Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 194 – 202

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

A case-based reasoning system to support the global software development

Rodrigo G. C. Rocha[a]*, Ryan R. Azevedo[a], Ygor Cesar Sousa[a], Eduardo de A. Tavares[a], Silvio Meira[b]

[a]*Federal Rural University of Pernambuco, Garanhuns 55292270, Brazil*
[b]*Federal University of Pernambuco, Recife 50670901, Brazil*

Abstract

In addition to the benefits brought by the use of Distributed Software Development, new challenges linked to its use also emerged. Due to lack of information companies and organizations around the globe, independently, solve these challenges in many different ways, each with their practices, some more some less efficient, where best practices are hardly widespread among DDS community. In this context, this paper aims to present a web system based on Case Based Reasoning and Natural Language Processing to extract information in text form of problems and solutions adopted by distributed software projects and to recommend similar past experiences in order to support the decisions and resolutions of problems arising from new situations in distributed projects. The feasibility proof of the method was made from experimental tests conducted to identify the success of the recommendations of previous valid cases, with a success rate of 90% for the sample used.

Keywords: Distributed Software Development; Case-Based Reasoning, Natural Language Processing.

* Corresponding author. Tel.: 55-87-**3764-5505**.
E-mail address: rodrigo@uag.ufrpe.br

doi:10.1016/j.procs.2014.08.099

1. Introduction

As a reflex of the business globalization and the evolution of the software market, new Software Engineering approaches are conceived and so are new ways to compete and collaborate. As a result, the Distributed Software Development practice evolves. In a DSD project, the teams working on it are dispersed in different locations[1].

Besides the benefits obtained by using this approach, new challenges linked to its use also emerged. Rocha et al.[2] states that in the distributed scenery, software projects assume different perspectives and consequently new risks. If good knowledge about the entire project does not exist and neither already known factors that may influence it, chances of failure will be higher. Due to lack of knowledge, companies and organizations around the world, independently attempt to patch these challenges in a variety of ways, each with its own practices; some more efficient than others, but even the efficient ones are hardly ever shared in the DDS field.

In this context, this work presents a method expressed in a web tool that uses the concept of Case-Based Reasoning and Natural Language Processing for extracting data from texts about problems and solutions adopted by companies and organizations in distributed projects. It recommends similar past experiences with the purpose of supporting decision making and solving problems raised from new situations in distributed projects. Thus, this tool recommends cases for those problems identified aiming to find solutions through the known cases by the system.

An experiment was necessary for the validation of the tool, which used 19 cases of problems and solutions retrieved from the event Distributed Software Development Workshop, from years 2007 to 2012, with a total of 51 papers analyzed. From the tests performance and the experiment, it is possible to state that the recommendation of previous valid cases was successful, with a 90% hit rate in the context of the sample used.

This work is organized as follows: Section 2 presents the background, in which the main concepts of the work are presented; Section 3 contains the proposed tool, its characteristics and features; in Section 4 the experiment used for the validation of the tool is discussed; Section 5 analyses related works, and finally, Section 6 comes with the final considerations

2. Background

This section presents the main concepts involved in this work, conceptualizing Distributed Software Development, Natural Language Processing Theory and Case-Based Reasoning.

2.1. Distributed software development

DSD is a software development model in which people involved with a certain project are in different locations[3]. According to Prikladnicki et al.[4], economy expansion, sophistication of communication means and cost pressure have encouraged the investment on DSD. The main advantages of this environment are: decrease in software development costs, access to human resources from other locations, and attending the needs of global clients[5].

However, there exists a series of challenges inherent to this context: physical distance, time zones and cultural differences. Carmel and Agarwal[6] evaluate how physical distance contributes to the complexity of the project developed under this kind of condition and define three main obstacles that DSD must overcome: the coordination, the information control and communication.

