Age-Inclusive Integrated Development Environments for End-Users

1st Katharine Kerr
Department of Computer Science
University of British Columbia
Vancouver, Canada
kekerr@cs.ubc.ca

2nd Reid Holmes
Department of Computer Science
University of British Columbia
Vancouver, Canada
rtholmes@cs.ubc.ca

***Abstract*—Computation increasingly pervades modern life in both the professional and personal realms. An example in the personal realm is the maker movement, which has helped expose many end-users to programming. To ensure equitable access to these new programming domains, it is important to ensure that the tools being promoted to these communities can be used broadly. In this paper, we investigate a tinkering-focused integrated development environment for makers who are engaged specifically in customizing designs for hand knitting. Through a controlled experiment with 91 end-users, 32 of whom were over age 50, we identified trends in how differently-aged participants worked through their maker design tasks with our integrated development environment. While older participants found it more challenging to complete assigned design tasks, participants of all ages were more likely to succeed if they decomposed tasks into partially correct programs. However, we found that successful participants of all ages exhibited common traits of engagement, experimentation, and curiosity. Users found the environment engaging and favoured visual feedback both when making progress and when stuck. Our results provide insights into how development environments can be designed to more inclusively support a broader cross-section of end-users.**

I. Introduction

The maker movement is one of many domains in which technology has moved into the personal realm. Many kinds of making require makers to interact with programmatic representations of their projects in a way that makes them effectively be end-user programmers. Specific tools are often used for helping these end-users interact with their projects. A variety of motivations drive makers, ranging from hobbyists, to those pursuing job training, through to entrepreneurs [1]. Unfortunately, younger male demographics have an outsized representation in the maker community [2], [3].

Age has been previously identified as an important design dimension that should be explicitly supported [4], and age is an increasingly important trait as the proportion of the population that is older continues to grow [5]. While 65 has previously been the threshold used to define 'older adult', it has been suggested that 50 may be a more appropriate threshold to avoid missing important members of the community [4]. Characterizing approaches as a 'solution' to age has been found to be problematic [6]; in this work we instead set out to design an inclusive environment for supporting maker tasks in a way that encourages positive tinkering behaviours that will be accessible to all users independent of their age.

Hand knitters are makers who design, modify, and create knitwear items by hand as hobbyists, designers, and professionals [7]. The designs knitters use are written in a domain specific language that is not easily amenable to customization. In this paper, we introduce StitchUp, a live programming environment for supporting knitters through tinkering as they create and evolve knitwear designs. Tinkering is a learning-rich style of work that is characterized by play, experimentation, iteration, and engagement [8]. Tools that have been purposely built to support tinkering have had incredible popularity, with Scratch having over 100 million users [9], [10]. Tinkering has also been shown to result in increased test performance in an educational setting [11] and has been tied to successfully testing and debugging [12].

For our research, we performed a between-groups controlled experiment with 91 participants: 88 women, 2 men and 1 non-binary person. Through a mixed-methods analysis we evaluated results across four age groups, where each age group contained 17 or more participants. Participants worked on two design tasks in StitchUp and completed a survey regarding their experiences with the tool. Participants used a familiar knitting pattern Domain Specific Language (DSL).We looked at participants' experiences in StitchUp to answer the following research questions:

RQ1: Can age-diverse end users successfully complete design tasks in StitchUp?
RQ2: What strategies do age-diverse end users use when performing tasks in StitchUp?
RQ3: Are StitchUp's tinkering features used consistently by all age groups?

We found that all participants, regardless of age, could successfully complete design tasks in StitchUp, but the oldest group (50+) had the highest rate of failure. The 50+ age group iterated less, took fewer risks, and got stuck at higher rates than their younger counterparts. Although younger participants encountered *more* errors than older groups, they were able to resolve them more quickly. Lastly, all ages preferred the visualization of the knitting pattern, as the visual representation gave them insights into the impact of their DSL changes,

encouraging them to explore design alternatives in an engaged way. Although older participants were less successful than younger participants, successful participants, regardless of age, exhibited common traits of engagement, experimentation, and curiosity. The contributions of this paper are:

- StitchUp, a tinkering-forward end user live programming environment for designing hand-knit pattern instructions.
- A study examining how 91 participants, primarily women ranging in age from 19 to 70+, performed domain-specific design tasks within this tinkering-forward environment.
- Observations about how participants of different ages engaged with tinkering features as they worked.

The results of our study suggest that investments in environments that support visualization, iteration, and engagement could increase how successful an age-diverse of end-user programmers could be which in turn help make these environments more inclusive.

II. Related Work

As society ages [5], it is increasingly important that older demographics are not excluded from technological advances [4]. One lens through which the importance of fairness and equity in access to technology can be viewed is Universal Usability [13]; this viewpoint emphasizes the importance of finding representations that help users to be successful with technology. Given that makers are disproportionately younger men, it is unsurprising that the participants for research in this space is more heavily represented by these groups [2], [3]. The primary contribution of this work is a controlled experiment with the opposite demographic, examining the influence of development environment features for end-user programmers with more diverse ages and genders.

Tinkering is a mindset that evolved from a *constructionist* style of education where students learn-by-making [14]. While tinkering encompasses a broad range of activities, the definition we use in this paper for tinkering describes it as "a valid and valuable style of working, characterized by a playful, exploratory, iterative style of engaging with a problem or project" [8]. Tinkering has also been defined more broadly as developers exhibiting exploratory behaviours, deviating from instructions, not relying on formal methods, and using trial-and-error techniques [15]. StitchUp was designed to support a tinkering-forward interaction mechanism to align the tool with the positive aspects of the traditional design process our makers would be familiar with to support Universal Usability [13].

To determine which tinkering features to support, we examined research from the live programming community (e.g., [16], [17]). Live-programming features [10], [18] provide immediate feedback and insight, an important aspect of tinker-focused design [8]. Tinkering is an effective approach for designing and building programs. For example, tools that have been explicitly designed to encourage tinkering have seen wide adoption [10], [18]. A core principle when designing these tools is immediate feedback [8]. Additionally, lightweight features to let users check their assumptions by evaluating design alternatives have been found broadly useful to quickly and concretely compare and contrast the impact of different design choices. In their study, Kubelka et al. showed that tools that provide immediate feedback and inspectors that showed the relationship between code and its impact were most widely used [19]. Note, in this study 100% of participants were male. StitchUp explicitly supports immediate feedback and assumption-checking features.

