



Article

# Crowdsourced Evaluation of Robot Programming Environments: Methodology and Application

Daria Piacun, Tudor B. Ionescu \*  and Sebastian Schlund 

Human-Machine Interaction Group, Vienna University of Technology, 1040 Vienna, Austria;  
e1402189@student.tuwien.ac.at (D.P.); sebastian.schlund@tuwien.ac.at (S.S.)

\* Correspondence: tudor.ionescu@tuwien.ac.at

**Abstract:** Industrial robot programming tools increasingly rely on graphical interfaces, which aim at rendering the programming task more accessible to a wide variety of users. The usability of such tools is currently being evaluated in controlled environments, such as laboratories or companies, in which a group of participants is asked to carry out several tasks using the tool and then fill out a standardized questionnaire. In this context, this paper proposes and evaluates an alternative evaluation methodology, which leverages online crowdsourcing platforms to produce the same results as face-to-face evaluations. We applied the proposed framework in the evaluation of a web-based industrial robot programming tool called *Assembly*. Our results suggest that crowdsourcing facilitates a cost-effective, result-oriented, and reusable methodology for performing user studies anonymously and online.

**Keywords:** robot programming; user interface evaluation; crowdsourcing



**Citation:** Piacun, D.; Ionescu, T.B.; Schlund, S. Crowdsourced Evaluation of Robot Programming Environments: Methodology and Application. *Appl. Sci.* **2021**, *11*, 10903. <https://doi.org/10.3390/app112210903>

Academic Editor: Carlo Canali

Received: 8 October 2021

Accepted: 9 November 2021

Published: 18 November 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The “fourth industrial revolution” or Industry 4.0 and its embodiment as the “digital factory” depicts the softening of limits between the physical, digital, and natural universes. Industry 4.0 combines advances in AI, IoT, robotics, 3D printing, genetic engineering, quantum computing, and other technologies. One of the pillars of Industry 4.0 is represented by human–robot collaboration. This form of collaboration requires extensive research, which is not focused on robotics alone but also on user-centered, simplified robot programming as a strong foundation for smooth human–machine interaction (HRI) [1].

The field of industrial robotics has progressed in the past decade, with companies looking to embrace this technology in diversified areas as it brings a high return on investment in a short time, increases productivity and velocity, and reduces the risk of human error caused mainly by the monotony and repetitiveness of tasks. Robots, in contrast to human workers, would mean a single more considerable initial investment in the hardware, which can later be employed 24 h/day, keep the entire machinery operating even without the physical presence of human workers, master repetitive tasks quickly, and even specialize in customizable ones.

Regarding customization, the ever rising demand for lot size 1 products is challenging this approach mostly from the software point of view, task changes within one industrial robot’s timeline being rare [2]. The complexity and high costs of programming and re-programming the robot are some of the main reasons for the low flexibility in these changes [3]. This emphasizes the relevance of creating a simple yet powerful and universally applicable robotic programming environment.

### Problem Statement

Currently, in the field of industrial robotics, the design and development of the robotic hardware has resulted in flexible machines with a high number of capabilities. As a result, relatively little attention has been granted to the underlying software components [4] that

animate robots. Aspects such as a visual representation of programming and interaction tools have recently started to become “first-class citizens” in robotics research. Previous research [5] has shown that older industrial robots can almost exclusively be introduced and controlled via proprietary, text-based, and predominantly low-level programming languages, derived from Pascal and BASIC and created back in the early 1990s. These programming languages have historically been designed by engineers for their own peers, which inevitably implies that writing or even reading these programs requires corresponding schooling or years of education and training. This furthermore means that only a minority of small- and medium-sized enterprises will be able to benefit from robotic automation since they cannot afford such highly educated workers [6].

As programming becomes ubiquitous and programming skills are increasingly required of employees with low or no technical background [4], developing simple and user-friendly robot programming interfaces to ease the programming task for non-experts has become an essential branch of research. There are already many robot programming environments available, but none have yet been established as the indispensable standard.

One of the particularities of UIs is that their validation cannot be carried out quantitatively and objectively—as opposed to other kinds of software, which can be validated through testing. Since such interfaces are designed to be used by humans, their evaluation must be carried out by their projected users. Graphical robot programming environments make no exception to this rule. Determining whether one programming environment is better than another is a matter of asking a group of users to try out the respective tools and then answer some questions. This methodology is part of the broader user studies field, which is the de facto standard in evaluating user interfaces. A major drawback of this approach is that established usability evaluation methods are both resource- and time-consuming [7] because they imply a series of qualitative process steps. First, user interface (UI) designers need to determine a relevant user group and approach a sufficient number of users from this group for the study to be relevant. The candidates must then be persuaded to participate in the study and supported throughout the process. Typically, user evaluations are performed in one location over 1–2 days, during which the researchers must accommodate and provide support to a high number of users (e.g., over 100 in the case of [4]). This can, for example, be achieved by inviting users to a designated location or by going to a specific location, where a pool of relevant users is available, such as a company [4]. One significant drawback of this approach is that the diversity within the user pool is inherently limited (i.e., the employees of a specific company, university students, workers of a specific factory, etc.).

In response to these issues, this work explores a more cost-effective and faster solution to the problem of evaluating the usability of simplified robot programming environments. The proposed methodological framework draws on crowdsourcing the evaluation of web-based robotic programming environments. Crowdsourcing comprises a number of methods for dividing and distributing tasks to a dynamic and diverse group of participants. The distribution and monitoring of task execution is usually performed on the Internet with the help of specialized platforms, such as Amazon Mechanical Turk (mTurk). The participants are paid for carrying out the specified tasks. Thanks to their large pool and great variety of participants, Internet-based crowdsourcing platforms provide an alternative to face-to-face user studies.

Due to the COVID-19 crisis, over the past year, crowdsourcing user studies have become more common than traditional user studies. In this context, potential adopters of this methodology would benefit from a systematic methodology and framework for conducting crowdsourced user studies in order to reduce their potential for errors and excessive costs. This paper introduces such a methodological framework, which was developed over the past year as part of a publicly funded project called “Cobot Meets Makerspace” (CoMeMak: <https://www.comemak.at> accessed on 11 November 2021). The CoMeMak project aims to democratize cobot technology for a wide variety of users, reaching beyond traditional industrial contexts, such as factories. The framework was

assessed by applying it in the evaluation of a simplified robot programming tool, called *Assembly*, which was developed as part of the CoMeMak project [8,9].

The work reported in this paper was guided by the following research questions:

1. Is the crowdsourcing approach a feasible and repeatable method of evaluation of web-based robotic programming interfaces?
2. To what extent can crowdsourcing support the usability evaluation of a web-based robotic programming interface?
3. How diversified is the audience in a crowdsourcing platform of choice (i.e., Amazon mTurk) both from a demographic and technical expertise point of view?
4. What are the strengths and weaknesses of the simplified robot programming tool used during the evaluation (i.e., *Assembly*—<https://assembly.comemak.at> accessed on 11 November 2021)?

Our results suggest that the proposed methodology is reusable and capable of setting up user studies over crowdsourcing platforms in a fast and cost-effective manner. To aid potential adopters of the framework, our evaluation includes aspects such as fraud detection and clarity of task definition. The early understanding of such issues is key to the successful application of the methodology.

The paper is structured as follows. First, it reviews the relevant literature on crowdsourcing in the context of user studies and then introduces the materials and methods used. We then introduce the proposed approach to evaluating robot programming environments using the Amazon mTurk crowdsourcing platform. We then apply the proposed framework to the evaluation of the *Assembly* web-based robot programming tools. Finally, we discuss the results of the evaluation and draw some lines of future research.

## 2. Related Work: Crowdsourcing in the Context of User Studies

The term crowdsourcing was coined quite far back, in 2006, with Jeff Howes' official definition of it being "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call [10]". Such a network of people is quite distinct from the actual internal employees of a given company since it can be composed of potentially any individual with access to the Internet. The members of the "crowd" may need to fulfil a set of requirements but are largely free to work under their own conditions. Usually, crowd members work alone and do not know the other members of the same crowd. Workers typically receive small remunerations for their work, if the results are acceptable. This way, anyone can outsource any task for the completion of which they do not have internal resources in terms of time and/or expertise.

Based on this emerging phenomenon, many web-based platforms and applications were created to host the solution. In the robotics field, crowdsourcing has received relatively little attention as a technology evaluation method. In this section, we review some of the approaches that use crowdsourcing in the context of user studies and technology evaluation, which reach beyond the field of robotics.

In terms of user experience (UX) evaluation, one can distinguish the following types.

- *Traditional user studies*: These studies are being conducted in the field or in the lab, requiring both study participants and evaluators to come to a certain location. This obviously takes more time to perform and means higher costs, since the required effort is higher, having a lower number of participants as a consequence. Applying this in the context of evaluation of robotic programming interfaces, additional limitations in terms of finding knowledgeable or voluntary study participants, robot access, safety, and platform availability emerge. As the authors of [11] define it, the typical research cycle passes through four main stages, starting with the method formulation, method implementation, conducting the user study, and, finally, presentation of the results. The last step is where the next delay due to lack of time mostly happens, because neither the presented results nor the lessons learned are significant if not implemented into the final product [11].

- *Crowdsourced user studies:* Following [11], in order to progress in robotics, it is indispensable to build professional-grade robots and create user evaluation research appropriate for and available to the masses. Conducting web evaluations seems an obvious answer to this issue, benefiting even more from the use of crowdsourcing platforms.

The authors in [12] conducted an experimental investigation of two distinct crowdsourced design evaluation approaches, one being the usual free evaluation in the crowdsourcing environment and the other using specially developed crowdsourced design evaluation criteria (CDEC). The outcomes of both approaches were finally benchmarked against a standard expert evaluation with a board of experienced designers. It is interesting that the outcome suggests that their CDEC approach produces design rankings with a strong correlation with those of the expert board. Their paper [12] assessment methodology demonstrates how crowdsourcing can be effectively used to evaluate, as well as generate, new design solutions. This paper followed their methodology, which needed to be adapted to the UI evaluation instead of design. Van Waveren et al. [13] have conducted a crowdsourced study aimed at exploring whether non-experts can successfully use off-the-shelf robot programming tools. These authors found that, with minimal instruction, the study participants were able to create simple programs using an off-the-shelf visual programming interface, a conclusion that is in line with other approaches to democratizing cobot programming [14–16].

The authors in [17] tackled the matter of the feasibility of conducting online performance evaluations of user interfaces via crowdsourcing platforms. As the experimental basis, they chose three distinguished, cognitively non-demanding, predominantly mechanical interface design tasks, namely Bubble Cursor [18], Split Menus [19], and Split Interfaces [20].

These tasks were then essentially re-evaluated through three performance experiments, one of which was conducted in a lab and the remaining two over the crowdsourcing platform. Their result analysis did not yield any evidence of significant or substantial differences in the data collected in the two settings, the raw task completion times, error rates, consistency, or the rates of utilization of the novel interaction mechanisms previously introduced in the experiments [17].

In [21], a web interface was created and used to invite crowdsourcing workers to move a mobile robot out of a maze. Crowdsourcing has also been used to test different robot teleoperation and training methods.

The results proved that the filtered image data alone shown to crowdsourcing participants were sufficient to train the robot.

The authors in [22] presented a framework for robot supervision through Amazon mTurk. The goal of the work was to design a model of the autonomous systems building upon asynchronous human computation through crowdsourcing, for which image labeling, object clustering, and the final model selection tasks were created. All of the tasks are confident of substantial human feedback as a strong foundation and an alternative to building a completely automatic system. The authors of [22] also recognized various challenges in the context of algorithm creation, user interface design, quality assurance (QA) policies, and learning. This paper will particularly focus on the user interface design creation challenge.

Another interesting crowdsourcing-based yet somewhat distinct idea of collecting substantial human input over already existing online communities was described by [23]. The author drew upon the immense online social network community and challenged the Twitter community to categorize and define actions performed by humans in a typical HRI interaction. This has a twofold advantage over mTurk, the first one being the fact that it does not cost anything. The second advantage is that the Twitter community can be targeted more straightforwardly because the followers of certain Twitter accounts about robotics customarily have some kind of knowledge or relation to it, therefore making their input more reliable than that of random mTurk workers. The goal of the paper seems very

high-reaching, intending to teach robots to learn and perform without direct instructions from a dedicated user.

The authors of [23] managed to present the potential of social media as a source of data collection, a knowledge base, and information gathering, all of them contributing to faster robot learning opportunities via crowdsourcing. Nonetheless, more complex actions will demand more sophisticated machine learning algorithms. Additionally, for this method to make a difference, an immense number of tasks need to be revised so that the robot has a higher probability of finding itself in a known environment. The final concern is the statistical significance of Twitter users' input since it is on a free basis, and additionally, it is impossible to analyze the crowd impact—specifically, how and if seeing other followers' responses impacts each user's individual response. None of the two concerns are relevant while using mTurk.

