Towards Pertinent Characteristics of Agility and Agile Practices for Software Processes

José Fortuna Abrantes, Guilherme Horta Travassos
Universidade Federal do Rio de Janeiro,
Rio de Janeiro, Brasil, 21941-972
abrantesjf@gmail.com ght@cos.ufrj.br

Abstract

Context: It is believed that agility in software processes can bring benefits to software development and lead to an economy of efforts when accommodating changes is needed. Objective: Assess pertinence and relevance of agility characteristics and agile practices for software processes. Method: From 18 agility characteristics and 17 agile practices applicable to software processes revealed through systematic literature reviews performed in 2010, a survey was conducted to assess their pertinence and relevance. Results: 16 agility characteristics and 15 agile practices were considered pertinent to insert agility in software processes. Conclusion: Results should be used sparingly. It would be interesting to replicate the study in other contexts, with different subjects, and compare them, to increase the generalization of their results.

Keywords: Agility in software processes, characteristics of agility, survey, evidence-based Software Engineering, agile practices, agile software development, agile methods.

1 Introduction

The last decades saw a need for software processes capable of adapting to last-minute unforeseen changes or modifications, especially in requirements, to meet specific needs of business in an increasingly dynamic market, that is competitive and opening itself to new possibilities. The most significant challenge for organizations in the twenty-first century seems to be their ability to adapt to rapid change and unpredictability, in a more appropriate and faster manner than their competitors [1]. Slowness to accommodate uncertainty and rapid changes inevitably lead to the obsolescence of software and customer dissatisfaction [2]. Constant changes, especially in requirements and in a business environment, become difficult when more rigid software processes are applied.

Some activities of software processes (e.g., testing activities) are amongst the costliest of all the activities which are part of software development [3]. Changes in requirements may turn artefacts already planned, designed, specified or even already implemented with their results analyzed, in obsolete or outdated artefacts. It is desirable that software processes could have sufficient flexibility to accommodate inevitable changes and uncertainties. At the same time, it is expected that these processes might be described, monitored and improved [4].

One of the most important innovations for software development approaches in recent years has been the introduction of agile principles [5]. However, despite a series of agile methods being proposed, little is known of how these methods are used in practice and what their effects are on software processes [6]. Thus, some concerns pop up when one thinks of defining software processes susceptible to changes: how to embed flexibility in a software process? What are the desirable characteristics in a software process so that it can be considered agile? These questions, although seemingly simple, have not had adequate responses described in the literature.

According to [7], agile development approaches involve the adoption of a set of practices that are mutually supportive. An alternative solution to embed characteristics of agility into software processes can be supported, hence, by the adoption of agile practices. So, exploring this alternative involves answering the following questions: (1) what characteristics of agility are pertinent and should or could be candidates for insertion in a software process in order to make it agile? (2) what are the agile practices that are pertinent and can be considered for adoption in software processes with the aim of trying to embed characteristics of agility in these processes? Despite the understanding that one practice alone may not be able to bring a degree of agility suitable for a software process, it is necessary to consider

the pertinence of each one, separately, to support the formation of an appropriate set of agile practices that can turn a software process to get the degree of agility desired for a specific software project.

The methodology used in this research is based on the concepts of experimental software engineering [8] and on a structure presented by [9] in which secondary studies and primary studies are included. As suggested by [9], the methodology is split into two phases: (a) definition of technology and (b) refinement of the technology.

In the first phase secondary studies are executed to find the knowledge necessary to support the development of a proposal technology to solve a problem. In the second phase out primary studies (concept proofs, case studies and experimental studies) are carried to evaluate the technology, which may then be refined and/or have their understanding expanded. According to [10] the design phase of new technologies involves performing secondary studies and/or primary studies with the objective of obtaining an initial proposal of the new technology. Figure 1 illustrates the steps involved in this stage, adopted in this research.

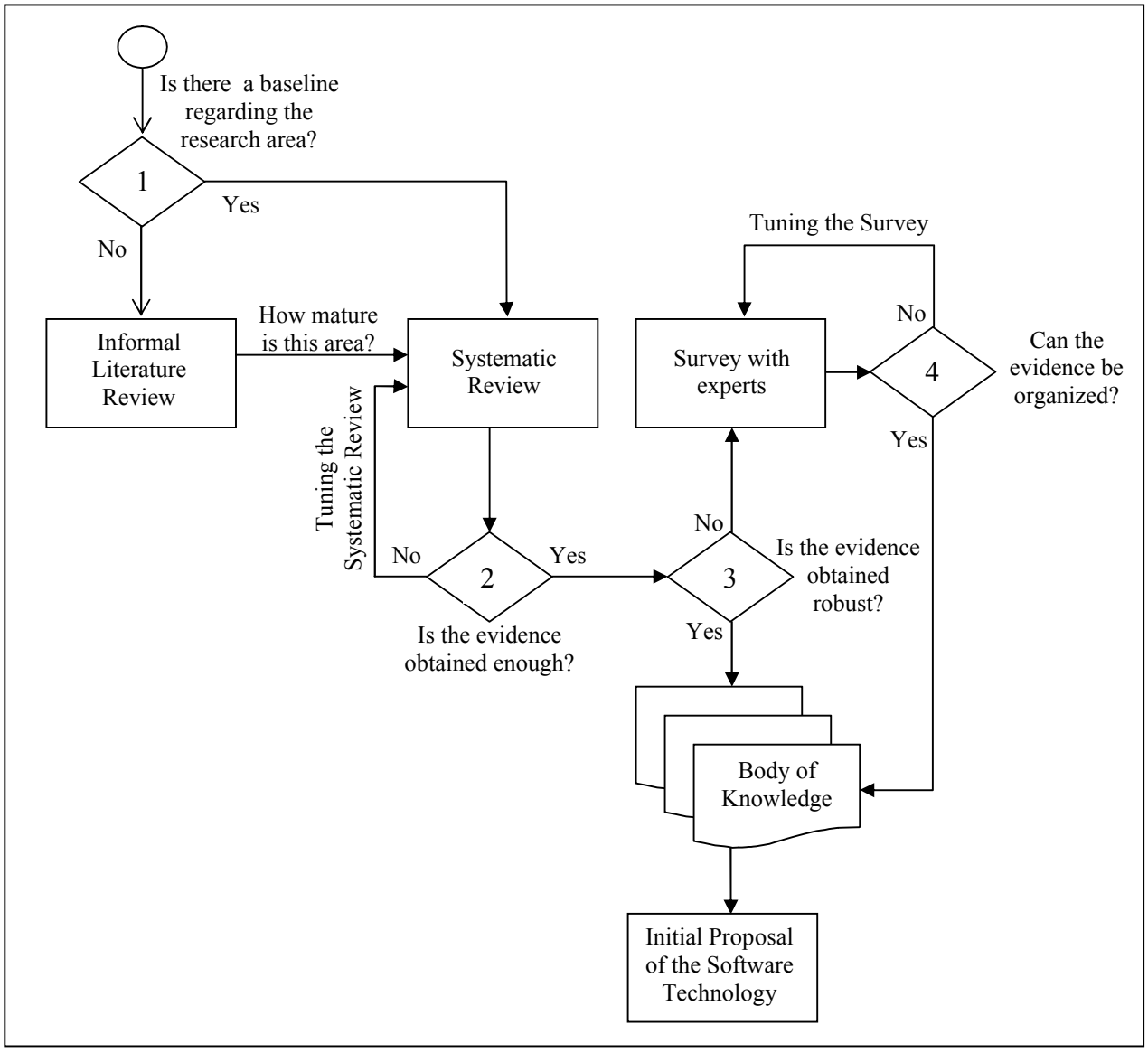

Figure 1: Research Methodology – Design Phase, adapted from [10]

So, to answer the raised questions, after initial informal literature reviews, systematic reviews were done to identify characteristics of agility to be expected in software processes and the agile practices that could be adopted in the process to embed the desired characteristics of agility. To increase confidence in these results, the characteristics of agility and agile practices identified through the systematic reviews became the subject of study in a survey to assess the pertinence and the relevance level of each characteristic and practice, in the context of the agile approaches.

This paper is structured as follows: Sections 2 and 3 present the results of systematic reviews conducted to identify characteristics of agility and agile practices for software processes, respectively. We carried out two separate systematic reviews as the needs for the knowledge they produce had arisen in different moments. Besides this, considering the high efforts required by systematic reviews, an informal literature review was initially tried to identify agile practices, whose results were not considered sufficient and did not meet the requirements to this research work. Section 4 presents results from a primary study (survey) performed to assess the characteristics and practices identified through secondary studies previously mentioned. In Section 5 the threats to validity of the studies are discussed. Section 6 has the concluding remarks as well as some possibilities for future work.

2 Characteristics of Agility – A Systematic Review

Software process agility has been defined as the capability of quickly adapting to changes in the requirements and environments involving software [11]. The agile software process perspective means not only fast application development, but mainly the capability for quick and flexible adaptation to changes in processes, products or environments. Lean and highly iterative development, putting strong emphasis on stakeholder involvement, have been pointed out as key characteristics of agile methods [12].

Agile methods have proposed a way of looking at software development, focusing on interactions among people which can collaborate to achieve high development productivity. Besides, like conventional methods, the goal of agile methods is to build high quality software to meet the users' needs [13]. Boehm and Turner [14], [15] suggest that agile development methods promise customer satisfaction in higher intensity, lower defect rates, faster software development and a solution for rapid requirements changes when compared to traditional ones. However, there is neither universal definition nor evidenced attributes for what could characterize a method or software process as agile [16], making hard the task of identifying agile software processes and/or decision-making when tailoring them to be agile. For the agile software development community, agility definition has been influenced by the values expressed in the Agile Manifesto, whereas in a business context, agility is usually defined in terms of timeliness and flexibility [17]. As observed by [18], the concept of agility has been approached in a number of disciplines and it originated, matured and has been applied and tested thoroughly over time in the manufacturing and management area.

The text in this Section aims at describing the results of a secondary study with the purpose of identifying a set of characteristics which could be used to characterize software processes as agile. A characteristic of agility represents a relevant attribute which a software process may present to make difference under the perspective of agility. Relevant attributes in the context of agile approaches will be considered as those attributes which contribute to achieve one or more of the values expressed in the agile manifesto [19], [20]. The basic research question is: What are the characteristics of agility in agile software process contexts? It is not intended to investigate characteristics for a particular agile method or software process, e.g. the six basic characteristics of ASD – Adaptive Software Development described by [21], but to identify what the desirable characteristics to support agility in software processes are. The objective of the study is to produce a characterization of the field and organize an evidence-based knowledge repository that can be used to support decision-making and knowledge management regarding agility in software processes.

2.1 Planning

A summary of the protocol, specifically designed for the study, is discussed in this Section.

The objective of the study is to identify the characteristics of agility for software development methods or processes, in a general sense, independently from a specific agile method.

The question was: What are the characteristics of agility in software development processes? The problem considered was to find characteristics of agility for software development processes. The research issue was structured by four elements (population, intervention, comparison, and outcome), according to [22]. The population considered was the set of software development projects. The interventions were the agile software development methods or processes. There was no comparison (alternative intervention). The outcome was a list of characteristics of agility

related to software development methods or processes. As there were no comparisons between intervention and alternatives [22], and also because it was not possible to undertake meta-analysis, this secondary study, although systematic, is considered by [23] as a *quasi*-systematic review. The application of the *quasi*-systematic review is to be used as a research support basis encompassing efforts to insert characteristics of agility in software processes.

Some documents were manually retrieved and analyzed before the execution of the initial searches. Three of them were chosen as controls: [24], [25], [26]. These controls were used to verify and calibrate the search string appropriately. The selected sources were the electronic databases available in [27], including conferences, journals and technical reports indexed by Compendex EI, IeeeXplore, Inspec, ISI Web of Science, the ACM digital library, Science Direct, and Scopus. The language expected for written documents was English. Any type of works or articles, mentioning and describing the meaning of characteristics of agility, on agile software processes, was considered.

The keywords selected for population were software development, project, system, application, engineering, building, and implementation. The keywords selected for intervention were agile, method, adaptive, rapid, approach, technique, environment, process, practice, and methodology. The keywords selected for outcome were characteristic, attribute, property, feature, characterization, aspect, idea, factor, dimension, driver, perspective, and requirement.

As inclusion and exclusion criteria, it was defined that documents should be available on the Web; documents should cover characteristics of agility for software development processes. The characteristics should be described or explained in the text. To avoid bias, documents reporting characteristics of specific agile methods were excluded.

