Critter: Augmenting Creative Work with Dynamic Checklists, Automated Quality Assurance, and Contextual Reviewer Feedback

Aditya Bharadwaj
Virginia Tech
Blacksburg, VA, USA
adb@vt.edu

Pao Siangliulue
B12
New York, NY, USA
pao@b12.io

Adam Marcus
B12
New York, NY, USA
marcua@marcua.net

Kurt Luther
Virginia Tech
Arlington, VA, USA
kluther@vt.edu

ABSTRACT

Checklists and guidelines have played an increasingly important role in complex tasks ranging from the cockpit to the operating theater. Their role in creative tasks like design is less explored. In a needfinding study with expert web designers, we identified designers' challenges in adhering to a checklist of design guidelines. We built Critter, which addressed these challenges with three components: Dynamic Checklists that progressively disclose guideline complexity with a self-pruning hierarchical view, AutoQA to automate common quality assurance checks, and guideline-specific feedback provided by a reviewer to highlight mistakes as they appear. In an observational study, we found that the more engaged a designer was with Critter, the fewer mistakes they made in following design guidelines. Designers rated the AutoQA and contextual feedback experience highly, and provided feedback on the tradeoffs of the hierarchical Dynamic Checklists. We additionally found that a majority of designers rated the AutoQA experience as excellent and felt that it increased the quality of their work. Finally, we discuss broader implications for supporting complex creative tasks.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**;

KEYWORDS

Quality assurance; checklists; creative work; feedback

CHI 2019, May 4–9, 2019, Glasgow, Scotland Uk

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300769

ACM Reference Format:
Aditya Bharadwaj, Pao Siangliulue, Adam Marcus, and Kurt Luther. 2019. Critter: Augmenting Creative Work with Dynamic Checklists, Automated Quality Assurance, and Contextual Reviewer Feedback. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland Uk.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300769

1 INTRODUCTION

Navigating constraints in complex creative tasks is challenging. Creative professionals in various domains such as web design, software development, and music composition have to balance their creative expression with practical constraints and quality considerations. On one hand, they are expected to create novel work. On the other hand, they need to ensure the quality of work that they produce. While we often pay more attention to the novelty of creative work, the quality and the repeatability of creative products is no less important.

To ensure quality in creative work, fields such as design and software engineering gravitate toward best practices such as design guidelines [21, 24] or coding style guides [44]. For example, web designers have to balance delivering websites that are aesthetically compliant with customers' requests while also following best practices around accessibility and responsivity. Design guidelines, while offering standards that should improve quality, pose their own challenges [30, 43]. Multi-page guidelines have multiple inherent usability drawbacks in areas like searchability, conflicts, and obsolescence. To our knowledge, there has been little exploration of how to effectively incorporate design guidelines into the design process without disrupting designer creativity [24].

Checklists are effective in adding structure and repeatability to complex processes, facilitating the enforcement of guidelines on a per-project basis [17]. Even in areas that require expertise like aviation and surgery, experts rely on checklists to ensure a standard of quality: pilots' every step in the cockpit is guided by a checklist [7, 20], and surgeons increasingly look to checklists to improve outcomes [4, 19]. While fields that require precision like automobile engineering [48] or construction are natural candidates for checklists

[17], there is also room for checklists in creative and semi-structured pursuits such as design or software engineering. Still, performing these creative tasks is not as simple as following protocol. The creative process is complex, dynamic, and non-linear [35, 41]. For example, successful designers can have drastically different workflows, and designers vary their approach on projects of different complexity or scope. Finally, some aspects of design are iterative, making the application of step-by-step checklists more challenging.

We investigate the iterative and real-world deployment of checklists for enforcing design guidelines with a team of expert web designers. In a needfinding study, we identify the root cause for the poor adherence to checklists and an effectively lower-quality work product: our initial checklist implementation lacked the dynamicity necessary for diverse design projects. While review and manual quality assurance helped improve quality, they proved expensive and did not promote an improvement in expertise.

Informed by these findings, we created Critter, a mixed-initiative system that helps experts efficiently create effective checklists that dynamically adapt to individual project requirements. Critter features three key components: Dynamic Checklists, AutoQA, and contextual reviewer feedback. Dynamic Checklists allow experts to create a customized checklist by skipping guidelines that they consider irrelevant to the project or their design process. Related checklist items are grouped and hierarchically nested to progressively disclose details. For each project, a checklist is automatically pruned to remove irrelevant guidelines (e.g., single-page website guidelines for a multi-page website). For bespoke requests, experts can use the Dynamic Checklists to add checklist items they will want to complete later in the project. Critter also features an automated critique system, called AutoQA, which performs an automated quality assurance check for common errors identified in the checklist (e.g., in the web design domain, AutoQA identifies errors in content, aesthetics, responsivity, and consistency). Finally, Critter allows reviewers to provide feedback around specific design guidelines that are highlighted in Dynamic Checklists on future projects to promote learning and iterative improvement.

To evaluate Critter's effectiveness, we conducted an observational study with professional web designers creating websites for 30 real-world clients at B12[1]. In the study, designers used Dynamic Checklists and AutoQA as part of their process while receiving contextual feedback from B12 employees. We found that Critter was able to effectively incorporate design guidelines into a designer's process and consequently create higher quality websites. Based on our findings, we discuss broader implications of our work to other creative domains with numerous best practices, idiosyncratic client requests, and nonlinear completion paths (e.g., designing, writing, programming).

In summary, our contributions include:

- A novel approach, informed by a needfinding study highlighting gaps in traditional checklists, that combines automation with human-driven planning and reflection to help experts navigate complex creative tasks. Specifically, we augment checklists with a) a hierarchical structure to mask complexity and offer navigability, b) automation to prune and promptly check guidelines whenever possible, and c) self-reflection through guideline-specific feedback.
- A system, Critter, that exhibits this approach with three components: a) Dynamic Checklists that are hierarchical and self-pruning, b) AutoQA to instantly identify common faults, and c) Contextually presented reviewer feedback.
- An empirical, mixed-methods evaluation of web designers using Critter to create websites for 30 B12 clients. We found that the more engaged a participant was with Dynamic Checklists and AutoQA, the fewer mistakes they made in following design guidelines. Participants rated the AutoQA and contextual feedback experience highly, and provided feedback on the tradeoffs of a hierarchical display.

2 MOTIVATING EXAMPLE: WEBSITE DESIGN

To motivate our work, we present an example based on the first-hand experience of some of the authors who work at B12, a company that works with web design experts to create websites for their customers. Initially, web designers creating websites would, upon starting a new project, receive a customer brief. This brief contained semi-structured aesthetic preferences, functional requirements, feedback, and content that a new customer provided by filling in a questionnaire. After reading the brief, designers followed different paths. Some would immediately start designing the website. Others would jot down reminders to key customer requests in a note-taking application of their choice. After completing a draft of the website, designer behavior again varied. Some designers simply submitted the draft website whereas others would go through the list of their reminders to ensure they hadn't missed any details.

