.1 Brief Description of Contextual graphs

A contextual graph (CxG) allows the representation of the different ways to solve a problem. It is a directed graph, acyclic with one input and one output and a general structure of spindle. Each path in a CxG corresponds to a practice, a way to fix the problem. Fig. 3 provides the definition of the four elements in a contextual graph. A more detailed presentation of this

formalism and its implementation can be found in [3], [4] and [11].
A contextual graph is composed of the following elements: actions, contextual elements, activities and temporal branching.
An action is the building block of contextual graphs at the chosen granularity. An action can appear on several paths but it will be in different contexts.
A contextual element is a couple of nodes, a contextual node and a recombination node. A contextual node has one input and N branches [1, N] corresponding to the N instantiations of the contextual element already encountered. The recombination node is [N, 1] and shows that, once items on the branch between the contextual and recombination nodes has been processed, it does not matter to know which branch was followed. Contextual elements are used to represent and implement context about the different events occurring in a given situation.
An activity is a contextual graph by itself that is identified by participants because it appears on different paths and/or in several contextual graphs. This recurring sub-structure is generally considered as a complex action. An activity is a kind a contextualized task that can be aggregated in a unit or expanded in a sub graph according to the needs [13].
A temporal branching expresses the fact (and reduces the complexity of the representation) that several groups of actions must be accomplished but that the order in which action groups must be considered is not important, or even could be done in parallel, but all actions must be accomplished before continuing the practice development. The temporal branching is the expression of a complex contextual element at a lower granularity of the representation.

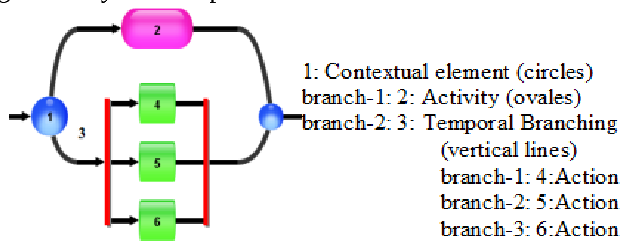
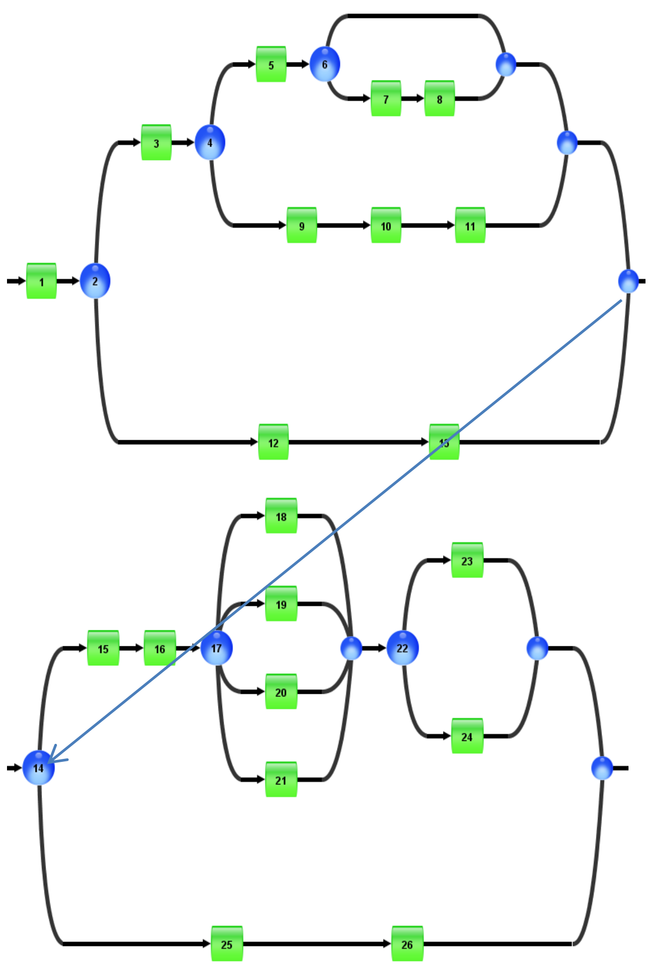

2
1
3
4
5
6
1: Contextual element (circles)
branch-1: 2: Activity (ovales)
branch-2: 3: Temporal Branching (vertical lines)
branch-1: 4:Action
branch-2: 5:Action
branch-3: 6:Action

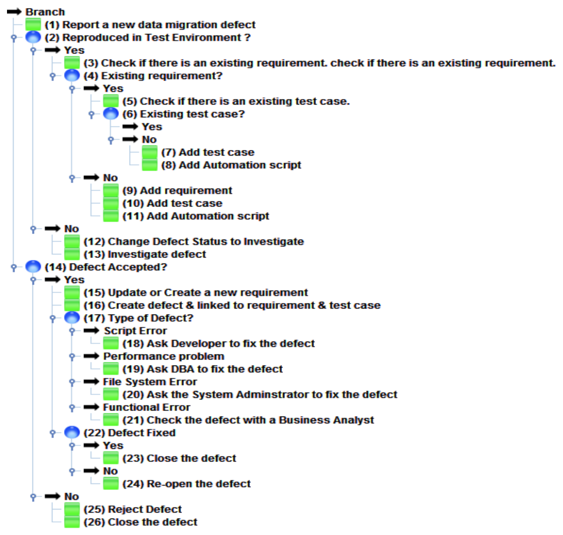
Fig. 3. Elements of a contextual graph.