A selection process for the studies was established. A researcher applied the search strategy to identify potential documents. Then, this researcher analysed the documents identified and applied the inclusion/exclusion criteria. The list of excluded documents was also analyzed by the second researcher, and in case of conflict, the document was included. The documents included were read by both researchers to extract information on the characteristics of agility.

It was also established that appraisals for the quality of the studies would not be conducted, due to their dealing with a characterization research and several kinds of studies. Most of the papers do not bring any evaluation through empirical studies, so extensive rigour could limit the capacity of observation necessary to identify the desired characteristics. The sources of documents were considered to provide initial confidence, since it was assumed the papers had been externally revised, so it could represent a 'quality' seal to justify their inclusion in this study.

It was established that the following data would be extracted: document title, author(s), document source, year of publication, and characteristics of agility. The results would be tabulated and an analysis would be done to identify commonalities as well as relevant agile software processes characteristics. The frequencies in which different authors approached each characteristic of agility were considered in assessing their relevance and as a basis for their ranking.

The search string was adapted according to specific search engines restrictions, observing: (1) the obtained search string should be logically equivalent to the basic search string; (2) if it was not possible to maintain exact equivalence, the obtained search string should be broader than the basic one, to avoid losses of relevant documents. This is to avoid changing the search string significance due to syntactical differences on the construction of its logical expressions among search engines and to avoid turning the search string less comprehensive due to specific engine restrictions. The basic search string adopted was:

(software development OR software engineering OR software building OR software implementation OR software projects OR software systems OR software application OR system development OR system engineering OR system building OR system implementation OR system project OR application development OR application engineering OR application building OR application implementation OR application project) AND ((agile OR adaptive OR rapid) AND (method OR process OR practice OR methodology OR approach OR technique OR environment)) AND ((agile) AND (characteristic OR attribute OR property OR feature OR characterization OR aspect OR idea OR factor OR dimension OR driver OR perspective OR requirement))

Some search engines offer particular operators which can be used in the construction of their specific search strings, such as Compendex EI's operator NEAR. Also, different search engines may adopt different syntax rules.

2.2 Execution

The secondary study covered 4 protocol trials until January/2010. In Nov-Dec 2006 the protocol was executed for the first time, using the Compendex EI, IeeeXplore, Inspec, ISI Web of Science, and ACM Digital Library databases. References to documents published from 1978 to 2006 were retrieved. The second protocol execution was carried out in December-2007 for the same five electronic databases, but with a limited time frame (2006 and 2007). The third protocol execution was also carried out in December-2007, for two additional electronic databases, Scopus and Science Direct, searching for documents published and/or indexed until 2007. Some issues occurred then with the ACM Digital Library search engine which did not allow it to process the same search string due to some restrictions imposed by its search engine, when the searches were first carried out (2007). To avoid it become a strong threat to validity, and trying not decrease recall, the search string was adapted to be processed by the ACM Digital Library search engine. To update the findings, a fourth protocol trial was carried out in January-2010, with all seven search engines, searching for documents published and/or indexed from 2008 until 2009. The protocol executions along time can be seen in Figure 2.

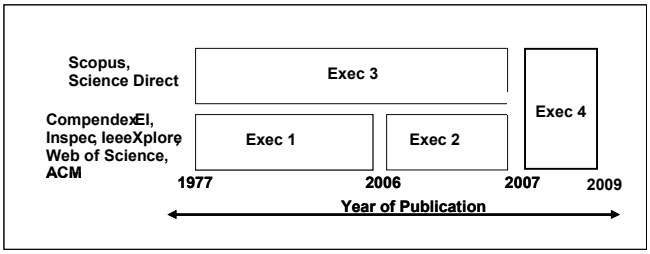

Figure 2: Protocol Executions along Time

To support the handling and treatment of retrieved items, the reference manager JabRef 2.5 (http://jabref.sourceforge.net/) has been used. Together, all protocol trials retrieved 6,602 papers. The second and fourth trial extended the search space 'chronologically'. The third trial extended the search space 'geographically', by using two new search engines. Table 1 shows paper distribution among the search engines for each protocol trial.

Table 1: Quantitative Summary for all Executions

Electronic Database	Exec 1	Exec 2	Exec 3	Exec 4	All Executions
ACM digital library	119	89			208
Compendex EI	303	179		327	809
IeeeXplore	299	124		201	624
Inspec	250	94		7	351
Science Direct			997	310	1,329
Scopus			1,864	1,339	3,203
Web of Science	45	22		33	78
TOTAL	**1,016**	**508**	**2,861**	**2,217**	**6,602**

2.3 Results and Discussion

In the set of retrieved references, duplicates were eliminated, the remaining papers being counted for the electronic database with most retrieved items. All references to documents adopted as controls were retrieved. References to documents published from 1977 to 2009 were found. Successive evaluations (title and abstract) were carried out and references to documents clearly dealing with aspects not concerning the research study were excluded. In some cases, also the document introduction was read in this initial evaluation. Further, in a more detailed evaluation, a set of 157 documents were selected and prioritized. These documents were thoroughly read while observing the criteria established in the protocol described in Section 2.1. Some of them lightly mention a possible characteristic of agility, but do not mention anything about its meaning. Fourteen documents contributed to the *quasi*-systematic review. This reduction in the number of documents is associated with part of the inclusion and exclusion criteria, as previously described in Section 2.1. Figure 3 shows the distribution of documents that contributed for the study, per year of

publication. It shows that it was not possible to find documents published in 1999, 2000, and 2007 the met the criteria established in the protocol.

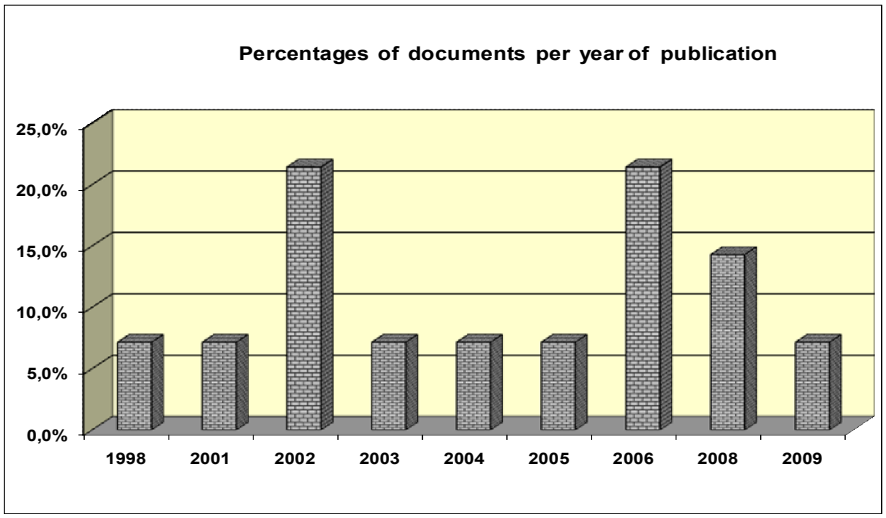

Figure 3: Documents contributing for the study per year of publication

The Scopus engine retrieved 9 out of 14 documents that contributed to this study. Inspec retrieved 5 of them. Compendex EI and IeeeXplore retrieved 4 documents each. Web of Science and Science Direct retrieved 2 documents each. The ACM digital library did not retrieve any reference among those 14 documents.

There was a concern to avoid repeated counts for one same characteristic of agility. To prevent this, two clusters of documents were constructed, each cluster being computed once for the incidence of the characteristics in its documents, avoiding repeated counts. A pair of documents was included in a cluster when they were approaching the same agility characteristic, and any author of one of them belongs to the set of authors of the other document (same characteristic, same author). Moreover, documents approaching the same characteristics, one of them referencing the other document, were also included into the same cluster. The clusters (with two documents each) were conceived as follows: cluster 1 [25], [28]; cluster 2 [15], [26].The characteristics of agility in the context of agile software processes were caught in 14 of the selected documents [28], [29], [30], [24] [11], [32], [33], [34], [25], [26], [15], [35], [36], [37].

Taromirad and Ramsin [38] propose 17 main features of agility recommended for an agile method. However, in this paper, they did not describe the perspectives they considered for these features. In their other work, Taromirad and Ramsin [35] present nine characteristics of agility that have been defined based on the Agile Manifesto, Agile Principles, and papers presenting common agile traits. The authors stated that these characteristics can be used to evaluate the degree of agility for any software development methodology, even non-agile ones, which are: Speed (related to how quickly the process can produce results); Sustainability (related to monitoring and controlling the maintenance of speed and quality until the end of the development process); Flexibility (related to the capability of capturing and handling in the project, expected or unexpected changes); Learning (related to the process ability to 'learn' from past projects and previous iterations); Responsiveness (related to the capability of the process to provide feedback); Leanness (related to how the process values shorter time spans, using economical and simple quality-assured means for production); Lightness and simplicity (related to how light and simple the development process is); Technical quality (related to how the technical quality is monitored and controlled during the development process) Active user collaboration (related to the degree of customers' involvement in the development process).

Rico [36] in his study about relationships between the use of agile methods to develop Internet websites and their quality pointed out four factors to characterize agile methods: iterative development, customer feedback, well-structured teams, and flexibility. Iterative development was defined by [39] as 'an approach to building software (or anything) in which the overall lifecycle is composed of several iterations in sequence'. Customer feedback according to [40], is 'a general term describing direct contact with users and covering many approaches' and it lies on the 'continuum from informative, through consultative to participative'. Well-structured teams are defined by [41] as 'work

groups made up by individuals who see themselves as a social entity, who are interdependent because of the tasks they perform, who are embedded in one or more larger social systems, and who perform tasks that affect others'. Flexibility is defined by [42] as 'the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed'. Rico [36] also presents, for each identified agility characteristic or factor, sub-factors selected from the technical literature concerned with agile methods.

Qumer and Henderson-Sellers [37] describe five key attributes of agility: flexibility, speed, leanness, learning, and responsiveness. They call these attributes of agility, as features, and describe them as follows. Flexibility - related to the capability of the process to accommodate expected or unexpected changes. It encourages the acceptance and accommodation of changes (generated from an internal environment or by a customer) in a software product, process, plan, development team(s) and development environment. Speed - related to the capability of the process to produce results quickly. Leanness - related to the capability of the process to follow the shortest time span, use economical, simple and quality instruments for production. Learning - related to the capability of the capability of the process to apply updated prior knowledge and experience to create a learning environment. Responsiveness - related to the capability of the process to exhibit sensitiveness. It refers to the fact that not only does it make it easy to accept the changes but the changes must be reflected and visible.

Analyzing the meaning of the characteristics described by the above authors of all 14 papers [28], [29], [30], [24], [11], [32], [33], [34], [25], [26], [15], [35], [36], [37], the domain values and the results of applying them to evaluate XP [24], as well as the sub-factors selected from the literature on agile methods pointed out by Rico [36], some matching can be detected among them. This leads to a consolidation of the meanings of all instances of the agility characteristics approached by the authors of all 14 papers.

According to [43], there are three context levels in which agility can be useful in a company: on the level of software development projects; in the portfolio of the level of software projects; and on the entire enterprise level where the company is challenged by its competitors. Simply using an agile approach on one level without considering the next higher one will not lead to the optimum return. This can be interpreted as how an agile approach should be aligned in these three context levels in order to achieve the best results for a software development organization.

Code reviews, mentioned by [32] as an agility characteristic, were analyzed and more appropriately considered as a practice, not a characteristic of agility, and for this reason, it was not included in the results of this study.

The being collaborative and being cooperative characteristics are sometimes confused in the technical literature, depending on their interpretation perspectives. However, in this *quasi*-systematic review, these two characteristics were not mixed, trying to preserve the original authors' ideas, besides the fact that strong and reliable foundations to mix them in a secure way were not caught. The difference between these two agility characteristics is neither easily noticeable nor highlighted in the retrieved and selected papers. The cooperation idea seems to be more directly connected to the relationship among customers and development teams; it is associated with the role of constant feedback [28], [29], [33]. On the other hand, the idea of collaboration seems to be associated with the relationship among the development team members and relates to the continuous integration of new software increments with modified or new functionalities [24], [30], [32].

When considering characteristics of agility, it is interesting to observe that deliverables should be quickly produced. However, they should aggregate value, by providing pragmatic and effective results, according to customer perception.

Thus, eighteen characteristics were identified and described in Table 2, as below. The sources for the descriptions for each characteristic are indicated in the left column, below each characteristic name.