Taking into account the fact that the most prevalent and, moreover, natural interaction with robots happens via verbal commands (e.g., the typical home assistant systems, such as Alexa from Amazon [24], Siri on Apple devices [25], or Google Assistant [26] on respective devices), the authors in [27] took such research to another level, using crowdsourcing to collect the significantly large amounts of training data needed to build and train speech models. They asked the workers crowd to watch a short video of an assembly robot—more specifically, a forklift executing an action. The workers were subsequently asked to type a one-sentence command that described the robot's action.

As opposed to this paper, which uses crowdsourcing for fairly simple and mentally non-demanding tasks, there are authors such as [28], who relied on this technique to provide solutions to computationally difficult problems and train algorithms. She proposed taking advantage of the existing large Internet gamer community and channeling their time and mental effort toward different kinds of games, called “games with a purpose” (GWAP). The set of guidelines for GWAP creation serve as the first known general method for seamlessly integrating computation and game-play, but it leaves a lot of questions open and many works to be done in this field [28].

The authors in [29] took GWAP to an even higher level by combining virtual and physical approaches in two essentially distinct domains. The former scenario was realized over a two-human-player online game called “Mars escape” and used to gather a large-scale set of unstructured HRI data, which then served as input to train a physical robot in a real-life museum scenario. The objective was to answer the question of whether the records of human-to-human interaction gathered in a virtual world can be used to design natural and robust human-to-robot interactive behavior in distinctive real-world environments [29], while performing similar tasks.

The results demonstrated similar behavioral patterns in both online and offline scenarios, therefore underlining once again the importance of the crowdsourcing approach in robotics.

The authors in [30] dealt with the question of a new goal-based imitation learning framework that utilizes both physical user input and crowdsourcing as a major source of demonstration input data, on a simple scenario where the robot learns to build 2D object models on a table from basic building blocks. As already discussed in [31], goal-based imitation learning has an advantage over other imitation learning approaches because it focuses on achieving the final task goal instead of solely imitating the human trajectories. This is important because robots obviously have end actuators distinct from the human ones and therefore face different obstacles, which call for different actions to the human ones. Crowdsourcing, on the other hand, has a clear advantage over traditional imitation learning approaches in the context of data and its availability by overcoming the issue of sparse, expensive, and time-consuming human demonstrations.

The results exhibit two proofs—firstly, that the robot indeed can use crowdsourcing data to grasp simple 2D models and, secondly, that the learning effects are better than the ones emerging from completely local user-provided object models. This paper did leave some open issues, such as crowdsourcing QA, the question of whether the framework is

applicable to more challenging models (e.g., more dimensions), more dangerous tasks such as assembly and tool use, and the possibility of self-learning and autonomous decision-taking in judgments, such as whether to crowdsource or ask the physical user for support.

As this paper aims to develop a framework for the usability evaluation of web-based robotic programming environments, ref. [32] designed an entire web-based framework built in a robot-, laboratory-, and interface-independent manner, which was created to greatly reduce the overhead and costs of running remote user studies, which also involve remote control of the robot [11]. In their work, they analyzed metrics across three user study conditions, namely co-present, in-lab, and remote groups. Their findings show non-significant statistical differences between distinctive usability patterns across the groups, which clearly favors the use of web-based crowdsourcing techniques for the considerable fragment of HRI evaluation studies. The most interesting finding is the one that showed no clear advantage of the co-present group, although it was inherent to expect this. Nonetheless, in addition to the disclaimer that the results require quite attentive analysis, ref. [32] discloses the shortcomings of this method by identifying the context in which a robot management system would not be useful, including most of the techniques and studies requiring either physical interaction (e.g., kinesthetic teaching and haptic input addressed in [33]), sensing or observing the user, and studies that have strict time requirements. Another downside of this system is that it requires supervision, whereas “pure” crowdsourcing does not.

Although the authors in [34] did not use crowdsourcing in their methods, their research offers interesting input. They developed a simple programming system for mobile service robots called *CustomPrograms* and based on Blockly [35], allowing both inexperienced workers and experienced programmers to use existing and design new commercially useful and easily deployable use cases. Additionally, they analyzed and classified the errors, distinguishing between minor and major ones, the former being easy to spot and fix and the latter representing an obstacle for the correct program execution. Despite the expected outcome of more experienced workers delivering faster and more correct output, the paper succeeded in proving their twofold goals, which are also relevant for this thesis:

1. possible and relatively simple to use for an inexperienced user programming interface;
2. complex enough for more advanced users to allow for developing more complex or completely new use cases.

One of the interesting takeaways for the future research and something that our *Assembly* is already trying to implement is the concept of the “copy and paste literacy” model of end-user programming [36], which would basically allow more experienced users to program templates and/or more complex cases, which can then be reused and, if necessary, customized by less experienced ones. The downside of their paper is that *CustomPrograms* was developed as proprietary software exclusively for one robotic hardware, although the authors of [34] claim that it could be easily adaptable. *Assembly* is implemented so that it can connect to all robots.

### 3. Materials and Methods

This section introduces the tools and methods used in the development and evaluation of the proposed approach.

#### 3.1. Assembly

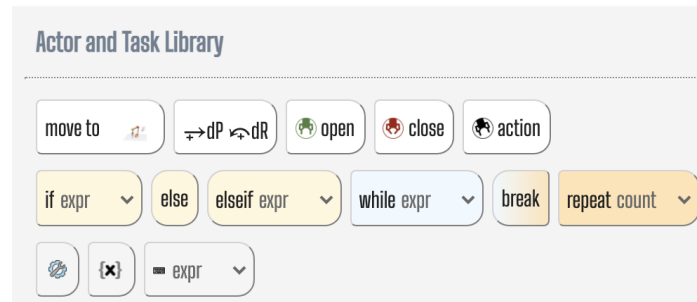
*Assembly* [37] is an open-source, simplified, web-based robotic programming environment developed as part of the CoMeMak project.

*Assembly* UI programming concepts rely on five main building blocks:

1. Actors (Blocks)  
*Assembly* implements block-based actors. As visible from Figure 1, most of the basic programming concepts already exist. The first row consists of the actors who are meant for precise movement of the entire robot (move to), adjusting the robot’s

gripper position (dp, dr), opening and closing the gripper, and defining a dynamic action. The second row consists of basic conditional statements, such as if, else, elseif, and also loops such as while and repeat. The final row is meant for explicit parameter, variable, and condition setting.

*Assembly's* core idea is to simplify block creation, use, and reuse by standardizing them.



**Figure 1.** *Assembly* actor and task library.

Additional help is offered to the end user via the “help option” (Figure 2). It can be easily accessed merely by clicking on each actor and appears as follows:

#### ▮ moveTo

Moves the robot to the current pose, as reflected by the position and rotation parameters in the robot control area ▮. This actor uses the following parameters:

- **Robot.tcp.[x, y, z]** → Tool center point (TCP) coordinates in Cartesian space
- **Robot.tcp.[rx, ry, rz]** → TCP rotation around the respective (Euler angles)
- **Params.motion.[acceleration, velocity]** → Movement acceleration and velocity

**Figure 2.** *Assembly* help option.

## 2. Blackboard

The blackboard is a behavioral design pattern used in computer programming, which is used to prioritize the actors of different systems that are bound to work either in parallel or sequentially in certain programs [38]. In *Assembly*, this concept is used to ease the creation and management of the actors.

As visible from Figure 3, the *Assembly* blackboard uses the following three objects:

- (a) “Robot” is a read-only object containing the state of the robot, including the joint angles and the coordinates of the end effector.
- (b) “Parameters” object contains a list of parameters used by the actors from the actor library. Only their values can be modified; the object’s name or structure cannot.
- (c) “Variables” object is editable by end users, and the input is then used for the conditional and loop enabling statements.

The “Parameters” and “Variables” objects can be edited by users using an embedded JSON editor [39].

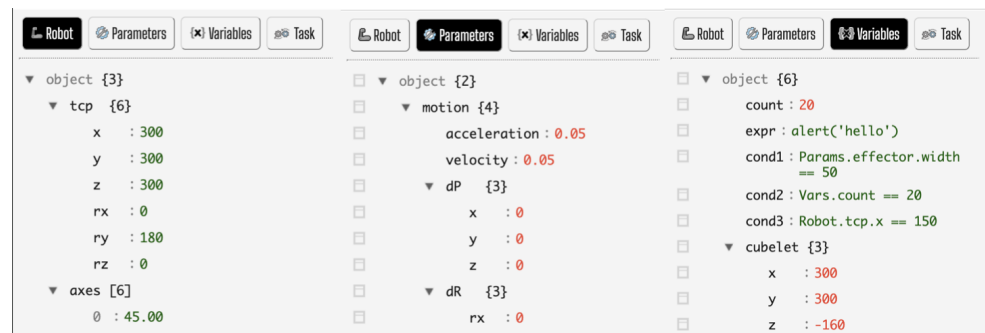


Figure 3. Assembly blackboard.

### 3. Tasks

A workflow can be created by linking a series of actors together. This workflow can then be easily saved via a bookmarking option and subsequently reused as a standalone task or part of another task. Creating a new task is as easy as dragging and dropping necessary actors between the start and stop buttons, automatically triggering the saving of snapshots of the “Parameters” and “Variables” objects. When loading a task, the content of the global “Parameters” and “Variables” objects is overwritten by the content of the saved snapshots.

When loading a bookmarked task, it will appear as an additional task actor in the actor and task library region. To create compound tasks by reusing, it is essential to load each task individually. The best practice suggests saving an empty task, loading it, and subsequently loading all the other saved tasks within it. This will be the final use case scenario.

The intention behind task creation is threefold:

- (a) simplified creation and organization of the robot programs within the browser;
- (b) reusing tasks via aggregation;
- (c) sharing with other users and exporting tasks effortlessly as browser bookmarks.

### 4. Simulator

*Assembly* uses an adapted version of a six degrees of freedom (DOF) robot simulator by [40]. The simulator is embedded in the right-hand side of the *Assembly* environment. The “Robot” object contains the axis angles corresponding to a certain tool center point (TCP) position. The robot can be controlled either using the text input boxes on the upper right side of the page or using the mouse. The TCP needs to be clicked to move the robot around. This currently only works on Windows systems. The exact position of the robot’s TCP is updated in the controls area and the “Robot” object on the blackboard:

A snapshot of each robot pose is stored as a thumbnail within the “move to” actor. When used in a workflow, this actor will move the robot to this pose. This way, different poses can be stored by first dragging the robot to the desired position and then dragging a “move to” actor to the workflow. This corresponds to a teach-in procedure, which is a common method of programming robot behaviors.

### 5. Robot

*Assembly* is set to be used both for online and offline robot programming. In the former case, the generated *Assembly* code will be sent to a web service, which will then trigger low-level robot handling functions. Currently, only Universal Robots [41] are supported. Connecting to the robot is relatively straightforward and is done by activating the “Connect to robot” switch in the upper part of the simulator. Connecting to an actual robot is beyond the scope of this paper.



### 3.2. Amazon mTurk

Owned by Amazon and operated under Amazon Web Services (AWS), mTurk [42] was launched in 2008 and is currently one of the largest operating micro-task sites [43]. It consists of numerous listed custom webpages nested within the platform, each of them consisting of a unique task. Some tasks may require workers to interact with web pages outside of the mTurk platform. Tasks are called “human intelligence tasks” (HITs) and are issued by unique users registered as so-called requesters. A requester can issue any number of different or the same tasks and can set certain filters to only allow suitable workers access to them. In order to become an official requester, typical account registration steps need to be carried out. The first one needs to be authenticated by providing the full name and additional institution and how often one plans on using the platform. After this, a credit card needs to be added and subsequently verified by sending and receiving a certain small amount of money. The entire workflow is depicted in Figure 4.

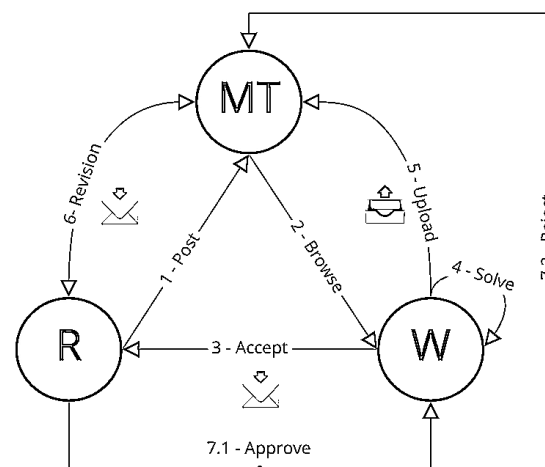


Figure 4. mTurk process workflow based on [12].

