A Nichesourcing Framework applied to Software Sustainability Requirements

Condori-Fernandez, Nelly; Lago, Patricia; Luaces, Miguel; Catala, Alejandro

published in
RCIS 2019 - 13th International Conference on Research Challenges in Information Science
2019

DOI (link to publisher)
10.1109/RCIS.2019.8877000

document version
Publisher's PDF, also known as Version of record

document license
Article 25fa Dutch Copyright Act

A Nichesourcing Framework applied to Software Sustainability Requirements

Nelly Condori-Fernandez[1,2], Patricia Lago[2]
Universidade da Coruña, Spain[1]
Computer Sciencedepartment
Vrije Universiteit Amsterdam, The Netherlands[2]
n.condori.fernandez@udc.es
{n.condori-fernandez, p.lago}@vu.nl

Miguel Luaces
Universidade da Coruña, Spain
Computer Sciencedepartment
miguel.luaces@udc.es

Alejandro Catala
Centro Singular de Investigacion en Tecnoloxias Intelixentes (CiTIUS)
Universidadede Santiago de Compostela, Spain
alejandro.catala@usc.es

***Abstract*— [Background] A software sustainability model is commonly defined in terms of social, economic, technical and environmental dimensions. A key challenge is its characterization regarding sustainability requirements and their dependencies since it depends on, and varies with, the type of software system, and changes in operational contexts. [Aims] With the purpose of facilitating the definition of a context-dependent software sustainability model, we introduce nichesourcing as a type of distributed problem-solving paradigm, named crowdsourcing. [Method] First a set of nichesourcing requirements were identified, and an existing nichesourcing methodology, originally used in the cultural heritage domain was analyzed with respect to its applicability in Software Engineering. [Results] As a result, we propose an adapted methodology for nichesourcing the knowledge on sustainability requirements and their dependencies. Strengths and challenges for crowdsourced software engineering are also identified.**

Keywords—crowdsourcing, dependency, sustainability requirements.

I. INTRODUCTION

Software systems are becoming more ubiquitous as people rely on them in their daily life, work and societal functions [1]. Accordingly, software sustainability is quickly becoming one of the key and urgent challenges of the 21st century.

In the last years, significant effort has been invested in providing principles and common basis of software sustainability (e.g. [2-4]). However, both assessment and design based on the notion of sustainability are still emerging [5-7]. A key challenge for software sustainability is its characterization as a software quality requirement [3]. Being able to identify the impact of quality requirements on sustainability is a first step towards developing software intensive systems that fulfill sustainability requirements.
ccordingly, and from a purely technical perspective, sustainability has been linked to the notion of software evolvability or longevity, e.g. [8]. However, software sustainability has a much broader scope. It is commonly defined in terms of economic, social, technical and environmental dimensions, which are tightly interdependent as argued in [3], [9].

This does not mean that all sustainability dimensions must be always addressed together to guarantee sustainability. In fact, the relevance of the different dimensions depends on, and varies with, the type of software system, and changes in operational contexts (e.g. depending on stakeholder goals). Hence, the identification of relevant sustainability requirements and their dependencies is not trivial since it often requires domain-specific knowledge. In order to gather people that can contribute from several perspectives, a possibility could be based on crowdsourcing, which is an emerging distributed problem-solving paradigm that has been successfully applied in certain software engineering activities [10].

Most crowdsourcing initiatives have been centered around users performing many but simple tasks, and their overall target was focused on "achieving quantity rather than quality" of participants. However, as we require groups of people with high expertise for contributing in the definition of a context-dependent software sustainability model, we introduce nichesourcing to address the growing demand for solving complex knowledge-intensive tasks without decreasing the quality of the final results. Nichesourcing is a specific type of crowdsourcing where complex tasks are distributed amongst a small crowd of experts rather than the faceless crowd [11].

In this paper we adapt a nichesourcing method, called Accurator [12], originally used in the cultural heritage domain. Our contribution is the adaptation of this method for crowdsourcing complex and indecomposable tasks highly demanded in the development of software systems. Specifically, we have applied it to support the definition of a software sustainability model for software-intensive systems, and we envision it can be applied to other relevant complex decision-making tasks (e.g. requirements prioritization, trade-offs analysis).

The paper is organized as follows. In section 2 we discuss the related works. Then we introduce the topic of software

978-1-7281-4844-1/19/$31.00 ©2019 IEEE

sustainability in section 3. Section 4 presents the nichesourcing method, which has been applied for supporting the identification of sustainability requirements and dependencies.