Table 2: Identified Characteristics of Agility

Characteristic	Description
Being Incremental [11][15][24][28][34]	Do not try to construct the system at once; the system will be partitioned in increments (small releases with new functionalities) developed in parallel and quick cycles; when the increment is completed and tested, it can be integrated into the system.
Being Cooperative [28][29][33][35]	Allow open iteration and proximity between all stakeholders (especially between customers and developers); the customer should be an active element in the development process and should provide feedback in a regular and frequent fashion.
Adaptability [11][24][28][30][33][35][36][37]	Ability and capability to quickly adapt the process to attain and react to last-minute requirements and/or environment changes as well as to attain and react to previously unforeseen risks or situations.
Being Iterative [15][30][24][32][35][36][37]	Using several short cycles guided by the product features, in which a given set of activities is completed in a few weeks; these cycles are repeated many times to refine the deliveries.

Characteristic	Description
Time-Boxing [11][24][32]	It is the establishment of a time frame for each of the programmed iterations. Big development efforts are divided in a predicted manner in multiple deliveries developed in incremental and concurrent ways,
Leanness (Also referenced as Parsimony) [11][24][33][35][37]	It is the elimination of losses and the ability to do more work with less effort; it represents an agile process characteristic that requires the minimal necessary activities to mitigate risks and achieve goals; all activities that are not necessary should be removed from the development process.
People-Orientation [24][33][34]	Privilege people over processes and technologies; developers that are empowered raise their productivity, quality, and performance; communication and cooperation are fundamental and necessary; stand up meetings and reflection workshops provide people with a chance to expose their concerns.
Being Collaborative [24][30][32]	It is the attitude by the development team members, among whom the communication is encouraged in order to disseminate information and support quick increments integration.
Transparency [24, 35]	The working practice is easy to learn and to modify, and it is documented accordingly.
Self-organization [15]	The teams decide the better ways to work; they are autonomous and can organize themselves in order to complete the work items.
Emergence [15]	The processes, principles, and work structures are recognized during project execution, and are not defined in advance; technologies and requirements are allowed to emerge during the product's lifecycle.
Reflection and Introspection [34][30]	There are meetings at the end of each subproject or iteration, in which team members can discuss what they are doing well and what needs to be changed.
Feedback Incorporation [29][30][35][37]	Teams should be able to receive and look for continuous feedback, in more frequent and quick ways.
Modularity [11][24]	This characteristic allows a process to be partitioned in components called activities, making it possible to add or remove these activities from a process when needed.
Convergence [24]	Proactively attacking risks moves the system closer to the reality sought with each of the iterations. As this action goes on, the system is delivered in increments. Everything within one's power is done to ensure success in the quickest way.
Small Teams [32][36]	A small number of people per project is needed to promote a collaborative environment and requires less planning to coordinate the activities of team members.
Constant Testing [32][35]	To prevent quality decay due to short release schedules, a high degree of emphasis is placed on testing the product throughout its lifecycle; integration testing must be automated with daily builds and regression tests to ensure that all the functionalities are properly working.
Local Teams [34]	For some methodologies this means the teams are located in the same or adjacent rooms; this works for teams from 8 to 14 people. All methodologies are sensitive to team location and are strongly grounded on rich and quick communication channels, supporting the reduction of documentation to be constructed and maintained.

Table 3 presents the final quantitative of papers in which the characteristics of agility were caught, according to the criteria established in the planned protocol for the *quasi*-systematic review.

Table 3: Incidence of Agility Characteristics on Papers

Ord	Characteristics	Number of Papers	Percentage
1	Adaptability	8	66.7%
2	Being Iterative	7	58.3%
3	Being Incremental	5	41.7%
4	Feedback Incorporation	5	41.7%
5	Leanness	5	41.7%
6	Being Cooperative	4	33.3%
7	Reflection and Introspection	4	33.3%
8	Being Collaborative	3	25.0%
9	People-Orientation	3	25.0%
10	Time-Boxing	3	25.0%
11	Constant Testing	2	16.7%
12	Modularity	2	16.7%
13	Small Teams	2	16.7%
14	Transparency	2	16.7%
15	Self-organization	1	8.3%
16	Convergence	1	8.3%

Ord	Characteristics	Number of Papers	Percentage
17	Emergence	1	8.3%
18	Local Teams	1	8.3%

Figure 4 below shows a graphical view of the information inserted in Table 3.

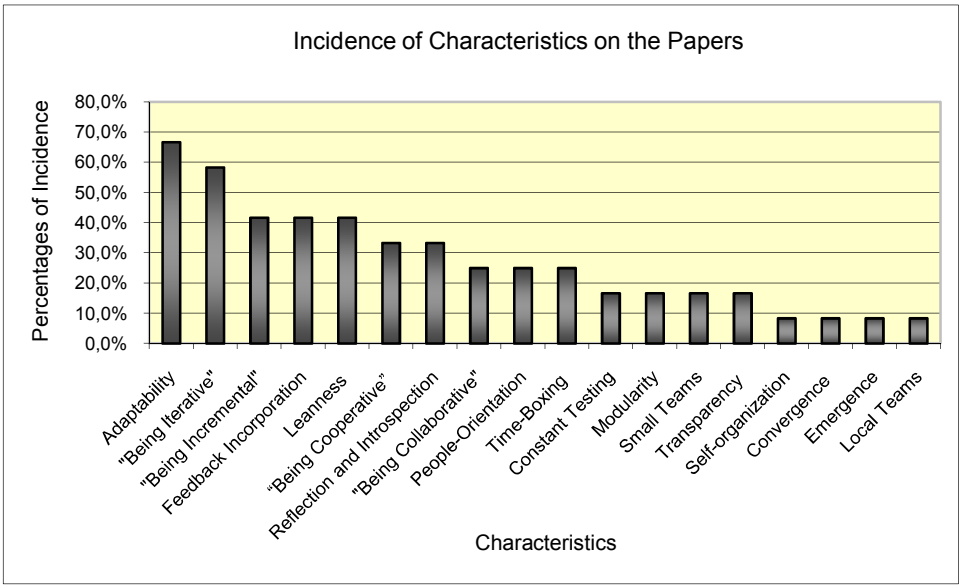

Figure 4: Incidence of Characteristics on the Papers

Observing the distribution of the eighteen characteristics found, it can be seen that the most frequent characteristics of agility treated in the papers were adaptability (66.7%) and being iterative (58.3%) followed by being incremental, feedback incorporation, and leanness with 41.7% of incidence each. The characteristics of agility with medium frequency were being cooperative, reflection and introspection, being collaborative, people-orientation, and time-boxing. The less frequent characteristics of agility treated in the papers were constant testing, modularity, small teams, transparency, self-organization, convergence, emergence, and local teams.

Table 4 shows quantitative paper distribution approaching characteristics of agility, according to the criteria adopted in the protocol, per year of publication range. While Table 3 shows the number of articles in which the characteristics of agility were addressed, in Table 4 the focus is to highlight the time range in which the characteristics of agility were the object of interest of authors, separating those which have been addressed only occasionally, from those which were in evidence in a broader range of time.

Table 4: Distribution of Characteristics per Publication Time Range

Characteristic	Range		Range	No. of Papers	Papers Per Year
	From	To			
Adaptability	1998	2009	12	8	**0.67**
Leanness	1998	2009	12	5	0.42
Being Iterative	2001	2009	9	7	**0.78**
Reflection and Introspection	2002	2009	8	4	**0.50**
Time-Boxing	1998	2005	8	3	0.38
Being Incremental	1998	2004	7	5	**0.71**
Being Cooperative	2003	2009	7	4	**0.57**
Transparency	2003	2009	7	2	0.29
People-Orientation	2001	2006	6	3	**0.50**
Being Collaborative	2001	2006	6	3	**0.50**

Characteristic	Range		Range	No. of Papers	Papers Per Year
	From	To			
Constant Testing	2005	2009	5	2	0.40
Feedback Incorporation	2006	2009	4	5	**1.25**
Modularity	1998	2001	4	2	**0.50**
Small Teams	2005	2008	4	2	**0.50**
Emergence	2004	2004	1	1	1.00
Self-organization	2004	2004	1	1	1.00
Local Teams	2002	2002	1	1	1.00
Convergence	2001	2001	1	1	1.00

This data distribution can be seen below in

Figure **5**. It shows characteristics of agility which have been under the focus of the authors only sporadically and which were addressed in a broader range of time.

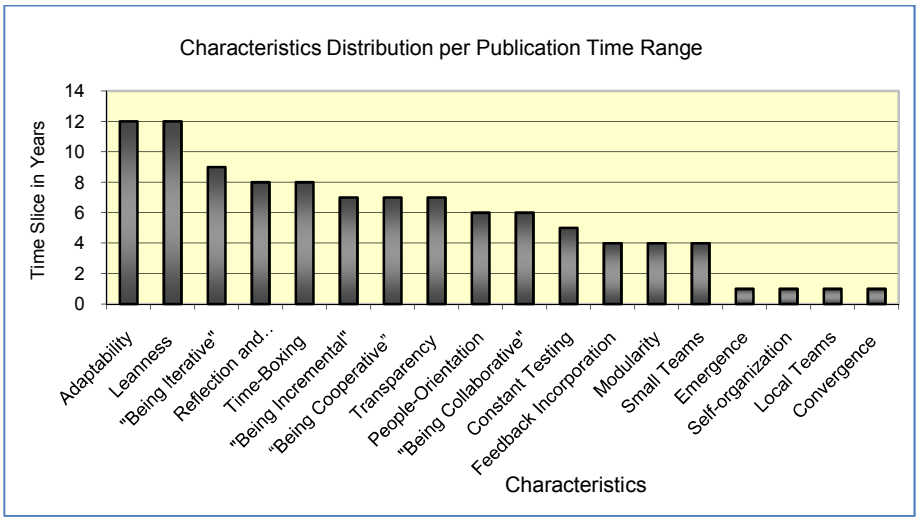

Figure 5: Distribution of Characteristics per Publication Time Range

When looking at Table 4, the characteristics of agility with the highest time range and highest density of publication in the period of time are: adaptability, being iterative, reflection and introspection, being incremental, being cooperative, people orientation, and being collaborative (all of these with half or more the time range and papers per year). Some characteristics of agility have a publication density at least 0.5 papers published per year: adaptability, being iterative, reflection and introspection, being incremental, being cooperative, people-orientation, being collaborative, feedback incorporation, modularity, and small teams. Emergence, self-organization, local teams and convergence were approached in only one paper each.

The goal of the study was not to map the characteristics of agility found for the values or principles of the agile manifesto [19], but rather, to identify which are the desirable characteristics to support agility in software processes.

Based on findings described in previous Sections, we can qualify some characteristics as core ones, to achieve agility in software processes. According to the context presented, and considering the consolidated descriptions for each characteristic of agility in Table 2, the characteristic that follows the Agile Manifesto most closely seems to be adaptability. After that, and considering more pragmatic aspects in software processes, the characteristics of being incremental and being iterative should be considered together. To these three characteristics, three others should be added, which are directly associated with the agile methods' core idea of focusing on people who conduct the activities in the processes. These three other characteristics are: being collaborative, being cooperative, and people-orientation.

Feedback incorporation and reflection and introspection, according to their descriptions in Table 2, are important characteristics as they can support lessons learned by teams and continuous improvement of the software development process. Both have more than 33% of incidence in the papers included in the study. Besides these, two necessary characteristics for the success of agility in software processes and essential to any process should be considered: leanness and time-boxing.

Other agility characteristics mentioned in this *quasi*-systematic review are also important for the success of agile software processes. However, they can be considered as subsidiary or derived from those previously referred as core ones.

3 Agile Practices – A Systematic Review

It has been demonstrated that values, principles and agile practices bring benefits to many organizations [44]. Values bring purpose to practices. Practices are evidence of values, which are universal and expressed on a higher level. Practices are clearer and more specific. Bridging the gap between values and practices are principles, which are domain-specific guidelines. Just as values bring purpose to practices, practices bring accountability to values [45]. Practices are activities that implement the principles governing the processes or methods. These in turn are ideas, understandings or goals that are behind the practices [46].

In the last 16 years, an interesting changing in the software development processes was the introduction of the 'agile' perspective. However, software development teams need support the choice of the right combination of agile practices based on their needs [47]. The aim of this study is to investigate software practices recommended in the context of agile approaches for software development. A research protocol has been formalized to conduct a systematic literature review. The searches were conducted in February 2010. The data was analyzed and the initial results summarized.