2.2. Natural language processing

NLP is a field of Artificial Intelligence and linguistics that consists in the development of computational models for performing tasks that depend on information expressed in a natural language[7]. In computing, the goal in using this concept is the evolution in the way in which the computer is used. Most of human knowledge is stored as natural language; computers able to understand natural language will be able to use this data with optimal speed and for a wide range of purposes[8]. Liddy[9] states that NLP is a set of techniques whose objective is to analyze and represent in a natural way texts that occur in one or more levels of linguistic analysis, with the purpose of reaching a language processing level similar to that of humans in different kinds of activities and applications.

A research performed by Vieira[10] highlights different applications derived from the study and development of the computational linguistics field. Some of those may be linked to NLP activities, for example: speak recognition systems and speak synthesizers, spellcheckers and grammar checkers, automated translators, text and summary generators, data extractors and natural language interfaces for specific domains.

2.3. Case-based reasoning

The concept of CBR is a paradigm for solving problems that emerge in systems of medical decision taking[11]. Pantazi, Arocha and Moerh[12] state that instead of depending solely on the general knowledge of a problem's domain, this approach uses the specific knowledge inferred from previous concrete problematic situations (so called cases) in order to face new ones.

In this context CBR is one of the most popular technologies for developing knowledge based systems using an approach for solving problems and learning by reusing previously known cases. Wangenheim[13] states that people on a day-to-day basis commonly use CBR when they solve their problems by recalling past experiences, comparing and adapting them to the context of the current problem.

The concept of cases is well described by Watson[14] and Aamodt[15]. They define case as the representation of the knowledge of an experience, containing its content and the context in which the lesson may be applied. Cases are stored in a database known as CBR, used as a case base[13]. When solving a new problem, a similarity calculation is performed between the new problem and the cases present on the base, so cases similar to the new problem may be retrieved. The solutions of the retrieved cases are then used for assisting solving the new problem in the current situation[16, 17].

So for developing a CBR system, initially, as one of the most complex and important activities, the modeling of the cases must be performed, which consists in identifying how to represent the knowledge. Then, a retrieving process must be developed and it must be decided how to retrieve cases as well as how to measure their similarities. Optionally, depending on the proposed solution, an adaptation and a learning mechanism may be developed. Then, the basic elements in which the operating cycle of a CBR system consists are: retrieving, adaptation/reutilization, review and retention[13].

3. A system proposal

The approach presented consists in a CBR system linked to a NLP component for the extraction of characteristics and recommendation of solutions for problems of the DSD team. As mentioned earlier, the purpose of the tool is to help solving new and/or old problems that may emerge in a distributed project. The architecture of the approach was divided in layers following the MVC (*Model-view-controller*) standard[18]. For the development of this tool, several technologies were used, such as the Java programming language, and The Stanford Parser 2.0.5[19].

The conceptual architecture of the developed approach is made up of two modules, the Natural Language Parser and the Case-Based Reasoning module. The operation flow of the approach consists in 4 steps, as showed below.

- firstly, the description of the new problem is fed on the interface for interaction between the user and the system via natural language through text;
- in the second step, the description of the new problem in DSD, in natural language through text, has its semantic characteristics extracted with the techniques applied in the NLP module;
- that’s the moment in which the characteristics retrieved in the second step are then used by the techniques applied to the CBR module as a means to represent the case and the similarity calculation with the cases in the case-base is performed; so that cases with similar characteristics to those described in the first step are retrieved, with the purpose of recommending solutions that will support the user in solving this new problem;
- finally, the last step of the approach is performing the revision and retention of this new experience (defined as a new case), expanding even more the case-base of the solution.

3.1. Natural language parser module

In this approach, as a grammar rule recognizer of the NLP an statistic method was used for the syntactic analysis and the grammar chosen was the *Probabilistic Context Free Grammar*[20]. English was chosen as the interaction language between the user and the system in natural language since it is the main language used in the academy context and also so it may be used by international industry.

The first activity of this module is Tagging, which is the lexical analysis process. In this moment words are identified (separated) in the text, and so are their grammatical categories (considered a *tag*). On the other hand, the objective of the Parsing activity is performing the syntactic analysis. In this moment the text is analysed thoroughly following the syntactic rules defined by the chosen grammar and its parse tree is retrieved, as mentioned in the Background chapter. As the last activity of the Natural Language *Parser* module, the Dependence activity performs the identification of the existing dependences among the words in the text and their types.

