<table>
<thead>
<tr>
<th><strong>Manuscript Number:</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full Title:</strong></td>
<td>PBURC: A Patterns-Based Unsupervised Requirements Clustering Framework for Distributed Agile Software Development</td>
</tr>
<tr>
<td><strong>Article Type:</strong></td>
<td>Original Research</td>
</tr>
<tr>
<td><strong>Keywords:</strong></td>
<td>requirements engineering; distributed software development; agile software process; requirements clustering</td>
</tr>
</tbody>
</table>
| **Corresponding Author:** | Petros Belsis  
TEI of Athens  
Aigaleo, Athens GREECE |
| **Corresponding Author's Institution:** | TEI of Athens |
| **First Author:**       | Petros Belsis |
| **Order of Authors:**   | Petros Belsis  
Anastasios Koutoumanos, Phd  
Cleo Sgouropoulou, Phd |
| **Order of Authors Secondary Information:** |
PBURC : A Patterns-Based Unsupervised Requirements Clustering Framework for Distributed Agile Software Development

Authors:

Petros Belsis*, Anastasios Koutoumanos, Cleo Sgouropoulou

Department of Informatics, Technological Education Institution of Athens, Agiou Spyridonos, Aigaleo, 12210 Greece

Emails: pbelsis@cs.teiath.gr, tkout@cs.ntua.gr, csgouro@teiath.gr

*Corresponding author:
Tel: 0030-2105910974
Fax: 0030-2105910975
**PBURC: A Patterns-Based Unsupervised Requirements Clustering Framework for Distributed Agile Software Development**

**Abstract**—Agile software development methodologies are increasingly adopted by organizations because they focus on the client’s needs, thus safeguarding business value for the final product. At the same time, as the economy and society move towards globalization, more organizations shift to distributed development of software projects. From this perspective, while adopting agile techniques seems beneficial, there are still a number of challenges that need to be addressed, such as the effective cooperation between the stakeholders and the geographically distributed development team. In addition, data collection and validation for requirements engineering demands efficient-processing techniques, in order to handle the volume of data as well as to manage different inconsistencies when the data are collected using online tools. In this paper we present “PBURC”, a patterns-based, unsupervised requirements clustering framework, which makes use of machine learning methods for requirements validation, being able to overcome data inconsistencies and effectively determine appropriate requirements clusters for optimal definition of software development sprints.

**Keywords**—requirements engineering; distributed software development; agile software process; requirements clustering;

**Introduction**

The requirements engineering process can be broken down into a structured set of activities which are carried out in order to derive, validate, and maintain system requirements [1]. The basic stages in the requirements engineering process include requirements elicitation, requirements analysis and requirements validation. It is essential to point out that there is a variation to the requirements engineering processes that would fit an organization, since each organization has different needs and preferences [2]. Requirements analysis is critical to the success of a development project. In particular, requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a product, taking account of the possibly conflicting requirements of the various user groups. Requirements must be documented, actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. In projects with many parties and many stakeholders involved, requirements elicitation and analysis face several challenges. These
challenges are even more apparent in multi-national projects where a large number of partners are involved and where the target groups of users and developers are characterized by diverse backgrounds and skills.

Lately, among a variety of available methodologies, agile methodologies have become very popular because they pay special attention to the client’s needs. A key characteristic of any agile approach is its explicit focus on creating business value for the clients [3]. Essentially, in agile software projects the development process is a value creation process that relies on active client participation. The value creation is ensured both through the final product as well as through the process itself [4]. Moreover, agile methodologies seem to be most effective in respect to keeping low the development times, as well as in respect to the harmonization of project goals with the project deliverables.

Albeit initially questioned for their potential drawbacks when applied in distributed development environments [5], agile techniques hold a lot of potential benefits for large-scale software development projects, being able to effectively focus on the estimation of the project’s progress. However, compared to agile methodologies for projects carried out on one site there are a number of challenges that need to be addressed before their efficient implementation. Among else, more documentation, coordination and management is necessary. The first iterations are also of vital importance since they lead the development to the following stages [6]. Within these stages, the requirements should be clearly understood and validated prior to proceeding. In addition, the role of technology is vital, as distance barriers, cultural and time differences are obstacles in respect to one site agile projects. Within this framework, we have utilized online tools for data gathering and accordingly we propose the use of machine learning in order to identify the most useful requirements from the user’s perspective; for example the requirements elicitation can be more effective and helpful for the developer if the most important requirements are classified in clusters, which represent specific user groups. In order to perform this process unbiased, effectively and without great effort, we propose “PBURC”, a patterns-based, unsupervised requirements clustering framework, which makes use of machine learning methods for requirements validation, processing and classification, being able to overcome data inconsistencies and effectively determine appropriate requirements clusters for optimal definition of software development sprints.