3.1 Planning

The objective of this secondary study is to identify the software practices usually used in the context of agile approaches for software development. The question made was: what are the software practices that can be considered agile in the context of software development approaches? The problem was finding and identifying agile practices for software development. The intended application of the results was to serve as a basis to support research involving software practices that can be used as alternatives to embed agility into software processes. The research issue, also in this review, was structured by four elements (population, intervention, comparison, and outcome), according to [22]. The population was the set of software development projects. The intervention was the agile software development processes. There was no comparison. The outcome was a set of agile practices. Three papers were used as controls: [25], [48], [29]. The sources were collected from the following digital databases, including conferences, journals, and technical reports indexed by Compendex EI, IeeeXplore, Inspec, Web of Science, Scopus, Science Direct, and the ACM digital library.

The keywords for population were "software projects", "software systems", "software development", "software engineering". The keywords for intervention were "agile approaches", "agile processes", "agile methods", "agile methodologies", "agile development", "agile software development", "agile projects". The keywords for outcome were "practices", "software practices", "agile practices", "sub practices", "agile techniques", "techniques".

The inclusion and exclusion criteria were: documents should be available on the Web; documents should include agile software practices in the context of agile approaches for software development; documents should provide a description of the agile practices they see. In order to select studies, a researcher applied the search strategy to identify potential documents. Documents identified by this researcher were analysed through the inclusion and exclusion criteria. Subsequently, each document excluded was analysed by a second researcher. In case of conflict, the document was included. Finally, documents were read by researchers to extract information about software practices in the context of agile software development approaches. The sources of the documents were assumed to be reliable, as their texts have undergone external reviews which serve as a filter so they present enough quality to contribute to this secondary study.

The following information was extracted from each paper, after running the selection process: document title, author(s), source, year of publication, and name and description of the agile software practices. The results were tabulated. An analysis was done to identify similarities among significant practices for software processes in the

context of agile software development. The frequency with which each practice has been reported by different authors was considered. If authors had more than one paper, repeated counts were avoided. The search string taken as the basis for all search engines, structured according to Pai et al. [22] was:

("software projects" or "software systems" or "software development" or "software engineering") AND ("agile approaches" or "agile processes" or "agile methods" or "agile methodologies" or "agile development" or "agile software development" or "agile projects") AND ("practices" or "software practices" or "agile practices" or "subpractices" or "agile techniques" or "techniques").

3.2 Execution

Search execution with the engines above returned 6,696 references, published between 1998 and 2010. Repetitions were eliminated, keeping the remaining paper counted for the digital library with the largest number of items recovered. All the controls were retrieved. From 5,093 references without repetitions, titles and summaries were assessed for each retrieved reference, and references that clearly did not address issues concerned with the research were excluded. After the removal of such items, 441 references were selected for the study.

3.3 Results and Discussion

Following the criteria established in the research protocol, 236 occurrences of practices were extracted from 24 papers published between 2001 and 2009 [25],[48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70] which were tabulated accordingly. In order to avoid bias, technical papers concerned with performance studies or specific issues of a particular practice were not considered. Some practices identified are associated to one or more commercial agile methods while others are not associated with any method. Some papers approached agile practices, but did not present an adequate description of its meaning. In order to identify repetitions, names and descriptions of the 236 occurrences of agile practices collected were analyzed and grouped, leading to 51 different agile practices. Figure 6 shows the distribution of the 24 documents, per year of publication.

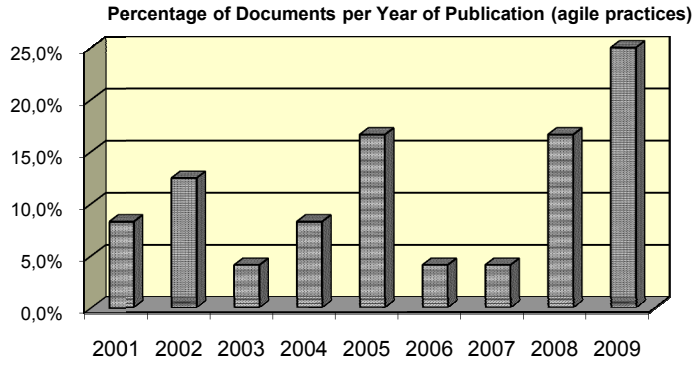

Figure 6: Documents Describing Agile Practices per Year of Publication

It is possible to see a significant increase of technical papers published in 2009 that meet the research protocol criteria, as well as the growing interest by the scientific community on agile approaches, considering the results of another secondary study on the characteristics of agility, previously made by the authors of this study, as described in Section 2.

From 51 different agile practices identified, 17 were approached in the technical papers more than once and they were selected to be further analyzed. Figure 7 shows the incidence of the agile practices in the papers.

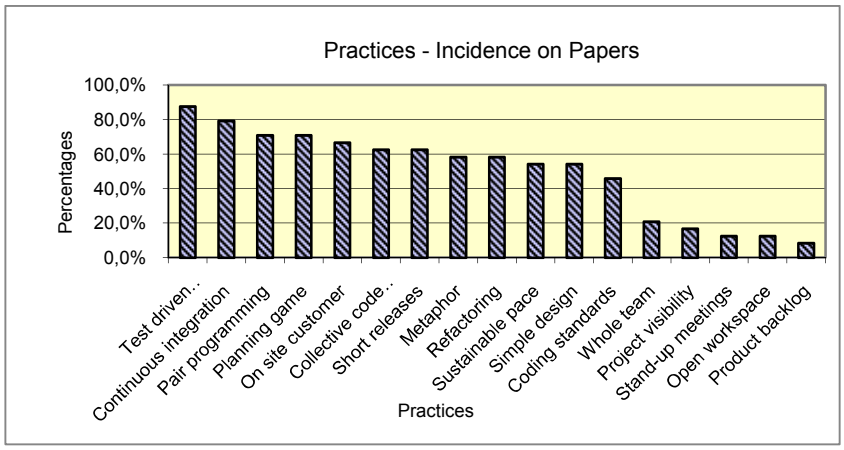

Figure 7: Incidence of Agile Practices on Papers

The descriptions for each practice from different papers were consolidated as follows:

Test driven development- Every programmer writes test cases before they write their production code. Developers write unit tests before coding and encourage customers to write acceptance test cases. Upon the initial execution of such a test case, it will fail as there is no corresponding code that implements the particular feature or condition being tested. Then the developer builds just enough code for the test to pass or to satisfy the current goal, followed by refactoring, to improve readability and remove duplications.

Continuous integration- The members of a team should frequently integrate their work, every time new changes or a task is completed, to reveal integration problems and to detect system failures as soon as possible. Usually each person integrates at least daily. All tests must still pass after integration or the new code should be discarded.

Whole team- Refers to the practice of including all necessary skills and perspectives in the team for it to succeed, emphasizing teamwork and that all team members share a purpose and support each other. Customers, end-users and other business stakeholders should have direct involvement in the project, to understand system behaviour early in the lifecycle.

Pair programming- All the source code produced should be written by two people working at the same time on the same machine. The programmers alternate the roles of driver and navigator throughout the day. The driver has the control of the keyboard and mouse; s/he implements the code, explaining it to the navigator. The driver's major responsibilities are development tests and algorithm construction, whereas the obligations of the navigator are to watch for both syntax and semantic defects, deciding if this is the best way to implement the functionality.

Planning game- Together, developers and customers play the 'Planning Game' where the customer chooses those User Stories that comprise the most important content for a short, incremental deliverable. Each short implementation increment is accepted and tried by the customer. Then, the remaining User Stories are re-examined for possible requirement and/or priority changes and the Planning Game is re-played for the next implementation increment. Planning is continuous and progressive.

On site customer- This practice indicates that the customer should be a member of the development team. To clarify and validate the requirements and set priorities, an on-site customer representative works with the team. Thus a customer works close together with the developers all the time, to answer the questions, resolve disputes, set small scale priorities, perform acceptance tests, and ensure that the development is progressing as expected.

Collective code ownership- The code repository should be freely accessible to all programmers, who are encouraged to make changes in all the code, anywhere and whenever they feel necessary, without asking 'permission' from anyone. Any programmers' pair who sees an opportunity to add value to any portion of the code can do so at any time. As the code is accessible to more minds, programmers should examine code that is written by others.

Short releases- Release working software frequently. This practice shortens the release cycle to speed the feedback from the customer. Given that requirements often change, one keeps release cycles short and ensures that each release produces an useful software system that generates business value for the customer. An initial version of the system is put into production quickly, after little iteration. At the end of each release, the customer reviews the interim product; identifying defects and adjusting future requirements.

Metaphor- This practice presents a simple shared story which explains the essence of how the system works to give both developers and customers a common understanding of the project. In a sense, the metaphor serves as the high-level software architecture. While thinking about an appropriate metaphor, developers must expand their perspective and analysis of the developed application.

Refactoring- This is a practice for restructuring an existing body of code, or constantly improving its understandability and maintainability, as well as its design, altering its internal structure without changing its external behaviour or system functionality. The different forms of refactoring involve simplifying complex statements, abstracting common solutions into reusable code, and removing duplicate code. When code is re-factored, it should still pass all the unit test cases in the code base.

Sustainable pace- This practice emphasizes working 'only as many hours as you can to be productive and only as many as you can sustain'. Work no more than 40 hours a week as a rule, no more than eight hours every day, without working overtime a second week in a row. During crunch periods when overtime is worked, the artefacts produced are of poor quality. Requirements should be selected for each iteration, such that developers do not need to put in overtime.

Simple design- The emphasis of this practice is on designing the simplest possible solution that is implementable at that moment. Unnecessary complexity and extra code will be immediately removed. One should not add additional features to one's artefacts unless they are justifiable. Programmers should not try to predict future needs. The most cost-effective development approach should focus on solving today's problems rather than designing for future changes.

Coding standards- As developers code different system parts with various team members, coding standards are a must. A coding standard makes the code easier to understand, increases readability and improves consistency among team members. The standard should be easy to follow and to be voluntarily adopted. It is agreed upon by the team to ensure that communication is made through code and lets developers easily understand each other's code.

Whole team- Refers to the practice of including all the necessary skills and perspectives in the team for it to succeed, emphasizing teamwork and the sense of all team members sharing the purpose and supporting each other. Customers, end-users and other business stakeholders should have a direct involvement in the project, to understand system behaviour early in its lifecycle.

Project visibility- Agile projects strive to provide immediate status and feedback to the teams. One can create project dashboards on the Web that host, at any point in time, all status and measurements regarding the progress of the project. A preview of the project in relation to the user stories that the teams have committed to deliver at the end of the iterations should be included. Models should be made accessible for the entire teams.

Stand-up meetings- Quick meetings (approximately 15 minutes) set to keep track of project progress, highlight issues and organize the daily activities. Each team member briefly states what he or she has been working on and what progress has been made. The standing component of the meeting tended to ensure that everyone be alert and attentive, besides intending not to go beyond the 15 minutes, and encouraging participants to conclude it quickly.

Open workspace- Developers work in a common workspace; a large room with small cubicles is preferred. Pair programmers should be placed in the centre of the space. Workplace layouts should have common areas that facilitate open communication. An alternative set with individual workstations at the fringe and common development machines in the centre could be appropriate.

Product backlog- This practice includes the tasks for creating the Product Backlog list, and controlling it consistently during the process by adding, removing, specifying, updating, and prioritizing items. Product Backlog defines everything that is needed in the final product based on current knowledge. It comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced.

From these, 12 practices have been addressed by technical papers published in at least 6 different years in the time interval of the selected publications: test driven development, continuous integration, pair programming, planning game, on-site customer, collective code ownership, small releases, metaphor, refactoring, sustainable pace, simple design, and coding standards. Five other practices, making 17 that were approached in the papers more than once were: whole team, project visibility, stand-up meetings, open workspace, and product backlog.

Among the 51 distinct practices identified in this secondary study, several of them were targets of interest to researchers only occasionally, the vast majority before 2004. The proposal now is to continue this research work, considering the aforementioned 17 agile practices. These results were already published in 2011 [71]. It is interesting to keep in mind that the performance of all practices, in terms of agility, depends on the environment in which they are applied, the project context, the way chosen to implement them in the process and the intensity with which they are applied and upheld by the team.