As a set of best practices emerged, B12 presented designers with design guidelines in the form of a Google Spreadsheet containing 153 distinct guidelines. These guidelines were curated by senior designers at B12 to serve as a template checklist for any new project. Guidelines included topics such as website structure, copy, imagery, and aesthetics, with major topic areas depicted in Figure 1A. The spreadsheet[2] used indentation and colors to reinforce a visual hierarchy, had a column to mark guidelines as completed, and offered macros for pruning irrelevant guidelines.

[1] https://www.b12.io/

[2] available at http://marcua.net/papers/chi2019-needfinding-checklist.pdf

While the guidelines would change with time, most designers rarely reviewed or checked off the guidelines after their first few projects. A few designers would create a short checklist at the start of each project and consult both the guidelines and their checklist before submitting a draft website for review. However, it was common for many designers to do neither of these. While designers provided expertise and kept key customer requests in their memory, they would still miss explicit customer preferences or B12 design guidelines. This poor adherence resulted in customer frustration, and researchers at B12 sought a system to better present guidelines, assure quality, and capture feedback.

3 RELATED WORK

Checklists and Todo lists

Checklists have been shown to provide multiple benefits, including reminding users of critical steps, creating consistency, enforcing regulation of policies, and offering a framework for evaluation [37]. The primary purpose of many checklists is essentially quality assurance through error reduction and guideline adherence [13]. Checklist usage varies considerably by domain. For example, in the aviation industry, checklists are standardized and compulsory [7], and completion of checklists from memory is considered a protocol violation [17]. In contrast, design guidelines are self-imposed and fungible (e.g., it is common for designs to hold aesthetics and structure in tension), reinforcing prior work that emphasizes the non-linearity of complex work [33]. In Critter, the checklists are dynamic and adapt to project requirements and prior experience. Additionally, we ask the experts to use the checklists at specificity level they see fit in their workflow.

With time, checklists have been digitized, showing advantages over traditional media. For example, the Boeing 777 Electronic Checklist, developed in the early 1990s, decreased errors by an additional 46% as compared to paper-based checklists alone [5]. Critter builds on this research by adding human-computer interactions like self-pruning and toggling that afford dynamicity to the checklists.

Design guidelines

Design guidelines are sets of rules designers should follow to ensure that their design artifacts are up to standards. Design guidelines are widely developed and applied in various fields such as user interface design [14, 21, 24], web design [27], interior design [29], and software development [44]. They can ensure the quality of design products and reduce stress of the designers [24]. Some design guidelines are products of expert judgement, common sense, and practical experience [6] while others are derived from more rigorous testing [27].

However, design guidelines alone are not always effective. While unclear guidelines discourage designer usage [42], long guidelines make it difficult to find relevant guidelines [24, 30]. Further, new design guidelines that differ from designers' previous experience are not effective even when the designers are motivated to follow the new guidelines [43].

Prior work explores various approaches to make design guidelines more effective. One approach involves integrating the guidelines to the design tools by detecting or preventing deviations from guidelines. For example, Merrell et al. implemented a system that proposes alternatives that comply with the guidelines [29]. With the guidelines embedded in the system, the designers do not have to double-check them, reducing the error in interpretation. This approach, however, only works well with computable guidelines and is less flexible because the guidelines are embedded tightly into the system. A different approach makes it easy to find relevant guidelines from a collection by reorganizing them [27] or making them searchable [21]. This approach helps designers save time and mental effort in locating relevant guidelines. Another approach adopts examples to clarify ambiguous guidelines [21]. In Critter, we use AutoQA to automatically identify deviation from guidelines, Dynamic Checklist to automatically prune irrelevant guidelines, and contextual reviewer-provided feedback to reminds the designers of guidelines that they missed. Finally, a hierarchical checklist structure affords progressive disclosure for non-linear workflows.

Feedback

Expert feedback is important in any job. It helps its recipients to grow their knowledge [18], learn about best practices [22], produce better results [8], and avoid mistakes in future [16]. It allows the recipients to reflect on their performance and identify their strengths and weaknesses [46]. A related thread of work considers how to provide effective feedback, especially to help novices. In general, good feedback is specific, actionable, and contains an explanation [31]. Kulkarni et al. used examples as feedback to improve creative work [25]. Yuan et al. used rubrics to structure design feedback [47]. However, in semi-structured tasks, human experts are susceptible to judgment errors [40], suggesting opportunities for mixed-initiative support tools. Specifically, experts like designers have a memory capacity and can forget a guideline while reviewing the work. Critter addresses this issue by using design guidelines as rubrics, ensuring consistent feedback supported by direct and actionable design guidelines.

Automated quality assurance

An automated critique system or quality assurance checker scans the work submitted by the user for possible violations of design guidelines in a given field [10]. Automated quality assurance is a common practice in domains ranging from structured tasks like manufacturing [15] and medical device engineering [11] to creative tasks like writing [45], software development [9], and design [10]. Embedding such automated critics in work practice can help designers learn how to identify and solve problematic situations early in the design process [10]. In design, trust plays a critical role in the effectiveness of such systems. For example, a study found that designers rated a critique tool as valuable, provided that they feel they are in control and the tool indicates the severity of the detected error [28]. The automated critic in Critter, AutoQA, helps designers to catch errors based on design guidelines. In this paper, we study the relative benefits of automated critique, self-review, and reviewer feedback.

4 NEEDFINDING STUDY

To understand how designers used existing quality assurance techniques — a spreadsheet-based checklist discussed in Section 2, an early prototype of AutoQA, and reviewer-provided feedback — we conducted a needfinding study in which we shadowed four web designers at B12. We observed their work on one website as the designers shared their screens with us and talked aloud about their design process. We also interviewed designers, asking them questions before and after the shadowing session. The interview focused on questions such as "What's the main motivation for not using the current checklist?", "When do you use automated quality checks?", "What are your thoughts on the feedback you receive from reviewers?" Each interview and shadow session was video-recorded and lasted approximately 3–4 hours.

Challenges and drawbacks

Our study uncovered some drawbacks of using traditional checklists like the spreadsheet-based version discussed in Section 2. We saw that projects had specific requirements that took priority even if they conflicted with some design guidelines. For example, "I'm a sole proprietor, but use 'we' in the copy" was a customer-specific requirement that conflicted with the design guideline "If the customer is a sole proprietor, do not use 'we' in the copy." Existing checklists do not address such diverse customer needs. We also saw that projects varied in complexity. For example, some customers wanted designers to focus only on content, while others wanted bespoke aesthetics for a multi-page website.