3.2. Case-based reasoning module

As a means of representing the cases, a data model was created in a way such that the characteristics extracted for future use are captured. Characteristics such as words, lexical classes and dependences among words in the text, as well as entire texts are the data that make up each case in a modelled relational database.

For the retrieval stage of the CBR, the retrieval approach used was the sequential retrieval[13]. This stage's purpose is to find similar cases in the case-base that may be easily adapted to the current problem, with the intent of solving it.

The approach used in the retrieval and revising stages of the proposed solution, based on the similarity between the current problem and the retrieved cases was that the knowledge engineer himself perform the action. In the last step of the CBR, the retain step is performed in a relational database created and modelled according to the cases' representation. The retaining (storage) happens after the user evaluates, repairs and revises the solution to the new problem.

The Stanford API was used for the extraction of characteristics from the text. As already mentioned, it used various parsers with the purpose of extracting characteristics from the description text of the new retrieved case.

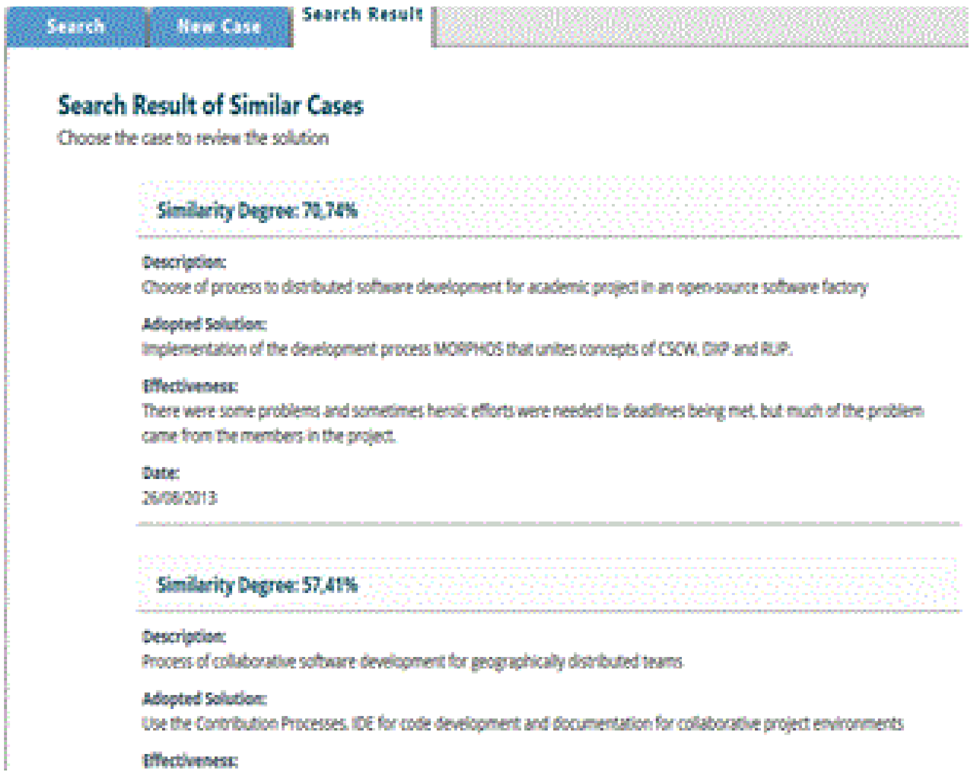
For the recommendation of cases based on the similarity, the *Weighted Nearest Neighbour*[21] was used, which can be observed in Equation. 1. It depicts two cases C1 and C2, with arbitrary attributes a1, a2, a3...ap. The approach performs the local similarities calculations for each of the cases' attributes, multiply them by their weights, calculates the summation presented in Figure 1, thus finding the similarity value. It is then divided by the sum of all weights, which characterizes the normalization of the weighted mean and the real similarity value among the cases. This value ranges from 1 to 0, where 1 is the total similarity (100%) and 0 is the absence of similarity (0%). The function sim_j represents the similarity measurement between C1 and C2 for attribute j, w_j is attribute j's weight for the determination of the similarity. If $w_j = 0$ the attribute is not taken into account for the similarity's calculation.

$$\mathrm{SIM(C1,C2)} = \sum_{j=1}^{p} w_j sim_j\,(C1, C2)$$

Equation 1. Nearest Neighbor weighted.