In section 5, we also discuss the advantages and challenges of implementing nichesourcing solutions, which give us an opportunity to extend the agenda of crowdsourced software engineering, when higher a knowledge-level is needed.

II. Related Works

In the last years, crowdsourcing has been intensively investigated. Some themes have been regarding the definition and characteristics of crowdsourcing in different fields (e.g., [17], [19]), taxonomies for crowdsourcing (e.g., [15], [16], [24]) platforms for crowdsourcing (e.g.,[18]). For example, crowdsourcing in requirements engineering (RE) has demonstrated to be a useful tool for eliciting requirements (e.g., [21],[22]) and conducting user studies within experimental contexts (e.g., [23]) for validating requirements. However, most of these types of activities have typically been limited to those tasks that are low in complexity, independent, and requiring low cognitive effort, and little time to complete them.

Although it is well-known the limitations of crowdsourcing platforms when complex tasks are deployed [10],[24], in this section we present related works on approaches that aim to address the solvability[1] feature in crowdsourcing.

CrowdForge [18] is one of the first frameworks developed for dealing with the complexity and interdependency of crowdsourced tasks using micro-task markets. However, CrowdForge is based on the assumption that complex tasks can be broken up into relatively small and independent pieces. This assumption can be violated when some work may not be easily decomposable. Another limitation is that CrowdForge does not support iteration.

Zheng et al. [20] proposed a general-purpose crowdsourcing platform, which is based on state machine workflow technology. This platform provides an automated support in the process of resolving and merging complex tasks, and also assumes the decomposition of task.

Lyu and Kantardzic [19] proposed the use of multiple views for complex tasks through crowdsourcing. The applicability of this multiple view strategy was investigated in evaluation tasks of programming projects or research papers.

Although all related works provide valuable contribution for providing support to solve complex tasks, most of them have yet some limitations regarding iteration, concurrency, and decomposition to address solvability of complex tasks.

Given that certain activities of the software systems development process might demand specific type of knowledge and skills due to their complexity, in this paper we aim to address the solvability feature of those complex tasks that cannot be broken down into more simple tasks, but they are still feasible for humans, by following a specific type of crowdsourcing called nichesourcing.

With the purpose of illustrating the application of nichesourcing, in the next section we briefly explain why the identification of sustainability requirements and their dependencies is a complex task.

III. Sustainability Requirements and Dependency

Sustainability is defined as the "capacity to endure" and "preserve the function of a system over an extended period of time."[25] The most frequently used definition of *software sustainability* refers to dimensions of economic, social, technical and environmental sustainability [9], [26]. According to Lago et al. [26], a sustainability model should include both traditional (system-related) quality requirements and sustainability-related ones. This does not mean that all sustainability dimensions must always be addressed together to guarantee sustainability. In fact, the relevance of the different dimensions depends on the type of software system. For instance, a software system designed to remind patients whether or not a medicine has been taken typically requires to address requirements of the social sustainability (like usability, safety, persuasion and health risk mitigation) and of the technical sustainability (like adaptability and availability, which are important to ensure user satisfaction and hence incentivize patients to use the application). Of course requirements of the environmental dimension could be met as long as required.

The identification of relevant requirements that are related to each sustainability dimension of software-intensive systems is not trivial since a variety of stakeholders with different background and specific domain knowledge is required (e.g. experts from climate science or experts from social sciences regarding psychology). The challenges arise when the stakeholders are not collocated since a close cooperation and collaboration is needed.

On the other hand, *requirements dependency* is addressed in different activities that demand a higher cognitive effort, such as requirements prioritization, software product release planning, change impact analysis, etc.

About 25 requirements dependency types were identified through a systematic review [13]. Table 1 shows some

[1] Solvability is the capability to be solved.

examples of requirements dependency types that can be addressed in different requirements engineering activities.

TABLE 1. EXAMPLES OF REQUIREMENTS DEPENDENCY TYPE

RE activity	Requirement dependency type
Change impact analysis	similar_to; resource; refined_to, generalize
Requirements evolution	replaces; satisfies; based on; changes_to
Product releasing	Positive_value; negative_value; requires: And; OR

According to Zhang et al. [13], requirements dependency identification is a subjective process impacted by people's understanding of requirements and relationships. Dependencies among sustainability requirements can exist at two different levels: i) among the four sustainability dimensions (i.e, economic, social, technical and environmental), and ii) between the sustainability requirements within a dimension [9]. Due to the interplay of both levels, the identification of both dependencies types is even more difficult.

