Human Factors in Software and Systems Engineering, Vol. 94, 2023, 86–94
https://doi.org/10.54941/ahfe1003773

Crowdsourcing for Second Language Learning

Abdelrahman Abounegm, Magomed Magomedov, Nursultan Askarbekuly, and Manuel Mazzara

Software Engineering Lab, Innopolis University, Tatarstan, 420500, Russian Federation

ABSTRACT

This work implements language acquisition and crowdsourcing techniques in a unique combination to aid with second language learning. It allows users to contribute linguistic assets, while other users vote on the quality of the assets. The system's goal is to provide a social platform for learning and contributing to underrepresented languages. The authors establish quality attributes for the system, namely: usability, scalability, security, and portability. The resulting system is tested against these quality attributes using quality scenarios and usability testing. The implemented system is shown to possess the quality of security, scalability, and portability. Usability testing highlights the importance of user interface for crowdsourcing systems and shows possible interface improvements.

Keywords: Empirical software engineering, Crowdsourcing, Language learning

INTRODUCTION

This work aims to utilize state-of-the-art crowdsourcing techniques in secondary language learning, which can benefit from a distributed crowd of volunteers and provide diversity and quality that would not otherwise be possible with a limited team of editors. This is especially important for minority and endangered languages, which are absent from mainstream applications and whose communities are eager to use opportunities to promote their heritage (Finardi, Leao, & Amorim, 2016). This work proposes a software system that will utilize the benefits of crowdsourcing for the purpose of second language learning (Solemon, Ariffin, Din, & Anwar, 2013). It has the following business goals: facilitate the crowdsourcing of linguistic assets through a highly usable software tool, provide a gamified experience for users to learn from the gathered assets, build an active community around the combination of crowdsourcing and gamified language learning, and gather linguistic assets for a wide variety of languages. Following the requirements process (Askarbekuly et al. 2021), we used the business goals to derive functionality and metrics to quantify the extent to which the project succeeded in attaining its targets. The methodology is a creative crowdsourcing solution that accepts contributions from anyone in any language and uses the learner's feedback to automatically rate these contributions and award high quality ones. The discussion and conclusion are proposed, stating limitations of the research, possible future improvements, and the significance of the results.

RELATED WORK

Crowdsourcing is a growing field of research that aims to advance the field by filling the gaps and initiating new potential directions. Recent research (Wang, Duy, Hoang, & Kan, 2010) groups crowdsourcing systems into three genres: Mechanical Turk, Games with a Purpose (GWAP), and Wisdom of the Crowds (WotC). To test the tendency, we develop a hybrid web-backed mobile application that attempts to fuse GWAP and WotC, taking fun and usability from the former and amplifying them with recognition and specialization of the latter. In terms of model, a crowdsourcing system can be competitive or marketplace, depending on the interaction between a task requester and provider. Observable prospects include reduced costs, subsequent diversification of the crowd via an opening of new markets, and the ability to select qualified participants through their previous results on the crowdsourcing marketplaces (Wang, Bohus, Kamar, & Horvitz, 2012). The research concludes by discussing a set of future challenges.

Language Learning

Language learning strategies include cognitive, memory-related, compensatory, affective, social, and acquisition-learning theories (Oxford, 2001). Cognitive strategies focus on directly consuming language material, while memory-related strategies focus on memorizing and compensatory strategies focus on compensating for missing knowledge. Affective strategies focus on the emotional side of the learner, while social strategies focus on interacting with other learners or native speakers.

Acquisition-learning theories include Stephen Krashen's five hypotheses (Krashen, Principles and practice in second language acquisition, 1982):

- Acquisition-learning: Acquisition is a subconscious process, while learning involves actively studying the language and its grammar rules.
- Monitor: learners pay attention to what they say, editing and correcting themselves, assuming they know the rules and have studied the language.
- Input: for language acquisition to take place, the learner must be exposed to input that is one level higher than their current stage of language competence
- Affective filter: a person's emotional state plays a big role in their language acquisition.
- Natural order: there is a natural order for learning grammatical structures of a language, independent of the learner's age.

Krashen stipulates that acquisition is more important than learning, monitoring is important, and over-using the monitor can hinder acquisition.

The theory of statistical learning states that learners use statistical properties of any linguistic input to recognize patterns in the language and generalize them to previously unseen inputs (Sembok, Abuata, & Bakar, 2011). Words can have different meaning depending on how they are used in a sentence. When learning a language, it is important to learn words in the contexts in which they appear. This can be done through stories written in the target language, which add the benefit of learning about the customs

and traditions of the people whose language one is learning (Mixon & Temu, 2006). Using stories and dialogues to learn a language engages both emotions and cognitive abilities, helping students identify the patterns present in the language (Lampariello, 2020). This method has seen numerous applications, and several authors have published books, YouTube videos, as well as podcasts employing the method of stories to teach languages.