The unique characteristics of the software requirements process, make the relevant data sets high dimensional, sparse, and often resulting from ambiguous expressions of need; thus they present difficult challenges for standard processing techniques [7]. Previous attempts to utilize machine learning or probabilistic techniques for requirements engineering focused on creating automatically requirements from different datasets; in these attempts it was proved that this approach does not produce high quality results to support the specific software engineering tasks [7]. However, we believe that the process should be directed bottom up instead of vice versa; we have identified different potential requirements using several semi-structured techniques, such as interviews with different stakeholders, discussion workshops, and using also various software tools. Accordingly using online questionnaires we have gathered the user’s opinion from a larger set of requirements. These data are more straightforward to process, however the large volume of
data requires intelligent processing and support from efficient data management techniques as well as specialized software tools.

The PBURC framework has been applied in a large scale software project, involving partners from eleven European countries, aiming to provide social services to communities of researchers and academics, along with mediated access to a federation of repositories with an open access policy. We, therefore, reflect on the effectiveness of the PBURC framework on a real-world application, where stakeholders’ participation has been the driving force of the software engineering process, ensuring that stakeholders actively contribute to requirements validation and prioritization. In effect, our approach with respect to the data-processing phase of the software requirements elicitation process, introduced PBURC as an unsupervised clustering framework for the identification of users’ preferences. An additional benefit of active user participation in the early stages of software development is that it guarantees higher possibilities of user satisfaction. To this end, we had to tackle a number of challenges, which typical clustering based approaches fail to address, such as handling of missing data and identifying the optimal number of clusters.

The paper is structured as follows: After this introduction, Section 2 provides a short review of related work, followed by a description of our research methodology, in Section 3, where the PBURC framework is detailed. Section 4 provides a reflection of the adaption of the PBURC framework for distributed software development of a large-scale international software project, while Section 5 provides a short discussion on the current outcomes of our work and an indication for future work which concludes the paper.

2. Related Work

Agile methodologies aim to maximize business value of the end-product by ensuring that stakeholders join early the software engineering process, during the requirements elicitation phase. Stakeholders aim to prioritize requirements based on their beliefs and perception about factors that are more relevant to them, such as importance, cost, time availability etc. Additionally, next to business value, stakeholders consider the associated risks and failure estimates that could potentially hinder fulfillment of their needs [8]. In non-agile approaches, for most of the requirements only the cost and value are considered; in the agile process there is also a risk estimation step, balancing effort with cost, which is brought to the client’s consideration while estimating the relevant priority of each requirement [9]. In summary, the identified requirements are prioritized based on cost benefit analysis and the expected gains from their implementation.

While prioritization of requirements has been of primary focus in agile literature [10], [11] and theoretical and methodological support has emerged, there is still a need to put these questions in a practical multi-party context. There is also a need to support the process with different software tools that enable distributed management and participation from groups of stakeholders with different backgrounds and skills.

Requirements management also demands efficient methods especially in respect to the data processing related tasks. User involvement results in a collection of large volume of data which
have to be processed in a manner that outcomes closely represent the needs of each stakeholder group. Hsia et al [12] proposed the idea of decomposing requirements into a certain number of useful, usable, and semi-independent partitions that would facilitate incremental development.

They describe an approach in which requirements are segmented through the application of a series of proximity thresholds. Yaung [13] in a similar context applies hierarchical clustering to explore the analogy between design modularity and requirements modularity, without however providing experimental evaluation or other kind of validation. Chen et al [14] apply requirements clustering to automatically construct models for software products analysis. They describe a graph based approach where edge weights are based on proximities and make use of an iterative graph-splitting technique to cluster requirements. The individual cluster quality is evaluated using an independency metric, being a graph theoretical metric that computes the ratio of the sum of outer edge weights over the sum of inner edges weights. Duan et al [7] describe the application of hierarchical clustering and probabilistic techniques to provide support for automated software requirements extraction. The experimental data that they used were large document collections and texts extracted from online user forums.