As with projects, designers also had diverse workflows. Traditional checklists, on the other hand, imply highly constrained workflows. Indeed, sometimes a designer may have different workflows for different projects due to project-specific requirements. For example, one designer said, "I will focus on aesthetics first since the customer was particular about them." Overall, our needfinding study identified a need for dynamicity that arose from the rapidly changing, diverse, and unpredictable nature of client needs.

We also observed challenges unrelated to dynamicity. We learned that designers who did not like to use the provided checklist often overestimated their capacity to remember customer-specific details and internalize the guidelines. These designers also acknowledged forgetting a few guidelines.

We also found that beyond technology, designers most desired and appreciated reviewer or peer feedback. Designers reported finding feedback useful because it helped them learn from their mistakes. They also felt that it could be frustrating if the reviewers do not review the project requirements before providing feedback. When asked about automated feedback systems, designers reported that an early version of AutoQA was unnecessary and raised unreasonable errors. Moreover, the designers strongly suggested that they favored human input on their work above generic design guidelines or algorithmic QA.

We distill these findings into the following challenges:

C1: Each project has specific requirements that must be noted and adhered to.
C2: Designers vary the order in which they complete a checklist depending on customer priorities and their areas of expertise.
C3: Only a subset of the design guidelines apply to a project.
C4: Designers overestimate their capacity to remember project-specific details and design guidelines.
C5: Designers appreciate high-quality human feedback.

5 CRITTER SYSTEM DESCRIPTION

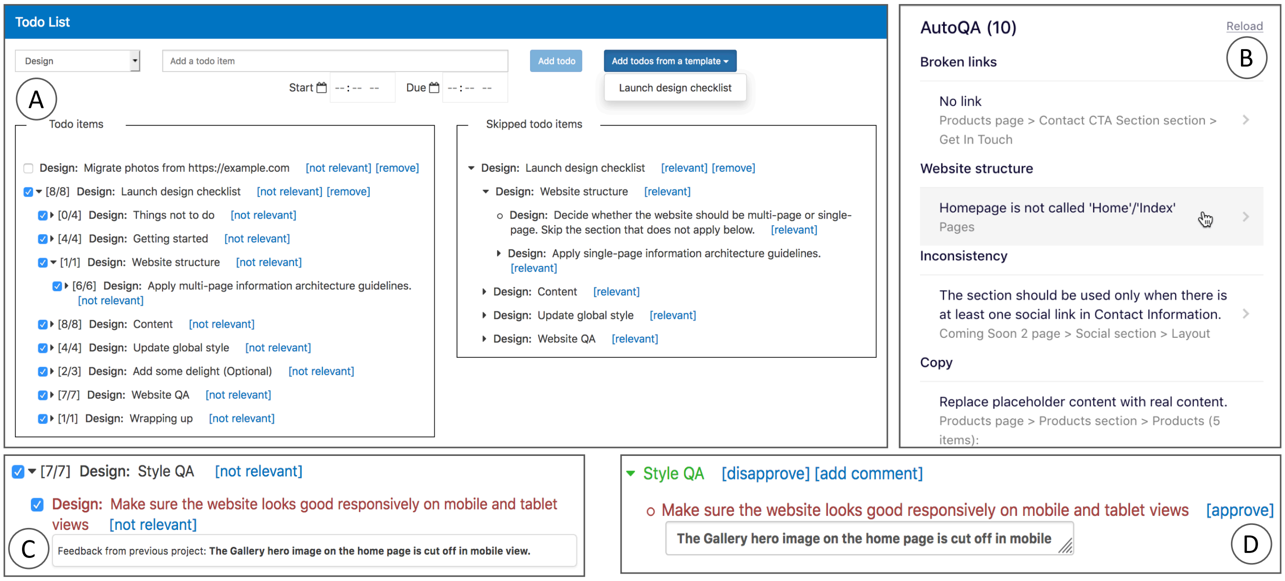
We built Critter with the aim of addressing the challenges (C1–C5) presented in Section 4, and now provide details on how each of Critter's component addresses the challenges, along with implementation details.

Dynamic checklists

Dynamic Checklists address challenges C1–C3 by providing dynamicity, incorporating the designer's work history, unique client needs, and customized templates to help experts focus on the most important guidelines for a project. We built Dynamic Checklists as an extension of the open source Orchestra project [2], a platform for managing flash teams [34] collaborating on creative and analytical projects.

Interface details. The Dynamic Checklists interface (Figure 1A) allows experts to manage checklist items called *todos*. An expert can add individual todos related to this project by writing its details and clicking "Add todo" (addresses C1). Experts can also add todos from a template by clicking "Add todos from a template." A template includes all relevant guidelines and can be applied to any project.

A left column contains todos the expert is responsible for completing. In Figure 1A, a designer has added a "Migrate photos..." todo to the top of their list. They have also added the "Launch design checklist," a collection of 153 nested todos that cover new website design guidelines. As an expert



Todo items

- ☐ **Design:** Migrate photos from https://example.com [not relevant] [remove]
- ☑ [8/8] **Design:** Launch design checklist [not relevant] [remove]
  - ☑ [0/4] **Design:** Things not to do [not relevant]
  - ☑ [4/4] **Design:** Getting started [not relevant]
  - ☑ [1/1] **Design:** Website structure [not relevant]
    - ☑ [6/6] **Design:** Apply multi-page information architecture guidelines. [not relevant]
  - ☑ [8/8] **Design:** Content [not relevant]
  - ☑ [4/4] **Design:** Update global style [not relevant]
  - ☑ [2/3] **Design:** Add some delight (Optional) [not relevant]
  - ☑ [7/7] **Design:** Website QA [not relevant]
  - ☑ [1/1] **Design:** Wrapping up [not relevant]

Skipped todo items

- **Design:** Launch design checklist [relevant] [remove]
  - **Design:** Website structure [relevant]
    - ○ **Design:** Decide whether the website should be multi-page or single-page. Skip the section that does not apply below. [relevant]
    - **Design:** Apply single-page information architecture guidelines. [relevant]
  - **Design:** Content [relevant]
  - **Design:** Update global style [relevant]
  - **Design:** Website QA [relevant]

AutoQA (10)

Broken links

No link
Products page > Contact CTA Section section > Get In Touch

Website structure

Homepage is not called 'Home'/'Index'
Pages

Inconsistency

The section should be used only when there is at least one social link in Contact Information.
Coming Soon 2 page > Social section > Layout

Copy

Replace placeholder content with real content.
Products page > Products section > Products (5 items):

☑ [7/7] **Design:** Style QA [not relevant]

☑ **Design:** Make sure the website looks good responsively on mobile and tablet views [not relevant]

Feedback from previous project: **The Gallery hero image on the home page is cut off in mobile view.**

Style QA [disapprove] [add comment]

○ Make sure the website looks good responsively on mobile and tablet views [approve]

The Gallery hero image on the home page is cut off in mobile

Figure 1: Critter components (HTML/CSS edited slightly for presentation): (A) Dynamic Checklists (B) AutoQA, with warnings about potential issues in the context of the B12 website building experience. (C) Checklist item-specific feedback for a project alongside feedback on recent relevant projects. (D) Reviewer interface, for reviewers to provide item-specific feedback.

works through a dynamic checklist, they can check off items in any order, or skip irrelevant items. Checklists are hierarchical, so experts can also choose to drill down to any level of specificity. For example, experts seasoned in information architecture can read less detail on “Website structure” while focusing on “Content” (addresses C2).