METHODOLOGY

Crowdsourcing mechanisms to language learning require a proper design and methodology to achieve the best possible outcome, guaranteeing all business goals.

Architecture Derivation

The method of Attribute-Driven Design (Wojcik, et al., 2006) was applied in order to derive a suitable architecture for the system. This is a four-step procedure that entails:

1. Identifying the business goals
2. Identifying the architectural drivers
3. Deriving the system architecture
4. Deriving the software design

After that comes the step of verifying the design to ensure that it meets the required quality attributes. This is primarily achieved by generating quality attribute scenarios and deriving various tests from them.

Business Goals

Business goals are the primary objectives for the authors to achieve in a specified time frame, and do not necessarily indicate a profit-oriented business.

Architectural Drivers

Architectural drivers are system requirements that are significant from the architectural perspective (Fröberg, Larsson, & Nordlander, 2013). They mainly include the quality attributes, technical and business constraints, and some functional requirements.

For the quality attributes, the ISO 25010 (ISO, 2011) defines a reference for the characteristics which a system can possess. The technical and business constraints are considered when choosing the most important quality attributes.

Functional requirements are the features that the system must offer for the users to be able to achieve their goals from using the system. They are commonly written in the form of user stories (Cohn, 2004) which outline the requirements from the perspective of the user. Some functional requirements may affect architectural decisions, while the majority are architecturally insignificant.

System Architecture

The attribute-driven design method (Bass, Clements, & Kazman, 2012) uses the previously outlined attributes to derive an appropriate architecture. First, a design roadmap is chosen, then a subset of the system's elements is chosen. The requirements for these elements are identified and a design concept is selected based on them. After that, architectural elements are instantiated. These decisions are then recorded, and the process is repeated until a condition is met, indicating that a satisfactory architecture is reached.

Architectural design concepts are methods by which the target requirement can be tackled. For each such architecturally significant requirement, a design concept is selected to be applied on the architecture generated thus far. This includes choosing the architectural elements, allocating their responsibilities, as well as defining their interfaces.

Software Design

After deriving the system architecture, the low-level software design becomes straightforward since it is a matter of picking the appropriate design patterns that serve the desired quality attributes. This gets reflected on the structure of the code in terms of module distribution.

Verification

After the architecture is developed and the software gets designed, these designs must be validated. In this case, validation means verifying that the system possesses attributes that promote the business goals stated earlier. Quality attribute scenarios and acceptance criteria are used to perform such verification.

For every selected quality attribute, a scenario is written as a test that verifies the presence of the aforementioned attribute. The framework of quality attribute scenarios can be summarized in the following table (Du, 2010):

Usability Testing

Usability testing (Lewis, 2006) is a technique used to evaluate a product by testing it on real users and collecting feedback on how they use it. It is most useful in highly interactive, user-centered designs (Pimenov, 2021) and involves identifying key stakeholders, designing tasks for each stakeholder group, and asking volunteers to complete the tasks. The goal is to understand how easy the users find the product to interact with without prior guidance. It is

Table 1. Quality attribute scenarios overview.

Scenario part	Description
Source of stimulus	The entity (human or otherwise) whence the stimulus is generated
Stimulus	The event that arrives at the system
Environment	The conditions of the system when the stimulus arrives
Artifact	The stimulated part of the system
Response	The activities to perform upon the arrival of the stimulus
Response measure	The metric that determines the success of the response

important to inform the volunteer users of the expected end result of their test task and not how to achieve it, since the goal is to figure out how the users think it is natural to achieve it. This method can be classified as unmoderated, assessment testing (8 usability testing methods that work (types + examples), 2022).

IMPLEMENTATION

This chapter discusses the implementation of the system and what was developed.

System Design

As outlined in the previous chapter, the Attribute-Driven Design methodology was followed in order to reach a suitable architecture for the system. That entailed identifying the business goals, deriving architectural drivers from them, using the architectural drivers to build an architecture, and finally designing the software that actually implements the architecture.

Architectural Drivers

The architectural drivers consist of system constraints, quality attributes (according to the ISO 25010 model), and architecturally significant functional requirements. There are no business constraints imposed on the system.