1. Starting from the lower left circle, marked with “R” for “Requester”, the work is published on the crowdsourcing platform in the form of a HIT. Requesters are usually companies, research institutions, or any other individual offering the digital task.
2. mTurk [42] (marked with “MT”) is the crowdsourcing platform of choice, which hosts and shows all available assignments to all potential mTurk workers. mTurk grants the “R” high-level control of who can see or accept their tasks by keeping a set of metrics for each registered worker. There are three system qualifications available free of cost:
  - (a) Number of HITs approved  
This is the number of HITs that one worker has successfully completed since their registration on mTurk. This is the base rate that requesters use to select workers with existing experience. According to [22], higher than the 90% mark provides a good rule of thumb.
  - (b) HIT approval rate  
This is the rate of accepted and completed HITs. For example, if a worker has completed 1000 HITs, out of which 150 were rejected, the worker’s approval rate is 85%.  
Experienced requesters report that the combination of these two qualifications usually delivers results of significantly higher quality.
  - (c) Location  
This qualification is important if a requester needs to filter workers by geographical location, which the workers usually specify in their profile. The location filter can be used to include or exclude workers from a specific continent, country, county, or even city.

Additionally, the possibility of setting up two premium or entirely customized qualifications exists. These kinds of qualifications induce additional charges and include age, gender, frequency of participation in the platform, or the type of smartphone owned by workers (important for smartphone software usability testing) [42].

3. To complete the first cycle, the mTurk workers (marked with “W”), on their own initiative, browse and decide to accept the offered assignment. “R” is notified about every assignment acceptance or rejection.
4. “W” starts solving the assignment immediately or according to the given time frame.
5. Subsequently, “W” uploads its results to the platform.
6. “R” is notified about the uploaded solution and has a limited, previously arranged, timeframe for revision.
7. The requester then has two options:
  - (a) In case of approved assignment, “R” sends the agreed payment to “W”, with the non-obligatory possibility of offering and later paying the bonus, and also pays the fee to “MT” for providing the platform.
  - (b) The other possibility is rejecting the assignment, which can only be authorized by “MT” if a valid argument for doing so is offered.

Next on the list is HIT creation. For this, numerous mTurk templates already exist. However, they are adapted to different use case scenarios (predominantly surveys, image or language recognition, or data collection templates). This meant choosing a customizable template sorted under “Other” and adapting it as well as possible to this paper’s needs.

#### 4. Crowdsourced Evaluation Methodology

The two most important techniques for evaluating a UI are heuristic and empirical evaluation [7]. The former represents a theoretical basis built upon a predetermined set of rules aimed at improving the usability of UIs. The latter is more practical and implies testing with real users. This paper reports on user studies conducted via crowdsourcing for the purpose of evaluating the usability of the *Assembly* programming environment. This first evaluation explores the potential and limits of the proposed methodology by showing how usability evaluations can effectively be performed using this method.

Nielsen [7] further differentiates between summative and formative usability evaluations. Summative evaluations produce quantitative, predominantly statistical data. Besides backing our findings via comparative analysis or performance measurement tests, quantitative data are useful, e.g., for identifying and rejecting outliers. In the case of crowdsourcing, this is likely caused by scammers. The results that are more than two standard deviations from the mean are usually excluded in human–computer interaction research [17]. On the other hand, the results of formative usability evaluation contribute via qualitative answers on how and why interface users experience a certain design or the entirety or chunks of the process flow. For this paper, both summative and formative usability evaluations are considered equally relevant and will be carried out.

For the purpose of evaluating *Assembly*, six experimental tasks have been designed and published on mTurk. Drawing on [17], the tasks progress from elementary mechanical ones, such as moving the robot in any direction, which demands a small amount of mental effort and concentration, to more cognitively and time-demanding ones, such as creating an algorithm.

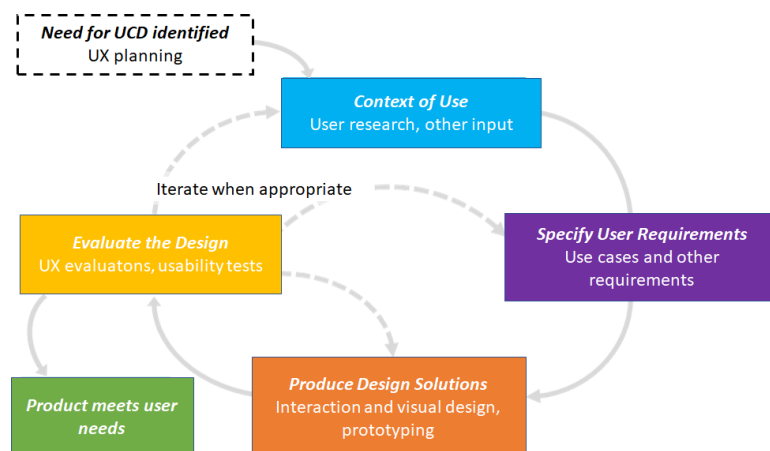
##### *Methodological Framework*

The basis for our framework is the iterative user-centered design (UCD) process for interactive systems described by the ISO standard 9241-210 [44].

As is visible from Figure 5, the following steps specified by the standard can be adhered to and re-iterated if necessary [44]:

1. Firstly, the need for a specific UCD task is identified through input and requirements from various stakeholders. In parallel, the work on the UX is planned considering time and other resources. This paper had a clear UCD task based on *Assembly*.

2. Secondly, the context of use is specified based on user background research or other input. Since the main goal of the CoMeMak project is ensuring all interested stakeholders access to cobots in makerspaces and additionally addressing other issues regarding robots, the context of the use of *Assembly* here is defined rather broadly, with user profiles being diverse and unspecified. The projected users of the *Assembly* tool should have basic computer skills and technical understanding. These prerequisites are also required of mTurk workers in order for them to be able to navigate the platform.
3. The third step handles user requirement specification. It is suggested to define and document relevant use cases in order to model the specifications. The use cases should be initially simple to allow the user to get to know and use the program and then gradually upgrading the complexity. The business relevance should not be disregarded here. One strategy is to use simple tasks as a part of the assessment test to encourage only qualified users to participate in the entire crowdsourcing project and later raise the difficulty bar. If every participant can solve all the tasks, then the use cases should be made more difficult; otherwise, the results will not be within the scale.
4. The fourth step delivers design solutions, such as UX concepts, tutorials, interaction evaluations, and prototyping.
5. Finally, the UX design is evaluated by testers or real users. This paper will focus only on crowdsourced results. Normally, if the users' design or functional needs are met, the entire UCD process is done. Otherwise, one or several iterations of one or more of the previous steps are necessary.
6. The end result should be a ISO 9241-210 standard [44] approved product.



**Figure 5.** UCD process according to the ISO 9241-210 standard [44].

The ISO standard 9241-210 was implemented as a strong foundation and enriched by the framework drawing on authors in [12]. The combination of the two frameworks generated a specific set of steps tailored to the task at hand, which is shown in Figure 6. This detailed process arguably provides a generic framework for the complete process of the crowdsourced evaluation of web-based programming environments. Using the color code in Figure 5, each extension of the original ISO process is traced in the detailed workflow models from Figure 6. Based on the top-down approach, our methodology is divided into four main stages, which are further refined into more detailed sub-stages where necessary. A complete description of this model is provided in Appendix A.

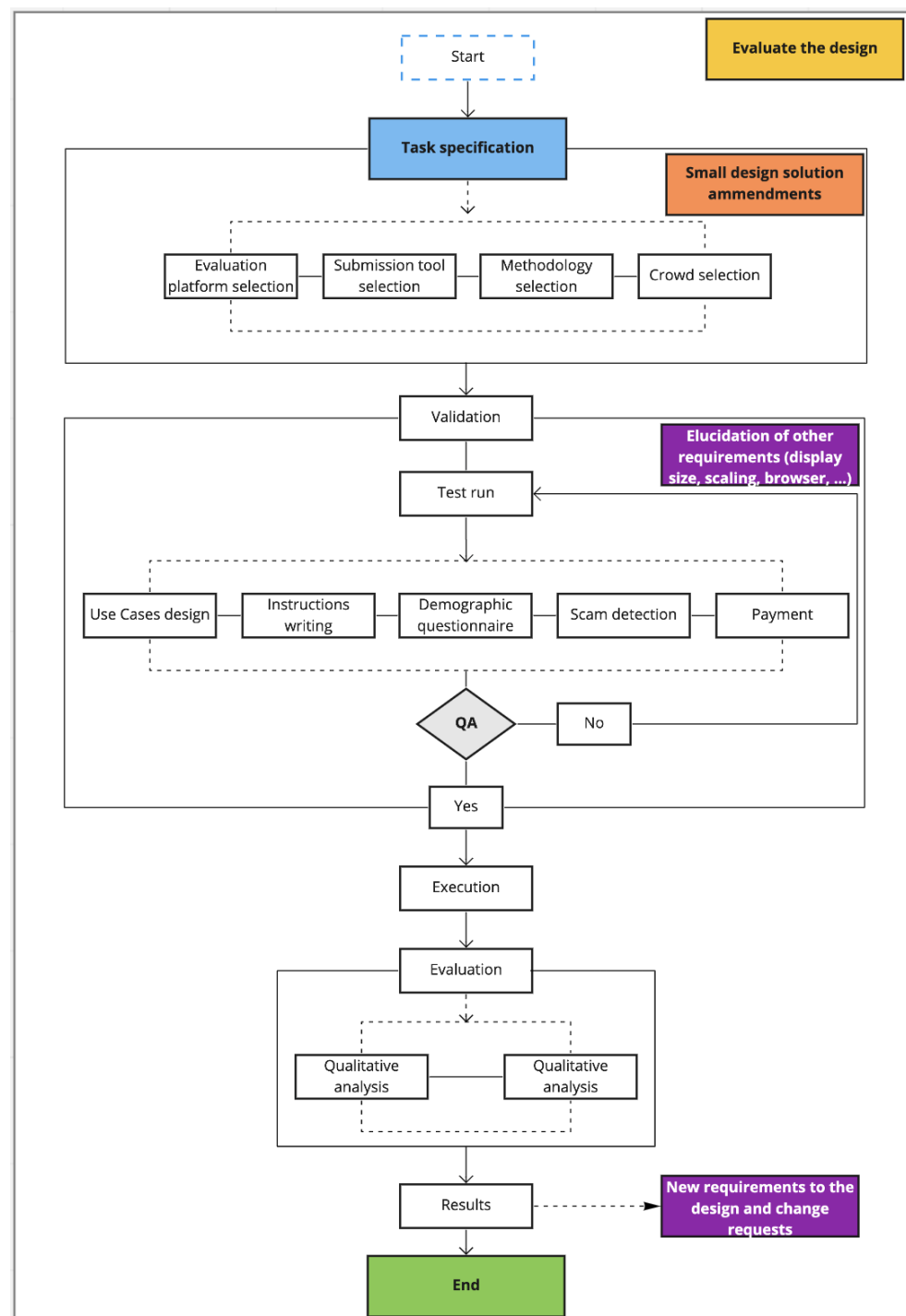


Figure 6. Crowdsourced evaluation methodology framework based on and adapted from [12].

### 5. Application of the Framework

In this section, for the purpose of validation, the proposed methodological framework will be applied to the web-based *Assembly* robot programming environment step by step.

#### 5.1. Task Specification: Evaluation of Web-Based Robotic Programming Interfaces

This paper focuses on web-based robotic programming environments, by the example of *Assembly*. Web-based robot programming tools are still rare, yet *Assembly* is not unique. For example, *drag&bot* is another web-based robot programming environment developed by a German startup (a spin-off of the Fraunhofer institute). This usability of simplified robot programming environments has practical relevance because it deals with the daily

routines of shop floor employees. The proposed framework aims to improve the usability of such tools with the aim of rendering robot programming more accessible to a broader and more diverse public in factories and elsewhere.

#### 5.1.1. Evaluation Platform Selection: Amazon mTurk

We chose Amazon mTurk as the crowdsourcing evaluation platform because it has already been used in web interface evaluations. Other platforms could have been adopted too.

#### 5.1.2. Submission Tool Selection: *ufile.io*

The appropriate methods for asserting the proof of work need to be established. They can be implemented as a generic questionnaire or by inserting questions into the task that can only be answered after a certain work step has been performed. Screen recording provides an additional option, but we opted not to use it in order to investigate the scamming phenomenon. For this paper, we opted for using file uploads over the *ufile.io* environment [45].

#### 5.1.3. Methodology Selection

For this task, the basic, non-iterative (or linear) competition approach was used.

#### 5.1.4. Crowd Selection: Worker Profile

For the online variant of the study, we recruited 117 mTurk participants from various countries. The HIT approval rate for all submitted HITs was set to higher than 90%. We imposed no age or gender restrictions.

### 5.2. Validation

There was no automatic validation system, so all the input had to be checked manually with a focus on:

1. number of submitted files;
2. content of submitted files;
3. content of the open-ended answers;
4. minimum time needed to finish the study;
5. overtaking placeholder values.