Four attributes were defined in the approach: attribute 0, which calculates the number of identical words between the two cases; attribute 1, calculates the number of words, if this number is equal between the two cases; attribute 2 calculates the number of identical verbs between the two cases; attribute 3 calculates the number of identical words that cause dependence in the description of the cases.

For a better contextualization, Fig. 1 depicts the result of a sample search performed by the system. The system presents the cases in a list, with all the relevant info of the retrieved cases, such as description, adopted solution, the effectiveness of the solution, inclusion day and similarity percentage of each case with the new problem, disposed in descending order according to the similarity level, from most to least similar.



Search Result of Similar Cases
Choose the case to review the solution

Similarity Degree: 70,74%

Description:
Choose of process to distributed software development for academic project in an open-source software factory

Adopted Solution:
Implementation of the development process MORPHOS that unites concepts of CSCW, DXP and RUP.

Effectiveness:
There were some problems and sometimes heroic efforts were needed to deadlines being met, but much of the problem came from the members in the project.

Date:
26/08/2013

Similarity Degree: 57,41%

Description:
Process of collaborative software development for geographically distributed teams

Adopted Solution:
Use the Contribution Processes, IDE for code development and documentation for collaborative project environments

Effectiveness:

Fig. 1. Example of Search Result

4. Experiments and results

To answer the research question “In what manner DSD teams may obtain effective solutions to everyday problems and extract knowledge to be applied in supporting new decisions?” the following hypothesis was defined: “Demonstrate that a CBR and NLP based method is a viable and effective solution to the extraction and recommendation of experiences in the support of decisions and solutions to problems in DSD”.

The sample is based on 19 cases of problems and solutions extracted from articles from the Workshop de Desenvolvimento Distribuído de Software (WDDS) event (Distributed Software Development Workshop), from 2007 through 2012, with a total of 51 analyzed papers.

The tests were performed with the intention of observing the possible behaviors of the systems; for example, some of the cases used were based on the description of the new problem (sentence to be processed), subtexts from the descriptions of cases mentioned earlier, so the tool would recommend exactly cases from which the descriptions of the new problems were retrieved. Totally new texts were also used as description of the new problem so the tool would recommend the cases from the case-base that were really more similar to the eyes of a knowledge engineer. And lastly, other tests used complete texts of cases present in the case-base as the description of the new problem, so the recommended cases were the same with a 100% similarity rate.

4.1. Tests and results

The Table 1 shown below presents the cases used in the tests and the similarity percentage of the most similar case. Here, the similarity value is that of only the most similar case, which does not mean that other cases were not recommended.

Table 1. Tests performed

Processed sentence	Most similar cases
communication among small team members in distributed project.	58.3%
Choose of the process of development software f or a small team.	63.2%
Lack of communication in software development teams separated geographically.	72.3%
Choose development process for distributed software project with small teams.	59.5%, 56.3%, 55.2%
Communication between members of a small team in a distributed software development project.	59.1%, 54.4%
Write the documentation requirements in a project distributed with national distance and high levels of temporal dispersion between team members.	52.1%

It is important to highlight that some of the processed sentences are similar and the reason for that was to show that depending on the words used, the sentences can receive different case recommendations. For example, the first and fifth sentences are similar but the first received just one recommended case ("Communication among small team members in distributed project using an adaptation of RUP") while the fifth sentence received two cases "Communication among small team members in distributed project using an adaptation of RUP" and "Communication between product owner and team project in a distributed project using an adaptation of SCRUM". The same thing happened to the second and fourth sentences: the former received as a recommendation "Choose the process of developing software for a small team using distributed software development" and the latter: "Process of collaborative software development for geographically distributed teams", "Process of software development with large and distributed teams" and "Choose of process to distributed software development for academic project in an open-source software factory". It has occurred for three reasons: minimum level of similarity was 50%, the used words are different and the case base is small. With a bigger database it will be possible to enhance the minimum similarity level and achieve better results.