As a crowdsourcing platform, the system is expected to possess the attributes of scalability, usability, portability, as well as security. The system must be able to **scale** depending on the demand levels since its main requirement is to gather resources from a large number of users. These users may not necessarily be tech-savvy, so the system must be easily **usable** for them. Similarly, they may not always have access to the same kind of device, so the system must be **portable**. Lastly, users expect their privacy to be protected, so **security** is essential. For each of these attributes, a quality attribute scenario is designed to ensure that it is promoted by the system architecture.

As for the architecturally significant functional requirements, the following can be identified:

- Users should be able to create stories that are persisted in a database and shared such that any other user can read them, while only the author can modify them.
- Users should be able to search through the stories using filters such as language and popularity.

The identified quality attribute scenarios are outlined in Tables 2–5.

System Architecture

From the identified architectural drivers, attribute driven design was followed, leading to a software architecture with the following components:

- Firebase: A Platform as a Service offering scalable backend products.
- Algolia: A Software as a Service that provides an efficient search engine.

Table 2. Scalability quality attribute scenario.

Scenario part	Description
Source of stimulus	A new user
Stimulus	The user registers and attempts to play a story
Environment	System already has a large number (>10000) of users and stories
Artifact	The whole system
Response	The requests are handled successfully
Response measure	The request is handled in the same amount of time it would if there were only a few (<100) users and stories

Table 3. Usability quality attribute scenario.

Scenario part	Description
Source of stimulus	The end user
Stimulus	The user wished to learn to use the system effectively
Environment	Normal system runtime and system configuration
Artifact	The mobile app through which the user interacts with the system
Response	The user interface provides easy access to all the features the user needs
Response measure	Usability testing was successful with positive feedback

Table 4. Portability quality attribute scenario.

Scenario part	Description
Source of stimulus	A registered user
Stimulus	Starts an action on one device, and wants to continue on another device
Environment	Under normal working conditions
Artifact	The whole system
Response	The user's actions are saved and they are able to continue on them on the other device
Response measure	No data is lost while switching to the other device

Table 5. Security quality attribute scenario.

Scenario part	Description
Source of stimulus	An unauthenticated user
Stimulus	Attempts to modify a story
Environment	Under normal working conditions
Artifact	The database
Response	Firestore security rules should block the request
Response measure	No data has been modified

- Flutter: A front-end framework that allows for rapid development of user-friendly cross-platform applications.

Firebase provides scalability, security, and a strong security system, while Algolia provides ease of access and search. Flutter helps build a portable application with an intuitive user experience. Together, these services provide a straightforward realization of the quality attributes.

Software Design

The software design is about how the code is structured to achieve the architecture above. The Feature Sliced Design methodology was applied to achieve a decoupled and highly maintainable codebase.

Verification

To verify the promotion of each quality attribute, tests must be derived based on response measures. Load testing, Usability Testing, and Portability Testing are used to verify scalability, usability, and security. Load testing is done by applying a high demand on the system and measuring its response. Usability Testing is done by performing tasks a typical user would perform on one device and ensuring that performing the same set of steps on another device with leads to the same results. Portability testing is done manually since it is difficult to find an automated tool that can be customized to the features of each specific system. Security testing is done through Firebase's SDK and Firestore's Rules Playground simulator to ensure no user can access or modify any date they should not.

Usability Testing

To perform usability testing, we identified the key stakeholders of the system, devised test tasks for each group of stakeholders, and recorded and analyzed the test sessions. These stakeholders included language learners, story narrators, translators, editors, and reviewers. Tests were designed to cover the most relevant features for each group, such as completing a store, translating an existing store, or creating a new one. After recording the test executions, they were reviewed and analyzed to extract the main pain points of the application and pinpoint areas where improvements are most needed.

RESULTS AND DISCUSSION

System tests and usability tests are used to verify the usability of a system. This chapter presents the results of these tests and discusses the implications.

System Verification

Armed by the Quality Attribute Scenarios designed in the Implementation section, the *Response Measures* were used as test cases for either automated or manual testing.

The response measure for scalability was that "the request is handled in the same amount of time it would if there were only a few users and stories". According to the Advanced Databases Project, Firestore guarantees being able to handle up to one million concurrent connections without affecting latency or error rate. The results of usability tests are detailed in the next section. The response measure for portability was to ensure that no data is

lost while switching to the other device, and that all features can be performed on both devices. All activities are portable and can be continued on a different device, granted that the user is logged in on both devices with the same account. The application also has working versions on Android, iOS, and Web, so it passes the portability test.

The quality of security was measured by the fact that "no data has been modified" after a user attempts to modify data they are not authorized to modify. This was tested by hand by enumerating possible scenarios and combinations of user authorization and data, judging whether the action should be allowed, and evaluating whether the security rules block that action. After testing and updating the rules multiple times, it was demonstrated the system is secure enough and no data gets modified by unauthorized users. Thus, the system is considered secure.