#### 5.2.1. Test Run

Before publishing the HIT, we worked with the mTurk developer requester sandbox, which is a simulated environment, where requesters can test their HITs prior to official publication in the marketplace. This is a free and useful feature provided by mTurk, allowing all requesters to test, practice, and redesign their surveys so as to make sure that all the tasks function correctly in advance. This saves time and money in preparation for the actual evaluation. In the case of evaluating *Assembly* without a fixed framework, we needed exactly 5 test runs, all of which yielded important findings and lessons learned. The test runs were executed in the period between 8 December 2020 and 30 January 2021, offering fees in the range of USD 0.80 to USD 3.00.

The importance of creating simple yet relevant use cases in order to obtain the user feedback was recognized from the first test run. Another immediate takeaway was the importance of creating required fields, since most of the workers left every optional field empty. Additionally, the significance of setting both overall and individual, clearly addressed and specific, evaluation criteria was recognized, reducing a Likert scale from 10 to 5 since it takes users a long time to fill out while it is not even the most important or reliable measure.

Reading user feedback on this test run pointed to some required UI adaptations so that the additional explanations about how to use the tool or the necessary time for solving

the HIT could be reduced, which in turn reduces the likelihood of workers giving up on the HIT with the argument that the program does not react as expected.

For the 3rd run, the same use cases were used as the ones in the TU Wien course “*Montage II: Advanced manufacturing*” [46]. As expected, these use cases were too complicated for the mTurk audience, which additionally had the disadvantage of no template solutions shown and a lack of expertise. Nevertheless, one worker managed to solve all the tasks, including the open-ended question, which asked the user to think of a way to solve a particular task rather than providing step-by-step instructions. Although this unexpected success is statistically insignificant, it demonstrated that it is possible to encounter skilled workers on mTurk. This run made clear the need to reduce the amount of data needed to provide a proof of work considering workers’ comments and the time required to check the quality of the results.

In the following test run, the *pick* task was decoupled from the *place* task to check which of them was easier for the mTurk workers. Moreover, the need to introduce standardized usability measurements became apparent in order to analyze the results more easily. The first choice was the Post-Study System Usability Questionnaire (PSSUQ). With 16 questions that can be answered with a 7-point Likert scale (N/A option) and three subscales evaluating system usefulness, information, and interface quality, this fine-grained usability test turned out to be too long for mTurk workers to fill out correctly and honestly.

The next run implemented the System Usability Scale (SUS) usability survey instead of the PSSUQ. Concerning the use cases, the *pick* task turned out to be easy to solve, so it was immediately expanded with the *place* task (supposing the same difficulty level) and seamlessly extended with the *reuse* task.

### 5.2.2. Use Case Definition

It was necessary to keep in mind the low payment, small amount of time on disposal, low attention span, no moderating possibilities, and general complexity of the topic. When defining the use cases, the requester must keep a series of constraints in mind concerning the relatively small payment and short time that workers dedicate to a hit as well as the short attention span of the majority of worker and the limited possibilities of moderation. These constraints must be placed in balance with the complexity of the task so as to obtain the most valuable output from as few use cases as possible. To achieve this goal, a gradual increase in use case difficulty is necessary. In doing so, the solvability rate (i.e., how many participants can solve the most complicated task) must be monitored. It is to be assumed that not all workers will be able to solve all use cases. However, the tasks and post-task surveys must be designed in such a way that the workers will be compelled to provide reasons for not doing so. Rejected HITs are just as important as the approved ones because they can provide useful insights, which can be used to adjust the difficulty of the tasks.

According to the International Federation of Robotics, *pick and place* is the most common application scenario for industrial robots. *Pick and place* is used in tasks such as loading and unloading machines, placing parts in a grid, or picking bulk material from a box with or without a camera support. The aim of the use cases was to start from basic tasks, such as moving a robotic arm in a certain direction, to more complex tasks, such as *pick* and *place*.

Based on these principles, six use cases have been designed in order to evaluate the efficiency and usability of *Assembly*, which were included in the HIT instructions.

### HIT Instructions

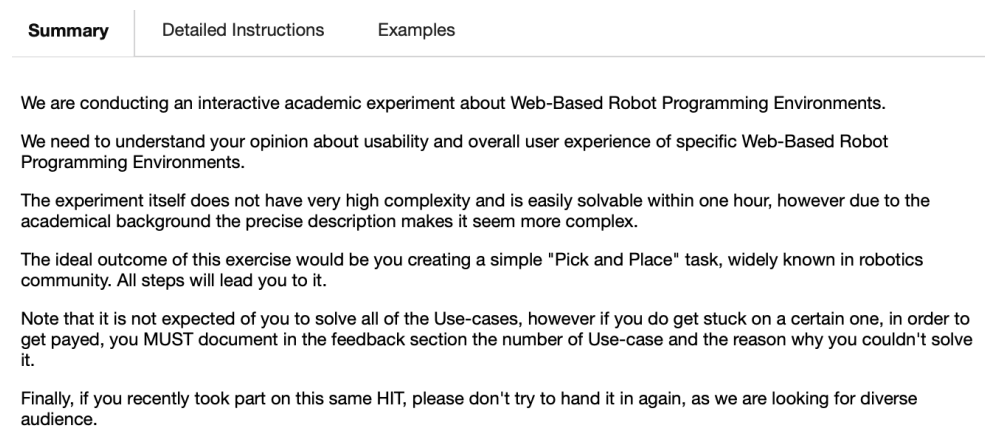
The step-by-step textual instructions, which did not contain any images, explained how to use the robot simulator and how to manipulate objects. This text could be reviewed at any point during the user study. During task execution, no means of support were offered. If a technical issue with mTurk occurs, the Amazon team is responsible for its resolution. Any issues with *Assembly* encountered by the workers were likely to provide

useful insights about the program being evaluated and were documented in the qualitative analysis part.

The *Assembly* tool did not provide any indications on which workers were supposed to deliver in each HIT. Moreover, explicit success or failure messages were not implemented in the interface. This deferred the decision concerning whether a certain HIT was completed or whether they should continue to the next use case to the worker. Although efforts were made to keep the instructions concise and informative, some workers did not follow them.

As visible in Figure 7, there were 3 tabs to be completed by the requester, namely *Summary*, *Detailed instructions*, and *Examples*. Workers were provided with the following instructions in the second tab:

Make sure you have a computer that supports the Google Chrome browser, and please use it for this survey. Before starting to solve each and every use case, make sure you click on the dashboard icon on the lower right part of the screen to show automatically generated JS code. Please read each use case carefully and remember to take a screenshot accordingly. (The following sentence was set as a reminder after every use case description: Make a screenshot of the current screen (with the visible code on the dashboard) and upload it.



**Figure 7.** mTurk summary tab text.

### Use Case Instructions

The program is drag-and-drop-based. The units used for the (x,y,z) coordinates are millimeters. The units used for the rotation angles along the spatial axes (RX, ry, rz) are degrees. By the workflow, the task line on the top starting with “start” and “stop” actors is meant. Useful hint: note that when you put your mouse over any component, a small “Help” window will appear.

#### Use Case 1: Robot Motion

- Create a program that performs a robot motion between two points.
- The first point should be set to  $x = 100$ ,  $y = 200$ ,  $z = 300$ .
- Subsequently, drag the “move to” actor to a location between the “Start” and “Stop” actors in the workflow (i.e., robot task or program) to create/memorize the first waypoint.
- You can freely choose the second waypoint, but it has to be different from the first one.
- Then, add another “move to” actor to the workflow to create/memorize the second waypoint in the program.
- Repeat the motion a few times by clicking the play button.
- Note the correct order of successfully creating instructions.
- Do not refresh or close the browser tab, as you will need the current state for the next use case.

#### Use Case 2: Robot Motion in a Loop

- As you could see in the previous use case, you could repeat the movement manually by repeatedly clicking on the start button.
- Now, you want to automatize this by creating a simple programming loop. For this, you will use the repetition block called “repeat count”.
- Note that, when added to a workflow, the “repeat” block has two parts—a head part (a yellow box with the text “repeat count” on it) and a tail part (empty yellow box).
- Put both of the previously created “move to” actors between the head and the tail of the “repeat” actor simply by dragging and dropping them at the corresponding place. This actor uses a variable named “count”. Open the variables panel (i.e., “xVariables”) and set count = 3.
- Make sure to store this value in the program by adding the “Set variables” (i.e., “x”) actor to the workflow before dropping the “repeat” block.
- Now, click on the play button. Both movements should now be repeated three times.

#### Use Case 3: Creating a Placeholder Task

- You are now familiar with the basic robotic arm operations.
- Refresh the browser window to start with a clean task. Please don’t skip this step.
- Then, simply rename the task from “Untitled” to “mturk” (double click) and save it by dragging the disc icon to the browser’s bookmarks bar to save it.
- You will need it in the final task.
- *No screenshot of this screen is necessary.*

#### Use Case 4: Creating a Personalized Task

- Refresh the browser window to start with a clean task. Please don’t skip this step.
- You will now teach a robot to pick up the small blue cube. Set it to the following position:  $x = 300$ ,  $y = 300$ ,  $z = 89$  and add the “move to” actor, same as in the 1st use case.
- Now, add the “close” actor to the workflow to close the gripper and grab the cube. Then, drive the robot to  $x = 300$ ,  $y = 300$ ,  $z = 300$ , and add another “move to” actor.
- Finally, rename the task from “Untitled” to *pick* and save it by dragging the disc icon to the browser’s bookmarks bar.
- Run the task to *pick* the blue cube up.
- You have now created your first example of a personalized task.

#### Use Case 5: Building Upon Personalized Task

- Refresh the browser window to start with a clean task.
- Now, drive the robot to a “pre-place” location at about  $z = 200$  above the table, then down to the table level ( $z = 89$ ), and use the “open” actor to the workflow to release the gripper.
- Then, use the “relative motion” actor ( $\rightarrow dP < dR$ ) to drive the robot back up to  $z = 200$  (the relative motion actor uses  $dP$ ,  $dR$  parameters).
- So, select the parameters tab and set  $z = 111$  so as for the robot to go to  $z = 200$ .
- After setting this parameter, drag and drop the “set parameters” actor into the workflow before the “relative motion” ( $\rightarrow dP < Dr$ ) actor.
- Rename the task to *place* and save it to the bookmarks, just as you did with the previous tasks.

#### Use Case 6: Building a Compound Task

- Refresh the browser window to start with a clean task.
- Now, load all of the previously saved tasks from the browser’s bookmarks, in the following order: *pick*, *place* and “mturk”.



- You will notice that they now appear in the “Actor and Task Library” and that you are currently in the “mturk” task. This allows you to load a fresh task without losing the *pick* and *place* tasks from the library.
- Finally, create one last task called *Pick and place*, merely containing the *pick* and *place* tasks as actors, and save it to the bookmarks bar.

### 5.2.3. Instruction Writing

In the *Summary* tab, a brief general explanation about the study was given:

This explanation was underpinned with concrete examples in the last tab, as suggested by the mTurk template:

- Read instructions carefully!
- Solve as many use cases as you can, documenting thoroughly in the feedback section if you got stuck somewhere.
- Before starting to solve the use cases, make sure you click on the dashboard icon on the lower right part of the screen.
- By doing so, the generated code will be shown in each screenshot you’re about to make.
- Make a screenshot after EVERY solved use case and UPLOAD it using [45].
- Make sure to refresh the browser after completing a program when you are asked to. You should upload 5 screenshots in total.
- Solve questionnaire.
- Skipping or not solving any use case without documenting the reason or missing 5 screenshot uploads will unfortunately result in a rejected HIT. Note that it is perfectly fine if you can’t solve one of the use cases but you MUST document the reason WHY you couldn’t finish it in the textual feedback section.
- Not refreshing the browser between the tasks when prompted will result in wrongly generated code.
- Not following all the instructions will result in not receiving credit for accomplishing our interactive experiment.

### 5.2.4. Demographic Questionnaire

For a better statistical evaluation, the following demographic information questions were asked at the beginning of the HIT:

- What is your age/gender/country you live in/highest finished level of education?
- Which university did you attend (university name + country)?
- On a scale of 1–5 (1 being the least and 5 the most), how much familiarity with programming/robotics do you have?

### 5.2.5. Scam Detection

A relatively large number of spammers who deliberately break the mTurk rules are active on the platform. In our case, scammers uploaded empty files and filled out the required fields only with sometimes completely random text. As [22] notes, a part of these workers are spotted automatically by analyzing the metadata associated with their submission. If requesters report specific workers that provided multiple false submissions, mTurk declares them as spammers and blocks them so that any future submissions by identified scammers are rejected automatically.

At the beginning of this work, we expected all workers to be human. However, this quickly turned out to be incorrect, as some submissions seemed to be automatically generated by software robots. A detailed analysis of the differences between a human and a robotic submission, however, is beyond the scope of this work. For the purpose of obtaining usable results, we performed an outlier detection analysis, while assuming that scammer submissions would be lower in number than usable submissions and that the scammers’ answers would differ significantly, both qualitatively and quantitatively, from the other participants’ answers.