4 Surveying Characteristics of Agility and Agile Practices

This Section presents the results of a primary study aimed at evaluating the characteristics of agility and agile practices identified through the systematic reviews presented in Sections 2 and 3. First, the study planning is described. Then, details about its execution are presented, followed by a discussion of the results of the study.

4.1 Survey Planning

The object of study includes two initial sets: the first one includes characteristics of agility identified through systematic literature review, to support the insertion of agility into software processes; the second one includes agile practices, also identified through systematic literature review. Both systematic reviews were held in 2010.

Using the GQM approach [72], the aim of this study was to analyze the characteristics of the agility set, with the purpose of characterizing them as regards their pertinence and relevance to characterize a software process as agile, as well as their corresponding relevance level as related to that characterization, from the point of view of researchers in agile software processes, in the context of software projects which adopt agile development approaches. And also to analyse the set of agile practices with the purpose of characterizing, as regards their pertinence and relevance in achieving agility in software processes, from the point of view of researchers in agile software processes, in the context of software projects which adopt agile development approaches.

The research questions and related metrics considered were:

- Q1: Are the characteristics of agility extracted from technical literature pertinent to characterize a software process as agile?
- M1: number of characteristics of agility classified as pertinent to characterize an agile software process, according to the opinion of the study participants.

- Q2: Are there any additional characteristics of agility which are pertinent to characterize an agile software process which are not present in the original set?
- M2: the number of additional characteristics of agility to be included in the initial set, according to the opinion of the study participants.

- Q3: Is there any characteristic of agility in this initial set that is not pertinent to characterize an agile software process?
- M3: number of characteristics of agility to be removed from the initial set, according to the opinion of the study participants.

- Q4: What is the order of relevance of the characteristics of agility in the final set when considering an agile approach for a software process to be applied in software projects?
- M4: the order in a set of characteristics of agility sorted by relevance level.

- Q5: Are the agile software practices, as extracted from technical literature, pertinent for agile approaches to software processes?
- M5: number of agile practices classified as pertinent as regards agile approaches for software processes, according to the opinion of the study participants.

- Q6: Are there any additional agile practices which are pertinent to be adopted by an agile software process which are not present in the original set?
- M6: number of additional agile practices to be included in the initial set, according to the opinion of the study participants.

- Q7: Are there any agile practices in this initial set which are not pertinent in the context of agile approaches for software process?
- M7: number of agile practices to be removed from the initial set, according to the opinion of the study participants.

- Q8: What is the order of relevance of the agile practices in the final set when considering their adoption in agile software processes?
- M8: order of each practice in a set of agile practices sorted by relevance level.

The following hypotheses were considered in this study:

Null Hypothesis 1 (H0 1):

- The initial set of characteristics of agility is complete, all the characteristics of agility present in the initial set are pertinent to the characterization of agile software processes, and there are no characteristics to be included or removed.
- Cr – characteristics classified as not pertinent and need to be removed from the initial set;
- Ci – characteristics not present in the original set and classified as pertinent, which must be included in the initial set of characteristics of agility;

$$(H0\ 1: |Cr| = |Ci| = 0).$$

Alternative Hypothesis (H1):

- There are characteristics in the initial set of characteristics of agility which were classified as not pertinent to the characterization of agile software processes. So, they should be removed from the initial set of characteristics of agility.

$$(H1: |Cr| \neq 0).$$

Alternative Hypothesis (H2):

- There are characteristics that are not present in the initial set of characteristics of agility that were classified as pertinent for the characterization of agile software processes. So, they should be included in the initial set of characteristics of agility

(H2: $|Ci| \neq 0$).

Null Hypothesis 2 (H0 2):

- All the characteristics of agility have the same relevance level to support agility insertion in software processes.
- CRLi – relevance level related to the agility characterization in software processes, for the characteristic "i", where i is a number between 1 and *n*, and *n* is the total number of characteristics of agility considered pertinent.

(H0 2: CRL1 = CRL2 = … = CRLn).

Alternative Hypothesis (H3):

- There is at least one characteristic of agility with a relevance level different from the other characteristics of agility related to the characterization of agile software processes.

(H3: $\exists$ i,j | CRLi ≠ CRLj, where "i" and "j" numbers between 1 and n, "i ≠ j").

Null Hypothesis 3 (H0 3):

- The initial set of agile practices is complete, all practices present in the original set are pertinent, as regards agile approaches to software processes and there are no practices to be included or excluded.
- Pc – initial set of agile practices;
- Pr – practices classified as not pertinent which need to be removed from the initial set;
- Pi – practices not present in the original set, classified as pertinent and that need to be included in the initial set of agile practices.

(H0 3: $|Pr| = |Pi| = 0$).

Alternative Hypothesis (H4):

- There are practices in the initial set of agile practices which were classified as not pertinent as regards agile approaches for software processes. So, they should be removed from the initial set of agile practices.

(H4: $|Pr| \neq 0$).

Alternative Hypothesis (H5):

- There are practices in the initial set of agile practices which were classified as pertinent as regards agile approaches for software processes. So, they should be included in the initial set of agile practices.

(H5: $|Pi| \neq 0$).

Null Hypothesis 4 (H0 4):

- All the agile practices have the same relevance level in the context of agile approaches for software processes.
- PRLi – The relevance level related to agile approaches for software processes, for practice "i", where i is a number between 1 and *m*, *m* is the total number of agile practices considered pertinent.

(H0 4: PRL1 = PRL2 = … = PRLm).

Alternative Hypothesis (H6):

- There is at least one practice with a relevance level different from the other agile practices related to agile approaches context for software processes.

(H6: $\exists$ i,j | PRLi $\neq$ PRLj, where "i" and "j" are numbers between 1 and m, "i $\neq$ j").

The questions answered by the survey participants included:

Step 1: Are the characteristics presented pertinent or not pertinent to characterize a software process as being agile? Are there new characteristics of agility to be included (with its meaning)?

Step 2:
After the definition of the final set of pertinent characteristics of agility, the goal is to set the level of relevance for each characteristic, in the context of agile software processes, according to the perceptions of the participants. Four relevance levels were established:

(0) Not relevant (no relevance) Lowest level of relevance and it means the characteristic would not have any influence on the characterization of a software process as being agile. The agility of a software process would not be affected if this characteristic were absent in the software process, independently of particular scenarios or development environments.

(1) Little relevant (insignificant) Indicates that the characteristic would not affect significantly or that it has a small influence on the characterization of a software process as being agile. The absence of the characteristic would not seriously compromise the agility of a software process in all or in most of the scenarios or development environments.

(2) Highly relevant (high relevance) Indicates that the characteristic has strong influence on the characterization of a software process as being agile. The absence of the characteristic would compromise the agility of a software process in all or in most scenarios or development environments.

(3) Absolutely relevant indicates that the characteristic is vital or imperative to characterize a software process as being agile. The absence of the characteristic would prevent the characterization of a software process as an agile one.

Step 3: Are the agile practices presented pertinent or not pertinent in the context of agile approaches for software processes? Are there new agile practices to be included (with its meaning)?

Step 4:
After the definition of the final set of pertinent agile practices, the goal is to set the level of relevance for each practice, in the context of agile software processes, according to the perceptions of the participants. Four relevance levels were established:

(0) Not relevant (without relevance) Lowest level of relevance, it means the practice would not have any influence on the adoption of an agile approach. The agile approach of a software process would not be affected if the referenced practice is absent in the software process, independently from particular scenarios or development environments.

(1) Little relevant (insignificant) Indicates that the practice would not significantly affect it, or that it has little influence on the adoption of an agile approach for a software process. The absence of the practice would not seriously compromise the agile approach for the software process in all or in most of the scenarios or development environments.

(2) Highly relevant (very relevant) Indicates the practice has strong or considerable influence on the adoption of an agile approach. The absence of the practice would compromise the agile approach for a software process in all or in most of the scenarios or development environments.

(3) Absolutely relevant: Indicates the practice is vital or imperative for the adoption of an agile approach. The absence of the practice would prevent the agile approach for a software process.

The questionnaire was available in the Internet, divided into five parts: subject characterization, identification of the pertinence for characteristics of agility, definition of the relevance level for characteristics of agility, identification of the pertinence for agile practices, and definition of the relevance level for agile practices.

To make different considerations on the answers from subjects, a weight will be attributed for each subject according to four perspectives: academic background of the subject, number of papers on agile processes published by the subject, experience level in the use of agile approaches in software projects, and total number of software projects using agile processes one has participated in. The formula used to define subject weight has been adapted from [73].

$$S(i) = f(i) + p(i) + e(i) + \frac{t(i)}{MedianTP}$$

S(i) is the weight attributed to subject i;
f(i) is the academic background. The options for this field are:
0, if the subject holds an Undergraduate degree;
1, if the subject holds a Specialization degree;
2, if the subject holds a Master degree;
3, if the subject holds a PhD or DSc degree;
p(i) is the indicator for the number of papers on Agile Processes or Agile Methods published by the subject. The options for this field are:
0, if the number of papers is between 1 or 2 papers;
1, if the number of papers is between 3 or 4 papers;
2, if the number of papers is between 5 or 6 papers;
3, if the number of papers is greater than 6 papers;
e(i) is the subject's experience level in the use of Agile Approaches in software projects. The options for this field are:
0, if the experience level is low;
1, if the experience level is medium;
2, if the experience level is high;
3, if the experience level is very high;
t(i) is the estimated number of software projects using Agile Approaches one has participated in.
MedianTP is the median for the total number of software projects using Agile Approaches, considering the answers of all subjects.

The values for indicator p(i) were defined after analyzing a sample of authors extracted from the papers that contributed with at least one characteristic of agility in the first execution of the protocol of the systematic review conducted by [74]. We considered a population of papers registered in the reference manager JabRef (http://jabref.sourceforge.net/), totalling 897 references (ACM Digital Library references not included). From these, we retrieved all the references on agile approaches published by each one in the sample of authors considered. This data led to following measurements: median = 2; mean = 2.18; mode = 1; standard deviation = 1.62; maximum value = 7.

Once the set of subjects that assess the characteristics of agility and agile practices is the same, the calculations to define what the characteristics of agility/agile practices are, and that are pertinent, are also the same, depending upon the different participant responses to the sets of characteristics and practices.

To determine which characteristics of agility/agile practices are pertinent in the context of agile software approaches, it is necessary to compute the responses of each participant and consider their respective weight for both characteristics and practices. The answers, with their respective weights were computed using the formulas given below, adapted from [73]:

$$Pertinence(j) = \sum_{i=1}^{m} (Answer(i, j) * S(i))$$

Pertinence(j) is the total value of the answers from all subjects (multiplied by their weights) on the pertinence of the characteristic/practice j in the agile software process context.
Answer(i,j) is the indicator of pertinence (1) or no pertinence (0) defined by subject i for the characteristic/practice j
S(i) is the weight attributed to subject i;
m is the total of subjects who participated in the survey

The definition of whether a characteristic of agility/agile practice is pertinent or not pertinent to characterize an agile software process should be based on a cut-off point, that is, a threshold indicating whether the characteristic/practice is included in the final set (value greater than, or equal to, the threshold). The threshold adopted is 50% of the maximum value that could be obtained for a characteristic/practice j in the variable Pertinence(j) if all subjects answer "YES" regarding its pertinence in the agile software process context.

$$Threshold = 0{,}5 * \sum_{i=1}^{m} S(i)$$

S(i) is the weight attributed to subject i;
m is the total number of subjects who participated of the survey

Thus, the criteria are:

if Pertinence(j) < Treshold, characteristic/practice j is classified as "not pertinent" and should be removed from the set.
if Pertinence(j) ≥ Treshold, characteristic/practice j is classified as "pertinent" and should be kept in the set.

In the set obtained from the initial set of characteristics/practices, according to the threshold established, the characteristics/practices indicated by the subjects to complete the initial set (characteristics/practices included by the subjects will be considered as pertinent) should be added.

To define the relevance level of a characteristic of agility/agile practice previously classified as "pertinent", it is necessary to first sum the answer from each subject (multiplied by its respective weight).

$$RLevel(j) = \sum_{i=1}^{m} (Scale(i, j) * S(i))$$

RLevel(j) is the total value of the answers from all subjects (multiplied by their weights) for the Characteristic/Practice j
m is the total number of subjects who participated in the survey
Scale(i,j) is the scale of relevance level (0-3) as defined by subject i for Characteristic/Practice j
S(i) is the weight attributed to subject i;

After this step, the characteristics/practices will be ordered by their RLevel(j). The most relevant characteristics/practices will be those with the higher value for RLevel(j).

