To Reuse or Not To Reuse? A Framework and System for Evaluating Summarized Knowledge

MICHAEL XIEYANG LIU, Human-Computer Interaction Institute, Carnegie Mellon University, USA
ANIKET KITTUR, Human-Computer Interaction Institute, Carnegie Mellon University, USA
BRAD A. MYERS, Human-Computer Interaction Institute, Carnegie Mellon University, USA

As the amount of information online continues to grow, a correspondingly important opportunity is for individuals to reuse knowledge which has been summarized by others rather than starting from scratch. However, appropriate reuse requires judging the relevance, trustworthiness, and thoroughness of others' knowledge in relation to an individual's goals and context. In this work, we explore augmenting judgements of the appropriateness of reusing knowledge in the domain of programming, specifically of reusing artifacts that result from other developers' searching and decision making. Through an analysis of prior research on sensemaking and trust, along with new interviews with developers, we synthesized a framework for reuse judgements. The interviews also validated that developers express a desire for help with judging whether to reuse an existing decision. From this framework, we developed a set of techniques for capturing the initial decision maker's behavior and visualizing signals calculated based on the behavior, to facilitate subsequent consumers' reuse decisions, instantiated in a prototype system called Strata. Results of a user study suggest that the system significantly improves the accuracy, depth, and speed of reusing decisions. These results have implications for systems involving user-generated content in which other users need to evaluate the relevance and trustworthiness of that content.

CCS Concepts: • **Information systems** → **Decision support systems**; • **Software and its engineering** → *Software design tradeoffs*; • **Human-centered computing** → Graphical user interfaces.

Additional Key Words and Phrases: Knowledge Reuse; Decision Making; Developer Tools; Sensemaking

ACM Reference Format:
Michael Xieyang Liu, Aniket Kittur, and Brad A. Myers. 2021. To Reuse or Not To Reuse? A Framework and System for Evaluating Summarized Knowledge. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 166 (April 2021), 35 pages. https://doi.org/10.1145/3449240

1 INTRODUCTION

Information and knowledge reuse has become a highly consistent paradigm across a wide range of fields and disciplines to advance their respective frontiers, such as reusing previous engineering best practices on future generations of products [20, 21], taking advantage of schemas and results from previous sensemaking episodes to create new representations and understandings of the world [41, 63, 92, 100], and plugging in previously written and well-maintained design patterns and code snippets to build novel software features and functionalities [3, 11, 46, 47, 69]. Reusing proven information and knowledge promises the benefits of potentially reduced workload and

Authors' addresses: Michael Xieyang Liu, Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA, xieyangl@cs.cmu.edu; Aniket Kittur, Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA, nkittur@cs.cmu.edu; Brad A. Myers, Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA, bam@cs.cmu.edu.

2573-0142/2021/4-ART166. https://doi.org/10.1145/3449240

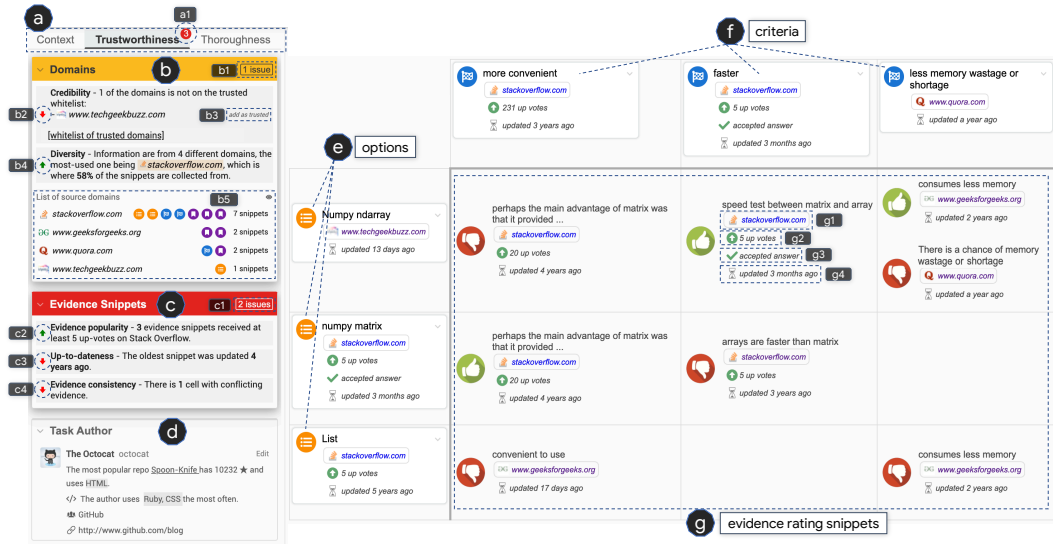


Fig. 1. Strata's user interface. Strata helps developers evaluate three main facets of appropriateness of reusing a Unakite comparison table with options (e), criteria (f), and evidence (g) through three overview panels: (a) the *Context* panel, the *Trustworthiness* panel, and the *Thoroughness* panel. Each panel contains the *groups* (such as (b), (c), (d)) of appropriateness properties to directly address developers' information needs. Developers will also be alerted of any potential issues with respect to each facet (e.g., b2, c3, c4).

development cycles [20, 69], improved quality and performance [47, 50, 125], and more time for creation and innovation [59, 84, 85, 125].