We performed outlier detection based on five measurements:

1. per-participant minimum and/or maximum processing time.
2. the question about the attended university as an adaption of the so-called “instructional manipulation check” (IMC) from [47]. An Ivy League university (Harvard) was set as the default placeholder answer here. The idea was that a worker who either did not read the question properly or used a software robot that solved the task would simply overtake the placeholder text. The assumption behind this test is that someone with a Harvard University degree is unlikely to be active on Amazon mTurk.
3. the presence of arbitrary text in required free-text fields.
4. the presence of fake uploaded files or no uploaded files at all.

#### 5.2.6. Payment

During the test runs, we experimented with different price ranges and finally decided that USD 5.50 per HIT was attractive for higher-rated workers and comparable to the rewards used in traditional usability evaluation methods. The goal of our study was not to lower the reward of participants but to reduce the preparation efforts for the researchers conducting the user study.

#### 5.2.7. Test Runs QA

As described earlier, altogether, five test runs were carried out, each of which provided some useful insights, which were used in the subsequent test runs as well as in the actual study.

### 5.3. Execution

The final run was performed in March 2021, offering USD 5.50 as a reward per assignment. UI errors found by this moment were corrected. This run had detailed instructions and explanations of 6 clearly defined use cases that were to be evaluated. The usability evaluation method was integrated and individual results were to be uploaded via the approved upload tool.

Even though all the elements of the final HIT were correctly defined, a rerun was necessary. All HITs have a “task expiry date”, which is basically the maximum amount of time for which each task will be available to workers on mTurk. However, independently thereof, we inferred from the experience from all the tasks that the peak in HIT acceptance is within the first three days. Interestingly enough, it is not immediately on the first day that most workers are approved, since the results showed that scammers are the fastest to accept the assignment. This phenomenon will be analyzed in more detail later in a section dedicated to fraud detection.

#### Test Setting and Procedure

As opposed to a laboratory or similar setting, which reduces all possible disturbances to a minimum in order to gain the most trustworthy results possible, the present study was conducted without any of the constraints of a laboratory setting. When using an online crowdsourcing platform, it is almost impossible to control the setting where the participants are located. Unlike video-filmed exams, which became common practice during the COVID-19 pandemic in universities worldwide, the participants in a user study cannot be controlled in this way because they are allowed to use any materials and take any amount of time to complete the test. It is also permitted for more than one person to act as a single participant.

Another argument in favor of an uncontrolled, crowdsourced user study is that, we believe, laboratory studies yield somewhat idealized results, where participants can also obtain immediate help from moderators. In the case of robot programming, this does not reflect reality, since such tools are likely to be used in factories and other real-life environments, where individuals do not always receive all the support they need. In this spirit, the participants in the current user study were asked to create what felt like a typical

working environment for them and to try to finish the tasks with as few disruptions as possible. Any unforeseen disruptions were included in the analysis of the results, which impacted the conclusions of the study.

#### 5.4. Evaluation

The evaluation method used in this work combined quantitative and qualitative methods. Quantitative methods were used to provide measurements of numerical metrics, such as the amount of time needed to handle certain tasks. Quantitative methods were used to prepare and evaluate the answers and results (e.g., files) produced by the participants.

A qualitative research approach was used in the design and documentation of the use cases as well as in the analysis of the test output and the observation of all side effects and events.

- **Qualitative analysis**

After the finished HIT, the following questions were asked per use case:

- How understandable was this task's description?
- Could you easily understand what was asked of you in this task?
- Did you manage to solve this task? (Yes/No)
- How difficult did you find this task? (Likert scale 1–5)
- If the response to the previous question was between 3 and 5: What was the most difficult thing about this task?
- Do you think this UI has improvement potential?
- What is one (any) thing you would like to change?

- **Quantitative analysis**

Crowdsourced evaluation works hand in hand with quantitative testing because it automatically measures the performance of the design and tracks usability metrics easily since the data are being recorded automatically [48]. Predefined SUS questions were as follows:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

#### 5.5. Results

The final results became available after the predetermined number of acceptable HITs were solved (i.e., more than 25 for a typical user study). Each of the 119 results obtained during the final run was checked by the researchers and only qualitatively satisfying ones were accepted. In order to analyze the results in more detail, they were divided into accepted and rejected HITs.

## 6. Study Results

In this section, we will first take a short look at the results of the pilot user study conducted with 12 students from the Vienna University of Technology, before discussing the overall findings relevant across all the tested groups. Then, the analysis will mostly focus on the accepted HIT results, accompanied by an analysis of the rejected ones.

### 6.1. Student Study Results

Prior to the execution of the mTurk study, a pilot study with a small cohort of technical university students was performed as part of an elective Master's course called "Advanced manufacturing" [46].

The main goals of the pilot study were to detect and solve as many potential issues as possible while intending to clarify and polish the research questions and use cases, while gathering data on the time and cost of the crowdsourced evaluation study. The pilot study was conducted with seven students enrolled on the course, who agreed to complete the questionnaire.

The students rated *Assembly* relatively highly. An average SUS score of 65 places the result at the beginning of the marginally high acceptability range, with a score of 68 lying above average.

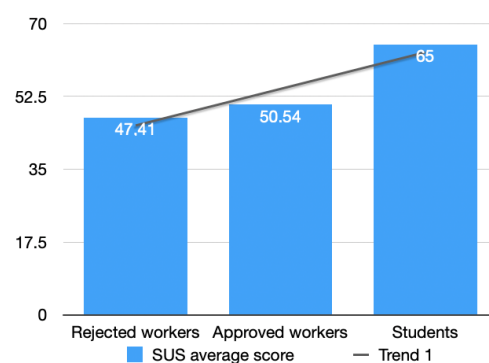
Considering that the participating students had already accomplished a basic programming course, all of them could master all the tasks. The issues that were signaled concerned the look and feel of the tool—notably, the way in which tasks could be saved as browser bookmarks. Although the students had no significant issues with solving the tasks, they did not rate the tool very highly, which suggests that their rating was not biased by the course setting. Their final course evaluation did not depend on the number of accomplished tasks or the SUS scores in their surveys.

The evaluation with students helped to clean up and decide on adequate use cases based on their ability to solve the tasks and the ideas from students' input.

### 6.2. Statistical Findings across All Groups

This section focuses on general statistical findings across all mTurk worker groups, which we did not find essential to observe separately per group. Furthermore, direct comparison between them is what makes these findings valuable. Note that we did not have such data for the pilot study.

Figure 8 depicts overall SUS scores for the student results, the accepted mTurk worker results, and the rejected mTurk worker results. The graphic shows a difference of almost 20% between the results of the rejected HIT workers and those of the pilot study. At the same time, the difference between accepted and rejected HIT workers was comparatively small, amounting to only 3.13%.



**Figure 8.** SUS scores across all groups.

This point raises at least two interesting questions:

1. Could the SUS evaluation results be considered relevant even if they originate from mTurk rejected worker's input?
2. Is there an observable pattern sample in rejected workers' SUS evaluation results that can lead to early scammer recognition?

The first question is beyond the scope of this paper. The answer to the second question, which will be discussed later in the paper, also provides a partial answer to the first one.

The average HIT approval rate is rather low, at 24.79%. This is something that needs to be considered when opting for crowdsourcing. The approval rate also implies a certain amount of time that needs to be invested in making the acceptance or rejection decision for each HIT. An individual analysis of each HIT is unavoidable if the researcher wants to make sure that all the input is valid.

Figure 9 focuses on the comparative difficulty level rating for accepted and rejected HITs. In terms of difficulty, the rejected workers rated all of the use cases more highly than the approved workers. N/A answers were considered in this rating as well. In our opinion, this measure is directly linked with the participants' previous experience with programming in general and, more specifically, with robotics. However, analyzing worker inputs on these two criteria turned out to be contradictory, since the rejected workers claimed to have considerably more experience in both of these fields, as is visible from Figure 10. In our understanding, this is likely to be the result of workers' dishonesty.

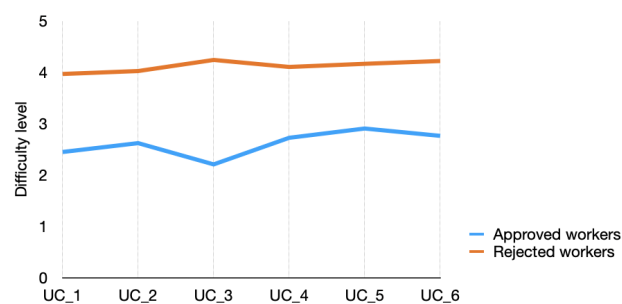


Figure 9. Use case difficulty evaluation among approved vs. rejected workers.

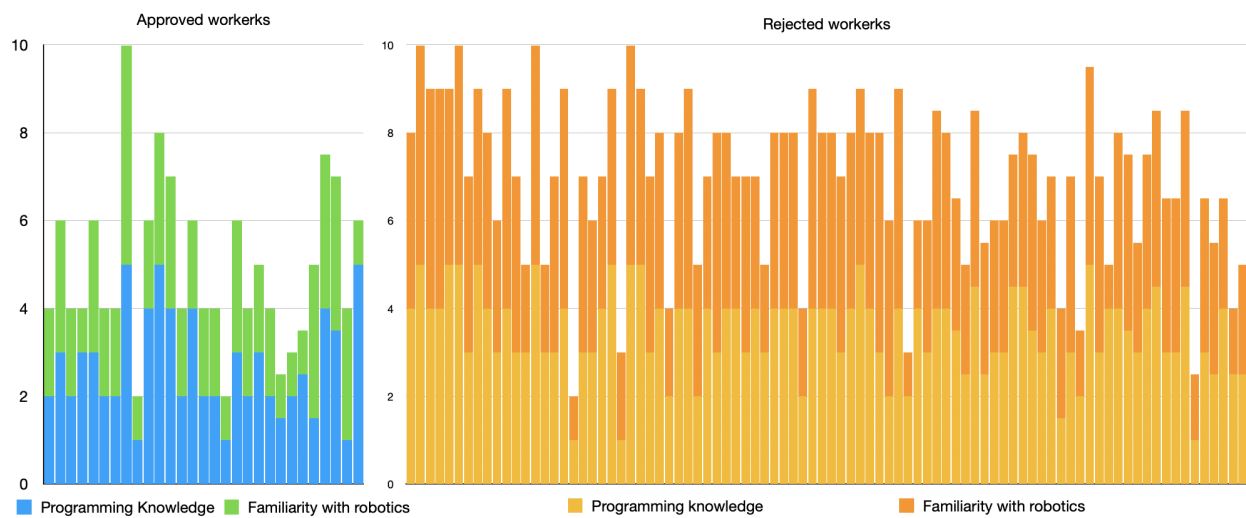


Figure 10. Previous knowledge in programming and familiarity with robotics among approved vs. rejected workers.

Another measurement that, in comparison, turned out as expected was the average duration for completing the assignment. The workers were restricted to a maximum of 2 h for providing their results. However, since the data here were not scaled (there were 59 workers more in the rejected group than in the accepted), the trend clearly shows that the approved workers on average took more than twice the time that the rejected workers did, which implies that they dealt with the tasks in greater depth.

When observing individual solving times in more detail, data scattering among rejected workers stands out. In the left part of Figure 11, completion times vary from a couple of minutes to the full 2 h. Similar to the conclusion in [32], we assume that the disparity in overall task completion times and results is a consequence of some participants skimming over the instructions or not reading them at all.

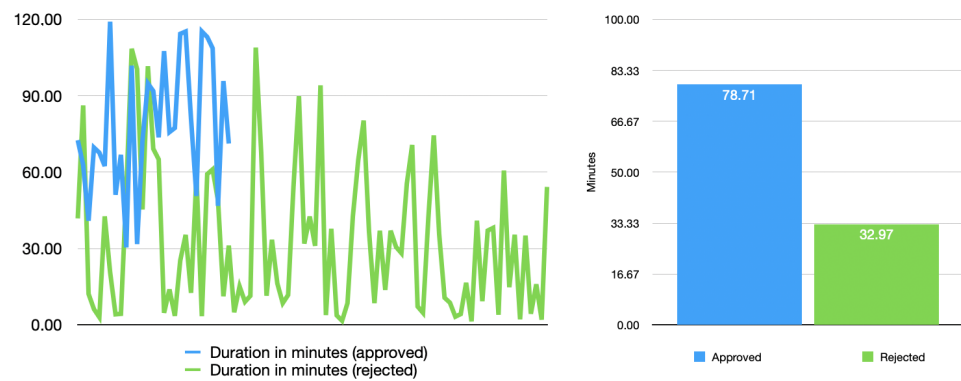


Figure 11. Duration in minutes for approved vs. rejected workers.

We consider it rather inconvenient that the individual task resolution time cannot be deduced from the automatic mTurk output.

### 6.3. Demographic Analysis across All Groups

This section presents the results of the analysis of the demographics and responses of the participants with both approved and rejected HITs. The similarities and differences between the same data from both participant groups are interesting to observe.

Starting with nationality, as Figure 12 shows, three distinct countries dominate both the approved and rejected HIT workers.