Prunable and self-pruning checklist templates. Checklist templates like the "Launch design checklist" in Figure 1A represent all design guidelines, but only a part of this template may apply to a given project. Dynamic Checklists offer a mixed-initiative system for pruning the irrelevant todos from these checklist templates (addresses C3).

First, the Dynamic Checklist analyzes customers’ request and automatically skips irrelevant todos and their children. For example, the design guidelines cover for both single-page and multi-page websites. If the customer specifically requests a multi-page website, the system automatically skips irrelevant todos that are specific a single-page website project. In Figure 1A, the “Apply multi-page...” todos remain in the left column, whereas the “Apply single-page...” todos have been skipped automatically based on customer requirements.

Second, after a designer adds a checklist template, they can further identify irrelevant todos that were not automatically detected as “[not relevant]”, moving a todo and its children to the right "Skipped todo items" column. This mixed-initiative approach aims to reduce a users’ mental load so that they can focus on tasks that require human judgment [49].

AutoQA

AutoQA (Figure 1B) is an interface embedded in B12’s browser-based website editor. When designers run AutoQA, they receive a report of the errors it detects in their project, identifying common aspects of the design guidelines for which experts overestimated their expertise (addresses C4). Once AutoQA identifies these potential issues, they are serialized to the AutoQA frontend as shown in Figure 1B. A designer is presented with the AutoQA issues (10 in the figure) and, upon clicking on an issue, is taken to the website editor screen in which they can address the issue.

There are 15 distinct AutoQA checkers, each of which flags multiple issues. Grammar checkers ensure proper spelling and grammar. Placeholder checkers ensure customer-provided content is preferred to placeholder content. Image checkers validate that images are not blurry, but also are not so large that they slow downloads. Consistency, layout, and text length checkers ensure collections (e.g., products) are treated with visual consistency and proper grid alignment. Finally, analytics, versioning, broken link, proper homepage, SEO, social link, and contact information checkers ensure that various specific modules are properly configured.

AutoQA checkers are Python functions that access an intermediate representation of a website’s structure, aesthetics, and content. Checkers also scan a structured client brief, which customers provide by filling out a form that includes desired structure and aesthetics alongside existing content. For example, the placeholder checker ensures that if the customer has provided product descriptions in their client brief, the website's content does not include more generic placeholder descriptions in the product section. AutoQA is limited to guidelines for which structured information is available. For example, AutoQA cannot be used to enforce free-text customer feedback like, "I don't care for the hero image."

Contextual reviewer-provided feedback

The Dynamic Checklists interface allows reviewers to provide human feedback on other experts' projects. The design guidelines serve as a rubric for feedback [47], with each piece of unstructured feedback (e.g., "The Gallery hero image on the home . . . ") linked to the relevant item in the checklist hierarchy (e.g., "Make sure the website looks good responsively . . . "). The interface offers reviewers a structured environment to provide the feedback for which experts expressed a preference (addresses C5) as shown in Figure 1D.

To promote a dialogue via feedback, Critter allows the reviewers to easily copy and send this feedback in markdown format along with a hyperlink to the Dynamic Checklist with item-specific feedback as depicted in Figure 1C. To further facilitate learning, the interface also displays the feedback an expert received on previous projects as a recommended todo. These recommended todos are highlighted in red along with the feedback (see Figure 1C). This allows an expert to keep an eye out for the mistakes they tend to make in practice.

In aggregate, AutoQA and contextual reviewer-provided feedback serve a more holistic purpose [10, 32]. When experts repeatedly encounter the same set of AutoQA errors or human feedback in particular areas of a checklist, the encounters serve as a reminder for experts to dedicate more attention to these areas in future projects.

6 EVALUATION

Our evaluation addressed the following research questions:

RQ1 How did designers' use of Critter affect the quality of their designs?

RQ2 What were designers' attitudes toward Critter and its three components?

Task. We selected website design as the creative task under study. This task aligns with our motivating example while also encompassing a range of technical and aesthetic tasks characteristic of many creative tasks.

Participants. We recruited six designers who regularly design websites on-demand for B12's customers, paying their normal hourly rate for all work and research activities. All designers worked remotely and were trained to use the B12 website editor and Critter. Some were also part of the initial needfinding study described in Section 4.

Procedure. We asked each designer to design five websites for actual B12 clients. The clients were generally small- and medium-sized businesses (SMBs), and their requirements generally ranged from 2–7 hours of design work, optionally requiring copywriting. For each website, the designers created an initial checklist using the "launch design checklist" shown in Figure 1A. The designers were allowed to add project-specific custom checklist items in addition to the templated items. After creating the checklist, the designers were instructed to use Critter as they saw fit. At the end of the design task, we asked designers to complete the checklist item that asks them to resolve all of the AutoQA errors.

After a designer completed a website, a reviewer reviewed their work and provided feedback via the Critter interface within 30–40 minutes. This turnaround was fast enough to allow the designer to consider the feedback before starting the next website in all but one case. The reviewer was an author of this paper with four years of experience in web development and two years of experience in user experience (UX) design. The reviewer evaluated the websites based on their adherence to the 153-item template checklist and provided structured feedback as described in Section 5.

To reduce demand characteristics, after completing all the websites, designers were asked submit an online survey where they rated their experience with Critter on several five-point Likert scales. The survey also asked open-ended questions about how they used Critter and how it impacted the quality of their work. We used the survey responses as the basis for follow-up semi-structured interviews [38] with each designer over video calls. All of the interviews were audio-recorded and partially transcribed based on detailed notes. We used a bottom-up approach to analyze the transcripts, organizing them around our three components. One of the designers opted out of the final user interview and online survey. We thus present our results based on the 30 websites delivered by six designers, with final survey and interview results based on five of the six designers' responses.

We also instrumented Critter to record user interactions and performed log analysis, as described in Section 7. Finally, designers provided partial or complete screen recordings for 27 of the 30 website design sessions to supplement the log analysis.