From 10 tests carried out, 9 recommended similar cases efficiently and successfully; cases that were truly similar to the new problem described by the user and had an useful solution that could be adapted and used, reaching a 90% hit rate. From the 9 successful tests, one hasn't given the best real recommendation possible, so from the successful tests, approximately 88.89% were real hits and from the complete set of tests (10), 80% were real hits. As a reminder, in all performed tests the entire database was used, with 19 cases.

Table 2, presents a sample result for the process of a sentence. Is it possible to visualize which sentence was used, which case or cases and the similarity rates obtained. It is also shown for each case found, its description, the adopted solution and its respective effectiveness, as well as a brief discussion in the end.

Table 2. Results from processing Case 4

Processed sentence:
Choose development process for distributed software project with small teams.
Results
The cases with most similar descriptions presented similar rates of 59,52%, 56,35% e 55,19%, respectively. -- Case 1 -- **Case description**: *Process of collaborative software development for geographically distributed teams.* **Adopted solution**: *Use the Contribution Processes, IDE for code development and documentation for collaborative project environments.* **Effectiveness**: *Uninformed.* -- Case 2 -- **Case description**: *Process of software development with large and distributed teams.* **Adopted solution:** *Use adapted concepts of SCRUM.* **Effectiveness**: *Uninformed.*

-- Case 3 --
Case description: Choose of process to distributed software development for academic project in an open-source software factory.
Adopted solution: *Implementation of the development process MORPHOS that unites concepts of CSCW, DXP and RUP.*
Effectiveness: *There were some problems and sometimes heroic efforts were needed for deadlines being met, but much of the problem came from the members in the project*.

The description of the most similar case, with a similarity rate of almost 60% was "*Process of collaborative software development for geographically distributed teams*", indeed a case very similar to the new problem, which shows that its solution may be easily adapted to the new problem. This also happens to the second and third cases of the list, since their similarity rates are quite close to that of the first case. For a knowledge engineer, from the results the most similar case to the one described would be: "*Choose of the process of development software for a small team using distributed software development*", i.e., the system recommended good similar cases but not the best one from the base. Other cases with smaller similarity rates were also recommended.

The system outputs the recommended cases for the processed sentences. For each of them there is a set of important information: a "Case description", that is, what cases were recommended for that sentence; "Adopted solution", representing how the problem was solved and what techniques or tools were used for that case; and the "Effectiveness", identifying the real result of adopting that solution.

For the validation of the experiments, a right-tailed binomial test[22] was applied to the sample of hits obtained in the tests. The retrieved results were: threshold of 80%, significance level of 5% and p-value of 10%. Since the p-value is smaller than the significance level, the null hypothesis is rejected, i.e., statistically, the proposed approach correctly recommends similar cases in more than 80% of the cases of use and it is viable ($0.8 < \text{hit rate} < 1$; p-value=0,0107). The hit rate of 90% was superior than 80% and indeed significant. Then, through the graphic it is possible to infer that in 90% of the 10 performed tests, the proposed approach extracted the characteristics of the description in text and recommended in a coherent way similar cases to help in solving new cases.

5. Related work

In this section, works with the same goal or theme of this paper are described. Based on the amount of related works found, it can be affirmed that relatively few works of Distributed Software Development supported by knowledge systems have been carried out.