As far as we are aware, only one tinkering study has explicitly looked at age-related factors. This prior study evaluated the effect of cognitive playfulness on microcomputer training performance; the average participant age was 41 and 89% of participants were female [11]. The authors found that increased cognitive playfulness resulted in increased performance. Our work extends this prior work by examining whether tinkering-forward features improves the ability of an age-diverse group of end-users to perform creative technical tasks.

III. Approach

This work investigates how an age-diverse group of end users can benefit from an environment designed for tinkering. The environment was custom-tailored to hand knitting tasks to enable us to recruit a diverse set of end users. There are millions of knitters, where the majority are female and the population exhibits broad age diversity [20], [21].

A. Knitters as End Users

The way knitters engage in their craft brings them to an intersection of tinkering and end user programming.

Knitters Tinker. The maker movement is a catch-all term that encompasses a broad cross section of creative activities; 'makers' are people involved in this space who design and build things that typically exist in physical space. Tinkering has deep routes in the maker movement, as makers often tinker as they explore and build [22]. Knitters are makers who design and create pieces, such as scarfs and sweaters, through knitting [7]. In this paper we focus on hand-knitters, rather than machine-knitters.

While knitters often start from existing designs, they frequently deviate from established designs to personalize or improve their final knitwear pieces [23]. In this way, knitters tinker with designs, testing out many solutions until they achieve an outcome they are happy with. For example, a knitter might tinker with the neck shaping of a sweater or different yarn colours in a motif. One of the most popular knitting patterns has a wiki page devoted to the many modifications on the original pattern [24].

Knitters Program. Knitting designs are specified in *patterns*. Patterns are like programs as they contain ordered, unambiguous instructions. In practice, almost all knitting designs are encoded in knitting-specific DSLs. The pseudo-natural language of knitting patterns has been processed on a large scale into a computer-readable representation [25] and programmatic DSLs have been created based on existing patterns [26], [27].

These patterns have an established notation, where symbols and shorthand are used to describe instructions [28]. This

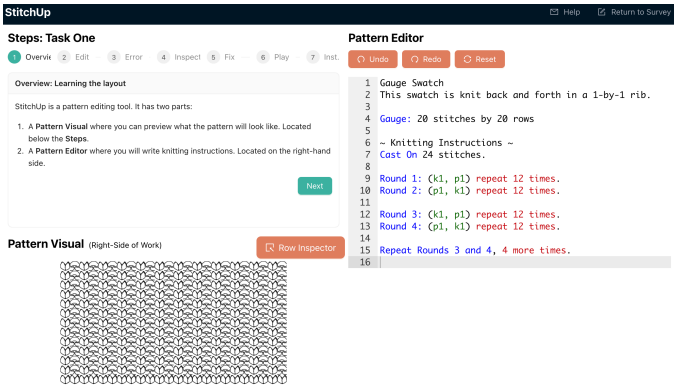


Fig. 1: The StitchUp IDE as configured for the first study task. The upper-left quadrant contains the study instructions, the lower-left quadrant shows a visual representation of the pattern, and the right-side contains the pattern editor.

notation is colloquially referred to as *knit-speak*. Each row in knitting is made up of stitches, most commonly knit and purl stitches, and each row is built off the row below it. Knit-speak is used to specify what stitches form a row and the order in which the rows should be knitted to form the knitted piece.

A definition of a round (row) in knit-speak might look like:

```
Round 1: (k1, p2) 3 times.
```

The `k` and `p` represent the type of stitch, knit and purl respectively. A purl stitch is the opposite of a knit stitch as it is a knit stitch viewed from the back of the finished piece. The numbers represent the number of times the stitch should be repeated. `k1` tells the knitter to knit one stitch and `p2` to purl two stitches. The round brackets represent a set of stitches to be repeated, so in this case the knitter should knit one stitch and then purl two stitches and repeat this three times.

Knitters interpret a written pattern and translate it into the physical space with yarn and needles. Due to the time required for this translation step, knitters spend considerable effort ensuring the design they are knitting (encoded in the pattern) matches their needs and expectations. Knitters create and modify these patterns (programs) to design pieces and then execute the programs to build them, similar to how a computer executes code. In this way, knitters are end user programmers, where they write, modify, and interpret programs to accomplish their domain-specific tasks.

B. Supporting Tinkering In StitchUp

The StitchUp environment was created to allow knitters to interactively design knitting patterns. The core design principles for the environment were chosen to enable tinkering. The environment aims to support immediate feedback, fluid experimentation, and open exploration, as these three properties have been previously identified as core principles for enabling tinkering [8]. The first two principles, immediate feedback and fluid exploration, complement each other as they allow a user to easily iterate through design alternatives and quickly see the results of these iterations. StitchUp supports these with a live-updating view that continually shows a visual representation of a pattern. StitchUp supports open exploration by allowing knitters to creatively express themselves in a knitting-specific DSL, allowing them to experiment with different sequences of stitches and rows to express alternate designs.

StitchUp consists of two primary parts: the Pattern Editor, where users modify knitting patterns, and the Pattern Visual, where users can view a fabric representation of the pattern; these are shown in Figure 1. The environment consists of four main features to encourage and support tinkering with designs.

1) Feature 1: Live Editing: As a user edits a design in the pattern editor, StitchUp updates the pattern visual every 200 milliseconds (ms), providing immediate feedback for any changes they make to their design. This interaction between editing the pattern (program) and the immediate update of the visual representation of the program makes StitchUp a live environment [17]. This liveness also represents the core technique enabling rapid tinkering in the environment supported through immediate feedback and fluid exploration.

The live visualization link between the pattern editor and the pattern visual is supplemented by three additional features. Each of these features is elaborated further below.

2) Feature 2: Row Inspector: The row inspector connects each row of text in the pattern editor to its representation in the pattern visual. A user enables the row inspector tool by clicking on the 'Row Inspector' button and then hovers over different rows in the pattern visual to see which lines are highlighted in the text editor. As the cursor moves over the rows they are highlighted in yellow and blue in the two representations as shown in Figure 2.

The row inspector allows users to inspect how the program is running. Seeing this link is a part of the first tinkering principle, immediate feedback, and inspectors are the most frequently used tools in live programming environments [19].

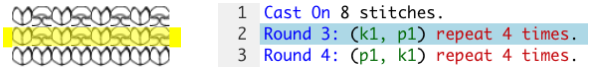


Fig. 2: The Row Inspector allows users to highlight over either a row of text in the DSL, or a portion of the diagram in the live view, and have the corresponding entity highlighted.