The PBURC framework as described in the following section is based on a user-directed approach in which the needs are initially documented using well established techniques, with major involvement by the end-users, while the requirements are not extracted randomly from large document collections, but are rather documented and accordingly prioritized using techniques based on machine-learning algorithms. We also describe a method that identifies the optimal number of clusters for requirements processing and prioritization. This last step is crucial when the software development team utilizes an agile software process. In essence, the prioritized clusters of user requirements can regularly guide the target for each of the software development sprints.

3. The PBURC framework

The next paragraphs describe the PBURC framework, which constitutes our proposal towards a methodological approach for requirements engineering especially suited for agile processes of distributed software development. This framework has evolved and has been validated by means of its application to an ongoing international project, involving partners from eleven European countries with diverse specialties, skills and background.

The main challenge that the PBURC framework aspires to address is that of the effective usage of agile processes for distributed software development. Indeed, distributed software development and agile approaches have been considered in the past as not compatible [5]; today, as more organizations become geographically dispersed, due to the globalization, cost reduction and other reasons, distributed approaches are continuously becoming common-place. At the same time, as organizations shift to agile implementations [15], [16], [17] there is a growing need to identify effective methodologies and tools that will enable the shift to both a distributed and agile paradigm. It is essential to note that the PBURC framework is focused on the requirements
engineering phase of the software development process. In effect it describes a mechanism which integrates machine-learning based algorithms for a Patterns-Based, Unsupervised Requirements Clustering (thus the PBURC acronym), for distributed agile software development. In short, PBURC proposes an intuitive mixture of established methods and activities, along with innovative usage of clustering algorithms, for prioritizing requirements, grouping them and driving the agile software process:

1) The first set of activities deal with the requirements identification, elicitation and gathering, more in the sense of an unstructured list of features, wishes and ideas, forming an extensive set of items that need to be further investigated, prioritized and grouped to form a proper specification of software requirements.

2) The next activities involve encoding of the collected feedback into raw data that can be processed by subsequent machine-based algorithms.

3) The third step employs data management techniques in order to achieve data imputation and handling of incomplete data.

4) The next step, being at the core of the PBURC framework, deals with the determination of the optimal number of clusters.

5) The fifth step, which is also machine-based, continues the workflow with the identification of clusters (groups) of requirements that serve as a prioritized list of features to be addressed by software development sprints of distributed agile teams.

6) The last stage of the workflow involves the verification of the outcomes, mainly based on human intuition and reflection, supported by tools and methods for distributed collaboration.

This workflow, presented in Figure 1 as a linear set of processes, should be considered as part of a repetitive, agile methodology towards effective software development.
The next paragraphs provide further details on each step of this workflow.

### 3.1 Requirements identification and elicitation

Requirements elicitation activities require several distinctive tasks, such as the organization of interviews, surveys, coordination of stakeholder workshops [18]. However, for these tasks to be effective, all qualified stakeholders need to be identified and engaged in the process, so that the project progress is maximized through the interaction of stakeholders with the software development team. In traditional requirements engineering several difficulties emerge, especially when it comes to manage distributed large-scale projects which affect a lot of different stakeholders with different backgrounds, skills, preferences and experiences. Very often such projects have a broader scope and complexity while their implementation remain subject to common constraints such as cost, time, and predefined outcomes in terms of cost-benefit estimation. To manage the increasing needs for such situations, it is essential for the implementation team to make use of flexible methodological and software tools for requirements elicitation, modeling and verification. In fact, many organizations are adopting wikis and other online collaborative tools [19] to facilitate the requirements elicitation process through capturing the initial needs of the stakeholders.
The PBURC framework identifies the importance of stakeholder’s participation from the early stages of the software process. In order to capture their perception of the desired functions, it is considered essential to promptly involve all interested parties in the requirements engineering process. In fact, our approach includes the following stages:

- Initially a number of key features are identified, after careful examination of relevant systems and existing experience.
- In addition, during workshops in which all members of the distributed developer teams participate, several value-adding features and services may be recorded. The material gathered by these different feedback-collection activities is then compiled and serve as a basis for the creation of a more sophisticated questionnaire-based survey.
- The next step involves the arrangement of several workshops, in several locations, bringing together different potential stakeholder groups, aiming to validate and prioritize identified requirements.