Most of the rejected participants declared to be residing in the USA, while this country was only third in the approved workers’ statistics. The exact reasons for this distribution are unclear. However, one possible reason may be that the rejected participants used software robots (a technology that is popular in technologically developed countries such as the USA), which led to their identification as scammers. The fact that participants from the USA account for 63% of all rejected participants suggests that user study participants from this country should be scrutinized more carefully because they show high potential for scamming. Since the UK does not figure among the approved participants, participants from this country are likely to use similar means for automatically completing assignments as those from the USA. Concerning India, it is difficult to deduce whether participants were rejected because they used software robots or did not invest enough time and effort in the completion of the assignment. A more detailed discussion about potential scamming is conducted later in the paper.

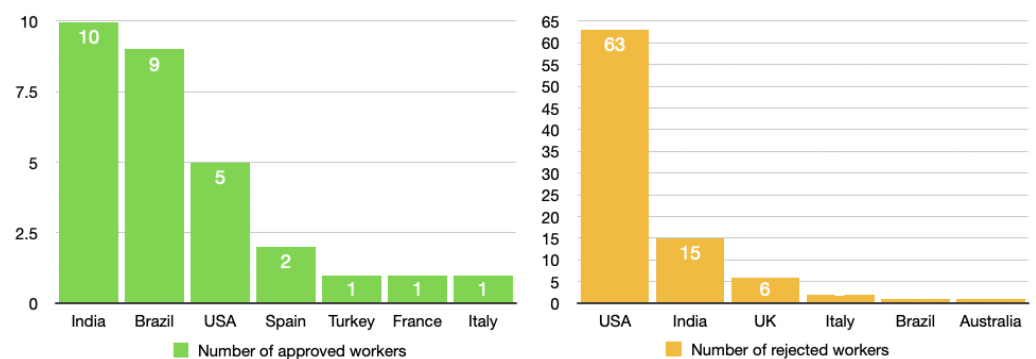


Figure 12. Approved vs. rejected workers’ nationality.

Concerning participant age, we did not impose any limits, as we wanted to be as inclusive as possible. The age of the participants ranged from 22 to 52, with 31 years as the average age. As Figure 13 shows, higher data scattering can be observed within rejected workers, with the participant age ranging from 18 to 63, with an average of 35.

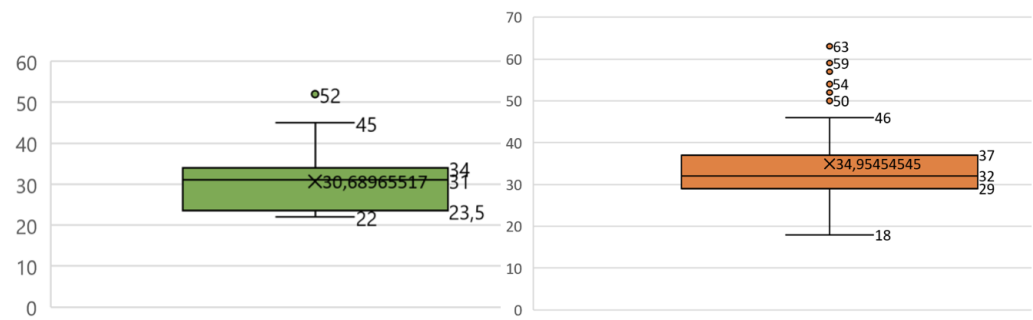


Figure 13. Approved vs. rejected workers' age.

In terms of gender distribution, there were 3 female and 29 male participants within accepted workers. The rejected workers were predominantly male (72%) as well; however, the sample was less biased towards male participants than in the case of the approved HITs. One participant did not declare their gender.

Concerning education, the majority of the workers listed Bachelor studies as their highest academical achievement. However, the field "Other" was highly diversified, from elementary school up to "Some college".

Educational statistics is particularly interesting considering the predominance of the Master's degree (68%) as the highest educational degree declared by the rejected participants. Considering the results of our study with Master's students, who were all able to solve the assignment, the fact that most participants who declared to have a Master's degree were rejected suggests high scamming potential for this group. By contrast, the majority (55%) of the approved participants declared to have a Bachelor's degree as their highest education. This suggests that a solid majority (i.e., 65% 55% declaring Bachelor's degree and 10% declaring high school as their highest level of education) of the approved participants are students in various phases of their studies.

To reduce the work of filtering out scammers, researchers who use crowdsourcing platforms to conduct user studies could consider filtering out participants who declare a Master's degree as their highest level of education from the results. Even if this declaration were to be true, people having high levels of education are not the best candidates in software user studies because these people tend to be more skilled in using software tools than others.

The demographic data collected suggest that, with the exception of gender, crowdsourcing platforms allow for the assembly of a diverse population of participants. Especially in terms of educational background, the diversity of the participants exceeds that of typical user studies conducted with students or company employees.

#### 6.4. Improvement Suggestions Proposed by Workers across All Groups

In order to understand and make better use of the workers' feedback, we performed a qualitative analysis of their textual inputs. In the course of this analysis, it became evident that most of the written feedback could be divided into five categories:

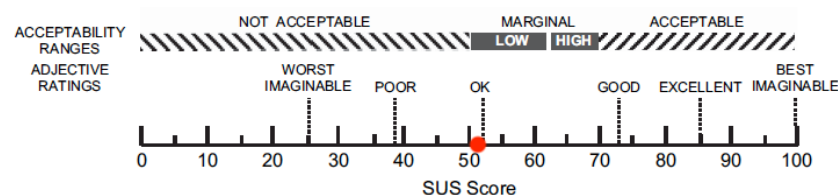
1. Did not understand what was meant by:
  - Driving the robot;
  - Taking a screenshot of the entire code or just the visible chunk.
2. Had a hard time figuring out how to:
  - Bookmark a task;
  - Understand when the code is generated and subsequently copy it from the clipboard;
  - Set variables, parameters, and the specific "relative motion" actor dp.
3. Not functioning properly:
  - Play button after the first use not executing correctly or at all;

- Browsers other than Google Chrome (robot disappearing from Safari);
  - Saving tasks and also disappearance of individual tasks when refreshing the page;
  - Cube on wrong coordinates;
  - Glitch between the execution of *pick* and *place* tasks where robot drops the cube while continues its waypoint.
4. Suggestions:
- The bookmarked actors should be shown separately after reloading the task;
  - More specific instructions needed; image or video tutorials suggested, such as simplifying the help section;
  - Less cluttered UI;
  - Robotic arm should move using the mouse and keyboard.
5. Uncategorized:
- Curiosity to observe the interaction with the real robot;
  - More than three workers claimed not to have an idea where to even begin;
  - Overtaking placeholder values was a recurring issue in open-ended questions;
  - Even though the workers were kindly asked not to repeat the HIT, two did anyway and reported high learning effects, a shorter completion time, and a higher rating of the tool in the SUS scale;
  - The file upload via external tool was cumbersome and probably more difficult than the entire task and took the longest time.

#### 6.5. SUS Score of Approved Tasks

To evaluate usability, we used the SUS survey defined by [49]. This survey defines ranges from 1 to 5, with 1 standing for “strongly disagree” and 5 for “strongly agree” with the respective statement. Additionally, the N/A option was offered. The participants had to fill it out in order to complete the assignment. According to [49], N/A answers should be placed at the center of the scale, which, in this case, was 3.

The results of the approved SUS survey, shown in Figure 14, provide at least two interesting insights. First, the lower overall score compared to the evaluation with students suggests that a more diverse background in terms of education, nationality, and age leads to more usability challenges for designers of robot programming tools. Although, in hindsight, such an outcome seems reasonable, the researchers’ expectation was that the results of the two cohorts (students and approved mTurk participants) would be somewhat closer. Second, the *Assembly* tool has great potential for improvement along the lines drawn by themTurk study participants in their textual feedback, which will be used to improve the next version of the system.



**Figure 14.** Our result (50.54) positioned in grade rankings of SUS score with an adjective rating scale based on [50].

The first insight raises additional questions, which, for the time being, will be deferred to future work: Are user studies conducted in controlled environments (e.g., laboratories, companies, university courses, etc.) systematically biased? If so, is this bias always positive with respect to the results of usability surveys? To tackle these questions, we aim to repeat both the study with students and that with mTurk participants using the next, improved version of *Assembly*.



### 6.6. SUS Score of Rejected Tasks

The average result of the SUS evaluation of rejected tasks is similar, yet lower than in the case of the approved participants. This result is concerning because it calls into question the SUS evaluation of the approved workers. Did the approved workers provide random answers clustered around the midpoint of the scale in the SUS survey or was the score centered around 50 coincidental in the case of the approved participants?

Considering that the approved mTurk participants had a lower average educational background than the Master's students, a lower SUS score in the case of the former was to be expected. Moreover, considering that the majority of the rejected participants were deemed scammers, we could simply dismiss their SUS scores. Nevertheless, the question remains: How should researchers deal with average SUS scores centered around 50 considering that such a value is obtained when providing random answers to the survey?

This points toward an important step that must be added to the methodological framework proposed in this paper: crowdsourced user studies should always be repeated at least once, whereby subsequent studies should be conducted after integrating the feedback from previous ones. This will provide a baseline trend for validating the study results, since usability scores after the improvements should be higher than before.

## 7. Discussion and Outlook

Our results suggest that the proposed approach and framework proved to be feasible considering the diverse and valuable input that we received from the mTurk participants obtained by investing a rather small amount of resources. Overall we spent EUR 223.23 and obtained around 117 results from the 29 accepted and 88 rejected participants in the final run. In addition, we obtained valuable feedback from several test runs. Considering the 29 approved results from the final run, the cost of one approved result amounts to EUR 7.69. This is far from the low rates promised when using crowdsourcing tools [43]. Our analysis supports the assumption that this can be attributed to the many trials and errors, and lessons learned, from the initial test runs.

Concerning the usefulness of crowdsourced usability evaluations of robot programming interfaces, our experience revealed that most usability issues can be identified when following the proposed methodology, with the exception of very specific issues and details.

Concerning audience diversity, both in terms of demographics and technical expertise, our results indicate that one can obtain a highly diverse group of participants, both in terms of demographics and expertise. In sum, we obtained data from five continents, from a total of nine countries, with ages ranging from 18 to 64. The gender distribution was predominantly male. The educational background was very variable. According to the self-assessment, previous knowledge and experience with programming and familiarity with robotics in general was diversified.

Concerning the strengths and weaknesses of *Assembly*, we obtained useful feedback. The SUS scores obtained from the mTurk participants were rather low, with 24.79% of workers managing to solve the tasks within the given timeframe of two hours. By contrast, all participants from the student group managed to solve all the tasks. Apart from the SUS score, the most valuable input was obtained through mandatory free-text fields.

The main takeaways from the participants' feedback were that participants requested tutorials (e.g., videos) that would have helped them to solve the assignment more easily. Whereas the students received a task description, which also contained some screenshots of the *Assembly* tool, mTurk participants only received textual descriptions.

The most frequently recurring issues concerned task saving and reusing, the failure of the "Play button" after one successful execution, and the error when attempting to connect to the robot. This suggests that more detailed instructions are needed to avoid confusion.

Considering the participants' diverse backgrounds, it is important to write instructions using a basic vocabulary and unambiguous meaning. One way to guard against this is to introduce a language test before reaching the HIT itself. The same applies in the case

of programming and robotic skills. We did not find English language proficiency to be important enough to justify a language test.

### 7.1. Limitations of the Approach and Results

Quality control is one of the most time-consuming and difficult challenges in handling the results, as is also noted in [30]. With the Internet acting as the communication medium, which allows for easy access to a wide audience, the abundance of scammers and low-quality results are inevitable. We divided the low-quality answers that we obtained into the following categories:

- data resulting from interactions that were not compliant with the instructions provided;
- solving the assignment using identical inputs from different mTurk accounts;
- random, automatically filled in content, mostly completely out of context.

The latter can be attributed to software robots.

There are no pre-built QA features in mTurk or in any other crowdsourcing tool to the best of our knowledge. This implies that QA must be dealt with explicitly by the researcher. Moreover, there exist commercial solutions that provide QA service for large-scale tasks, such as CrowdFlower [51] or HitBuilder [52]. These solutions, however, cannot be used in the robotics context because this is still a very specific domain [22]. Furthermore, as was recognized by [53], quality tests cannot be automated, which means that heuristic score functions need to be used to predict crowd success. A possible solution for this would be exact physics models for all the objects used in the task [53], but this is beyond the scope of this paper.

Another issue that needs to be addressed is the identification of scamming attempts and software robots. Currently, crowdsourcing platforms provide insufficient instruments for software robot and scam detection. The requesters must implement their own means to tackle this problem. In this context, the distinction between honest and dishonest HIT results must be made using qualitative means, e.g., by manually checking the results and using key questions in surveys that cannot be answered pertinently by software robots.

Finally, ethical concerns must be addressed throughout the process, starting with the question of what represents a fair payment for a HIT. Moreover, tasks should not be designed such that they induce physical strains (e.g., click-intensive tasks).