3) Feature 3: Error Inspector: There are two kinds of errors in StitchUp: syntactic errors and semantic errors. Syntax errors occur when the user writes a code fragment that does not conform to the DSL. Semantic errors occur when the knitted pattern is physically not possible to hand-knit. In knitting, each row builds on the previous row and unless new stitches are added (via increases) or are removed (via decreases), the number of stitches in the previous row must be the same as the number in the current row [25]. If the expected number of stitches in a row, which is the number of stitches in the previous row plus any stitches increased minus any stitches decreased, does not equal the actual number of stitches, a semantic error occurs.

Semantic errors are highlighted to the user in two places: a red highlight will colour the row of stitches in the visual representation and a red box with an 'X' beside the corresponding line in the text editor, as shown in Figure 3. For syntactic errors, only the red box with an 'X' beside the corresponding line will appear as that line cannot be parsed so it will not appear in the visual representation. In both cases, if the user hovers over the 'X' with their cursor, a message will appear explaining the reason for the error. Visual error signaling enables users to quickly determine when they had an issue and the cause of the error.

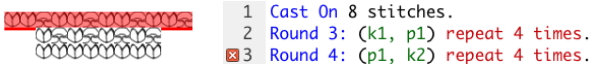


Fig. 3: The Error Inspector helps the user discover the correspondence between a DSL error and where this problem is manifested in the visual output of the pattern.

4) Feature 4: Undo, Redo, and Reset: Undo, redo, and reset features were included to increase risk taking and to support *repeated tinkering*. Repeated tinkering occurs when a user makes an action and their next action immediately reverts that change [12]. Undo reverts the last text change, measured as the text that changed within a 200ms interval. Lastly, redo reverts undo and reset actions in reverse-chronological order.

C. DSL Design

The semi-standardized notations used by knitters allowed us to build a knitting-specific DSL with which knitters would be immediately familiar [28]. We created our own DSL because while there are existing knitting DSLs, current solutions are either not formally defined [26], or contain notations designed for professional programmers rather than end-users [27].

The primary design goal for our DSL was to build a similar language to knit-speak which is used in practice. To build our knitting DSL, we reviewed the most popular patterns on Raverly, a pattern database which contains over 500,000 knitting patterns [29]. We examined several of the top patterns and created a DSL that minimally bridged the differences between these patterns. We added flexibility to our language by ensuring that capitalization, commas, and periods did not affect compilation [26]; lines that did not start with a DSL keyword were comments, as is common domain practice.

Figure 4 contains an example of how the text from a knitwear pattern is represented in our DSL. The Raverly instructions and StitchUp version create the same fabric in the visual representation. StitchUp uses colour to visually signify parts of the language. The beginning of rounds were coloured blue, the stitch specifications green, and the repeats red.

IV. Methodology

We created the StitchUp environment to enable knitters to interactively design and explore knitting patterns with tinkering-inspired features. To answer our research questions, we performed a controlled experiment with a between-groups design where every participant used the StitchUp environment and were assigned to groups based on their age. The age groups were 19–29, 30–39, 40–49, and 50+. These age groups were chosen between participants needed to be 19 to take part in the study and prior research [4] suggested that 50 is a more appropriate age threshold to avoid minimizing the concerns of older users. The experiment consisted of a 25-minute online study where participants performed two tasks using StitchUp and answered survey questions about their experience.[1] We chose this methodology in order to better compare age-based cohorts in a controlled setting. The findings from this experiment will be used to improve the StitchUp environment before evaluating more open-ended design tasks in a future study.

A. StitchUp Prototyping

Before releasing the study to the public, the study, and consequently StitchUp's design, went through five testing iterations to ensure participants could understand the task descriptions and understand how to use StitchUp. These iterations were crucial to ensure that the tasks and tool were easy enough to understand to complete in the time available to our experimental participants. Each iteration involved a tester whom had no prior knowledge of the study or StitchUp. During testing, the tester completed the study while an author observed, and the tester was encouraged to speak aloud as they worked. At the end of each tester's session, the study and StitchUp were revised based on their feedback. For example, we removed a confusing interactive tutorial and replaced it with a short video introducing StitchUp. We stopped after five rounds of updates as the last tester did not encounter challenges that required changing the protocol or tool. The testers were diverse, the youngest in their 20s and three being 50+ and had varying knitting and programming experience.

B. Tasks

The study consisted of two tasks. The first was a *training* task that introduced StitchUp before giving participants open-ended prompts so they could explore and tinker within the environment. The second *experimental* task reviewed StitchUp's features again, before asking participants to evolve a pattern design. Before joining the study, participants were provided with a consent form outlining what they would expect, the risks associated with the study, and a description of the study compensation. For compensation, participants could enter a raffle for one of five knitting prizes such as yarn or specialty bags for holding knitting supplies. After consenting to the study, participants completed an entrance survey that collected demographic information and played a short video demonstrating how StitchUp worked. They filled out a short survey after each task that reflected on the task. At the conclusion of the study participants completed an exit survey that contained open-ended questions about their experience with StitchUp. Both the training task and experimental task were broken down

[1]The study was approved by the University of British Columbia's Research Ethics Board [Certificate #H22-03556].

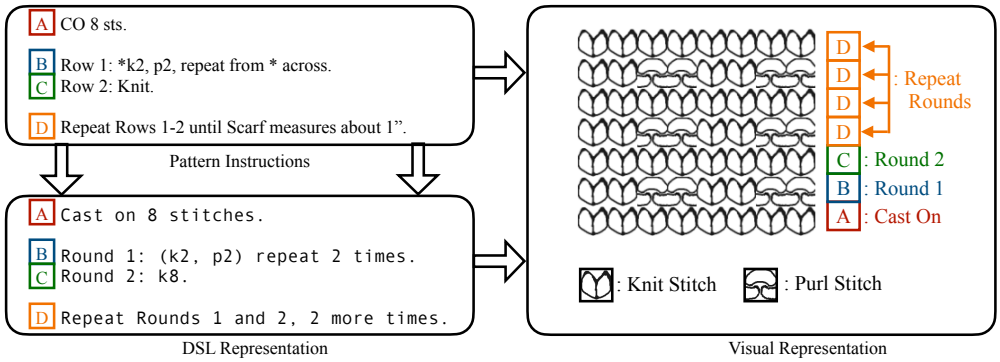


Fig. 4: A diagram representing how instructions from a pattern are represented in our DSL and the corresponding visual shown in StitchUp. The pattern instructions are from the Heartwarming Knit Scarf by YarnInspirations and have been slightly modified to reduce the size and complexity [30].