Inevitably, an interesting challenge is that during these workshops new ideas often emerge and, moreover, this challenge is coupled with the usual issue of requirements-conflicts management.

The issue of conflict emerged in the requirements engineering domain as the interest shifted from software and hardware alone to socio-technical systems. It was early discovered in the requirements engineering domain that systems must account various (im)precise and (in)consistent expectations of the stakeholders, including future users, owners and so on [8]. An understanding of the functions of the system-to-be could only be sought after beliefs, desires, and intentions of stakeholders had been grasped to a feasible extent. The management of various types of conflicts was identified as an essential one in our approach.

During the workshops often the opinions were different and many requirements that were examined for possible implementation were often conflicting. The developer’s opinion on the one hand is essential since it has a focus on both the functional and non functional requirements; on the other hand agile methodologies tend to put focus on the user’s side. Thus it is essential to identify effective techniques to solve the conflicts by invoking the users in a way that the real needs would emerge and ambiguity would be clarified. For this reason a number of such techniques are suggested by the PBURC framework, including dotmocracy and use of online tools that allow all participants to express their opinion and feedback for the prioritization of identified requirements.

- Dotmocracy is a decision making process that separates the participants of a session to two groups: the facilitators which are trained individuals who manage the process and remain neutral on the content and to the members which are the people who express their agreement on a scale, represented as a dot on a specially designed sheet of paper. These sheets are designed to compliment a consensus decision-making process by providing a simple way to visibly document levels of agreement among participants on a large variety of ideas.
- A similar approach is that of brainstorming, where the stakeholders use an iterative and cooperative approach to capture user requirements in a mind map representation [20].
• In addition to participating to such consensus decision process, online questionnaires allow stakeholders to grade well formulated requirements. The difference between online questionnaires and consensus based approaches such as the one described is that the questionnaires are very well structured and aim in formulating the stakeholders opinion on a well clarified requirement, while, for example, dotmocracy allows a less structured recording of tendencies rather than clearly formulated opinions.

• The list of essential requirements can be further elaborated with feedback from face to face interviews with representatives from different stakeholder groups.

### 3.2 Requirements processing

After the completion of the requirements identification and elicitation activities, it’s time for the developers’ team to work in groups in order to decode the main findings and integrate them in the elicitation documents. At first the fundamental requirements should be properly identified and documented, as those for which there was almost consensus between the different stakeholders.

Additionally, a review and technical comparison with other similar or relevant systems can provide insight for some further requirements that should be added to the existing list.

There are different approaches when there is a need to create clusters from research data or data collected by user questionnaires. Machine learning algorithms which are used most commonly for this purpose can be either supervised or unsupervised. In supervised algorithms, classes need to be predetermined.

Typically this involves a human expert who is arranged to label a certain amount of data with specific classifications. Unsupervised algorithms allow the determination of clusters without previous guidance provided by experts. Several research approaches have focused on applying semi-supervised techniques in order to guide the clustering process through expert feedback.

The final purpose is to classify data as similar, using a mathematical framework which is evaluated based on its ability to predict the membership of a certain subgroup of data to a certain cluster, using the value of a specific measure of variance. Recent research has investigated the use of semi-supervised clustering techniques for automated software requirements extraction from texts [7], which is guided by prior knowledge of constraints collected through human expertise in the form of constraints.

These constraints may be considered as cluster level or instance level where cluster level constraints specify relationships between a pair of elements. Most popular are the instance level constraints, which can be further categorized in two types, namely must-link ML and Cannot-link CL constraints indicating whether different instances must be classified in the same or in different clusters. When constraints are gathered directly from users there is a high probability of inconsistencies, since different constraints can be treated as ML while some of these constraints may be treated as CL. There is a high cost especially when we are talking about complex projects with a large set of requirements and therefore high probability of such inconsistencies.