Limitations

One of the limitations of our study is that we do not have quantitative data on the control condition: how did designers perform with traditional flat checklists? This limits our ability to make broader claims about the use of traditional checklists and their impact on quality of work in creative tasks. However, our needfinding study helped us uncover challenges of using a traditional checklist in the context of creative tasks, and made it clear that adherence and understanding of the traditional checklist was low.

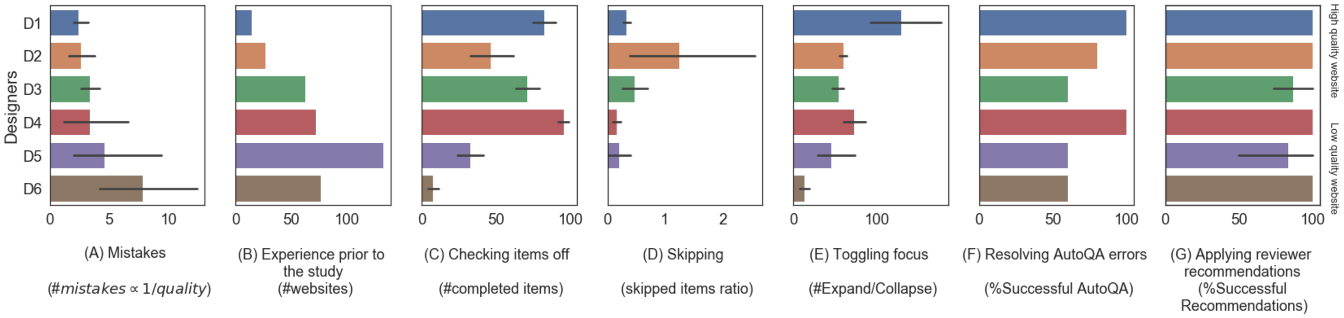


Figure 2: Bar plot of mean values for user engagement measures and mistakes defined in Section 7. The y-axis represents the designers arranged in increasing order (top to bottom) of the mean number of mistakes made by each expert.

Another limitation of our work is the study sample size. Because we studied six professional web designers as they created 30 websites for B12's customers, we were limited to a comparative study between different designers. A larger sample would allow us to perform more robust quantitative analysis and make stronger claims to generalization. However, the rich performance data on diverse real-world websites, interviews, surveys, and instrumented interaction logs enabled us to triangulate our claims across multiple data sources.

7 RQ1: HOW DID DESIGNERS' USE OF CRITTER AFFECT THE QUALITY OF THEIR DESIGNS?

We report the number of mistakes per designer in Figure 2A, and sort each subfigure in Figure 2 by the mean number of mistakes per designer across the websites they designed. The number of missed design guidelines proxy for the quality of a designer's work on a project. This follows a rich history of using guidelines to evaluate quality in design [47] and other domains [26].

While design guidelines may not capture all of a client's quality considerations, we ensured that the guidelines were reasonably comprehensive and captured common design mistakes. Additionally, only 8.5% of the design guidelines call for subjective opinion like, "Is the image appropriate for this client?" Furthermore, only 11% of mistakes correspond to these subjective guidelines, and the designers pushed back on reviewer judgment in only one instance. This shows that the missed guidelines captured reasonable mistakes, and we use this insight to identify a website with a lower number of missed design guidelines as a higher-quality website.

The results in Figure 2 suggest that despite similar instructions, different designers designed websites of varied quality. Across several measures of engagement with Critter (Subfigures C–G), we found that the more engaged a designer was with Dynamic Checklists, AutoQA, and reviewer feedback, the fewer mistakes they made in following design guidelines. Figure 2B, which provides a count of the number of websites designed by each designer for B12 prior to the study, suggests that novice designers who adhered to the system made fewer mistakes than experienced designers who did not.

We next explore how engagement with each of the three component of Critter affected participants' website quality.

Dynamic checklists

Checking items off. Marking a checklist item as complete is the most essential interaction in Dynamic Checklists. We asked all of the designers to only check off items they have read and addressed while designing the website.

In Figure 2C, we note that the designers with fewer checked off items generally made more mistakes, whereas designers with more checked off items generally made fewer mistakes. D1, who checked off the second largest number of items and had the lowest average mistakes per task said, "I like that it guides me through the whole process, making sure that I don't forget anything."

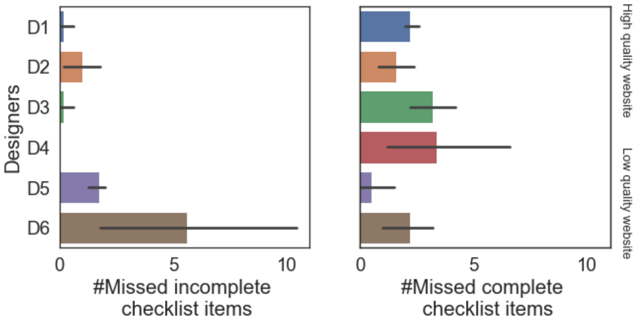


Figure 3: Bar plot of the average number of missed incomplete/complete design guidelines. The y-axis represents the designers arranged in increasing order (top to bottom) of the mean number of mistakes made by each expert.

We noticed that 61% of the missed guidelines were marked as complete by the designers. In Figure 3, we see that the majority of these mistakes were made by designers who had larger checked-off item counts (Figure 2C). When asked about this oversight during the interview, one participant (D5) mentioned having "limited time," and identified situations in which it was "very hard to pay attention," especially to design guidelines that were missed rarely.

In contrast, designers with fewer completed items not only missed some completed items, but also missed a lot of incomplete items (see Figure 3). Some designers explained that they felt that they were experienced enough to have the steps memorized and did not need to explicitly check off items. In practice, these designers appear to have overestimated their capability to remember design guidelines for each task.

Skipping. Skipping checklist items is one of the core operations which allows Critter users to prioritize the design guidelines that are relevant to the task. To measure the level of user's engagement with the skipping functionality, we compute the ratio of the number of checklist items marked as irrelevant to the size of their completed checklist.

In Figure 2D, we see that designers who skipped fewer items generally made more mistakes than designers who skipped more items. We note that 0.6% of mistakes were on items that designers skipped as irrelevant. This shows that designers were successfully able to prune irrelevant items, reducing their chance of missing an item due to an oversight.

Toggling focus. The accordion-style hierarchical checklist allows designers to switch and compartmentalize their focus on a subset of checklist items at a time. To estimate engagement with this functionality, we measure the number of times a participant toggled (expanded or collapsed) a checklist item.

The results in Figure 2E indicate that participants who used the toggling functionality the most also delivered the highest-quality websites. Asked to explain their toggling behavior, one participant (D1) said, "It was convenient because it was logically divided and allows the designer to open up one branch and close it when it is done."