into steps which would progress the participant through the task. The survey questions, and training tasks have been made publicly available.[2]

Training Task. Participants were first introduced to the StitchUp environment with an 80 second video[3] and an interactive guided walkthrough tutorial. The walkthrough was anchored by a small pane that provided instructions for each step and how the participant could use the tool to perform it (see the top-left pane of Figure 1). Participants could move between the steps in the tutorial with next/previous buttons present below each set of instructions, but could also navigate to any past or future steps by selecting them from the list of steps above the instructions. The tutorial described each feature of StitchUp while asking the participant to complete a step that used a specific StitchUp feature:

1) Edit the pattern to introduce a semantic error, which triggered a red highlight row in the pattern visual and a red 'X' in the pattern editor.
2) Use the Row Inspector to highlight the error visually.
3) Use the Error Inspector by hovering over the red 'X' to see the error message.
4) Use the 'Undo' or 'Reset' buttons to remove the error.

After completing these steps, participants were provided with a few open-ended design suggestions that they could use to try to tinker with their design from the tutorial task.

Experimental Task. A guided walkthrough pane was also used to describe the experimental task, but in this condition the steps only included instructions that described what the participant was to perform, not how to accomplish it. As with the tutorial task, participants were able to navigate between the instructions for the different steps at any time. The steps for the experimental task were:

1) Review the tool by editing a specific line in the pattern and viewing the change in the pattern visual.
2) Read a short pattern for a sweater sleeve.
3) Solve a problem: Add a cuff to the sweater sleeve pattern, with a specified stitches (width), specified rows (length), and given texture (stitch combination).

For the experimental task, we developed a 5-point rubric to objectively score how participants performed:

1) Failure: The cuff had the wrong texture, the wrong number of rows (too short or too long), or was missing.
2) Wrong Texture: The cuff has the correct width and length, but it had the wrong texture. We asked participants to create a cuff of one-by-one ribbing which is `k1, p1`, so an incorrect texture might be `k2, p2`.
3) Too Short: The cuff has the correct width and texture but too few rows.
4) Too Long: The cuff has the correct width and texture but too many rows.
5) Success: The cuff had the right width, length, and texture.

Both the training task and experimental task were timed and the participants were directed to the survey after their time expired. To minimize frustration for participants, those who wanted more time to explore their designs could repeatedly extend the experimental task by two minute increments.

C. Participant Recruitment

The target population for our study was people who are familiar with knitting instructions. Since StitchUp is a browser-based tool, the study was run remotely and asynchronously. By building a browser-based environment and using online surveys we were able to have an easily communicated and shared link. To reach our target audience we deployed two recruitment strategies: snowball recruitment, and sharing on social media.

For the first strategy, snowball recruitment, we asked our personal network of knitters and friends to share the study with those they knew were knitters. The final page of the study thanked participants and asked them to consider sharing the study with one or two knitters. Also, one author worked at a knitting store and the store owner distributed the study information to all other staff members.

[2]www.katkerr.ca/projects
[3]https://www.youtube.com/watch?v=3oZ0SgzW0m0

We also used social media by publishing the study details on Instagram and YouTube. The five prizes for our study were knitting related and were purchased from fellow local artisans. Many participants chose to re-share our Instagram posts, including two of the creators of our prizes and a prominent local knitting conference account. In addition, the website for our study hosted a free pattern which the researcher had designed themselves. Through all the above methods, 155 people started our study.

D. Tool Data

The StitchUp environment was heavily instrumented to learn how participants interacted with the environment; we measured all clicks, hovers, and typing events. Each action was recorded as an event containing the timestamp of the action, who performed the action (via an anonymized id), which task the the action was performed in (training or experimental), the name of the action, and a general field for any additional information about the action (like the text in a typing event).

Each environment feature had corresponding actions. The Row Inspector had click and hover actions as a user clicks to enable the feature, hovers over the visual rows to inspect them, and then clicks again to disable the feature. The Error Inspector had hover actions as a user hovered over the error to see the detailed error message. However, errors arose through a participant typing a semantically- or syntactically-invalid pattern. Through analyzing the text within the editor, we can determine when an error occurred and whether it was semantic or syntactic. Undo, redo, and reset were recorded through click actions as they were performed by clicking a button. In addition, there were start task, end task, and extend task events. Lastly, we recorded which step of the tasks users were on, by recording click events on the next/previous step buttons.

E. Data Analysis

All open-ended questions were analyzed with an open coding approach [31]. Responses were split into one-sentence segments. Three coders met together and collaborated as a group to assign all sentences to codes, creating new codes as needed. Two subsequent rounds of theoretical coding were subsequently performed by one author. During this process the initial codes were organized into a hierarchy, and then each quote was reviewed to ensure it was assigned a code that fit the quote given the final hierarchy.

The collected tool usage data was analyzed for each participant for each task to identify which features they engaged with. The experimental task rubric was manually applied to the final programs for each of the participants to assign them a performance score for this task.

V. Results

To answer our research questions about participant's StitchUp experiences, we performed a mixed-methods analysis combining the qualitative data for each participant's thoughts and quantitative tool data for each participant's actions.

A. Participant Demographics

Over 6 weeks, 155 people started our study; however, we required complete responses for our analysis, so the 55 participants who did not finish the study were removed (65% completion rate). The remaining 100 participants who completed the study were reviewed; an additional 9 were removed due to: bad participant fit (4 people), choosing not to answer the age question (1 person), having no interaction with the training or experimental task measured by no typing events (3 people), or taking a break over 10 minutes while working on a task (1 person). The reasons for removing a participant due to bad fit were: being unfamiliar with knitting instructions (2 people), completing the study on a tablet (1 person), or completing the study with a friend (1 person). Ultimately, 91 of the original 155 participants remained in the study.

The final pool of participants were age-diverse: ages 19–29 (17 people), ages 30–39 (21 people), ages 40–49 (21 people), ages 50–59 (13 people), ages 60–69 (15 people) and age 70+ (4 people). In alignment with prior findings [4], we combined the final three groups into one 50+ age group (32 people). Of the 91 participants, 88 were women, 2 were men and 1 was non-binary. Participants self-reported their experience in two dimensions that could influence their ability to use StitchUp: domain experience (knitting instruction familiarity) and programming experience. Chi-square tests showed no significant differences between the age groups with respect to knitting instruction experience; however, the 19–29 group's programming experience was significantly different than the others ($p = 0.02$), with a trend towards more programming experience. Participants are referred to as $p1 - p91$.