Exactly for addressing such issues, the PBURC framework adopts a different approach proposing a methodology ranging from the formation of process-able data to the final identification of requirements clusters (Figure 2).
Initially, after identifying a large set of requirements with the assistance of stakeholders, using interviews, group sessions and online tools, we carry along to validate them and identify the most important ones using online questionnaires where the system’s users are accordingly asked to grade the importance of each requirement. Thus, we create a n*m matrix consisting of the n different users and the m different grades for each of the m requirements. Therefore, we receive n*m vectors of data that need to be classified. However, data are not ready for clustering, since some contain empty spaces and clustering algorithms are not compatible with null values. This makes necessary to identify a way to process data that will make the data compatible with clustering algorithm’s requirements as well as not to alter the quality of data either by ignoring some of the data or by filling them. We have determined a methodology that makes data gathered from the user’s participation in the requirements gathering phase available for clustering; our approach focuses not only on the preparation phase, but also on identifying the optimal number of clusters k prior to utilizing appropriate clustering algorithms.

3.3 Handling incomplete data

One major problem in applying the aforementioned algorithms has to do when missing data are present; in our case the feedback we received contained such data especially when some of the questions were not answered. The problem of handling with missing or incomplete data is common in various fields of scientific research. When data are collected using questionnaires, it is often that some of the respondents chose not to answer specific questions. The missing data constitute a problem for researchers who cannot analyze the data due to the missing data sets. Different approaches can be followed in order to handle such situations. One approach is to disregard missing data with most common techniques applied list-wise deletion and pair-wise deletion. Another approach is to try to approximate the values for the missing data; a very common approach is based on the calculation of maximum likelihood. If certain restrictions are satisfied, maximum likelihood produces values for the missing values which are consistent with the rest of the data, have minimal standard errors.

In order to calculate maximum likelihood, we need a likelihood function that produces probability of data as a function of the missing parameters. Considering we have two discrete variables X and Y with a joint probability function denoted by p(x, y| v) where v is a vector of parameters. Then p(x, y| v) gives the probability that X=x and Y=y. In case there are no missing data and all the observations are independent the likelihood function is given by
L(v) = \prod_{i=1}^{N} p(x_i, y_i | v). In order to calculate maximum likelihood estimates this function needs to be maximized. Considering that the data are not following some pattern but are missing at random, we consider that r values are missing from Y and the next s values are missing from X variable. If 

\[ g(X|v) = \sum_{y} p(x, y | v) \] is the marginal distribution of X (sum over Y) and

\[ i(y|v) = \sum_{x} p(x, y | v) \] the marginal distribution of Y (sum over X), then the maximum likelihood can be calculated by the formula: 

\[ L(v) = \prod_{i=1}^{N} g(x_i | y_i) \prod_{i=s+1}^{r+s} i(y_i | y) \prod_{i=s+1}^{N} p(x_i, y_i | y). \] This is typically the calculation of the products of the likelihood function for all the different variables.

This formula can be extended to include more variables. To implement maximum likelihood one common approach is to use the expectation-maximization (EM) algorithm.

We consider a vector v which needs to be estimated and which consists of the unknown parameters; within the EM framework the vector v can be estimated by means of the complete-data log likelihood (of both the observed and missing data). We let \( g_c(x; \Psi) \) denote the probability distribution function of the random vector \( \Psi \) corresponding to the complete vector x. Then the complete-data log likelihood function for \( \Psi \) if x was fully observable will be given by 

\[ \log L_c(\Psi) = \log g_c(x; \Psi). \]

There are two steps for each iteration of the EM algorithm, called the expectation (E) step and the maximization (M) step:

- The E-step, which involves the computation of the so-called \( Q \)-function, which is given by the conditional expectation of the complete-data log likelihood given the observed data and the current estimates.

- The M-step, which updates the estimates that maximizes the \( Q \)-function over the parameter space.

On the \((k + 1)\)th iteration of the algorithm, the E-step computes the \( Q \)-function:

\[ Q(\Psi; \Psi^{(k)}) = E_{\Psi^{(k)}} \{ \log L_c(\Psi) | y \} \] with \( E_{\Psi^{(k)}} \) denoting the expectation operator using the current value \( \Psi^{(k)} \) for \( \Psi \). The M-step of the algorithm updates \( \Psi^{(k)} \) taking \( \Psi^{(k+1)} \) to be the value of \( \Psi \) that maximizes \( Q \) over all admissible values of \( \Psi \); thus, in the applied version of the EM algorithm the complete log-likelihood function is approximated by replacing the random vector \( \Psi \) by its conditional expectation.