### 7.2. Conclusions

The present work introduced a methodological framework for conducting user studies with the help of online crowdsourcing platforms, such as Amazon mTurk. Considering the current restrictions regarding face-to-face engagements, we conclude that the online crowdsourcing evaluation of robot programming environments provides a viable alternative to traditional user studies conducted in controlled environments. In consideration of our findings, and the open questions and limitations that we identified, several strands of research are conceivable.

First, we found that crowdsourced user studies yield a lower SUS score than other evaluations (in our case, with Master's students). This raises questions concerning the reliability of the method when it is not being used in a comparative way, i.e., when the study only involves one tool. To tackle this question, we intend to repeat the evaluation in the same way, including with Master's students, in order to better understand the differences in the obtained SUS score. This will also provide some insights into whether controlled user studies of robot programming tools yield SUS scores that inspire very optimistic expectations of robot programming in general. By contrast, our findings suggest that participants would benefit from some kind of training before engaging with robot programming.

Second, there are many technical caveats that we identified when using Amazon mTurk. Further research could look into how scamming could be detected automatically without filtering out a high number of HITs. As we suggested, by filtering out participants

that claimed to possess a Master's degree and to reside in the USA or the UK, we could have reduced the number of rejected results by around 70%. This, however, would have singled out some groups of participants in an unfair way. This indicates the need for better ways to detect scammers while avoiding unfair treatment, e.g., by creating more refined and detailed scammer profiles.

Third, we will integrate the feedback obtained during this study into the next version of the *Assembly* tool and will conduct another study, which will help to further validate the proposed framework and to tackle some of the remaining open questions.

**Author Contributions:** Conceptualization, T.B.I., D.P. and S.S.; methodology, D.P. and T.B.I.; data collection and processing, D.P.; data analysis and interpretation, D.P. and T.B.I.; master thesis supervision, T.B.I. and S.S.; draft paper preparation, D.P. and T.B.I.; final paper preparation, all authors. All authors have read and agreed to the published version of the manuscript.

**Funding:** Open Access Funding by TU Wien.

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. Detailed ISO 9241-210 Framework

This appendix provides a detailed description of the workflow steps from Figure 6.

### Appendix A.1. Evaluation Platform Selection

The first step in the process is choosing the correct platform depending on the nature of the task. The concept of crowdsourcing is much more common in the context of ideation and design (website, graphic, product) than user studies. Websites such as DesignHill [54], Designcrowd [55], 99Designs [56], or Guerra Creativa [57] are examples of forerunners in these categories.

For the less specific tasks, or simply because they cover a broader range of services, other pages are definitely a better choice. In this section, websites such as microworkers [58], Crowdspring [59], or mTurk are considered to be good candidates for tasks related to user studies.

### Submission Tool Selection

In the event that the assignment requires the uploading of results, if not already included in the chosen platform's service, an external web page with any cloud storage system will be necessary. mTurk claimed to offer the possibility of file uploading, but, as will be discussed later, this turned out to be a piece of outdated information.

### Appendix A.2. Methodology Selection

According to [12], there are two main generic crowd design methodologies that can be drawn upon at the high-level stage of creating the task:

1. Iterative improvements  
Especially suitable for design assignments, these tasks are designed as competitions so that the workers are building upon each other's solutions. In this model, workers are anonymously shown previous results from other workers and are basically asked to improve or build upon them.
2. Non-iterative (linear) competition  
This kind of competition can be performed as a single or multistage task, which either compensates workers with rank payments or the best-rated result takes the largest financial award. In this model, competing workers do not collaborate, which is more appropriate for the purpose of evaluating software tools.

### *Appendix A.3. Crowd Selection*

The participants' profile can but does not have to be defined, allowing every individual with Internet access to participate in the task and consequently making each participant's evaluation uniquely diversified.

In the event that the requester decides to define the participants' profile, some of the typical filters that they could use are participants' location, age, gender, education level, experience with the matter in question, or user rating on the selected platform (e.g., number of previous successfully solved tasks or even the quality of the solution).

Additionally, being intentional and inclusive with participant samples generally has positive effects on products or designs since the input comes from diverse participant profiles who are likely to see and do things differently [48]. This is one of the major arguments in favor of leveraging crowdsourcing platforms for user studies.

Nevertheless, even if user diversity is desirable, crowd selection needs to be performed in order to obtain statistically significant results. The authors in [60] conducted a study to show how many participants are necessary to deliver qualitatively good input from the broadest possible point of view on the issue of detecting usability problems. They found that the information gain decreases after reaching the number of five participants. Up to this number, as many as 90% of the issues are likely to be already covered. To account for the remaining 10%, and considering that the present study required workers to solve a specific set of tasks, valid results from at least 25 participants should be aimed for.

### *Appendix A.4. Validation*

This stage deals with the validation of the task specification, its relevance, and the affirmation of the test run. During this phase, the requester may find that, for as relevant as their task initially seemed, there currently exist neither applicable use cases nor appropriate evaluation methods via crowdsourcing.

A further realization might occur later in the process, after having designed the use cases. If the use cases turn out to be rudimentary and seemingly too easy to solve, the task definition should be revisited to avoid wasting funds. Conversely, if the use cases turn out to be too complex to solve, the task should be redesigned and, possibly, broken down into simpler sub-tasks.

### *Appendix A.5. Test Run*

Ahead of uploading an assignment to the crowdsourcing platform, it is recommendable to run a small pilot test with a small batch of participants from one's own organization, preferably familiar with the topic yet not too involved with the assignment. This gives an opportunity to identify some of the obvious flaws in the design of the assignment [48].

The evaluation basis is composed of test use cases that will be used for the test run. The results of this run are a good indicator of what is missing, misformulated, redundant, or irrelevant. One or more test runs can be undertaken depending on the relevance of the participants' input. For example, should the tasks be too repetitive or irrelevant, the test run phase can be ended.

Test runs can also be carried out with members of the crowd. These runs are carried out with a smaller participant sample. While, usually, in usability studies, the user interface should be self-explanatory, to make test runs more effective, it is advisable to provide additional instructions to the participants, since this has a psychological effect on the participants, encouraging them to provide remarks while pointing out that the tool being evaluated is still under development.

Crowdsourced evaluations do not entail less work than in-person testing when performed without a validated methodology, as with the one presented in this paper. Crowdsourced user studies require more explicit planning and communication in order to make sure the tools and devices that will be worked with are working as expected [48]. In a face-to-face setting, improvisation can be used to mitigate poor planning, and communication issues (e.g., task understanding, clarity, difficulty, etc.) are solved informally.

### *Appendix A.6. Use Case Definition*

Use case diagrams are officially defined by [61] as behavior diagrams that are used to describe a specific set of actions that one or more systems should or can perform in collaboration when triggered by actors that are external users of the system. Each use case is meant to provide stakeholders with perceptible and beneficial results.

In the context of robotics, the participants will likely be people with limited or no programming or robotic background and limited attention spans, since they are working behind their displays. As [22] suggests, simplifying complex and difficult to describe problems into smaller, simpler, and easier to describe sub-problems is the key to lowering the rate of unusable participant output.

To improve the yield of evaluation runs in terms of usable outputs, individual use cases should cover the essence of the functionality or process flow being evaluated as succinctly and simply as possible. Ideally, the use cases should progressively grow in complexity. Starting with simple use cases can help to ensure that participants become familiar with the system and are comfortable with using it. This will increase the likelihood of participants also solving more complex use cases.

Ongoing research in the area investigates how to define tasks in such a way that enables the crowd to accomplish complex and expert-level work. Various approaches and workflows can be used to break a complex piece of work into approachable parts, while also using members of the crowd to check the quality of other members or of their own work [62].

### *Appendix A.7. Instruction Writing*

This point is crucial in remote testing methods because they are mostly not moderated, so it is highly important to write the instructions as clearly and straightforwardly as possible so as to avoid misunderstandings, raising questions, or double meanings.

It is important to keep the instructions simple, using basic wording or exclusively technical vocabulary, the latter of which should only be used if the participants will be filtered accordingly. Any ambiguity has to be avoided. Long and complex sentences should be avoided because they provide more opportunities to make mistakes, impede the understanding due to complexity, and tire out the participants faster. Concerning the last point, it is very important to try to keep the participants captivated by the task and highly motivated. One of the suggestions for doing so is mixing the type of feedback requests between audio or video output and textual, which can vary between open-ended or single or multiple-choice questions. The same applies to the input from the requesters' side.

As suggested in [48], usability testing should only be as long as it needs to be in order to gain confidence in the results. In the event that the requester is uncertain of the test length, they should pilot it and collect feedback. Running multiple smaller separate tests is recommended over trying to obtain useful feedback from participants who are cognitively exhausted by the length of the assignment [48].

### *Appendix A.8. Demographic Questionnaire*

In every study, a demographic questionnaire is required in order to obtain a clear picture of the participants, understand their motivation, and better assess the relevance to their input. In crowdsourced evaluation, this point is even more important than in face-to-face settings.

Demographic questions are usually placed at the beginning or at the end of the survey and should ask participants about their background. There is a variety of questions that can be asked, but the idea is to restrict them to avoid overloading the participants while gathering useful information.

### *Appendix A.9. Scam Detection*

The availability of assignments and the reachability and diversity of the participant pool also have their downsides. One of the major issues with crowdsourced evaluations

is the potential for scamming. The workers are trying to finish as many assignments in the shortest amount of time possible in order to earn as much as possible. This is often achieved by using software robots, which can automate user tasks. Scamming can be tackled by using Captcha tests and trick questions in the surveys, as well as by filtering results.

#### *Appendix A.10. Payment*

Crowdsourcing is infamous for rather low payments because it mostly provides a quick method for outsourcing numerous basic tasks to as many participants as possible. According to [43], depending on the method used, the mean hourly wage per worker on mTurk lies between USD 3.13 and USD 3.48, while the median hourly rate lies between USD 1.77 and USD 2.11. This is significantly below the minimum wage of economically developed countries.

From the requester's point of view, cost-effectiveness and speed are significant advantages of crowdsourcing over other evaluation methods, but it is important to find the right balance here. Very small payments will probably attract participants having low educational backgrounds, coming mostly from underdeveloped countries, or insufficient participants. Ethical issues should be considered here, such as not offering less than a participant would be offered in a face-to-face setting. In Austria, for example, the usual payment for participating in a face-to-face user study ranges from EUR 5 to EUR 20. This includes the participants' overhead of traveling to a designated location at a certain hour of a certain day.

On the other hand, paying too much does not guarantee the right crowd. The problems change their nature, such as attracting wrong participant profiles, with an increased percentage of scammers because of the higher payoff. Ultimately, high payments will reduce the attractiveness of crowdsourcing for the requester since the costs do not offer benefits over conventional evaluation methods. The payment should be sufficiently high to provide participants with enough motivation for solving the assignment properly and to avoid ethical concerns, while not canceling out the benefits of the crowdsourcing evaluation approach.

For example, the authors in [63] reported that there is a weak correlation between the level of payment (per worker and per task) and the quality of results. Other researchers have found a strong correlation between the level of payment and the number of designs generated [43]. This is confirmed by [64], adding that higher payment definitely incentivizes more participants, which results in quicker recruitment and higher willingness to deliver more output. They also state that there is some weak proof that bonuses and penalties do contribute to higher overall output quality.

#### *Appendix A.11. Execution*

Generally speaking, usability evaluation task execution can be:

- **Moderated**  
In moderated remote usability testing, the test facilitator interacts in real time with the participant at a remote location [65]. This is easily realized via any video conferencing or remote application sharing tool, such as WebEx [66].
- **Non-moderated**  
The most obvious advantage of non-moderated usability testing is that no moderator or location (such as lab spot) is necessary, which entails lower costs and quicker implementation. This implies that participants can complete the assignments at the time and place of their choosing, which is the definition of an uncontrolled environment. This setting also resembles more closely how a real user would interact with the program in their natural environment and therefore generates more authentic results. As with the remoteness factor, non-moderated execution contributes to the speed of result collection merely because it is not necessary to find a moderator or set a location and time for each session.

Nonetheless, this usability testing execution method comes with disadvantages, notably in terms of less control over the entire test environment and evaluation procedure as a whole. In order to minimize the risk of misunderstanding and the need to ask questions during the evaluation itself, a great deal of thought and effort needs to be invested in the instruction writing phase. The author in [48] suggests writing an introductory context and guidelines at the beginning of the test to inform the participant of the session's goals and to provide them with relevant contextual information that clears the participants' doubts.

In [65], both study participant groups delivered rather similar arguments, with non-moderating users reporting a higher percentage of high-relevance input. This could be the result of the positive effect of not being monitored, which seems to make participants more relaxed and creative.