Each subject should fill the information regarding his/her characterization; after that, he/she should indicate the pertinence of a set of characteristics of agility (presented in alphabetical order) to characterize a software process as an agile one; in the sequence, he/she should define the relevance level of these characteristics; then he/she should indicate the pertinence of a set of software practices (presented in alphabetical order) regarding the adoption of an agile approach for software processes; and finally, he/she should define the relevance level of these practices.

The population of this survey is comprised by selected authors of scientific papers published in three years (2008-2010) regarding agile approaches identified by, and referenced in, systematic reviews on characteristics of agility and agile practices in software processes published by [71]. To avoid bias, authors of papers describing characteristics of agility and/or agile software practices identified and considered in the systematic reviews were not invited to participate. The subjects were contacted by email and could access a website with the questionnaire using a login and password sent in the body of the contact email.

The independent variables in the study are the initial set of characteristics of agility and the initial set of agile practices. Dependent variables are the final set of characteristics of agility, the relevance level for each characteristic of agility included in the final set concerned to their relevance to characterize a software process as an agile one, the final set of agile practices, and the relevance level for each agile practice included in the final set concerned to their relevance to the adoption of an agile approach for a software process.

Concerned with validity conclusion, and attempting to avoid bias, the authors of papers describing characteristics of agility and/or agile practices identified in the systematic reviews were not invited to participate. It is assumed that the remaining authors constitute a population which is representative in the context of researchers on agile approaches, and they answered the questionnaire using their background and experience in this field.

After running the survey, the confidence level (1-E_0) of the data obtained was evaluated using the following formula, adapted from [75]:

$$E_0 = \sqrt{(N - n)/(N * n)}$$

- E_0 = Confidence Level (e.g.: 0.05 → 95%)
- N = Population Size
- n = Sample Size

The verification of the Null Hypothesis 1/Null Hypothesis 3 was done by simple verification of the number of characteristics of agility/agile practices in the set of characteristics/practices to be included and in the set of characteristics/practices to be removed from the initial set. The final set of characteristics/practices was defined as follows: the initial set of characteristics/practices will be changed by adding the characteristics/practices present in the set of characteristics/practices to be included and removing the characteristics/practices present in the set of characteristics/practices to be removed. The result of these two operations will produce the final set of characteristics of agility/agile practices.

The criteria used to define the items in each set (characteristics/practices to be included and characteristics to be removed) were:

1. For the set of characteristics/practices to be removed, the pertinence for the characteristic/practice should be lower than the established threshold previously described;
2. For inclusion in the set of characteristics/practices to be included, a characteristic/practice should be indicated for inclusion at least by two subjects.

The verification of the Null Hypothesis 2/Null Hypothesis 4 was done by the simple observation of whether all the characteristics of agility/agile practices have the same relevance level in the context of agile approaches for software processes. If they have the same relevance level, in this sense there is no difference among them, and a characteristic/practice more relevant than another cannot be indicated. The relevance level (Rlevel) of a characteristic of agility/agile practice is calculated using the formula already described in this Section.

4.2 Survey Execution

The instrument was available from January 6, 2011 to March 6, 2011, at http://lens-ese.cos.ufrj.br/surveyagile/. By email, 117 subjects were invited. By the end of January 2011 there were 31 accesses, amongst which 21 subjects recorded their responses. Among the respondents there were subjects from Austria, the US, Canada, Jordan, Denmark, Finland, and Italy, amongst others. Unfortunately there are no mechanisms to check whether all messages sent by email reached their recipients. Some may no longer be in use.

In Figure 8 and Figure 9 we show an idea for the login screen, with subject characterization and agreement screen.

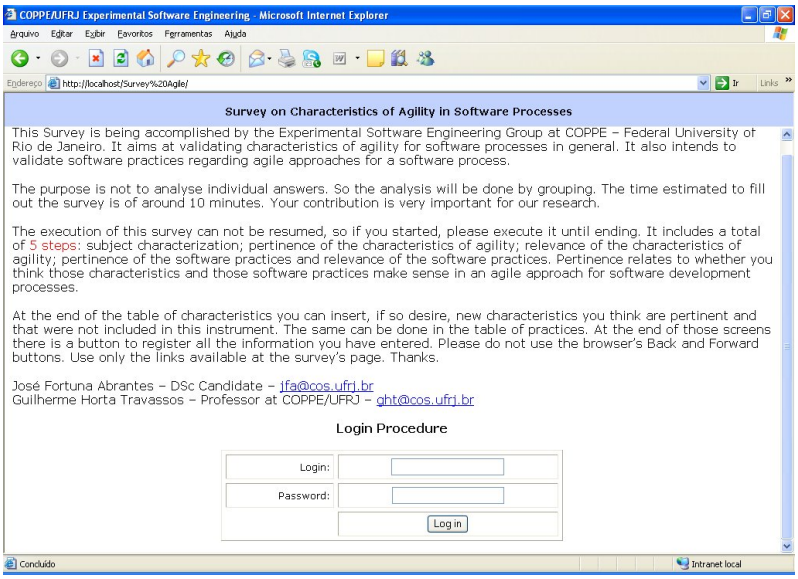

Figure 8: Login Screen

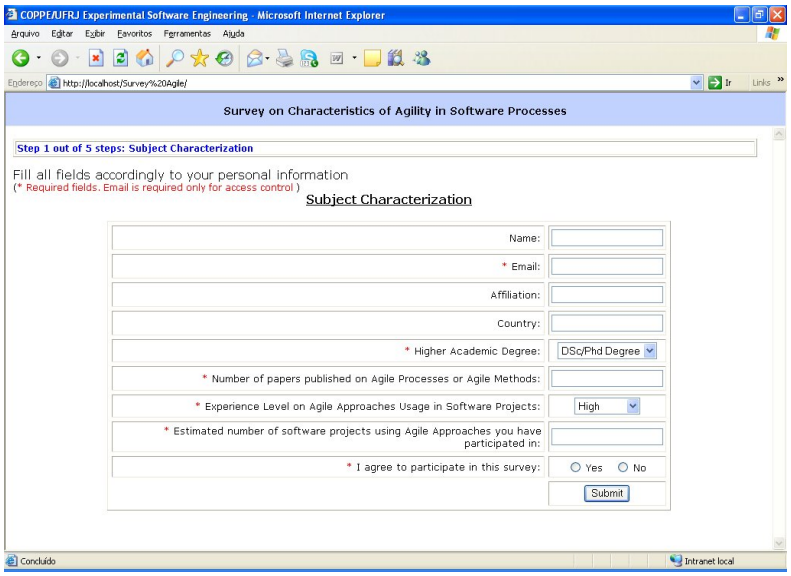

Figure 9:Subject Characterization and Agreement Screen

These screens were followed by more 4 screens: pertinence for characteristics of agility, relevance level for characteristics of agility, pertinence for agile practices, and relevance level for agile practices. Figure 10 and Figure 11 give an idea of the screens: relevance level for characteristics of agility and pertinence for agile practices. The relevance level screen for agile practices is similar to the relevance level screen for characteristics of agility. The pertinence screen for characteristics of agility is similar to the pertinence screen for agile practices.

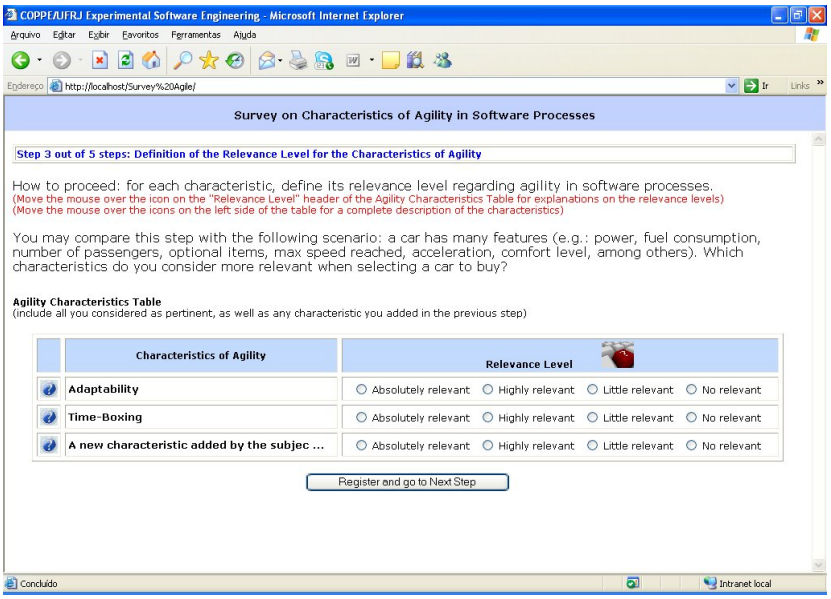

Figure 10: Relevance Level Screen for Characteristics of Agility

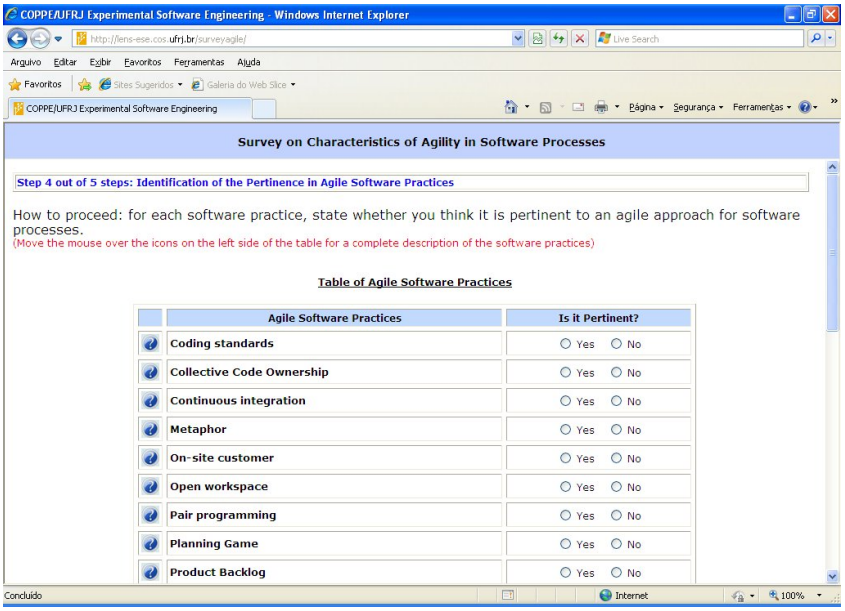

Figure **11**: Pertinence Screen for Agile Practices

The last effective screen was the Thank You screen, where the subject could freely place comments. This screen is shown in Figure 12. The Thank You screen was followed be a final screen advising the subject the survey was completed and that the instruments were being closed.

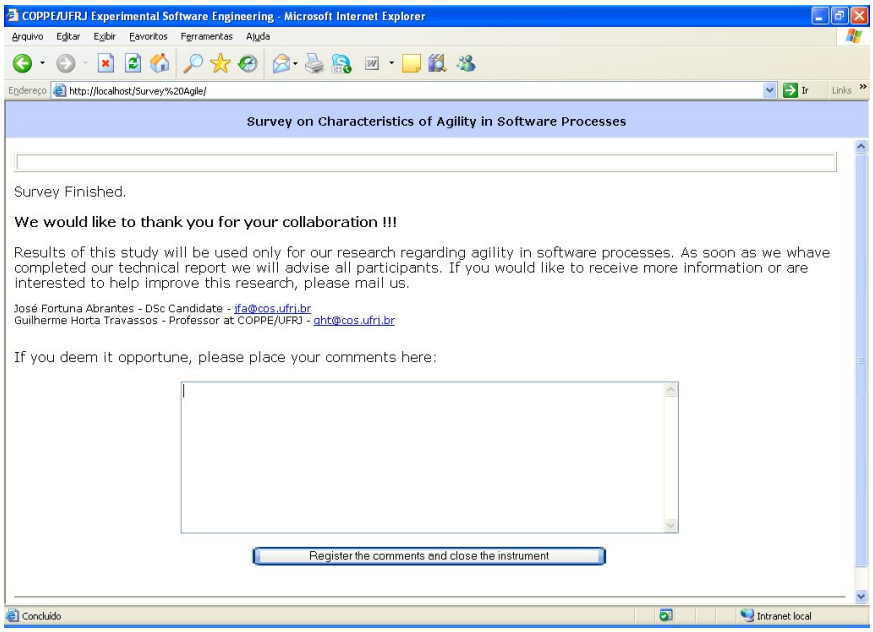

Figure 12: Thank You Screen

4.3 Survey Results and Discussion

Each subject had one's characterization attributes tabulated and an individual weight computed. Answers were weighted individually, according to subject experience levels and/or skills.