### 3.4 Determining the optimal number of clusters

In order to determine the most desirable user features, we need to classify the vectors created by the user’s questionnaires. One very common approach is to use the k-means clustering algorithm [21]. K-means is a clustering technique which partitions n observations into k clusters with each observation assigned to the cluster with the nearest mean. However, there is no guidance
provided by the algorithm as to the optimal value for the number of clusters. One technique that can be applied towards this direction is Ward’s algorithm. Ward’s method considers the distance between two clusters A and B by calculating the sum of squares when we merge them. The merging cost \( D \) of the two clusters is given by

\[
D(A,B) = \sum_{i \in A \cup B} \left\| x_i - \overrightarrow{m_{A \cup B}} \right\|^2 - \sum_{i \in A} \left\| x_i - \overrightarrow{m_A} \right\|^2 - \sum_{i \in B} \left\| x_i - \overrightarrow{m_B} \right\|^2
\]

\[
= \frac{n_A n_B}{n_A + n_B} \left\| \overrightarrow{m_A} - \overrightarrow{m_B} \right\|^2,
\]

where \( \overrightarrow{m_j} \) is the center of cluster \( j \), \( n_j \) is the number of points in it.

Ward’s method is both greedy, and constrained by previous choices as to which clusters to form. Therefore, the sum-of-squares for a given number \( k \) of clusters is usually larger than the minimum for that \( k \), and even larger than what \( k \)-means will achieve. But we can use this algorithm as a tool to identify the optimal number of \( k \), by determining the value of \( k \) where the merging cost will start to rise significantly. In our case when the step takes the value 98 the cost rises significantly. So \( k=100-97=3 \) is an expected suitable value for the number of clusters.

Fig. 3. Determination of optimal number of clusters by calculating the distance coefficient in relation to the merging cost.

### 3.5 Identifying clusters using the k-means algorithm

The next step is to classify the existing vectors as gathered during the requirements phase into clusters, using appropriate algorithms that minimize distance metrics. One of the most popular algorithms is the \( k \)-means algorithm. In general, the algorithm operates as follows: Given a set of observations \( (x_1, x_2, ..., x_n) \), where each observation is a \( d \)-dimensional real vector, \( k \)-means clustering aims to partition the \( n \) observations into \( k \) sets \( (k \leq n) \) \( S = \{ S_1, S_2, ..., S_k \} \) so as to minimize the within-cluster sum of squares (WCSS): initially the algorithm needs to define \( k \) centroids, one for each cluster. Then each point is associated with the nearest centroid. When there
is no unassigned point left, a first grouping is completed. The next step is to recalculate k new
centroids using the clusters from the previous iteration. The whole process is repeated until the
centroids are not moving; the aim of the algorithm is to minimize an objective function, in this
case a squared error function. The objective function \( J = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_{ij} - c_j \|^2 \) is used as a
measure of the distance of the n points from their cluster centers.

In order to verify the results we measure the variance which shows the total within-cluster
distance. Minimal values for this criterion mean that the grouping of points in a cluster ensures
minimal within-cluster distance. The parameter used to calculate the variance is the squared
Euclidean distance between points:

\[
d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2.
\]

Fig. 4 presents the mean variance between clusters for a number of clusters equal to k=2, 3, 4
and 5 presenting average values for within cluster variance equal to 1.05 exp(-3), 1.0 exp(-3),
1.175 exp(-3) and 1.36 exp(-3) respectively. This proves that our initial selection choice for k=3
complies with the minimum variance criterion.

![mean variance](image)

Fig. 4 Mean variance values for k=2,3,4 and 5.

### 3.6 The verification process - tools and methods

Accordingly, after being elicited from the previous phase, the key requirements are recorded in
use case diagrams. In order to be able to verify the requirements, a list of use cases are created
describing the different system functionality. A number of indicative mockups also need to be
created to further facilitate the verification process.
In order to overcome distance barriers, tools for web-based developments are used; in our application case we used a web-based software requirements tracking tool through which all developers could collaborate. In addition, a wiki was set up to facilitate the communication between the stakeholders and the developers. The software engineers frequently commented on the wiki in order to resolve conflicts and eliminate discrepancies.

**Coordination**

One of the big challenges in a distributed project is to ensure coordination between participants. The role of the project coordinator is essential at this point: the coordinator is reminding to the other team members their responsibilities, the time schedules and deadlines and helps conflicts to be resolved very fast. In order to achieve this, during the various phases of requirement engineering process, we used mind-maps and quality procedures, such as workflow diagrams and other established cost and time-scheduling monitoring tools.