IV. NICHESOURCING REQUIREMENTS AND METHOD

We first present the nichesourcing requirements taken into account for adapting the Accurator method [12].

A. *Nichesourcing requirements*

- *Iterative.*- Given its complexity, the nichesourcing task can be executed as many times as the requester considers necessary to reach the campaign goal as long as deadlines are met.
- *Skills matching.*- It is important to match the knowledge required by the nichesourcing task with the skills of the selected niche communities. In contrast to crowds, these communities have a common purpose, members have an identity and affinity with the purpose at hand, and regular interactions engender social trust and reputation [11].
- *Computer mediation.*- Similarly to crowdsourcing, tasks must be remotely executed using the Internet for communication and coordination.
- *Continuous interaction.*- In the execution of nichesourcing tasks, experts can use different channels of communication with i) requesters, when they, for instance, may require more explanations to understand the tasks; ii) peers, when for instance they need to reach each other. The communication can be asynchronous (e.g. email, forum) or synchronous (e.g. chat).
- *Blindness.*- In contrast to crowdsourcing, requesters must know the identity and skills of the contributors. They can monitor contributors' performance. However, anonymity should be kept when contributors interact among them (*single-blind*). This allows to avoid that strong contributors might influence others.
- *Flexible planning.*- Although we can count on contributors with similar domain knowledge, both planning and scheduling should adapt to the availability of the experts without affecting the achievement of the goal.
- *Trustness.*- For the assignment of tasks, requesters trust on contributor's capability to deliver solutions with good quality in terms of effectiveness[2].

B. *The Nichesourcing method*

Figure 1 shows the nichesourcing method adapted for the completion of complex tasks, such as the identification of sustainability requirements dependencies, for which decomposition is non-trivial and high domain-specific knowledge is required.

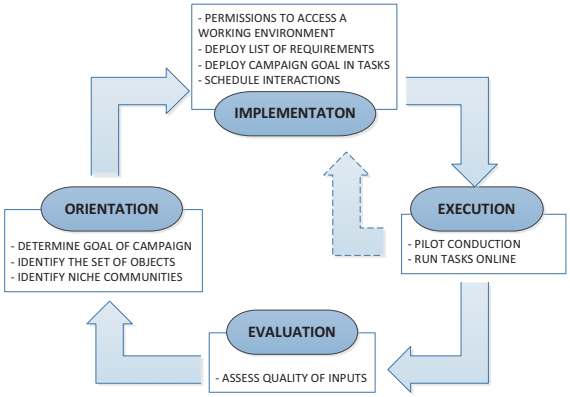

Fig 1. Nichesourcing methodology, adapted from[12]

It consists of four phases: Orientation, Implementation, Execution and Evaluation. All phases must be supported by a software tool.

1) Orientation:

a) Determine goal of campaign. The requester first determines what he or she wants to achieve with the sustainability requirements and dependencies identified during a nichesourcing campaign. It is important to remark that the goal has to be formulated in such a way that contributors deem it worthy to invest time into, fitting with their domain interest. E.g., the goal of a campaign could be as follows:

"Contribute to the definition of a software sustainability model for assessing software –intensive systems"

b) Identify the set of objects to analyze. The requestor carries out this step manually. However, as a dependency

[2] Degree to which something is successful in producing a desired result

identification task can be time consuming and require specific domain knowledge, requesters must consider a preliminary set of (candidate) requirements to address sustainability. For instance, traditional quality requirements that were identified as relevant for each sustainability dimension through a survey research method [9] could be considered as objects to analyze. Moreover, with the purpose of getting a common understanding among contributors, definitions of sustainability dimensions and requirements must also be given.

c) Identify niche communities to select potential contributors. To ensure the feasibility of a nichesourcing study, the set of objects to analyze (i.e., sustainability requirements of software-intensive systems) should match with the expertise of a niche community. It will facilitate to select potential contributors (professionals, senior researchers with background in Software Engineering and Sustainability). Examples of niche comunities in this topic are: members of the Karlskrona Manifesto for Sustainability Design [14], senior researchers from the Requirements Engineering community working on requirements for sustainability IT projects, etc.

According to Dijkshoorn et al. [12], a common feature of niche communities is that they can be divided into even more specialized sub-niches. It is useful to identify such sub-niches because it will help select most knowledgeable contributors. Table 2 shows examples of sub-niches (SN) that can be identified based on who can contribute the best professional knowledge needed for achieving the campaign goal. Each sub-niche consists of a set of candidate experts, contributing to at least in two of the four sustainability dimensions defined in previous works [9], [26]: Technical (T), Economic (E), Social (S) and Environmental (Env).