Contextual graphs represent the set of known practices (strategies) in order to solve a given problem. They also allow incremental acquisition of practices and provide an understandable way to model context-based reasoning. A practice is the path from input to the output of a contextual graph. The problem solving process is guided throw a specific path by the evolution of context over time. Adopting a given practice or strategy among the others is dictated by the values of the different contextual elements forming the situation. However, it is not always obvious for a user to select one of these values. For example, in the area of database administration, to solve a serious performance problem within a given critical situation and context, a DBA (Database Administrator) may have different options when asking this question: what causes the slow response time of the system? Is it a network problem? Is it a bad database configuration? Is it a bad query in the application programs? Etc.
User practices are added and stored in an experience database. They may differ from each other because of their contexts that are slightly different where users used different actions at a step of the problem solving. The process of practice acquisition by the CxG system concerns the new action to integrate and the contextual element that discriminates that action with the previous one. The integration of the new practice requires either adding a new branch on an existing contextual node, or introducing of a new contextual node to distinguish the alternatives. The phase of incremental acquisition of practices relies on interaction between the CxG system and the users in order to acquire their expertise, which consists of a context-based strategy and its evolution along the process of the problem solving.

4.2 Sharing Context: Data Project Manager and DBA

Sharing context in defect tracking means that actors' contexts have a non-empty intersection. The shared context corresponds to the validity context of the design focus. It is built from contextual elements coming from the different members. The shared-context building results from an incremental enrichment of contextual elements coming from individual contexts. Thus, a contextual element proposed by an actor will enter the shared context if accepted (validated) by other actors. Individual contexts are mental representations of the design focus and of its validity context (the shared context). A contextual element provided by an actor must be integrated in other experts' mental representation, i.e. each expert must find a translation of this shared contextual element in his mental representation.
The following contextual graphs present the views of the Data Project Manager (Main Actor) and one of the team members (DBA, a Database Administrator). Fig. 4 illustrates the Project Manager view

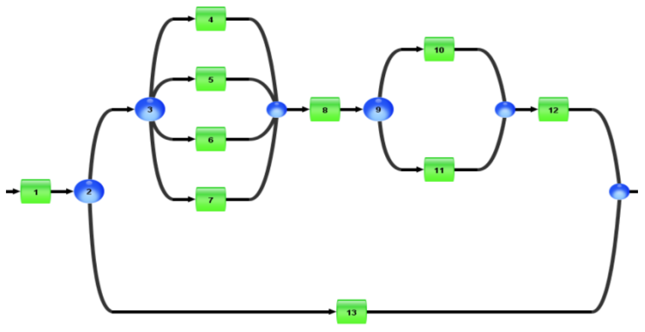


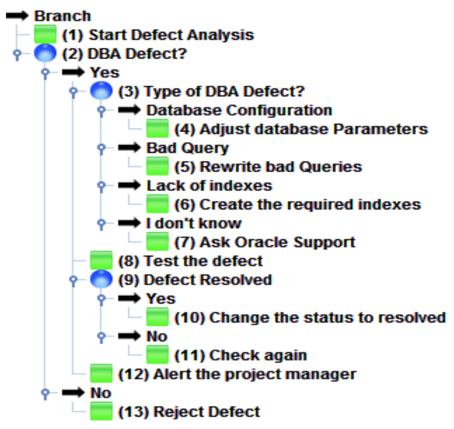


```
➡ Branch
  (1) Report a new data migration defect
  (2) Reproduced in Test Environment ?
    ➡ Yes
      (3) Check if there is an existing requirement. check if there is an existing requirement.
      (4) Existing requirement?
        ➡ Yes
          (5) Check if there is an existing test case.
          (6) Existing test case?
            ➡ Yes
            ➡ No
              (7) Add test case
              (8) Add Automation script
        ➡ No
          (9) Add requirement
          (10) Add test case
          (11) Add Automation script
    ➡ No
      (12) Change Defect Status to Investigate
      (13) Investigate defect
  (14) Defect Accepted?
    ➡ Yes
      (15) Update or Create a new requirement
      (16) Create defect & linked to requirement & test case
      (17) Type of Defect?
        ➡ Script Error
          (18) Ask Developer to fix the defect
        ➡ Performance problem
          (19) Ask DBA to fix the defect
        ➡ File System Error
          (20) Ask the System Adminstrator to fix the defect
        ➡ Functional Error
          (21) Check the defect with a Business Analyst
      (22) Defect Fixed
        ➡ Yes
          (23) Close the defect
        ➡ No
          (24) Re-open the defect
    ➡ No
      (25) Reject Defect
      (26) Close the defect
```

Fig. 4. Contextual graph for defect tracking (from Project Manager View).