**Communication and knowledge sharing**

Another big challenge in distributed agile projects is to circulate fast all the outcomes of a specific stage, minimizing at the same time the overhead to participants [22]. The role of knowledge sharing activities and tools has long been accredited to leading software industries, using knowledge management tools [23]. In order to efficiently leverage and circulate knowledge between partners, evolving requirements were informally recorded using online requirements gathering and changes tracking tools, while team members were continuously sharing opinions and exchange views using forum and online discussion tools. All these together have boosted cooperation and knowledge exchange among the distributed groups.

4. **PBURC FRAMEWORK VALIDATION**

In this section we describe the usage of the PBURC framework for a real case study carried out for the needs of a distributed agile project. The main focus is to describe the key points of the applied algorithms. Before we carry on with this description, we need to mention that an agile approach was selected, due to the emphasis it pays on communication with the client in contrast to the focus on documentation that characterizes non-agile approaches.

The system under development is part of the VOA3R project (Virtual Open Access Agriculture & Aquaculture Repository), which aims at re-using existing and mature metadata and semantics technology to deploy an advanced, community-focused integrated portal for the retrieval of relevant open content and data. To this end a new platform will be developed that will facilitate explicit models of scholarly research methods and procedures while at the same time will promote the practical procedures targeted by applied research, suitable for the needs of practitioners and other information-seekers, consumers of research output. Towards these goals, the portal will enable researchers to formulate their information needs in terms of elements of the scientific methods established in their field (variables, techniques, assessment methods, kinds of objects of interest, etc.) combined with topical descriptions as expressed in metadata. The community
approach will enable the enhancement of information seeking with extended criteria (i.e. ratings, public reviews, social tagging and links to supporting or conflicting reports) that complement and go beyond the traditional, anonymous peer review process which results are not made available openly. The technology used will itself become open source, so that the model of the service can be adopted by enterprises (including small medium enterprises – SME’s) or other kinds of institutions as a value-added, community-oriented model for open access content.

In addition to the aforementioned challenges for a distributed agile project, we also had to cooperate with a large number of stakeholders with different needs and background; thus, in order to increase the possibilities of creating an acceptable product, the development team was urged to involve the stakeholders from the early phases; this decision would enable the formulation of requirements tailored to the user’s needs, so as to make the system valuable for future and continuous use. One major obstacle was that different tasks were divided between groups with diverse skills, backgrounds and preferences. In addition, the locations of the partners which spanned across different countries left no choice except distributed implementation. Thus, a proper selection of development methods and tools, facilitating the distributed nature of the project’s team while effectively following the agile paradigm, was necessary.

The tools selected, featuring a mostly social dimension, include traditional interviews with stakeholders, whiteboard mapping of primary requirements, workshops organization. These tools required the presence of a number of people at the same time and discussion and interaction with the developers. In addition, the requirements elicitation process was facilitated using online questionnaires while discussions about important issues raised during the process were also supported by asynchronous tools such as discussion forums and online discussion software tools. The use of a wiki based tool was also very important for the support and evolution of requirements as well as for the coordination of the process. From the developer’s perspective, the requirements were also recorded using a tool that allowed distributed collaboration between the different members of the team and also allowed the monitoring of recent changes in the recorded requirements.

In summary, the main requirements identification and elicitation activities included the arrangement of 8 different workshops organized in different countries each, and more than 20 interviews with people involved in different organizations or universities. The main groups engaged through these events were:

- Academics in fields related to the project
- Researchers in fields related to the project
- Open Access / Metadata / Semantics Experts
- Key authorities (advisor, consultant, etc.)
- Practitioners
- Students (undergraduate and postgraduate)
- Academic and vocational training providers
- Library personnel
In terms of numbers, the activities included different people from the aforementioned disciplines, working in different organizations with different background and skills. Table 1 summarizes the aggregated profiles of the participants.

<table>
<thead>
<tr>
<th>Group</th>
<th>Number of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Researchers</td>
<td>65</td>
</tr>
<tr>
<td>Practitioners</td>
<td>4</td>
</tr>
<tr>
<td>Students</td>
<td>45</td>
</tr>
<tr>
<td>IT experts</td>
<td>7</td>
</tr>
<tr>
<td>Professors</td>
<td>17</td>
</tr>
</tbody>
</table>

The described process facilitated a first draft recording of the initial requirements. The next important step was requirements prioritization and elicitation. In order to extract the most important requirements input from different stakeholders was collected using online questionnaires, which allowed the grading of each requirement.