B. Code Book

We coded over 1,200 sentences from 534 participant responses to develop a hierarchical code book containing 88 codes. The topmost categories were *Experiences*, *Tool Suggestions*, *Study*, and *Noise*. Since we coded individual sentences from long answer responses, the noise category was necessary to code sentences that were unrelated to the study, tool, or domain; for example, when a participant's response could not be understood (*Unable to Decipher*), or if they mentioned they had answered a question previously in the survey (*Already Answered*). Only eight responses were solely categorized as noise ($<2\%$).

To answer our three research questions we focus on *Experiences*, the most popular category of responses. The *Experiences* category contains 51 codes split into the following subcategories: *Overall Experience*, *Task Completion Experiences*, *Feature Experiences*, *Domain Experiences*, and *Emotions*.

To investigate **RQ1**, we examined the success rate for the experimental task. **RQ2** examined participant's *Task Completion Experiences* and how they iterated towards successfully completing the experimental task. Lastly, to investigate **RQ3**, we examined participant's *Feature Experiences* for all tasks.

C. RQ1: Age-Diverse Task Success

RQ1: Can age-diverse end users successfully complete design tasks in StitchUp?

Our first research question sought to investigate whether all age groups could be successful at the experimental task; to do this we compared the results from each of the groups from our between-groups experimental design with respect to whether they could complete their assigned task. Although the goal of design exploration tasks is not typically *time minimization*, the experimental task had a clearly-defined definition of success; consequently we measured success with respect to whether the stated task was accomplished in the provided time.

To incorporate both success and time to task completion, we performed a *Survival Analysis* on participant data [32]. In the context of our study, survival means that a participant has *not* solved the task at a given point in time. The survival curves for our participants, split by age, can be seen in Figure 5; although the curves in this figure have significant overlap, one can observe that the 95% confidence intervals for the 50+ group have minimal overlap with the confidence intervals for the other groups. As participants could extend the time they had available for each task as they liked, we chose 19 minutes, the maximum time that any participant decided to spend on the task, as the censoring threshold for our analysis. As the flat tails of the curves suggest, few participants benefited from taking extra time and none made additional progress after 8 minutes on the experimental task. A log-rank test shows that there is a statistically significant difference between one or more of the survival curves for the different age groups ($df = 4, \chi^2 = 12.41, p = 0.01$). A pairwise analysis further shows that there is a statistically significant difference between the 50+ age group's survival curve and all remaining groups ($19-29 : p < 0.005, 30-39 : p < 0.005, 40-49 : p = 0.02$). The remaining pairwise combinations amongst all other age groups are not statistically significant.

This analysis demonstrates that participants in the 50+ age group were meaningfully different in terms of experimental task success than their 19–49 year old peers. Additionally, the 50+ age group struggled more than the other groups, exhibiting the lowest success rate with the experimental task (9% success) compared to participants aged 19-49 (34% success). Given this difference (and in the interest of space), for the remainder of the paper we report all results with these two groups in mind: 19–49 (59 participants) and 50+ (32 participants).

RQ1 Summary

While participants from all age groups were able to succeed, the task completion analysis showed that the 50+ age group was less likely to complete the experimental task, as the rate at which they completed the task was significantly different than the other age groups.

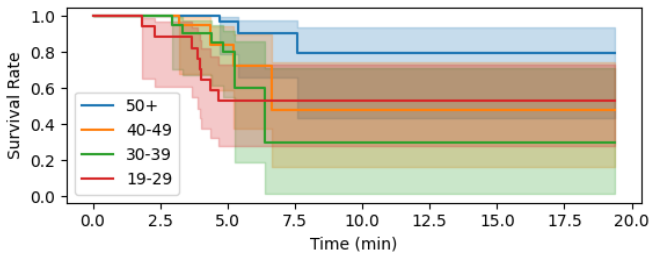


Fig. 5: Kaplan Meier Survival Curve for the experimental task. The curves are split by participant age group; shaded areas are 95% confidence intervals for each curve. The difference between the 50+ curve and each other curve is statistically significant, but the remaining curves are not statistically significantly different from each other.

D. RQ2: Participant Task Strategies

RQ2: What strategies do age-diverse end users use when performing tasks in StitchUp?

The two sub-codes below *Task Completion Experiences* revealed opposing states that participants transitioned through while working on their tasks: *Tinkering* and *Stuck*. We examined the experimental task data to gain insights into how participants progressed through their tasks.

1) Code Book Analysis: When a participant is *Tinkering*, they feel like they are making progress on their task. When they are *Stuck*, they enter a state where they are not making effective progress. Both the *Tinkering* and *Stuck* codes contained two subcategories each: *State of mind* and *Enabling*. The majority of participants made comments related to *Tinkering* (60%) and almost half to being *Stuck* (48%). Next we further examine what *State of mind* the participants were in and what *Enabled* them to have these *Task Completion Experiences*.

2) Tinkering: In terms of their *State of mind*, participants made comments about being *Engaged*, *Curious*, and *Confident* while they were *Tinkering* with their designs. *Engaged* was the most commonly mentioned state of mind, with roughly a third of participants making a statement that showed engagement. We modeled engagement with the 'Tinkering Learning Dimensions', where engagement is characterized as an investment in time, thought, and emotion and can be indicated by play, exploration, and emotional investment [33]. Not all emotional investment needs to be positive:

> "*Great interest, with a sprinkling of frustration at getting used to the interface and syntax*" - $p88$ $(50+)$

> "*Just by visualising a sleeve for instance and how different it may look with different ribbed patterns made me smile and wanted to play with it*" - $p47$ $(40-49)$

Curiosity was a prominently mentioned emotion:

> "*Puzzled but not frustrated*" - $p40$ $(40-49)$

Several *Enabling* codes, which capture entering the *Tinkering State of mind*, captured how participants felt they were able

to make progress. For example, *Experimenting* was often mentioned as it helped with alternative evaluation:

> "*You can see in the visual how the pattern changes and you can see if something is wrong with the stitch count right away. And it encourages to try something new.*" - $p85$ $(50+)$

The *Low risk for changes* code was frequently cited as a benefit of digital experimentation relative to the effort required to actually knit a physical piece. Finally, several participants reflected on how they were able to *Become unstuck*, which often related to reviewing error-free parts of the pattern or the instructions for syntax hints.