TABLE 2. SUB-NICHES DISTRIBUTION PER DIMENSIONS

	T	**E**	**S**	**Env**
T	0	SN1	SN2	SN3
E		0	SN4	SN5
S			0	SN6
Env				0

2) Implementation:

In this stage, as shown in Figure 2, tasks are designed and assigned to the selected contributors. This stage consists of the following steps:

a) Permissions to access a working environment. Permissions are granted to all contributors who accept participating in the nichesourcing campaign. According to the expertise of the contributor, the corresponding working environment is loaded. For this, it is very important to ask an explicit confirmation for addressing the "skills matching" requirement.

b) Deploy list of requirements per sustainability dimension in Tool. In this step, the tool is deployed and all relevant requirements identified in the orientation stage are loaded. All definitions of sustainability requirements and dimensions are also loaded.

c) Deploy campaign goal in Tasks. During this step, the goal of the campaign is translated into a set of tasks. With the purpose of ensuring the quality of solutions/inputs, the campaign is organized in a set of sessions. For example, by applying the Delphi technique, the requester can organize the campaign in three sessions as shown in Table 3.

TABLE 3: NICHESOURCING TASKS PER SESSIONS

	Purpose
Session 1	Identify dependencies among each pair of requirements (x,y) that belongs respectively to the sustainability dimensions (A and B)
Session 2	Review your dependency matrix by paying first attention to possible missing or incorrect dependencies
Session 3	Reach a consensus by discussing with your peer(e.g. via chat) to solve conflicts.

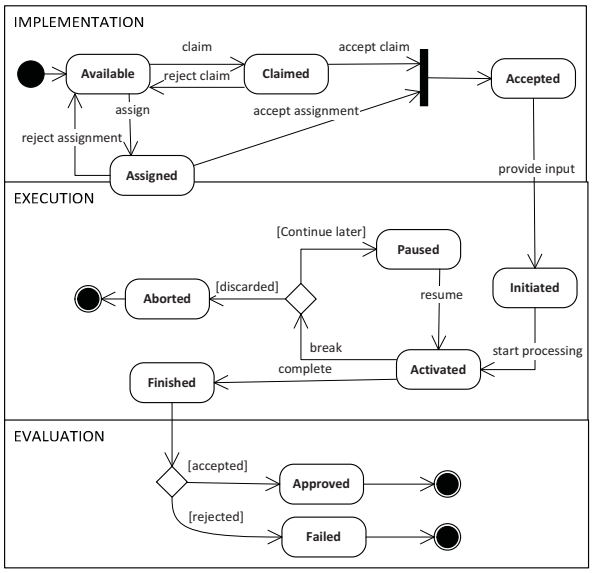

Fig. 2. Possible states of tasks during implementation, execution and evaluation stages of nichesourcing

d) Schedule interactions between contributors and requesters. To facilitate the interaction between contributors and requesters, the nichesourcing tool must support both synchronous and asynchronous communication. Scheduling synchronous communication is harder because of different

time zones and the diversity of work. However, scheduling this type of communication upfront with each sub-niche is needed especially for some sessions (e.g. session 3).

3) Execution:

a) Pilot conduction. To test the nichesourcing tool, the formulated tasks, and the corresponding planning, a pilot is conducted with a limited number members of the targeted niche community.

b) Run tasks online. Once contributors accept a task, they can start providing their inputs in an asynchronous way. With the purpose of avoiding/reducing aborted tasks, continuous support should be provided to contributors. Moreover, as shown in Figure 2, interruptions might occur during the study. If so, the tool should enable to resume suspended tasks in case requesters decide to pursue it later. Reminders to meet the set deadlines of the campaign should be sent for the task completion.

4) Evaluation:

Once a task is finished, the quality of the inputs provided in each session should be verified by the requester. This assessment can be manual or automated. An example of automated assessment is the identification of conflicts in the third session.

The requester decides, based on the assessment, to approve or to reject the solution(responses). It is failed if the contributor is unable to deliver the desired output. A task is successful if most of the outcomes of a contributor are approved by the requester.

Table 4 shows examples of external applications that can be used for supporting each nichesourcing stage. Moreover, based on our own experience in running a pilot study with 12 experts that implemented the method presented, averaged estimated durations are also proposed.