Usability Testing

The feedback gathered from the first round of usability testing was indispensable to the improvement of the product of finding out areas where it lacked. The tests were conducted with four participants, all of whom opted for the Learner role, with one of them additionally testing the Narrator role, and another testing the Translator role. More rounds of testing will be done in the future after addressing issues highlighted by the first round.

CONCLUSION

In this work, a cross-platform application was developed that combines Krashen's theories for language learning with state-of-the-art crowdsourcing mechanisms to promote second language learning. One significant outcome is that the system allows for the democratic contribution of resources for underrepresented languages while ensuring that all contributed assets remain of a high quality, as determined by the users. The resulting application provides a gamified experience for users with a social element where users can interact together by viewing and liking each other's stories. This was supported by a flexible infrastructure and a number of volunteers who helped test the system. Verification methods were also employed to measure the success of the developed platform. In general, the system passed all the criteria except for usability, which will be improved in the future. In summary, other than the minor usability issues, the system can be said to have achieved its business goals.

REFERENCES

8 usability testing methods that work (types + examples). (2022, February). *8 usability testing methods that work (types + examples)*. Hotjar. Retrieved from https://www.hotjar.com/usability-testing/methods/

Askarbekuly, N., Solovyov, A., Lukyanchikova, E., Pimenov, D., & Mazzara, M. (2021). *Building an educational product: constructive alignment and requirements engineering*. In Advances in Artificial Intelligence, Software and Systems Engineering: Proceedings of the AHFE 2021 Virtual Conferences on Human Factors

in Software and Systems Engineering, Artificial Intelligence and Social Computing, and Energy, July 25-29, 2021, USA (pp. 358–365). Springer International Publishing.

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice* (3rd ed.). Addison-Wesley Professional.

Cohn, M. (2004). *User stories applied: For agile software development*. Addison-Wesley Professional.

Du, W. (2010). Understanding quality attributes. *Understanding quality attributes*. University of New Brunswick. Retrieved from https://www.cs.unb.ca/wdu/cs6075w10/sa2.htm

Finardi, K. R., Leao, R. G., & Amorim, G. B. (2016). Mobile assisted language learning: Affordances and limitations of Duolingo. *Education and Linguistics Research, 2*, 48–65.

Fröberg, J., Larsson, S., & Nordlander, P.-Å. (2013). A method for analyzing architectural drivers when engineering a system architecture. *2013 IEEE International Systems Conference (SysCon)*, (pp. 711–717).

ISO. (2011). *System and software quality models*. ISO, International Organization for Standardization, Geneva. Retrieved from https://www.iso.org/standard/35733.html

Krashen, S. (1982). Principles and practice in second language acquisition.

Lampariello, L. (2020, July). The remarkable power of stories and storytelling as ways to learn languages. *The remarkable power of stories and storytelling as ways to learn languages*. Retrieved from https://www.lucalampariello.com/storytelling-learn-languages/

Lewis, J. R. (2006). Usability testing. *Handbook of human factors and ergonomics, 12*, e30.

Mixon, M., & Temu, P. (2006). First Road to Learning: Language through Stories. *English Teaching Forum*, *44*, pp. 14–19.

Oxford, R. (2001, January). Language learning styles and strategies: An overview.

Pimenov, D., Solovyov, A., Askarbekuly, N., & Mazzara, M. (2021, December). *Data-Driven Approaches to User Interface Design: A Case Study*. In Journal of Physics: Conference Series (Vol. 2134, No. 1, p. 012020). IOP Publishing.

Sembok, T., Abuata, B., & Bakar, Z. (2011, May). A Rule and Template Based Stemming Algorithm for Arabic Language. *International Journal of Mathematical Models and Methods in Applied Sciences, 5*.

Solemon, B., Ariffin, I., Din, M. M., & Anwar, R. M. (2013). A review of the uses of crowdsourcing in higher education. *International Journal of Asian Social Science, 3*, 2066–2073.

Wang, A., Duy, C., Hoang, V., & Kan, M.-Y. (2010, August). Perspectives on Crowdsourcing Annotations for Natural Language Processing. *Language Resources and Evaluation, 47*. doi: 10.1007/s10579-012-9176-1.

Wang, W., Bohus, D., Kamar, E., & Horvitz, E. (2012, December). Crowdsourcing the acquisition of natural language corpora: Methods and observations., (pp. 73–78). doi: 10.1109/SLT.2012.6424200.

Wojcik, R., Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R., & Wood, B. (2006). *Attribute-driven design (ADD), version 2.0*. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.