> "*I got stuck on the problem of making the row repeat, I figured it out by looking back at how the pattern coded row repeats and taking that action.*" - $p70$ $(50+)$

3) Stuck: Almost half of participants made references to being *Stuck* (48%). The *Stuck State of mind* codes are *Confused* and *Stuck*. These codes relate to participants exclaiming they are confused or stuck, whereas the *Enabling* codes relate to how they got stuck or what lead to the confusion. 42% of all participants mentioned encountering confusion due to syntax, the visual representation, or the domain. Syntax confusion was the most common, with 32% of participants mentioning some problems forming the correct syntax:

> "*A couple times I tried putting in notations that weren't supported by the program, I tried working around this by trying to put it in terms I thought the program would understand.*" - $p1$ $(19-29)$

Visual confusion occurred when there was a divergence between what the participant expected to see and what StitchUp actually displayed for them:

> "*Why did it show the first stitches at the bottom vs. the top? maybe because that's what will happen as you knit the piece. I kind of thought that it's order would line up with the instructions.*" - $p61$ $(50+)$

4) Experimental Task Data Analysis: We examined the experimental task data in two ways: first, by analyzing the *Tinkering* and *Stuck* states in terms of sessions (Figure 6) and secondly, by analyzing the states participant's moved through to complete the task based on our success rubric (Figure 7).

We partitioned each participant's environment event streams into sessions. 'Tinkering' sessions occurred when a sequence of events occurred that contained no errors. 'Stuck' sessions occurred when at least one error was being displayed to the participant. In this way, a participant could iterate between tinkering and stuck sessions multiple times while completing a task. In terms of average sessions, the 50+ group had fewer tinkering and stuck sessions (avg 2.8 and 2.2 sessions) than 19–49 group (avg 4.3 and 3.7 sessions). The 50+ group had more events in both their tinkering and stuck sessions (median 19 and 17 events), relative to the 19–49 group (median 15 and 14 events). There was one session with greater than 100 events in the 50+ group and two sessions with greater than 100 in the 19–49 group. Figure 6 shows box plots of the tinkering and stuck session lengths and average events per session per participant. This shows that while younger participants introduced more errors, they also required fewer events to resolve them. These quick transitions through stuck states resulted in the younger participants being able to explore more alternatives, which may have contributed to their overall higher success.

Based on the survival analysis, we examined how participants moved through the experimental task for two groups, those aged 19–49 (59 participants) and those aged 50+ (32 participants). Figure 7 depicts how these two groups progressed through the experimental task. We evaluated the intermediate states of the 91 programs according to our rubric, described in Section IV, to determine how each participant generated their final pattern. While the results show a higher proportion of participants failing to make meaningful progress in the 50+ group, a more interesting finding is that the 19–49 group were better able to transition through partially-successful states by trying different alternatives. For example, for the 19–49 group we observed participants making designs that transitioned from `wrong texture` to `too short` to `to long` before finally reaching `success`. These incremental transitions through multiple partial success states was far more common for the 19–49 group (83%) than the 50+ group (34%).

RQ2 Summary

Participants in the 50+ age group were less likely to evolve their patterns through different partially-correct solutions than their younger counterparts (Figure 7). Also, the 50+ age group spent longer in each *tinkering* and *stuck* session and iterated less between the *tinkering* and *stuck* sessions (Figure 6).

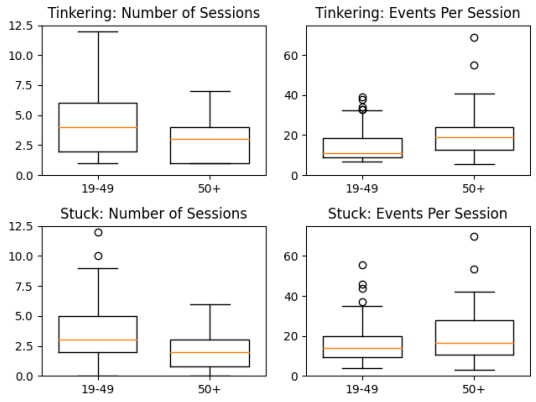
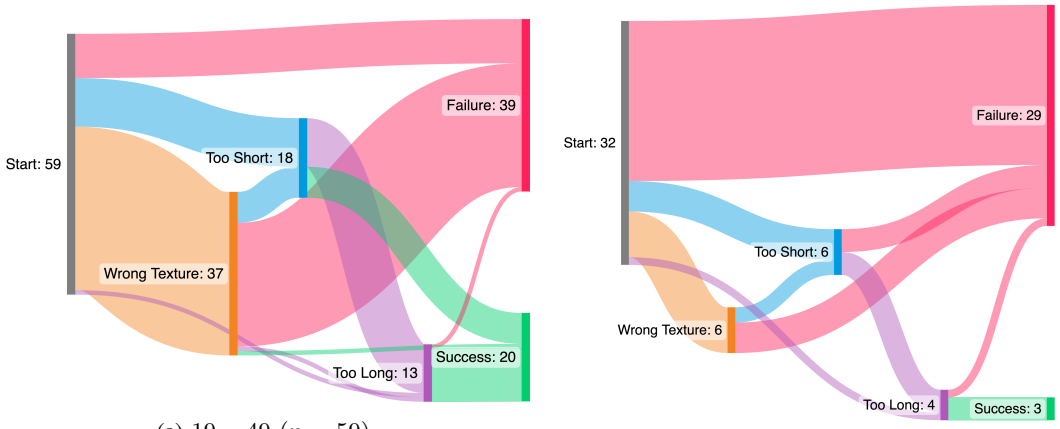


Fig. 6: Box plots of the number of sessions and the events per session for tinkering and stuck states. Three outliers were elided from 'Stuck: Events Per Session' for consistent y-axes: 109 and 319 from 19–49 and 139 from 50+.



(a) $19-49$ $(n=59)$ (b) $50+$ $(n=32)$

Fig. 7: Sankey diagrams showing how the different groups evolved their solutions during the experimental task. Most successful solutions evolved through more than one partial-success state, which was more prevalent for the 19–49 group.

E. RQ3: StitchUp Tinkering Feature Usage

RQ3: Are StitchUp's tinkering features used consistently by all age groups?

Participants provided a broad array of comments about the tinkering environment features.