The length of the blue bars corresponds to 8 (A, B, D), 4 (E), and 3 (C,F) working days. The lighter blue bars represent the additional time that should be considered at i) orientation stage when more contributors were needed to be included; ii) execution stage when issues detected in the pilot may require more time to be solved.

TABLE 4. ESTIMATED PLANNING OF A NICHESOURCING CAMPAIGN

Campaign	Support
Orientation (A)	Linkedin, Twiter, ResearchGate
Implementation (B)	Tool, email
Execution:	Tool,
Pilot (C)	Email, chat,
Run tasks:	
First session (D)	
Second session (E)	
Third session (F)	

V. ISSUES AND CHALLENGES

Mao et al.[10] identified a series of issues regarding the application of crowdsourcing in software engineering. Nichesourcing is a natural evolution of crowdsourcing that emerged to address some of these issues shown in Table 5.

TABLE 5. STRENGTHS AND CHALLENGES FOR IMPLEMENTING NICHESOURCING IN SOFTWARE ENGINEERING

Crowdsourcing Issues [10]	**Nichesourcing**	
	Strengths (S)	**Challenges(CH)**
Task decomposition	S1: The assigned complex tasks are carried out in an iterative way.	CH1: Tools or platforms should provide continuous communication and coordination support for making right assignments of complex tasks, and monitoring contributors during the "n" iterations of the study. One question that arise is: How many iterations would be enough for considering a task as successful?
Skill barriers	S2: Involvement of contributors with specific knowledge selected from niche communities.	CH2: Matching skills of potential contributors with specific knowledge required for solving a task can demand a lot of effort. Platforms/tools should provide (semi-)automated support for identifying "suitable contributors". Considering additional selection criteria can be even more challenging, such as availability, enthusiasm or motivation on the theme, that would ensure high levels of engagement.
Motivation	S3: Social trust and reputation are key drivers of niche communities. S4: Traditional motivation techniques from crowdsourcing like remuneration are not needed anymore.	CH3: Keep the motivation of contributors in long-term studies is a well-known challenge, especially when they are locally distributed (cultural aspects should be considered). It is needed to empirically investigate the applicability of new motivators for nichesourcing.
Careful planning	S5: A flexible planning makes feasible the completion of tasks.	CH4: Adapting continuously the planning and schedule according to the availability of contributors could affect to other ones. So, some mechanisms to facilitate agreeing on deadlines and schedule for synchronous communication should be provided.

Crowdsourcing Issues [10]	Nichesourcing	
	Strengths (S)	**Challenges(CH)**
Quality assurance	S6: Higher quality individual results because nichesourcing tasks are performed by members from socially-trusted communities,	CH5: Determination of minimum level of skills and knowledge to ensure an enough group of skilled contributors for each required sub-niche. CH6: Provide opportune support and positive feedback to contributors for increasing the quality of individual results.

It also summarizes corresponding strengths and challenges for implementing nichesourcing studies in software engineering. The conduction of the pilot has been helpful on realizing these strengths and challenges. It is also important to remark that although our method has been applied specifically to support the identification of sustainability requirements and dependencies, we consider that the method might be applicable to any type of complex knowledge-intensive task. As a future work, we are going to analyze the collected data from our pilot and apply our method for identifying sustainability requirements of a specific healthcare software.

ACKNOWLEDGMENT

This work has been partially funded by the Spanish Ministry of Economy, Industry and Competitiveness with the projects: TIN2016-78011-C4-1-R, TIN2016-77158-C4-3-R; and Conselleria de Cultura, Educacion e Ordenacion Universitaria (accreditation 2016-2019, ED431G/08) and the European Regional Development Fund (ERDF).

REFERENCES

[1] M. Wieczorek, D. Vos, H. Bons. Systems and Software Quality. Springer, Germany. 2014

[2] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler,N. Sey, C. C. Venters, Sustainability design and software: The karlskronamanifesto, in: Proceedings of the 37th International Conferenceon Software Engineering - Vol 2, ICSE SEIS, IEEE Press, 2015, pp. 467-476.

[3] P. Lago, S. A. Kocak, I. Crnkovic, B. Penzenstadler, Framing sustainabilityas a property of software quality, Communications of the ACM 58 (10) (2015) 70-78

[4] M. Razavian, G. Procaccianti, D. A. Tamburri, Four-dimensional sustainablee-services, in: U. V. A. W. B. R. N. G. Jorge Marx Gmez,Michael Sonnenschein (Ed.), BIS-Verlag, 2014, pp. 221-228.