Another representation (Fig. 5) is that extracted from the viewpoint of a database administrator and how is solving the defect. This can be helpful for sharing context. Other examples using contextual are graphs can be found in our research papers [16] and [17].





```
➡ Branch
  (1) Start Defect Analysis
  (2) DBA Defect?
    ➡ Yes
      (3) Type of DBA Defect?
        ➡ Database Configuration
          (4) Adjust database Parameters
        ➡ Bad Query
          (5) Rewrite bad Queries
        ➡ Lack of indexes
          (6) Create the required indexes
        ➡ I don't know
          (7) Ask Oracle Support
      (8) Test the defect
      (9) Defect Resolved
        ➡ Yes
          (10) Change the status to resolved
        ➡ No
          (11) Check again
      (12) Alert the project manager
    ➡ No
      (13) Reject Defect
```

Fig. 5. Contextual graph for defect tracking from DBA View.

4. CONCLUSION

This paper has presented how to contextualize the management of defect tracking in data integration. We have used Contextual graphs formalism to illustrate how it is easy to represent different viewpoints and practices when actors are communicating and interacting to resolve defect. Our study is in the framework of designing context-based systems for defect tracking. It can also be extended to several other computing areas such as monitoring systems, computer security and network management.

REFERENCES

[1] S. Abhiraja et al., "Defect Prevention Technique in Test Case of Software Process for Quality Improvement", Int. J. Comp. Tech. Appl, Vol 3 (1), 56-61, 2012.

[2] K. Ansar, "Establishing a Defect Management Process Model for Software Quality Improvement",

International Journal of Future Computer and Communication, Vol. 2, No. 6, December 2013.

[3] P. Brézillon, “Task-realization models in Contextual Graphs” in Modeling and Using Context (CONTEXT-05), A.Dey, B.Kokinov, D.Leake, R.Turner (Eds.), Springer Verlag, LNAI 3554, pp. 55-68 (2005)

[4] P. Brézillon, and J.-C. Pomerol, “Contextual knowledge and proceduralized context”, Proceedings of the AAAI-99 Workshop on Modeling Context in AI Applications, Orlando,Florida, USA, July. AAAI Technical Report (1999)

[5] A. Carneiro, and R. Passos et al., “DBSitter: An Intelligent Tool for Database Administration”, Berlin-Heidelberg, Springer-Verlag (2004)

[6] O. Don, “Defect Management in an Agile Development Environment”, The Journal of Defense Software Engineering, September 2003.

[7] S. Elfayoumy, and J. Patel, “Database Performance Monitoring and Tuning Using Intelligent Agent Assistants”, In: H.R. Arabnia, L. Deligiannidis, R.R. Hashemi (eds). Proceedings of the 2012 International Conference on Information & Knowledge Engineering, IKE 2012, WORLDCOMP’12. July 16-19, Las Vegas Nevada, USA, CSREA Press (2012)

[8] N. Gopalakrishnanm, “Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels”, World Academy of Science, Engineering and Technology 42, 2008.

[9] R. JÖRG, "Handling of Software Quality Defects in Agile Software Development", Fraunhofer Institute for Experimental Software Engineering (IESE), 2005.

[10] K. Macros, and T. Guilherme, “Towards a Defect Prevention Based Process Improvement Approach”, 34th Euromicro Conference Software Engineering and Advanced Applications, IEEE, 2008, DOI 10.1109/SEAA.2008.47.

[11] J.-C. Pomerol, P. Brézillon, “Context proceduralization in decision making”, In: Modeling and Using Context (CONTEXT-03), P. Blackburn, C. Ghidini, R.M. Turner and F. Giunchiglia (Eds.). LNAI 2680, pp. 491-498, Springer Verlag (2003)

[12] Rajni et al., "Defect Analysis and Prevention Techniques for Improving Software Quality", International Journal of Advanced Research in Computer Science and Software Engineering, 2013.

[13] J.F. Sowa, “Knowledge Representation: Logical, Philosophical, and Computational Foundations”, Brooks Cole Publishing Co., Pacific Grove, CA, (2000)

[14] V. Suma,"Defect Management Strategies in Software Development", Wseas Transactions on Computer, 2011.

[15] Sydney et al., "Agile-Why the fear", Planit Software Testing, 2009

[16] H. Tahir, and P. Brézillon, “Contextual graphs platform as a basis for designing a context-based intelligent assistant system”, In: P. Brézillon, P. Blackburn, and R. Dapoigny (Eds.): CONTEXT 2013, LNAI 8175, pp. 259-273, 2013.

[17] H. Tahir, P. and Brézillon, “Individual decision making based on a shared context”, in Frontiers in Artificial Intelligence and Applications, DOI: 10.3233/978-1-61499-073-4-63

[18] Conference: International Conference on Decision Support Systems: “Fusing DSS into the Fabric of the Context”, At Anávissos, Greece, June 2012.