To invite the subjects, 117 emails were sent with presumably valid addresses. There were responses from 21 subjects, which led to a confidence level of around 80.23% for the collected data.

Once the subjects' weight was computed, the analysis of the results from the evaluation of the pertinence of the characteristics of agility could proceed, applying the criteria and procedures described in the survey plan. After calculations for each characteristic of agility, a graph was obtained showing the individual levels of pertinence achieved. Figure 13 presents a graphical representation of pertinence levels.

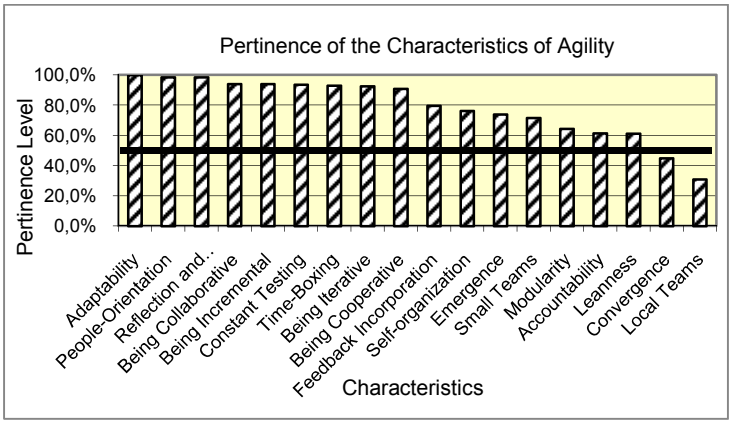

Figure 13: Pertinence for Characteristics of Agility

As the lower limit for a characteristic of agility to be considered relevant was 50%, we then have the characteristics Convergence and Local Teams (pertinence levels 45.0% and 30.9% respectively) were not considered pertinent and will not be part of the initial knowledge repository to be adopted for further studies in a broader research

context. It was observed that the participants did not suggest any new characteristics of agility with their respective descriptions, differently from the initial set. The H0 1 presented in the survey plan was not observed. However there is a risk in these findings, as the confidence level for the sample was of 80.23%.

Applying the same criteria and procedures, the calculations for agile practices were carried out. Figure 14 shows the pertinence levels obtained for each agile practice.

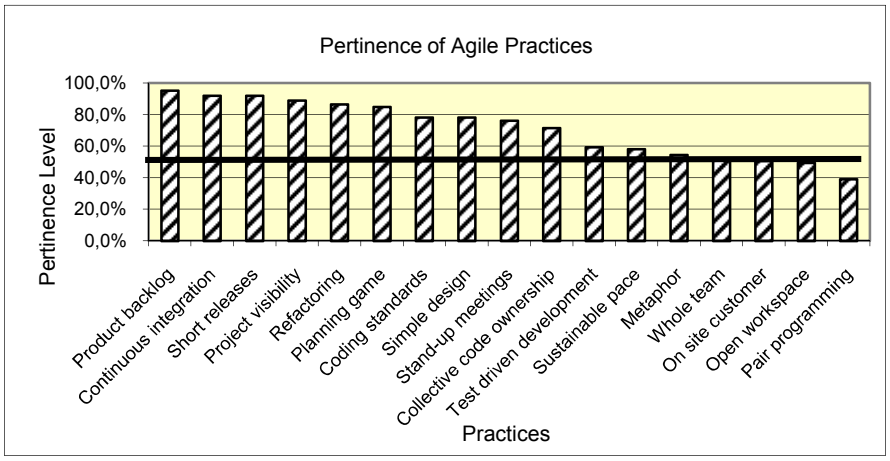

Figure 14: Pertinence for Agile Practices

Two agile practices did not reach a minimum of 50% for pertinence level: open workspace and pair programming. For this reason, these practices were removed from the initial knowledge repository knowledge repository to be adopted for further studies in a broader research context.

There was no indication of new agile practices by the subjects. However, H0 2 was not observed, as two agile practices were removed from the initial set. However, the same reservations already made in relation to risk in not observing the H0 1, are valid now for the non-observation of H0 2 too (confidence level for the sample is around 80.23%).

Once the pertinence for each characteristic and for each practice is identified, their relevance level may be computed, according to the procedure described in the survey plan. The results are shown in Table 5.

Table 5: Relevance Level for Characteristics of Agility and Agile Practices

Characteristic	Relevance Level	Practice	Relevance Level
People-orientation	89,1%	Continuous integration	92.1%
Adaptability	88.7%	Product backlog	82.7%
Being iterative	84.8%	Short releases	82.4%
Being collaborative	84.7%	Refactoring	80.2%
Constant testing	84.1%	Project Visibility	75.8%
Feedback incorporation	84.1%	Planning game	70.7%
Being incremental	83.3%	Simple design	62.9%
Reflexion and introspection	80.8%	Stand-up meetings	57.7%
Being cooperative	79.3%	Coding standards	55.7%
Time-Boxing	63.5%	Collective code ownership	51.7%
Transparency	62.8%	Test driven development	46.2%
Emergence	57.7%	Sustainable pace	42.3%
Self-organization	57.2%	On site customer	37.3%
Leanness	56.5%	Whole teams	34.9%
Modularity	54.0%	Metaphor	34.6%
Small teams	43.2%		

H0 3 was not observed, as the characteristics of agility did not present the same relevance level. The same caveats already made in relation to risks related to not observing H0 1 and H0 2 also apply to H0 3.

H0 4 also was not observed, as the agile practices did not present the same relevance level. But the same risks arise, due to the sample confidence level.

One of the characteristics of agility that had more than 90% for pertinence (being iterative) is considered a commonplace amongst the most known agile methods.

Although in different order, the 10 characteristics of agility/agile practices that had higher pertinence also presented higher relevance levels.

Based on results of this study, 16 characteristics of agility and 15 agile practices, recorded in Table 5 above will proceed in the next steps of this research.

5 Threats to Validity

This study was based on initial sets of characteristics of agility and agile practices identified based on what was found in the technical literature, from the point-of-view of the researchers. This translates into a limitation of the study. One cannot be sure that the same initial sets would be obtained if an attempt had been made to identify characteristics and practices in agile software projects which employ real ideas of agility in software development. This could lead this study to different results if a more 'industrial' view had been used to identify the initial sets of characteristics and practices.

Also, limitations or threats to the validity of this study are associated with the criteria that led to the construction of search strings used in both the systematic reviews that led to the initial set of characteristics of agility and agile practices included in the study. On those opportunities, there was an attempt to avoid a loss of documents relevant to the study. However, due to the choice of search terms, and also because of some difficulties with the search engine in the ACM digital library (as described in Section 2.2), especially during the review of characteristics of agility, there is the risk that relevant studies have been left out of the review.

In the case of the systematic review to identify characteristics of agility, the criteria established in the protocol (deleting retrieved references to documents reporting characteristics of an agile method in particular), though avoiding influences (biases), may have left out of the initial set some characteristic, differently from those identified with the execution protocol planned for this systematic review.

The experience with agile projects was weighted with the weights associated to each participant. However, many projects nowadays that are said or considered to be agile are actually not, which could skew the results. It was not possible to evaluate the profile of the respondents in the study, to exclude those who do not have sufficient practical experience to answer the questions.

It should also be considered that the confidence level obtained for the sample used in this study was of about 80.23%, which represents a risk, even considering that some messages might not have been received (because, for example, of the possibility of emails that are no longer being used by the researchers invited to participate in the study). Thus, the present results can be used, but with due caution.

As for *constructo* validity, this study is characterized by a supposed validity of two initial sets of characteristics of agility and agile practices identified through systematic reviews of the technical literature. One purpose of the study is to confirm the validation of the initial set of characteristics of agility, and enable its evolution, besides getting, at the same time, a suitable set of agile practices related to the adoption of an agile approach to software processes.

Regarding the external validity of this study, participants are considered representative for the population of researchers in agile approaches, as they were identified through systematic reviews of literature, including studies published since 2008. Potential risks or threats to validity arising from the criteria used in systematic reviews are partially reduced by not including, in the survey, authors of articles from which we extracted characteristics and practices, once it could be some argument that the views of such researchers could eventually influence the results, as they themselves have contributed to the identification of characteristics / practices in their papers. The objects used in this study (initial set of characteristics of agility and agile practices) can be considered real and representative to the problem being studied, as these objects have been defined using different scientific papers published in the technical literature on agile approaches. Once he/she started filling out the survey instrument, each participant had the opportunity to answer the questionnaire without a limitation of time.

The instrumentation used in this study was designed to be as simple as possible and require the least amount of time from participants to answer questions. The language of the survey instrument was the English language, for being the most accessible one and considering the various nationalities of the participants invited to join the study.

6 Conclusion and Future Works

From the participants' responses, two characteristics of agility originated by the systematic literature review were excluded from the initial knowledge repository: Convergence and Local Teams. The same happened with two agile practices originated from the systematic literature review: open workspace and pair programming. Sixteen characteristics of agility and fifteen agile practices became part of an initial knowledge repository to support the continuation of studies about agility in software processes. Out of sixteen characteristics of agility which became part of the knowledge repository, only one (small teams) had its relevance level below 50%.

Out of fifteen agile practices which should be part of the knowledge repository, ten had a relevance level over 50%. Among these, Continuous Integration is the practice with the highest level of relevance, at 92%. The practices with the lowest levels of relevance were: whole team and metaphor.

The use of the knowledge repository constructed from the initial results of this study has a confidence level of 80.23%, achieved from the sample and calculated according to the plan described in Section 4.1. The result of the study steers to sixteen characteristics of agility and fifteen agile practices which initially can be part of a knowledge repository to be used in the next phases of this research, aiming at inserting characteristics of agility in software processes:

- 16 Characteristics of Agility: Being collaborative, being cooperative, being incremental, adaptability, self-organization, being iterative, constant testing, emergency, feedback incorporation, leanness, modularity, people orientation, reflection and introspection, small teams, time-boxing, and transparency.

- 15 Agile Practices: Coding standards, collective code ownership, continuous integration, metaphor, on-site customer, planning game, product backlog, project visibility , refactoring, simple design, short releases, daily meetings, and sustainable pace.

The descriptions of these characteristics of agility and agile practices have been consolidated and presented in Sections 2.3 and 3.3. respectively.

Future work in the context of agile software processes may include aspects such as estimation of the degree of agility achieved by methods or software processes, design and execution of experimental studies to detect and/or implement agile practices related to software process activities, which can promote agility for a specific software process in a given development environment.

The studies more closely related to this work are those identified through the systematic reviews conducted in order to identify characteristics of agility and agile practices. Also, there are surveys about agile practices such as those done by [31]. At a glance, what differentiates this work is its contribution, in the agile context, consolidating characteristics of agility and agile practices, both identified and evaluated through studies planned and executed with some level of formalism and rigour, towards the construction of an evidence-based knowledge repository.

This work is part of a research effort in a broader context [76], including other phases which contemplate establishing a relationship between characteristics of agility and agile practices as well as between agile practices and process activities, to support the selection of agile practices to be adopted in software processes. These relationships, after evaluation, along with the information captured by the studies presented here, form a knowledge repository to support an agility framework to guide teams on calibrating their processes according to their needs.

References

[1] B. Bohem, “Making a difference in the software century,” *IEEE Computer*, pp. 78-84, Mar. 2008.

[2] O. Ktata, G. Lévesque, “Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects,” In: *2nd Canadian Conference on Computer Science and Software Engineering*, Montreal, 2009, pp. 59-66.

[3] B. Beizer, *Software Testing Techniques*, Boston: International Thomson Computer Press, 1990

[4] A. M. J. Hass, "Testing processes". In: IEEE International Conference on Software Testing Verification and Validation Workshop ICSTW, 2008.

[5] K. Vlaanderen, S. Jansen, S. Brinkkemper, E. Jaspers, "The agile requirements refinery: Applying SCRUM principles to software product management," *Information and Software Technology*, vol. 53, pp. 58–70, 2011.

[6] T. Dyba, T. Dingsoyr, "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology*, vol. 50, pp. 833-859, 2008.

[7] H. Sharp, H. Robinson, M. Petre, "The role of physical artefacts in agile software development: Two complementary perspectives," *Interacting with Computers*, vol. 21, pp. 108–116, 2009.