Drilling deep. The hierarchical nature of Dynamic Checklists allow users to explore and drill down to the level that fits their design process. We estimate the level of drilling interaction by the median depth of the checked off items in the checklist.

While not plotted due to space constraints, designers who drilled deep generally made fewer mistakes than the designers who only used higher-level design guideline topics.

AutoQA

AutoQA provides near instantaneous feedback to participants. In Figure 2F, we show the percentage of tasks in which participants eliminated all AutoQA-reported errors, a measure of engagement with AutoQA.

We note two observations from this data. First, AutoQA had relatively high usage among designers, with all designers eliminating all AutoQA-reported errors on the majority of their projects. Second, we notice a slight trend in which experts with higher AutoQA engagement made fewer mistakes.

Reviewer feedback

As explained in Section 5, recommendations from reviewers allow designers to keep an eye out for mistakes on previous tasks. When a designer deviates from a guideline, the reviewer points out the mistake in the feedback and the guideline gets highlighted on the designer's next project. In Figure 2G, we measure the percentage of recommendations that participants adopted in their next project.

We observed strong adherence to the recommended todos: 88 out of 91 recommendations across 30 websites were successfully applied. In three cases where participants repeated a mistake, the designers explicitly differed from and pushed back on reviewer judgment. Due to the strong adherence to recommendations across all participants, we were unable to observe its effect on mistakes based on engagement.

Usage model

In *The Checklist Manifesto* [13], Gawande discusses two checklist usage patterns: 1) *do-confirm*, where a user can perform (*do*) one or many tasks and check off (*confirm*) the corresponding items; and 2) *read-do*, where a user reads (*read*) a checklist item and performs (*do*) the corresponding task.

We found two primary usage patterns for Dynamic Checklists amongst participants. The primary pattern is a hybrid of the two usage models: *do-confirm-read-skip-do*. In this model, the designers finished (*do*) majority of the tasks, checked (*confirm*) corresponding checklist items, went through (*read*) the list of remaining incomplete items, actively *skip*ped irrelevant items, and finished (*do*) relevant checklist items.

The second pattern of *read-do* manifested in two newer experts to web design with B12 (D1and D4). Early in their experience with checklists, they serially *read* the guidelines and completed them (*do*). As they gained confidence with the guidelines, they switched to *do-confirm-read-skip-do*. The designers did not explicitly identify these models and there is no evidence to suggest that either of the two models is better than the other in terms of the designer's quality of work. The designers reported that they chose to switch to *do-confirm-read-skip-do* model because it allowed them to design and cover majority of the checklist items from their working memory without switching focus to the checklist.

8 RQ2: WHAT WERE DESIGNERS' ATTITUDES TOWARD CRITTER AND ITS THREE COMPONENTS?

Overall experience

As depicted in Figure 4, Critter received a moderately good overall experience rating of 3.4 on a 5-point scale where 5 is an excellent experience and 1 is a poor experience. D2 summarized, “It is a helpful reminder of things that can easily get lost in all that has to be done to build a website.”

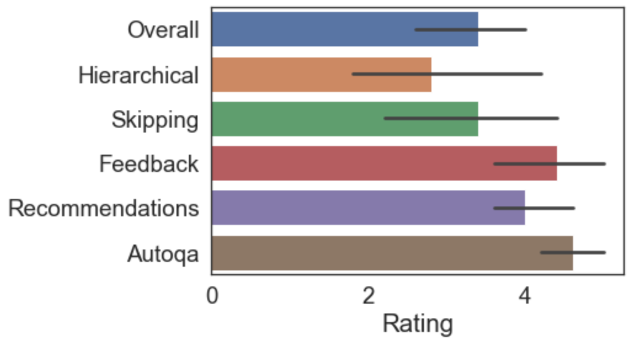


Figure 4: Bar plot of the average rating given by the designers on a five-point scale where 1 represents poor and 5 represents excellent user experience.

Dynamic checklists

Designers reported that they liked the idea of automatically skipping irrelevant checklist items (mean rating = 3.4) from their checklist as it saves time. As described in Section 5, Critter automatically pruned 20.4% of checklist items for each project on average (std dev 5.1). D5 explained, “I thought it was smart and worked well. In one project I reviewed these automatically skipped items just in case, and I don’t think there was anything there that should have been included.” Other designers described not noticing automatically pruned items, with D6 saying, “You are not meant to notice it anyway.” These comments support our design goal to move irrelevant checklist items out of focus.

Some designers reported that they liked the hierarchical presentation of checklist items because it allowed them to compartmentalize their focus. D1 explained, “I think they are logical and help concentrate on one aspect of the website at a time. It was convenient because it was logically divided and allows the designer to open up one branch and close it when it is done.” They found the hierarchical format easier to use than the original flat list of checklist items, which were perceived as hard to navigate through and overwhelming.

However, other designers were less enthusiastic about the hierarchical checklists. One designer (D6) felt that it did not fit into their design flow and felt like a chore. D5 maintained their stance that, as an experienced designer, they did not need to use the thorough checklist and spend time on moving out irrelevant items from their checklist. Instead, D5 expressed the desire to use a reusable, shorter *cheatsheet checklist* that they can update based on reviewer feedback. Furthermore, designers unanimously reported readability issues with hierarchical checklists and therefore rated it poorly in comparison to other components (mean = 2.8). They suggested different fonts, borders, or color schemes to more clearly delineate different sections of the hierarchical view.

AutoQA

Four out of five designers rated the AutoQA experience as excellent (mean rating = 4.6) and felt that it improved the quality of their work. Designers reported two key benefits of using AutoQA. First, they felt it helped them catch errors they would miss otherwise. D4 noted, “I look forward to it each time. It makes me accountable.” Second, they felt AutoQA also acted a learning tool for understanding B12’s design guidelines. In D1’s words: “Now I understand what kind of mistakes I can make. It helped me learn more about my design process. Now I try to make sure I don’t repeat AutoQA errors before running AutoQA.”

Reviewer feedback and recommendations

All of the designers expressed appreciation for the human reviewer’s feedback (mean rating = 4.4) as it helped them learn about B12’s expectations and improve their design process. D1 said, “It actually helped me to better understand some todo items and and what results are expected.” Designers also liked the fact that the feedback appeared on their checklist, helping them to understand the approach the reviewer took to give feedback. Given the hierarchical nature of the checklist, the feedback next to the checklist item not only helped them identify what needed to fixed, but also the general area in their design process that needs attention.

Designers also scored the automatically recommended todo items from previous projects highly in the survey (mean = 4.0), but were more reserved in their interview comments. Most designers did not feel that it directly impacted the quality of their work. However, all of them acknowledged that highlighting the todos in red and attaching reviewer feedback caught their attention. D1 felt that it indirectly prevented mistakes, saying, “It makes me pay more attention to the particular todo items so that I don’t make the same mistakes again”. Some designers also acknowledged its potential as a learning tool. D5 said, “It would be an issue you were struggling with and by showing up, it reminds you to work on it”.