Wongthongtham et al.[23] present the project and implementation of a social network approach as a mean to support the sharing and evolution of a Software Engineering ontology. A multi-agent recommender system that uses the 'Software Engineering Ontology' and 'SoftWare Engineering Body of Knowledge (SWEBOK)' as sources of knowledge is designed within multi-site communities of software engineers and developers working on related projects as the target audience. Though a big challenge faced by this approach is ensuring that the knowledge bases of different agents are coherent and consistent with one another, as stated by Dilon and Simmons[24].

Ankolekar et al.[25] consider as one of the toughest problems faced by online professional communities the fact that the vast amount of data generated as a result of their interactions is not well-linked on the basis of the meaning of its content. With the assumption that a better semantic support can bring improvements to these communities, a prototype Semantic Web system was developed. Such task required a way of describing the semantic content retrieved from the data obtained from these communities, which was accomplished through the use of ontologies. The large amount of data generated was a large obstacle as the parsers used were unable to reason efficiently for large amounts of data.

Rocha et al.[26] presented a research about the use of ontologies supporting DSD projects. So, the important characteristic is proposal of the use of best practices for the challenges found by any member, thus, they can be use the DKDs (Distributed Knowledge Development System) to check or consult all knowledge stored looking for possible best practices. It also allows the creation of a list of possible problems during the initial phases, so the manager or developers can avoid some challenges. Other interesting resource is the creation of a list of possible developers who may be able to help solve technical problems through their skills.

The essential difference of this work is the concept that permits the use of successful cases for challenges found. So, the goal of this research is that new challenges and problems may be solved through the use of known cases, once these cases are saved on the knowledge base. And it is possible through natural language, just the description of simple texts. This research presents a complete model of a web tool supported by NLP and CBR theories, using their concepts and characteristics.

6. Conclusion

Motivated by some opportunities, companies have been starting to work with geographically distributed development teams, adopting the Distributed Software Development approach, though by using it, software projects assume new risks and new challenges will arise. If good knowledge about the whole project does not exist nor already known factors that can influence the project, the chances of failure will be higher.

In another context, the combination of the Natural Language Processing and Case-Based Reasoning techniques brings the freedom that the expression through natural language enables, as well as the use of unique knowledge that may be reused or not, according to the case.

In this way, this work presented a web tool that used the CBR and NLP concepts for extracting information from texts of problems and solutions adopted by distributed projects, for recommending similar past experiences with the purpose of supporting decisions and solving problems brought by new situations in distributed projects. The experiment performed used 19 cases of problems and solutions extracted from papers of a specific event from the Distributed Software Development Field, from the years of 2007 through 2012, with a total of 51 papers analysed. Based on the experiment, it is possible to state that there was success (approximately 90%) in the recommendations of valid past cases.

In future work, the increasing the number of Software Engineering and DSD events used for the extraction of the cases; as well as having a specific DSD ontology to be used as database, since the tool presented may use the semantic web proposed by an ontology, improving the recommendation of more similar cases. And for that, this approach[27] may be used and compared with Rocha's work[26], to create and test the ontology. Another possible work is testing other options for similarity algorithm aiming at the best rates