#### Appendix A.12. Evaluation

Establishing appropriate evaluation criteria is an essential factor in the success of crowdsourced user studies. If this is not done properly, even with all the previous steps done correctly, one may end up with a high amount of useless data. The requester should make clear from the beginning how a successful run can be defined and identified (i.e., definition of success) and have explicit ways to distinguish success from failure [48]. To help with this task, several methods can be drawn upon:

- **Qualitative analysis**  
The qualitative data are usually obtained using individual, subjective satisfaction questionnaires [67] or observations and analysis of participants' input, in the form of task solutions or open-ended answers. Based on the requester's own expertise in UX, and keeping the program's known UX difficulties and limitations in mind, acknowledging and comparing these insights with the participants' output should lead to clear ideas if a certain part of the UI needs to be redesigned [68].
- **Quantitative analysis**  
The quantitative data include all numerical data, such as the time necessary to complete a task, amount of usability defects identified [67], or task completion rate. Crowdsourced evaluation works hand in hand with quantitative testing because it automatically measures the performance of the design and tracks usability metrics easily, since most of the data are calculated and reported automatically [48]. Usability quantification can be performed using post-experiment surveys, such as PSSUQ [69] or SUS [49].

#### Appendix A.13. Results

This final step should deliver all smaller change requests or larger redesign requirements. These should ideally be implemented immediately in the version of the system being evaluated. Subsequently, the same use cases should be run through by the requester or the same group of people from inside the organization who were gathered for the pilot study during the test run.

At this stage, the UI usually shows improvements. If this is not the case, some part of the methodology was not implemented or evaluated correctly, especially if the same errors are repeated. In the case of new errors, another user study should be performed.

## References

1. Huber, A.; Weiss, A. Developing Human-Robot Interaction for an Industry 4.0 Robot: How Industry Workers Helped to Improve Remote-HRI to Physical-HRI. In Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, Vienna, Austria, 6–9 March 2017; pp. 137–138. [\[CrossRef\]](#)
2. Linsinger, M.; Stecken, J.; Kutschinski, J.; Kuhlenkötter, B. Situational task change of lightweight robots in hybrid assembly systems. *Procedia CIRP* **2019**, *81*, 81–86. [\[CrossRef\]](#)
3. Lehmann, C.; Städter, J.P.; Berger, U. Anwendungsbeispiele zur Integration heterogener Steuerungssysteme bei robotergestützten Industrieanlagen. In *Handbuch Industrie 4.0 Bd. 2*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 45–58.

4. Weintrop, D.; Afzal, A.; Salac, J.; Francis, P.; Li, B.; Shepherd, D.C.; Franklin, D. Evaluating CoBloX: A comparative study of robotics programming environments for adult novices. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–12.
5. Biggs, G.; Macdonald, B. A Survey of Robot Programming Systems. In Proceedings of the Australasian Conference on Robotics and Automation, CSIRO, Taipei, Taiwan, 14–19 September 2003; p. 27.
6. Pan, Z.; Polden, J.; Larkin, N.; van Duin, S.; Norrish, J. Recent Progress on Programming Methods for Industrial Robots. In Proceedings of the ISR/ROBOTIK, Munich, Germany, 7–9 June 2010.
7. Nielsen Norman Group. Available online: <https://www.nngroup.com> (accessed on 30 June 2021).
8. Ionescu, T. Leveraging graphical user interface automation for generic robot programming. *Robotics* **2021**, *10*, 3. [CrossRef]
9. Ionescu, T. Adaptive Simplex Architecture for Safe, Real-Time Robot Path Planning. *Sensors* **2021**, *21*, 2589. [CrossRef] [PubMed]
10. Howe, J. The rise of crowdsourcing. *Wired Mag.* **2006**, *14*, 1–4.
11. Toris, R.; Chernova, S. RobotsFor.Me and Robots For You. 2013. Available online: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.365.38> (accessed on 11 November 2021).
12. Wu, H.; Corney, J.; Grant, M. An evaluation methodology for crowdsourced design. *Adv. Eng. Inform.* **2015**, *29*, 775–786. [CrossRef]
13. Van Waveren, S.; Carter, E.J.; Örnberg, O.; Leite, I. Exploring Non-Expert Robot Programming Through Crowdsourcing. *Front. Robot. AI* **2021**, *242*. [CrossRef] [PubMed]
14. Ionescu, T.B.; Fröhlich, J.; Lachenmayr, M. Improving Safeguards and Functionality in Industrial Collaborative Robot HMIs through GUI Automation. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 557–564.
15. Ionescu, T.B.; Schlund, S. A participatory programming model for democratizing cobot technology in public and industrial Fablabs. *Procedia CIRP* **2019**, *81*, 93–98. [CrossRef]
16. Ionescu, T.B.; Schlund, S. Programming cobots by voice: A human-centered, web-based approach. *Procedia CIRP* **2021**, *97*, 123–129. [CrossRef]
17. Komarov, S.; Reinecke, K.; Gajos, K.Z. Crowdsourcing Performance Evaluations of User Interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 207–216. [CrossRef]
18. Grossman, T.; Balakrishnan, R. The bubble cursor: Enhancing target acquisition by dynamic resizing of the cursor’s activation area. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Portland, OR, USA, 2–7 April 2005; pp. 281–290.
19. Sears, A.; Shneiderman, B. Split menus: Effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact. (TOCHI)* **1994**, *1*, 27–51. [CrossRef]
20. Gajos, K.Z.; Czerwinski, M.; Tan, D.S.; Weld, D.S. Exploring the design space for adaptive graphical user interfaces. In Proceedings of the Working Conference on Advanced Visual Interfaces, Venezia, Italy, 23–26 May 2006; pp. 201–208.
21. Crick, C.; Osentoski, S.; Jay, G.; Jenkins, O.C. Human and robot perception in large-scale learning from demonstration. In Proceedings of the 6th International Conference on Human-Robot Interaction, Lausanne, Switzerland, 6–9 March 2011; pp. 339–346.
22. Sorokin, A.; Berenson, D.; Srinivasa, S.S.; Hebert, M. People helping robots helping people: Crowdsourcing for grasping novel objects. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 2117–2122.
23. Emeli, V. Robot learning through social media crowdsourcing. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 2332–2337.
24. Amazon Alexa. Available online: <https://developer.amazon.com/en-US/alexa> (accessed on 1 May 2021).
25. Apple Siri. Available online: <https://www.apple.com/siri/> (accessed on 1 May 2021).
26. Google Assistant. Available online: <https://assistant.google.com> (accessed on 1 May 2021).
27. Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M.; Banerjee, A.; Teller, S.; Roy, N. Understanding natural language commands for robotic navigation and mobile manipulation. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011; Volume 25.
28. Von Ahn, L.; Dabbish, L. Designing games with a purpose. *Commun. ACM* **2008**, *51*, 58–67. [CrossRef]
29. Chernova, S.; DePalma, N.; Morant, E.; Breazeal, C. Crowdsourcing human-robot interaction: Application from virtual to physical worlds. In Proceedings of the 2011 RO-MAN, Atlanta, GA, USA, 31 July–3 August 2011; pp. 21–26.
30. Chung, M.J.Y.; Forbes, M.; Cakmak, M.; Rao, R.P. Accelerating imitation learning through crowdsourcing. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 4777–4784.
31. Verma, D.; Rao, R.P. Goal-based imitation as probabilistic inference over graphical models. In *Advances in Neural Information Processing Systems*; Citeseer: Princeton, NJ, USA, 2006; pp. 1393–1400.
32. Toris, R.; Kent, D.; Chernova, S. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *J. Hum.-Robot Interact.* **2014**, *3*, 25–49. [CrossRef]
33. Kormushev, P.; Calinon, S.; Caldwell, D.G. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Adv. Robot.* **2011**, *25*, 581–603. [CrossRef]



34. Huang, J.; Lau, T.; Cakmak, M. Design and evaluation of a rapid programming system for service robots. In Proceedings of the 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Christchurch, New Zealand, 7–10 March 2016; pp. 295–302.
35. Blockly. Available online: <https://developers.google.com/blockly> (accessed on 1 June 2021).
36. Perkel, D. Copy and paste literacy? Literacy practices in the production of a MySpace profile. *Informal Learn. Digit. Media* **2006**, 21–23.
37. Ionescu, B.T. Assembly. Available online: <http://assembly.comemak.at> (accessed on 30 April 2021).
38. Blackboard Design Pattern. Available online: <https://social.technet.microsoft.com/wiki/contents/articles/13215.blackboard-design-pattern.aspx> (accessed on 1 June 2021).
39. de Jong, J. JSON Editor Online. Available online: <https://jsoneditoronline.org/> (accessed on 30 June 2021).
40. Beck, M. Glumb. Available online: <http://robot.glumb.de/> (accessed on 30 April 2021).
41. Universal Robots. Available online: <https://www.universal-robots.com/> (accessed on 30 April 2021).
42. Amazon Mechanical Turk. Available online: <https://blog.mturk.com> (accessed on 30 June 2021).
43. Hara, K.; Adams, A.; Milland, K.; Savage, S.; Callison-Burch, C.; Bigham, J.P. A data-driven analysis of workers' earnings on Amazon Mechanical Turk. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–14.
44. Usability Is a Key Element of User Experience. Available online: <https://eu.landisgyr.com/better-tech/usability-is-a-key-element-of-user-experience> (accessed on 30 June 2020).
45. ufile. Available online: <https://ufile.io> (accessed on 1 June 2021).
46. Montage II: Advanced Manufacturing (TU Wien). Available online: <https://tiss.tuwien.ac.at/course/courseDetails.xhtml?courseNr=330288&semester=2021W&dswid=5005&dsrid=624> (accessed on 5 August 2021).
47. Oppenheimer, D.M.; Meyvis, T.; Davidenko, N. Instructional manipulation checks: Detecting satisficing to increase statistical power. *J. Exp. Soc. Psychol.* **2009**, *45*, 867–872. [CrossRef]
48. Sirjani, B. maze.co. Available online: <https://maze.co/guides/usability-testing/> (accessed on 30 June 2020).
49. Brooke, J. Sus: A 'Quick and Dirty' Usability. *Usability Eval. Ind.* **1996**, *189*, 4–7.
50. Bangor, A.; Kortum, P.; Miller, J. Determining what individual SUS scores mean: Adding an adjective rating scale. *J. Usability Stud.* **2009**, *4*, 114–123.
51. Crowdfunder. Available online: [https://visit.figure-eight.com/People-Powered-Data-Enrichment\\_T](https://visit.figure-eight.com/People-Powered-Data-Enrichment_T) (accessed on 1 July 2021).
52. HitBuilder. Available online: <https://ga-dev-tools.appspot.com/hit-builder/> (accessed on 1 July 2021).
53. Forbes, M.; Chung, M.; Cakmak, M.; Rao, R. Robot programming by demonstration with crowdsourced action fixes. In Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, Pittsburgh, PA, USA, 2–4 November 2014; Volume 2.
54. Design Hill. Available online: <https://www.designhill.com> (accessed on 1 June 2021).
55. Designcrowd. Available online: <https://www.designcrowd.com> (accessed on 1 June 2021).
56. 99Designs. Available online: <https://99designs.at> (accessed on 1 June 2021).
57. Guerra Creativa. Available online: <https://www.guerra-creativa.com/en/> (accessed on 1 June 2021).
58. Microworkers. Available online: <https://www.microworkers.com> (accessed on 1 June 2021).
59. Crowdspring. Available online: <https://www.crowdspring.com> (accessed on 1 June 2021).
60. Nielsen, J.; Landauer, T.K. A mathematical model of the finding of usability problems. In Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems, Amsterdam, The Netherlands, 24–29 April 1993; pp. 206–213.
61. UML. Available online: <https://www.uml-diagrams.org/use-case-diagrams.html> (accessed on 4 May 2021).
62. Zaidan, O.F.; Callison-Burch, C. Crowdsourcing Translation: Professional Quality from Non-Professionals. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, 19–24 June 2011; Association for Computational Linguistics: Portland, OR, USA, 2011; pp. 1220–1229.
63. Wu, H.; Corney, J.; Grant, M. Relationship between quality and payment in crowdsourced design. In Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Hsinchu, Taiwan, 21–23 May 2014; pp. 499–504.
64. Shaw, A.D.; Horton, J.J.; Chen, D.L. Designing incentives for inexpert human raters. In Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, Hangzhou, China, 19–23 March 2011; pp. 275–284.
65. Hertzum, M.; Borlund, P.; Kristoffersen, K.B. What do thinking-aloud participants say? A comparison of moderated and unmoderated usability sessions. *Int. J. Hum.-Comput. Interact.* **2015**, *31*, 557–570. [CrossRef]
66. Albert, W.; Tullis, T. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*; Elsevier: Amsterdam, The Netherlands, 2013.
67. Nielsen, J.; Levy, J. Measuring usability: Preference vs. performance. *Commun. ACM* **1994**, *37*, 66–75. [CrossRef]
68. Quantitative vs. Qualitative Usability Testing. Available online: <https://www.nngroup.com/articles/quant-vs-qual/> (accessed on 30 June 2021).
69. PSSUQ. Available online: <https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/> (accessed on 30 June 2021).