9 DISCUSSION AND CONCLUSION

In this paper, we argue that Critter augments creative work through Dynamic Checklists, AutoQA, and contextual reviewer feedback. We found that the more engaged a designer was with Critter’s self-pruning hierarchical checklists and AutoQA, the fewer mistakes they made in following detailed design guidelines. We noticed that novice designers who adhered to the system made fewer mistakes than experienced designers who didn’t. Survey and interview data suggest that the designers generally liked Critter (mean rating = 3.4) and found it to be a useful guidance tool. We now discuss some practical implications for systems that aim to augment creative work and speak to the generalizability of Critter.

Implications for practice

A diversity of approaches. Systems that aim to support high-quality creative work should include more than one way to ensure the quality. To handle the different forms of mistakes experts might make across projects, Critter uses complementary quality assurance techniques like Dynamic Checklists and AutoQA to eliminate these mistakes. The various components of Critter address various scalability issues with feedback. At one extreme, AutoQA and automatic pruning offer instantaneous feedback and review scalability, but are limited to computable guidelines and requires subject matter-specific automation. At the other extreme, Dynamic Checklists that embed reviewer feedback cover the long tail of guidelines and quality assurance without the implementation challenges, but lack the immediacy of automation and are susceptible to errors inherent to human judgment [40]. Notably, experts identified each of Dynamic Checklists, AutoQA, and reviewer feedback as useful learning tools.

Automated critics are effective when accurate. Corroborating prior work [13], automated critics like AutoQA should only report high-confidence errors transparently identifying doubt in errors. While participants trusted and appreciated reviewer feedback, they quickly grew suspicious of automated errors with high false positives in early prototypes. After we reduced the AutoQA false positive rate, participants' opinion changed, and the majority rated their experience with AutoQA as excellent.

Design guidelines as rubrics lead to learning and better work. Guideline-specific feedback and automated quality checks helped improve designers' capacity to self-reflect using a checklist. This finding corroborates prior work which shows that utilizing specific criteria as a rubric for self-assessment has a learning effect [8]. Similarly, we found that by resolving errors reported by AutoQA, experts felt more confident going through the checklist, with some experts expressing a better understanding of the process. Still, guideline-specific feedback and AutoQA can only assist self-reflection, whereas the designers' ability to learn from self-reflection also depends upon other factors like experience and motivation [36].

Hierarchical checklists support task non-linearity and varied expertise. The hierarchy of Dynamic Checklists affords both a broad and deep exploration that is ideal for non-linear tasks like writing, poster design, or filmmaking. Hierarchical checklist organization also allows experts to stay at a different level of granularity than novices. During our needfinding study, we learned that web designers did not like a traditional flat checklist because it did not align with their process. In larger projects, they found themselves scrolling the checklist multiple times to revisit different parts of the checklist to explore various subtasks. A hierarchical view facilitated a quick scan with deeper exploration into the relevant area of focus.

Alternative design opportunities

Critter asks experts to build a checklist by pruning the template checklist down to relevant items in a top-down fashion. An alternative design could allow the experts to build their checklists from scratch by adding relevant items in a bottom-up fashion instead of skipping them. We selected a top-down approach to enable experts with less experience in the guidelines to explore and internalize them with time. For experts who have already internalized most of the checklist and do not want to spend time on removing irrelevant items, a bottom-up approach that enables them to add relevant items might fit their workflows better. Designing a dynamic system that supports such interactions while ensuring the adherence to the guidelines is an interesting future direction.

Generalization to other domains

At their core, the design guidelines, automated checks, and reviewer feedback from our motivating example have analogies in many creative fields. The web design guidelines in this paper are akin to written style guides [12, 39] (e.g., "use the oxford comma") or programming style guides [1, 44] (e.g., "Class names should be nouns in UpperCamelCase"). Similarly, automated systems like Grammarly [3] for writing or linters [23] for programming play a similar role to AutoQA. Finally, reviewer feedback can take the form of an editor for writing or code review for programming.

With these analogs in mind, one can imagine generalizing Critter to other creative fields with numerous guidelines, idiosyncratic client requests, and nonlinear completion paths like designing, writing, programming. For example, before submitting a code review, a software engineer might be presented with a hierarchically organized style guide for their programming language that has been pruned to relevant aspects of the task, checking off key areas they have completed. If the engineer has received code review feedback on certain aspects of the style guide in the past, those areas can be recommended for deeper inspection. Before submitting code for review, the engineer must ensure their code free of linter errors. Finally, a code review can enforce quality while informing future tasks. As the programmer internalizes the checklist, they can inspect it at coarser levels of granularity.

With Critter, we have showcased that creativity can be augmented by structure. While this work warrants more exploration, we believe Dynamic Checklists, AutoQA, and contextual reviewer feedback to be part of a brighter future of work.

10 ACKNOWLEDGEMENTS

We're grateful to B12's design experts and our B12 peers. This research was supported by NIH grant 1UH2CA203768-01.

REFERENCES

[1] 2009. Google Java Style Guide. https://google.github.io/styleguide/javaguide.html

[2] 2015. Orchestra. http://orchestra.b12.io/

[3] 2015. Write your best with Grammarly. https://www.grammarly.com/

[4] Jochen Bergs, Johan Hellings, Irina Cleemput, Ö Zurel, Vera De Troyer, Monique Van Hiel, J-L Demeere, Donald Claeys, and Dominique Vandijck. 2014. Systematic review and meta-analysis of the effect of the World Health Organization surgical safety checklist on postoperative complications. *British Journal of Surgery* 101, 3 (2014), 150–158.

[5] Daniel Boorman. 2001. Today's electronic checklists reduce likelihood of crew errors and help prevent mishaps. *ICAO Journal* (2001).

[6] C Marlin Brown. 1999. *Human-computer interface design guidelines.* Intellect Books.

[7] Asaf Degani and Earl L Wiener. 1991. Human factors of flight-deck checklists: the normal checklist. (1991).

[8] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work.* ACM, 1013–1022.

[9] Elfriede Dustin, Jeff Rashka, and John Paul. 1999. *Automated software testing: introduction, management, and performance.* Addison-Wesley Professional.

[10] Gerhard Fischer, Kumiyo Nakakoji, Jonathan Ostwald, Gerry Stahl, and Tamara Sumner. 1993. Embedding critics in design environments. *The knowledge engineering review* 8, 4 (1993), 285–307.