References

1. Carmel, E. Global Software Teams – Collaborating Across Borders and Time-Zones. Prentice Hall, EUA. 1999
2. Rocha, R. G. C. ; Costa, C. ; Rodrigues, C. M. O. ; Azevedo, R. R. ; Farias Junior, I. H. ; Prikladnicki, R. ; Meira, S. R. L. . Collaboration Models in Distributed Software Development: a Systematic Review. CLEI Electronic Journal, v. 1, p. 1-12. 2011
3. Prikladnicki, R. MuNDDoS - Um modelo de referência para desenvolvimento distribuído de software. Master Thesis, Pontifical University of Rio Grande do Sul (PUC/RS) at Porto Alegre. 2003
4. Prikladnicki, R., Marczak, S., Conte, T., De Souza, C., Audy, J., Kroll, J., Marques, A.B. e Orsoletta, R. The Evolution and Impact on research in Distributed Software Development in Brazil, XXV Brazilian Symposium on Software Engineering, special track: SBES 25 years (In Portuguese), 126-131. 2011
5. Carmel, E., Tjia, P. Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce, UK: Cambridge. 2005
6. Carmel, E., Agarwal, R. Tactical approaches for alleviating distance in global software development. J. of IEEE Software. 18, 2 (Mar./Apr. 2001), 22–29. DOI= http://dx.doi.org/10.1109/52.914734. 2001
7. Pereira, S. L. Natural Language Processing [S.1]. Available: <http://walderson.com/2012-1/ia/ApIA07-ProcessamentoDeLinguagemNatural .pdf>. 2012
8. Allen, J. Natural Language Understanding. Redwood City, CA: The Benjamin/Cummings Pub. Co. 654 p. 1995
9. Liddy, E. D. Enhanced Text Retrieval Using Natural Language Processing. In: Bulletin of the American Society for Information Science, v. 24, n. 4. 1998
10. Vieira, R.; LIMA, V. L. S. Lingüística Computacional: Princípios e Aplicações. In: JAIA, SBC, Fortaleza, Brasil. 2001
11. Bichindaritz, I. And Montani, S. Report on the Eighteenth International Conference on Case-Based Reasoning, AI Magazine 33. p 79-82. 2012
12. Pantazi, S. V., Arocha, J. F. and Moehr, J. R. Case-based medical informatics. BMC Medical Informatics and Decision Making. 2004
13. Wangenheim C. G. V.; Wangenheim, A. V. Based Case Reasoning. 2003
14. Watson, I.; Marir, F. Case-based reasoning: A review. Knowledge Engineering Review, v. 9, n. 4, p. 327-354. 1994
15. Aamodt, A.; Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications, v. 7, n. 1, p. 39-59. 1994

16. Thé, M. A. L. Raciocínio Baseado em Casos Uma Abordagem Fuzzy para Diagnóstico Nutricional. 170f. Ph.d Thesis. Universidade Federal de Santa Catarina, Florianópolis - SC, 2001
17. Leake D. B. Case-Based Reasoning: Experiences, Lessons and Future Directions. MIT Press Cambridge, MA, USA. 1996
18. Krasner, G. E.; POPE S. T. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. ParcPlace Systems, Inc, Mountain View, CA, USA. 1988
19. The Stanford Parser: A statistical parser, In: The Stanford Natural Language Processing Group. [S.1]: Stanford University. Available: <http://nlp.stanford.edu/software/lex-parser.shtml>. 2013
20. Chi, Z.; Statistical Properties of Probabilistic Context-Free Grammars. Computational Linguistics, vol. 25, nº 1. 1999
21. Cover, T.; Hart, P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, IEEE Information Theory Society, vol. 13, p. 21–27. 1967
22. Gamerman, D.; Migon, H. S. Inferência estatística: uma abordagem integrada. Instituto de Matemática, Universidade Federal do Rio de Janeiro, 1993
23. Wongthongtham, P., Chang, E. and Dillon, T. Ontology-based Multi-agent System to Multi-site Software Development. In Proceedings of the Workshop on Quantitative Techniques for Software Agile Process (QUTE-SWAP). (Newport Beach, USA). 2004
24. Dillon, T. and Simmons, G. Semantic Web support for Open-source Software Development. In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS). 2008
25. Ankolekar, A., Sycara, K.,Herbsleb, J., Kraut, R and Welty Chris. Supporting online problem-solving communities with the semantic web. Internactional Conference on World Wide Web. Pg 575-584. 2006
26. Rocha, Rodrigo G. C., Azevêdo, Ryan., Costa, Catarina., Duarte, Marcos., Fechine, João Paulo., Freitas, Fred., Meira, Silvio. An Ontology-based System to Support Distributed Software Development. In The Eighth International Conference on Software Engineering Advances (ICSEA). Venice, Italy. 2013
27. Azevedo, R. R; Freitas, Fred; Rocha, R. G. C; Menezes, J. A. A; Rodrigues, C. M. O; Silva, G. F.P; An Approach for Learning and Construction of Expressive Ontology from Text in Natural Language. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). 2014

.