[8] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslén, *Experimentation in Software Engineering – An Introduction*, Kluwer Academic Publishers, 2000.

[9] S. N. Mafra, R. F. Barcelos, G.H. Travassos, "Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software", In: *Proc. of the XX Simpósio Brasileiro de Engenharia de Software (SBES)*, Florianópolis, Brasil, 2006.

[10] R. O. Spinola, A. C. Dias-Neto, G. H. Travassos, "Abordagem para Desenvolver Tecnologia de Software com Apoio de Estudos Secundários e Primário," In: *Experimental Software Engineering Latin American Workshop (ESELAW),* Salvador, Nov. 2008.

[11] M. Aoyama, "Agile Software Process and Its Experience," In: *International Conference on Software Engineering*, 1998, pp. 3-12.

[12] M. A. Noor, R. Rabiser, P. Grunbacher, "Agile product line planning: A collaborative approach and a case study". *Journal of Systems and Software* vol. 81, pp. 868--882, 2008.

[13] D. Sato, D. Bassi, M. Bravo, A. Goldman, F. Kon, "Experiences Tracking Agile Projects: an Empirical Study". *Journal of the Brazilian Computer Society*, vl. 12, n.3, Dec. 2006.

[14] B. Boehm, R. Turner, "Observations on balancing discipline and agility". In: *Agile Development Conference*, 2003.

[15] B. Boehm, R. Turner, "Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods", *Institute of Electrical and Electronics Engineers Computer Society*, Piscataway, NJ 08855-1331, United States, vol.26, pp. 718—719, 2004.

[16] K. Conboy, B. Fitzgerald, "Toward a conceptual framework of agile methods: A study of agility in different disciplines". *Association for Computing Machinery*, New York, NY 10036-5701, United States, pp.37—44, 2004.

[17] S. Adolph, "What lessons can the agile community learn from a maverick fighter pilot?", In: *AGILE 2006 Conference*, 2006, pp. 94--99.

[18] K. Conboy, "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development". *Information Systems Development*, vol 20, n. 3, pp. 329-354, 2009.

[19] Agile Manifesto, 2001. http://agilemanifesto.org

[20] M. Fowler, J. Highsmith, "The agile manifesto". *Software Development*, vol. 9, pp. 28–32, 2001.

[21] J. Highsmith, *Agile Software Development Ecosystems*. Boston, MA: Pearson Education, 2002.

[22] M. Pai, M. McCulloch, J.D. Gorman, *et al*., "Systematic Reviews and meta-analyses: An illustrated, step-by-step guide", *The National Medical Journal of India*, vol. 17, n.2, 2004.

[23] G. H. Travassos, P. S. M. Santos, P. Mian, A.C. Dias Neto, J. Biolchini, "An Environment to Support Large Scale Experimentation in Software Engineering"In: *IEEE International Conference on Engineering of Complex Computer Systems*, Belfast, 2008.

[24] Miller, Granville G.: The Characteristics of Agile Software Processes. In: 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (2001).

[25] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, "Agile Software Development Methods. Review and Analysis", Espoo. VTT Publications 478, 2002.

[26] M. Lindvall, V. Basili, *et al*. "Empirical Findings in Agile Methods". In: *Extreme Programming and Agile Methods – SP/Agile Universe*, 2002, pp. 197—207.

[27] Capes BR Portal- *Portal de Periódicos da CAPES*, 2011. http://www.periodicos.capes.gov.br

[28] Abrahamsson, P.; Warsta, J.; Siponen, M.T. & Ronkainen, J.: New directions on agile methods: a comparative analysis. IEEE Computer Society, pp. 244--254 (2003)

[29] Meso, Peter. Jain, Radhika.: Contemporary practices in systems development. Agile Software Development: adaptive systems principles and best practices. Information Systems Management, summer. (2006)

[30] Holmstrom, Helena. Fitzgerald, Brian. et al.: Contemporary practices in systems development. Agile Practices Reduce Distance in Global Software Development. Information Systems Management, summer. (2006)

[31] Ambler, S.W. (2009) "Agile Practices Survey Results: July 2009", available at http://www.ambysoft.com/surveys/practices2009.html, accessed in 2010, Feb. 25.

[32] Coram, Michael. Bohner, Shawn.: The Impact of Agile Methods on Software Project Management. In: 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05). (2005)

[33] Hansson, C. Dittrich, Y. Gustafsson, B. Zarnak, S.: How agile are industrial software development practices?. The Journal of Systems and Software 79, 1295--1311. (2006)

[34] Cockburn, A.: Agile Software Development Joins the 'Would be' Crowd. Cutter IT Journal, 15, 2. (2002)

[35] Taromirad, Masoumeh. Ramsin, Raman.: CEFAM: Comprehensive Evaluation Framework for Agile Methodologies. In: 32nd Annual IEEE Software Engineering Workshop (2009)

[36] Rico, David F.: Effects of Agile Methods on Website Quality for Electronic Commerce. In: 41st Hawaii International Conference on System Sciences (2008)

[37] Qumer, A. Henderson-Sellers, B.: A framework to support the evaluation, adoption and improvement of agile methods in practice. The Journal of Systems and Software, 81, pp.1899--1919 (2008)

[38] Taromirad, Masoumeh. Ramsin, Raman.: An Appraisal of Existing Evaluation Frameworks for Agile Methodologies. In: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems. (2008)

[39] Larman, C.: Agile and iterative development: A manager's guide. Boston, MA: Pearson Education. (2004)

[40] Kujala, S. User involvement: A review of the benefits and challenges. Behaviour and Information Technology, 22, 1, pp. 1--16. (2003)

[41] Guzzo, R. A. Dickson, M. W.: Teams in organizations: Recent research on performance and effectiveness. Annual Review of Psychology, 47, 1, pp. 307--338. (1996)

[42] Institute of Electrical and Electronics Engineers.: IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990). New York, NY (1990)

[43] Steindl, C.: From agile software development to agile businesses. In: 31st EUROMICRO Conference on Software Engineering and Advanced Applications. (2005)

[44] Glazer, Hillel.: Love and Marriage: CMMI and Agile Need Each Other, CrossTalk: The Journal of Defense Software Engineering. Volume 23, No. 1, Jan/Feb. (2010).

[45] Beck, K. Andres, Cynthia.: Extreme Programming Explained: Embrace Change, Second Edition, Addison Wesley Professional (2004).

[46] Jiang, Li. Eberlein, Armin.: An Analysis of the History of Classical Software Development and Agile Development, In: IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA – October (2009).

[47] Abbas, Noura. Gravell, Andrew M. Wills, Gary B.: Using Factor Analysis to Generate Clusters of Agile Practices, In: AGILE Conference, August 9 - 13, Orlando, Florida (2010).

[48] Cohen, D. Lindvall, M. Costa, P.: An Introduction to Agile Methods, in: M.V. Zelkowitz (Ed.), Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam (2004).

[49] Koehnemann, Harry. Coats, Mark. (2009), "Experiences Applying Agile Practices to Large Systems", In: 2009 Agile Conference.

[50] Fruhling, Ann. De Vreede, Gert-Jan. (2006) "Field Experiences with eXtreme Programming: Developing an Emergency Response System", Journal of Management Information Systems / Spring Vol. 22, No. 4. pp. 39-68.

[51] Paige, Richard F. Brooke, Phillip J. (2005) "Agile Formal Method Engineering", J. Romijn, G. Smith, and J. van de Pol (Eds.): IFM 2005, LNCS 3771, pp. 109–128.

[52] Jureczko, Marian. (2008) "The Level of Agility in Testing Process in a Large Scale Financial Software Project", In: Software engineering techniques in progress, Oficyna Wydawnicza Politechniki Wroc?awskiej, 139-152.

[53] Xu, Bin. (2009) "Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects", College. of Computer Science & Information Engineering, Zhejiang Gongshang University, Hangzhou China.

[54] Stolberg, Sean. (2009) "Enabling Agile Testing Through Continuous Integration", 2009 Agile Conference.

[55] Martin, Angela. Biddle, Robert. Noble, James. (2009) "XP Customer Practices: A Grounded Theory", 2009 Agile Conference.

[56] Zhou, Yinghua. (2009) "UniX Process, Merging Unified Process and Extreme Programming to Benefit Software Development Practice", 2009 First International Workshop on Education Technology and Computer Science.

[57] Cannizzo, Fabrizio. Marcionetti, Gabriela. Moser, Paul. (2008) "The Toolbox Of A Successful Software Craftsman", 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.

[58] Huo, Ming. Verner, June. Zhu, Liming. Ali Babar, Muhammad. (2004) "Software Quality and Agile Methods", Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04).

[59] Maurer, Frank. Martel, Sebastien. (2002) "Extreme Programming Rapid Development for Web-Based Applications", IEEE Internet Computing, January - February.

[60] Paulk, Mark C. (2001) "Extreme Programming from a CMM Perspective", IEEE Software, November/December.

[61] Vejandla, Pavan K. Sherrell, Linda B. (2009) "Why an AI Research Team Adopted XP Practices", Proceedings of the 47th Annual Southeast Regional Conference, ACM-SE 47, March 19-21, Clemson, SC, USA.

[62] Concas, Giulio. Francesco, Marco Di. Marchesi, Michele. Quaresima, Roberta. Pinna, Sandro. (2008) "Study of the Evolution of an Agile Project Featuring a Web Application Using Software Metrics", A. Jedlitschka and O. Salo (Eds.): PROFES 2008, LNCS 5089, pp. 386-399

[63] Hazzan, Orit. Tomayko, Jim. (2003) "The Reflective Practitioner Perspective in eXtreme Programming", Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in

Bioinformatics), 2753, 51-61.

[64] Xiaohua?Wang. Zhi, Wu. Ming, Zhao. (2008) "The Relationship between Developers and Customers in Agile Methodology", 2008 IEEE International Conference on Global Software Engineering.

[65] Aiello, G. Alessi, M. Bruccoleri, M. D'Onofrio C. Vella, G. (2007) "An Agile methodology for Manufacturing Control Systems development", Engisud S.p.a., Palermo, Italy.

[66] Svensson, Harald. H¨ost, Martin. (2005) "Introducing an Agile Process in a Software Maintenance and Evolution Organization", Proceedings of the Ninth European Conference on Software Maintenance and Reengineering (CSMR'05).

[67] Mills, David. Sherrell, Linda. Boydstun, Jeff. Wei, Guoqing. (2005) "Experiences Using Agile Software Development for a Marketing Simulation", Dept. of Comput. Sci., Memphis Univ., TN USA.

[68] McKinney, Dawn. Denton, Leo F. (2005) "Affective Assessment of Team Skills in Agile CS1 Labs: The Good, the Bad, and the Ugly", SIGCSE '05, February 23-27, 2005, St. Louis, Missouri, USA.

[69] Ramachandran, Vinay. Shukla, Anuja. (2002) "Circle of Life, Spiral of Death: Are XP Teams Following the Essential Practices?", Dept. of Comput. Sci., North Carolina State Univ., Raleigh NC USA

[70] Williams, Laurie. Upchurch, Richard. (2001) "Extreme Programming for Software Engineering Education?", 31st ASEE/IEEE Frontiers in Education Conference, October 10 - 13, 2001 Reno, NV.

[71] J. F. Abrantes, G. H. Travassos, “Common Agile Practices in Software Processes”. In: 5th International Symposium on Empirical Software Engineering and Measurement, Banff, Alberta, Canada, Sept. 2011, pp.355-358.

[72] V. R. Basili, C. Caldiera, H. D. Rombach, “Goal/Question/Metric Paradigm”, *Encyclopaedia of Software Engineering*, vol. 1, pp. 528-532. John Wiley & Sons, New York, 1994.

[73] L. D. Farias, “Planejamento de Riscos em Ambientes de Desenvolvimento de Software Orientados à Organização”, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, 2002.

[74] J. F. Abrantes, G. H. Travassos, “Caracterização de Métodos Ágeis de Desenvolvimento de Software”. In: *Primeiro Workshop de Desenvolvimento Rápido de Aplicações – VI Simpósio Brasileiro de Qualidade de Software*, Porto de Galinhas-PE, Brasil, 2007.

[75] M. Hamburg, “Basic Statistics: A Modern Approach”, *Journal of the Royal Statistical Society, Series A* (General), vol. 143, n.1, 1980.

[76] J. F. Abrantes, “Estudos Experimentais sobre Agilidade no Desenvolvimento de Software e sua Utilização no Processo de Teste”, Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2012.