[11] Werner Funk, Vera Dammann, and Gerhild Donnevert. 2007. *Quality assurance in analytical chemistry: applications in environmental, food and materials analysis, biotechnology, and medical engineering.* John Wiley & Sons.

[12] Jose L Galvan and Melisa C Galvan. 2017. *Writing literature reviews: A guide for students of the social and behavioral sciences.* Routledge.

[13] Atul Gawande. 2010. *Checklist manifesto, the (HB).* Penguin Books India.

[14] Jun Gong, Peter Tarasewich, et al. 2004. Guidelines for handheld mobile device interface design. In *Proceedings of DSI 2004 Annual Meeting.* 3751–3756.

[15] Andrew Grochowski, Debashis Bhattacharya, TR Viswanathan, and Ken Laker. 1997. Integrated circuit testing for quality assurance in manufacturing: history, current status, and future trends. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 44, 8 (1997), 610–633.

[16] Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. 2015. Argonaut: Macrotask Crowdsourcing for Complex Data Processing. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1642–1653. https://doi.org/10.14778/2824032.2824062

[17] Brigette M Hales and Peter J Pronovost. 2006. The checklist – a tool for error management and performance improvement. *Journal of critical care* 21, 3 (2006), 231–235.

[18] John Hattie and Helen Timperley. 2007. The power of feedback. *Review of educational research* 77, 1 (2007), 81–112.

[19] Alex B Haynes, Thomas G Weiser, William R Berry, Stuart R Lipsitz, Abdel-Hadi S Breizat, E Patchen Dellinger, Teodoro Herbosa, Sudhir Joseph, Pascience L Kibatala, Marie Carmela M Lapitan, et al. 2009. A surgical safety checklist to reduce morbidity and mortality in a global population. *New England Journal of Medicine* 360, 5 (2009), 491–499.

[20] Robert L Helmreich. 2000. On error management: lessons from aviation. *Bmj* 320, 7237 (2000), 781–785.

[21] Scott Henninger, Kyle Haynes, and Michael W Reith. 1995. A framework for developing experience-based usability guidelines. In *Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques.* ACM, 43–53.

[22] Mariana G Hewson and Margaret L Little. 1998. Giving feedback in medical education: verification of recommended techniques. *Journal of general internal medicine* 13, 2 (1998), 111–116.

[23] Stephen C Johnson. 1977. *Lint, a C program checker.* Citeseer.

[24] Huhn Kim. 2010. Effective organization of design guidelines reflecting designer's design strategies. *International Journal of Industrial Ergonomics* 40, 6 (2010), 669–688.

[25] Chinmay Kulkarni, Steven P Dow, and Scott R Klemmer. 2014. Early and repeated exposure to examples improves creative work. In *Design thinking research.* Springer, 49–62.

[26] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R Klemmer. 2013. Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction (TOCHI)* 20, 6 (2013), 33.

[27] Sri Kurniawan and Panayiotis Zaphiris. 2005. derived web design guidelines for older people. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility.* ACM, 129–135.

[28] Jonas Löwgren and Ulrika Laurén. 1993. Supporting the use of guidelines and style guides in professional user interface design. *Interacting with Computers* 5, 4 (1993), 385 – 396. https://doi.org/10.1016/0953-5438(93)90003-C

[29] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. In *ACM transactions on graphics (TOG)*, Vol. 30. ACM, 87.

[30] Jane N Mosier and Sidney L Smith. 1986. Application of guidelines for designing user interface software. *Behaviour & information technology* 5, 1 (1986), 39–46.

[31] Tricia J Ngoon, C Ailie Fraser, Ariel S Weingarten, Mira Dontcheva, and Scott Klemmer. 2018. Interactive Guidance Techniques for Improving Creative Feedback. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, 55.

[32] Henry Petroski. 1985. *To engineer is human: The role of failure in successful design.* St Martins Press.

[33] Daniela Retelny, Michael S Bernstein, and Melissa A Valentine. 2017. No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Constrain Complex Work. *Proceedings of the ACM on Human-Computer Interaction* 1, 2 (2017), 23. https://doi.org/10.1145/3134724

[34] Daniela Retelny, Sébastien Robaszkiewicz, Alexandra To, Walter S Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S Bernstein. 2014. Expert crowdsourcing with flash teams. In *Proceedings of the 27th annual ACM symposium on User interface software and technology.* ACM, 75–85.

[35] Dominique L Scapin. 1990. Organizing human factors knowledge for the evaluation and design of interfaces. *International Journal of Human-Computer Interaction* 2, 3 (1990), 203–229.

[36] Donald A Schön. 2017. *The reflective practitioner: How professionals think in action.* Routledge.

[37] Michael Scriven. 2000. The logic and methodology of checklists. (2000).

[38] Irving Seidman. 2013. *Interviewing as qualitative research: A guide for researchers in education and the social sciences.* Teachers college press.

[39] Mary Shaw. 2003. Writing good software engineering research papers. In *Software Engineering, 2003. Proceedings. 25th International Conference on.* IEEE, 726–736.

[40] Barry G Silverman. 1991. Expert critics: operationalizing the judgement/decisionmaking literature as a theory of "bugs" and repair strategies. *Knowledge Acquisition* 3, 2 (1991), 175–214.

[41] Michael Terry and Elizabeth D Mynatt. 2002. Recognizing creative needs in user interface design. In *Proceedings of the 4th conference on Creativity & cognition*. ACM, 38–44.

[42] Linda Tetzlaff and David R Schwartz. 1991. The use of guidelines in interface design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 329–333.

[43] Henrik Thovtrup and Jakob Nielsen. 1991. Assessing the usability of a user interface standard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 335–341.

[44] Guido van Rossum, Barry Warsaw, and Nick Coghlan. 2001. PEP 8: style guide for Python code. *Python. org* (2001).

[45] Mark Warschauer and Paige Ware. 2006. Automated writing evaluation: Defining the classroom research agenda. *Language teaching research* 10, 2 (2006), 157–180.

[46] Jane Westberg and Hilliard Jason. 2001. *Fostering reflection and providing feedback: Helping others learn from experience.* Springer Publishing Company.

[47] Alvin Yuan, Kurt Luther, Markus Krause, Sophie Isabel Vennix, Steven P Dow, and Bjorn Hartmann. 2016. Almost an Expert: The Effects of Rubrics and Expertise on Perceived Value of Crowdsourced Design Critiques. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1005–1017. https://doi.org/10.1145/2818048.2819953

[48] Myung Hwan Yun, Heecheon You, Wooyeun Geum, and Dongjoon Kong. 2004. Affective evaluation of vehicle interior craftsmanship: systematic checklists for touch/feel quality of surface-covering material. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 48. SAGE Publications Sage CA: Los Angeles, CA, 971–975.

[49] Haoqi Zhang, Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz. 2012. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 217–226.