The *Pattern Visual* feature was mentioned by two-thirds of all respondents (66%). The most common kind of feedback about the live updating visual was that participants *Enjoyed* having visual feedback for their changes (36% of participants). For example,

> "*I love that there's a visual representation that updates as you adjust the pattern.*" - $p4$ $(19-29)$

22% of participants specifically mentioned enjoying the live nature of the visual view and its immediacy after changes:

> "*I really liked how edits to the pattern were immediately reflected in the visual. That was very helpful.*" - $p24$ $(30-39)$

Many participants (20%) talked about how they used the visual representation to either confirm a hypothesis or that they learned something from the visual aspect of a change.

> "*That the visual is very important in ensuring the accuracy of the written pattern*" - $p52$ $(40-49)$

The *Error Inspector* (21%), *Row Inspector* (13%), and *Undo, Redo, and Reset* (9%) were mentioned less frequently, although this feedback often mentioned how they helped with iteration:

> "*it was easy to change things or undo if I didn't like it. The visual updating of the row inspector made it easier to see what was happening.*" - $p29$ $(30-39)$

Examining the tool usage data, we see that participants infrequently used the Undo, Redo, and Reset feature (only 14% used it more than once for the experimental task). The row inspector, which had to be manually engaged, was used more frequently (55% invoking it more than once). When the row inspector was invoked, participants used it to examine how an average of 6 lines of DSL mapped to their visual representation. The error inspector, which highlighted syntactic and semantic errors in the DSL, was used the least (12% invoking it more than once); however, it was used in ways we did not expect. Given the difference in solving the errors introduced by the different age groups in Section V-D highlighting these errors in the visual view was likely more helpful for the younger participants than the older ones. The data shows that stuck sequences were longer (67 events on average) when the error message was examined, compared to stuck sequences where the error message was not examined (16 events on average). This further reinforces the hypothesis that the visual view was more likely used to resolve errors, as they error inspector was turned to as a last resort.

When examining the tinkering and stuck sessions in Section V-D, a primary difference between the two groups is not specific tool features that are used, but how the tools are used. This is most prominently visible in terms of the iterations (sessions) that the two groups underwent, as is shown in Figure 6. From these box plots we can see a decreasing trend in the number of iterations for the groups as the age increases. Some older participants explicitly expressed concern about introducing errors while they worked:

> "*I didn't play because I thought I would mess it up*" - $p69$ $(50+)$

RQ3 Summary

Across age groups, the immediate visual representation was the most used and best perceived StitchUp live programming feature. Supporting rapid iteration seems to be more important than specific environment features.

VI. Discussion

Our goal was to evaluate a tinkering-focused environment with an age-diverse group of users to learn whether live programming features could provide insight into the age-inclusivity of tooling for supporting making activities.

a) Mindsets While Working: While working through the tasks, participants reported states of mind and working styles analogous to tinkering: engagement, curiosity, experimentation and working through problems (becoming unstuck) [8]. Interestingly, participants who were able to successfully complete the experimental task were more likely to mention engagement (3.6 times), curiosity (1.8 times), and experimentation (1.1 times) when compared to unsuccessful participants. These results corroborate previous findings where cognitive playfulness was linked to increased performance [11].

b) Iteration and Success: While the analysis in this work has focused mostly on age differences while engaging with tinkering environment features, participants of all ages were successful with the experimental task. This shows that age is not a discriminator that makes in impossible to succeed, suggesting that perhaps *how* successful participants worked could also provide clues for success. Comparing successful participants to unsuccessful participants, ignoring age, reveals some additional interesting features: Successful participants iterated more when working through their tasks, on average moving between two partially-successful solutions before ending at success as seen in Figure 7. Unsuccessful participants moved through an average of 0.5 states before ending at failure. Successful participants also had more stuck and tinkering sessions (avg 12) than unsuccessful participants (avg 5).

Importance of Iteration. While participants of differing ages interacted with the provided tools in unequal ways, unsuccessful participants, across all ages, had similar behaviours. This suggests that encouraging successful behaviours, in particular encouraging iteration, may be one way to improve overall tool effectiveness providing an important pathway for older participants to be successful.

c) Relationship to Programming IDEs: StitchUp draws inspiration from existing live programming tools that are built into Integrated Development Environments (IDEs). How the end-users in our study interacted with StitchUp's live programming tinkering features could provide insight into how older end-user programmers might interact with these kinds of features in more traditional IDEs. The same tensions that led us to examine how older people should be supported in maker communities also applies to software programming as populations age.

A. Threats to Validity

Our goal with this work was to discover how environment features supported or inhibited users of differing ages.

Construct Validity. We stratified ages into four groups, although we acknowledge the cutoff between each of these groups mean that a 39 year-old and a 41 year-old are treated more differently than their two year age gap might suggest is relevant. While it may not be surprising that participants under 50 had greater task success than those over 50, a meaningful portion of software developers are above 50 years of age and supporting them can only help tools to be more effective for the overall population [34].

Internal Validity. Selection biases may have influenced who chose to take our study. While technically-literate knitters may have been more likely to choose to take the study, the balanced set of ages suggests this is either unlikely, or that the demographics of knitters more broadly skews older, increasing the pool from which the participants in the older age groups could be drawn from. Additionally, we relied on participant-supplied data for their ages, genders, experience with programming, and experience with the knitting domain. Lastly, over a third of participants chose to extend the time within the tool while working on the experimental task, which shows the task time was short and might have added time pressure and stress on participants. We alleviated this stress by allowing participants to extend their time.

External Validity. There are three primary threats to the generalizability of the results in this work. First, the primary demographic factor we considered was age. Unfortunately, as 97% of our respondents were women, we were unable to consider gender demographics, and since we did not collect cultural background demographics, we were unable to consider them within our analysis. Second, our study only analyzed users in the specific domain of knitting design. Finally, it remains future work to investigate how our analysis will hold for other programming domains.

VII. Conclusion

As populations age it is crucial to ensure that older users remain equitably supported by technical innovations. We created the StitchUp environment to evaluate how an age-diverse set of end users were able to perform maker tasks within an environment that encouraged and supported tinkering. Our between-groups study investigated how 91 participants interacted with the StitchUp environment to complete knitting design tasks. Through a mixed-methods analysis, we found that participants enjoyed the immediate visual feedback provided by the environment which enabled them to rapidly experiment with their designs and gain insight and confidence in how their changes to the DSL for their pattern corresponded to changes in their design. Participants of all ages were able to complete their tasks, but the oldest participants had the highest failure rate. Examining their task completion behaviour, we found that older participants took fewer risks and iterated less on their designs, while spending more effort trying to get unstuck when they encountered problems. Younger participants were more willing to move through partial-success states while working. This suggests that one approach for improving the age-inclusivity of programming environments may be to encourage iteration through mechanisms that are perceived positively, enabling users to see their progress as they work.