[5] S. Betz, C. Becker, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, C. C. Venters, Sustainability debt: A metaphor to support sustainability design decisions, in: RE, 2015.

[6] A. Raturi, B. Penzenstadler, B. Tomlinson, D. Richardson, Developing asustainability non-functional requirements framework, in: Proceedings of the Int. Workshop on Green and Sustainable Software, GREENS 2014, ACM, New York, NY, USA, 2014, pp. 1-8.

[7] H. Koziolek, Sustainability evaluation of software architectures: A systematicReview. In: Proceedings of the Joint ACM SIGSOFT Conference-QoSA and ACM SIGSOFT Symposium -ISARCS, ACM, New York, USA, 2011, pp. 3-12.

[8] P. Avgeriou, M. Stal, R. Hilliard, Architecture sustainability [guest editors'introduction], Software, IEEE 30 (6) (2013) 40-44.

[9] N. Condori-Fernandez, P. Lago, Characterizing the contribution of quality requirements to software sustainability, Journal of Systems and Software,Volume 137,2018, Pages 289-305.

[10] Ke Mao, Licia Capra, Mark Harman, YueJia. A survey of the use of crowdsourcing in software engineering. The Journal of Systems & Software (2017) 126:57-84.

[11] V. de Boer et al. (2012). Nichesourcing: Harnessing the Power of Crowds of Experts. In Knowledge Engineering and Knowledge Management. EKAW 2012. LNCS, vol 7603. Springer, Berlin.

[12] C. Dijkshoorn, V. De Boer, L. Aroyo, G. Schreiber. Accurator: Nichesourcing for Cultural Heritage, Human Computation. pp. 101–131, 2014.

[13] H. Zhang, J. Li, L. Zhu, R. Jeffery, Y. Liu, Q. Wang, M. Li, Investigating dependencies in software requirements for change propagation analysis, Information and Software Technology, Volume 56, Issue 1,2014. pp. 40-53.

[14] Karlskrona manifesto for sustainability design: http://sustainabilitydesign.org/ (Last visit 30th December, 2018)

[15] D. Geiger, S. Seedorf, T. Schulze, R. C. Nickerson, and M. Schader, "Managing the crowd: Towards a taxonomy of crowdsourcing processes." in AMCIS, 2011.

[16] M. Hosseine, K. Phalp, J. Taylor, R. Ali. The four Pillars of Crowdsourcing: a Reference Model. 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS). Marrakech, Morocco. 2014.

[17] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, and M. Stade, "The Crowd in Requirements Engineering: The Landscape and Challenges" IEEE Softw., vol. 34, no. 2, pp. 44–52, 2017.

[18] A. Kittur, B. Smus, S. Khamkar, R. E. Kraut. CrowdForge: Crowdsourcing Complex Work. 24th ACM Symposium on User Interface Software and Technology. Santa Barbara. USA. October 2011.

[19] L. Lyu, M. Kantardzic. Evaluating Complex Task through Crowdsourcing: Multiple Views Approach. International Journal of Crowd Science. 1(2). 2017

[20] Q. Zheng, et al. Crowdsourcing Complex Task Automatically by workflow Technology. Management of Information, Process and Cooperation. Springer. 2017.

[21] M. Hosseini, K. Phalp, J. Taylor, and R. Ali. "Towards crowdsouricing for requirements engineering". In Procesedings of the 20th Internationational working conference on Requirements Engineering: foundation for software quality (REFSQ)- Empirical Track, 2014.

[22] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, R. Ali and F. Dalpiaz, "Configuring crowdsourcing for requirements elicitation," 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), Athens, 2015, pp. 133-138.

[23] L. Layman and G. Sigurdsson, "Using Amazon's Mechanical Turk for User Studies: Eight Things You Need to Know," 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement(ESEM), Baltimore, MD USA, 2014, pp. 275-278.

[24] Hosseini, M.; Shahri, A.; Phalp, K.; Taylor, J. & Ali, R. Crowdsourcing: A taxonomy and systematic mapping study Computer Science Review, 2015, 17, 43 – 69.

[25] Hilty, L.M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., and Wäger, P.A. The relevance of information and communication technologies for environmental sustainability: A prospective simulation study. Environmental Modelling & Software 21, 11 (Nov. 2006).

[26] P. Lago, S. A. Kocak, I. Crnkovic, B. Penzenstadler, Framing sustainability as a property of software quality, Communications of the ACM 58 (10) (2015) 70-78.