From this process the output of prioritization was given as a large number of multi-dimensional vectors, which required efficient processing methods. In order also to verify patterns in user’s preferences a clustering approach was selected. One of the main problems related to vector processing was the fact that some of the data were missing. The followed approach allowed us to overcome this issue and to proceed by applying reliable clustering techniques. The next phases of software requirements made use of various tools for online collaboration between the team members. Use case diagrams and web based tools that allowed tracking of changes, knowledge management techniques which made use of both explicit as well as tacit knowledge were used; for example, a knowledge base facilitated problem solving, while online collaboration tools allowed cooperation between expert and junior developers. The integration of quality procedures as workflow diagrams and cost and time-scheduling monitoring tools ensured a positive outcome of the overall process.

5. CONCLUSION

As more and more organizations shift to agility [15], [16], [17], it is inevitable to try to adjust the distributed implementation process to agile methodologies. A number of challenges are necessary to be tackled in order to make such a shift feasible; among else, the most notable are establishment of trust between the stakeholders, coordination, proper use of knowledge sharing activities and tools and efficient communication between the participants, as well as intelligent processing techniques to efficiently manipulate the input from the stakeholder’s participation in the requirements gathering process. These challenges can be managed as long as there is organizational commitment and willingness to overcome the long-distance obstacle and appropriate support using technology means. Mutual trust can be established between project partners as long as there are organizational processes that are followed but not strict enough to limit the flexibility of partners; in respect to the trust between developers and users, this can be achieved at the meetings and workshops and by using transparent techniques that record the user’s
opinions and also facilitate consensus between contradictory opinions; to this end, online grading tools for already formulated requirements or dotmocracy for evolving ideas has been proved as a helpful solution in our case.

In order to efficiently process the data in a timely manner, we have proposed a machine learning approach; this approach handles several issues in respect to the nature of the problem and the available data: first the often missing data problem was missing with an imputation method. Second, in order to identify the most important requirements for different categories of users, a clustering based approach was applied. The number of clusters however cannot be known in our case in advance. In order to estimate the number of clusters, Wards algorithm was applied. After determining the number of clusters, the k-means algorithm helped us identify the key requirements for the 3 identified user-categories. These requirements were recorded by means of use case diagrams in software modeling tools that allowed also distributed collaboration to support the needs of the software development team.

Communication when the members of the development tools are distant can be facilitated by synchronous communication tools when necessary and provided that time-zone limitations enable such a solution. In any case asynchronous communication using mailing lists or forums has also proved very beneficial especially for minor issues that do not demand the concurrent presence of all the members. Knowledge sharing activities are very strongly supportive in successful requirements engineering processes, as participants in the process may leverage of other members experience using online repositories, wikis and other similar tools. The input from stakeholders demands also an efficient processing technique that utilizes advanced categorization algorithms. At the final stage, recording the requirements with a tool that allows tracking changes is also helpful. We need though to remark that the methodology or the tools that will finally be used to record the requirements come to second priority since the nature of agile processes gives priority to the people rather than the process itself.

We have so far applied our methodology to a large international project; we seek to expand its applicability to more development scenarios. We are also planning to experiment with other clustering algorithms that are able to treat data inconsistencies using fuzzy techniques.

ACKNOWLEDGEMENTS

The research presented in this paper has been carried out within the VOA3R project (Virtual Open Access Agriculture & Aquaculture Repository – http://voa3r.eu), which has received funding from the European Commission’s ICT Policy Support Programme (ICT PSP), Theme 4 - Open access to scientific information.

REFERENCES


mean variance

- $k=2$
- $k=3$
- $k=4$
- $k=5$
TABLE I. Summary of participant groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Number of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Researchers</td>
<td>65</td>
</tr>
<tr>
<td>Practitioners</td>
<td>4</td>
</tr>
<tr>
<td>Students</td>
<td>45</td>
</tr>
<tr>
<td>IT experts</td>
<td>7</td>
</tr>
<tr>
<td>Professors</td>
<td>17</td>
</tr>
</tbody>
</table>