References

[1] J. S. Hui and E. M. Gerber, "Developing makerspaces as sites of entrepreneurship," in *Proceedings of the Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, 2017, pp. 2023–2038.

[2] T. Capel, B. Ploderer, and M. Brereton, "The wooden quilt: Carving out personal narratives in a women-only makerspace," in *Proceedings of the Designing Interactive Systems Conference (DIS)*, 2020, pp. 1059–1071.

[3] D. Dougherty, "Maker market study," 2012. [Online]. Available: https://cdn.makezine.com/make/bootstrap/img/etc/Maker-Market-Study.pdf

[4] A. Lazar, M. Diaz, R. Brewer, C. Kim, and A. M. Piper, "Going gray, failure to hire, and the ick factor: Analyzing how older bloggers talk about ageism," in *Proceedings of the Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, 2017, pp. 655–668.

[5] GBD 2017 Population and Fertility Collaborators, "Population and fertility by age and sex for 195 countries and territories, 1950-2017: A systematic analysis for the global burden of disease study 2017," *Lancet*, vol. 392, no. 10159, pp. 1995–2051, Nov. 2018.

[6] J. Vines, G. Pritchard, P. Wright, P. Olivier, and K. Brittain, "An age-old problem: Examining the discourses of ageing in hci and strategies for future research," *ACM Transactions on Computer-Human Interactions (TOCHI)*, vol. 22, no. 1, Feb 2015.

[7] D. Dougherty, "The maker movement," *Innovations: Technology, Governance, Globalization*, vol. 7, no. 3, pp. 11–14, 07 2012.

[8] M. Resnick and E. Rosenbaum, "Designing for tinkerability," in *Design, Make, Play: Growing the Next Generation of STEM Innovators*, M. Honey, Ed. Routledge, 2013, p. 19 pages.

[9] Scratch. (2015) Scratch statistics - imagine, program, share. [Online]. Available: https://scratch.mit.edu/statistics/

[10] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, p. 16, 2010.

[11] J. J. Martocchio and J. Webster, "Effects of feedback and cognitive playfulness on performance in microcomputer software training," *Personnel Psychology*, vol. 45, pp. 553–578, 2006.

[12] L. Beckwith, C. Kissinger, M. Burnett, S. Wiedenbeck, J. Lawrance, A. Blackwell, and C. Cook, "Tinkering and gender in end-user programmers' debugging," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, 2006, pp. 231–240.

[13] B. Shneiderman, "Universal usability," *Communications of the ACM (CACM)*, vol. 43, no. 5, p. 84–91, May 2000.

[14] S. Papert and I. Harel, "Situating constructionism," in *Constructionism*, S. Papert and I. Harel, Eds. Ablex Publishing, 1991.

[15] S. Krieger, M. Allen, and C. Rawn, "Are females disinclined to tinker in computer science?" in *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)*, 2015, pp. 102–107.

[16] M. M. Burnett, J. W. Atwood Jr, and Z. T. Welch, "Implementing level 4 liveness in declarative visual programming languages," in *Proceedings of the Symposium on Visual Languages (VL/HCC)*, 1998, pp. 126–136.

[17] S. L. Tanimoto, "A perspective on the evolution of live programming," in *Proceedings of the International Workshop on Live Programming (LIVE)*, 2013, pp. 31–34.

[18] Y. Dong, S. Marwan, V. Catete, T. Price, and T. Barnes, "Defining tinkering behavior in open-ended block-based programming assignments," in *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)*, 2019, pp. 1204–1210.

[19] J. Kubelka, R. Robbes, and A. Bergel, "The road to live programming: Insights from the practice," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2018, pp. 1090–1101.

[20] Wool and the Gang. (2015) Our survey results are in! [Online]. Available: https://www.woolandthegang.com/blog/2015/02/our-survey-results-are-in

[21] M. CX. (2017) 2016 creative products size of the industry study. [Online]. Available: https://craftindustryalliance.org/wp-content/uploads/2021/11/AFCI-Presentation_012317_FINAL.pdf

[22] S. Martinez and G. Stager, *Invent to Learn: Making, Tinkering, and Engineering in the Classroom*. Constructing Modern Knowledge Press, 2016.

[23] B. Jones, Y. Mei, H. Zhao, T. Gotfrid, J. Mankoff, and A. Schulz, "Computational design of knit templates," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 2, 2021.

[24] Ravelry. (2023) Baby suprise jacket wiki page. [Online]. Available: https://www.ravelry.com/patterns/library/baby-surprise-jacket/wiki

[25] M. Hofmann, L. Albaugh, T. Sethapakadi, J. Hodgins, S. E. Hudson, J. McCann, and J. Mankoff, "Knitpicking textures: Programming and modifying complex knitted textures for machine and hand knitting," in *Proceedings of the Symposium on User Interface Software and Technology (UIST)*, 2019, pp. 5–16.

[26] J. Briar. (2023) Stitch maps. [Online]. Available: https://stitch-maps.com/

[27] C. Battell, "Domain specific language for modular knitting pattern definitions: Purl," 2016.

[28] C. Y. Council. (2023) Knitting abbreviations master list. [Online]. Available: https://www.craftyarncouncil.com/standards/knitting-abbreviations

[29] Ravelry. (2023) About ravelry. [Online]. Available: https://www.ravelry.com/about

[30] J. Farmer. (2011) Heartwarming knit scarf. [Online]. Available: https://www.ravelry.com/patterns/library/heartwarming-knit-scarf

[31] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, ser. Observations. Aldine Transaction, 1967.

[32] M. K. Goel, P. Khanna, and J. Kishore, "Understanding survival analysis: Kaplan-meier estimate," *International Journal of Ayurveda Research*, vol. 1, no. 4, p. 274, 2010.

[33] B. Bevan, J. P. Gutwill, M. Petrich, and K. Wilkinson, "Learning through stem-rich tinkering: Findings from a jointly negotiated research project taken up in practice," *Science Education*, vol. 99, pp. 98–120, 2015.

[34] S. Baltes, G. Park, and A. Serebrenik, "Is 40 the new 60? how popular media portrays the employability of older software developers," *IEEE Software*, vol. 37, no. 6, pp. 26–31, 2020.