There have been various commercial and research information gathering and sensemaking systems that help people with creating reusable knowledge by helping with capturing [16, 23, 73], organizing [29, 51, 52], disseminating [32, 76, 102], and understanding [29, 62, 91, 92, 100] information. One that is relevant to the context of programming, our Unakite system [83], enables developers to collect and organize information online into comparison tables with options, criteria, and evidence to help with making decisions (see Figure 1-e,f,g). Systems like these often support keeping track of information for sharing with others later [27, 52, 83, 92, 100]. For example, Unakite might present a comparison table authored by an initial developer (who we call the *author*) to help subsequent developers (who we call the *consumers* or *readers*) pick an API to represent matrices in Python (as in Figure 1 and Figure 2), or to choose the best JavaScript framework to build a website. Unakite is designed to help consumers reuse the decisions and trade-offs identified by the author [49, 57, 72, 83, 103] instead of spending the time to discover them from scratch.

However, a major challenge to such a knowledge artifact actually being suitable for reuse is that the consumers do not know if it is *appropriate* to use it or not [85, 125]. Prior research suggested that when checking if a piece of online information can be reused or not, people primarily focus on verifying its *correctness*, and often use *credibility* as a surrogate for correctness because it is easier to check and is highly correlated with correctness [55]. For example, signals that can be leveraged to judge credibility include whether the information came from credible sources, whether the way it was presented looked credible, and what the author's qualifications and credentials were [44, 87, 108, 124]. In addition, the correctness of information can not always be measured objectively, but rather often depends on the situation [42]; for example, a statement that a sorting

algorithm is "fast" may depend on the size of the data it is applied to. Furthermore, the knowledge artifacts shown by previous systems are usually a collection and synthesis of different individual pieces of information from different sources. They often capture the author's *opinion* about whether a decision should be made in one way or another, and there is not likely to be a single correct answer but multiple valid options with trade-offs [83]. Unlike general web pages and their content, such knowledge artifacts require many more types of judgements in addition to credibility for someone to decide whether it is appropriate to reuse them or not, including whether the goal and context of the author matches that of the consumer's [55, 85], how thorough was the author's research [37, 100], etc.

Another challenge identified in sensemaking research is that, in reality, consumers often opt to start from scratch rather than reusing previous users' work because of the high costs associated with 1) systematically identifying all of the potential aspects of the work to verify, and 2) obtaining access to properties that could help with the verification [41, 79, 80, 87]. For example, when checking the thoroughness of an author's research, the list of search queries used, the web pages visited, the pages that the author spent the most time reading, and the potential alternatives that were overlooked can all be valid properties to help with the assessment, but are currently not kept track of (even by systems such as Unakite) and hence are not available to the consumer.

In this work, we explore these challenges in the context of reusing the comparison tables created using the Unakite system, where the consumer developer needs to evaluate the *appropriateness* of reusing the table authored by the initial developer. We perform our research through a *user-centered design* approach. From the vast body of prior work discussing frameworks and measurements for issues of trust and reuse, we extracted properties of importance to developers. We then conducted formative needs-finding interviews with developers about their information needs when evaluating appropriateness. We then synthesized all this information together, resulting in the three key facets of the author's *context*, and the *trustworthiness* and *thoroughness* of the resulting knowledge artifacts, each with a collection of the consumers' specific information needs, which are summarized in a framework in Table 1. We then devised various key signals and properties that can be used to address those needs as well as mechanisms to automatically identify, compute or keep track of them as an author collects information, which are summarized in the last column of Table 1. Then, we iteratively designed a hierarchical presentation of the information that lets consumers view and explore those signals and properties interactively, by augmenting the original Unakite tables. These were implemented in a prototype system called Strata[1], which consists of a browser plugin for Google Chrome and a web application (see Figure 1). Finally, we conducted a user study to evaluate Strata's effectiveness.

The primary contributions described in this paper include:

- a formative **study** showing developers' needs for support with reusing previously-generated knowledge,
- a synthesized **framework** (Table 1) for augmenting judgements of appropriate reuse including three major facets: context, trustworthiness, and thoroughness,
- a prototype **system** called Strata that automatically records, computes, and visualizes many of the appropriateness signals described in the framework,
- an **evaluation** of the prototype system that offers insights into its usability, usefulness, and effectiveness.

[1]Strata is named after a series of layers of rock that shows the history of a geographical location. It stands for "**S**idebar **T**owards **R**euse and to **A**ssess **T**rustworthiness and **A**pplicability".

Facet	Information Need	Selected References in Prior Research	Sample Quotes in Formative Study	Selected Supporting Features in Strata
Context	Goals of the original decision	• Search queries are useful for encoding task goals & contexts in various settings like asynchronous collaborations [23, 92, 100, 101, 111, 126].	• *"This looks like it's trying to pick a speech recognition API, but what I want is actually text to speech."*	• Keeping track of the author's search queries to reflect his or her task goal.
	Explanation or contextualization of information	• Recontextualization of information helps with understanding [83, 85]. • Clarity and informativeness of website content improves understanding [43, 118].	• *"What does this 'very efficient' mean, is it 'memory' or 'time' efficient?"* • *"Is it [a sorting algorithm] 'fast' only when there're a few hundred data points or also when there are millions of data points?"*	• Keeping track of the surroundings along with the information snippets and presenting them as contextual explanations.
	Situational awareness	• Awareness of common ground facilitates sensemaking handoff [33, 109, 111]. • Users need awareness of each others' actions in order to perform their tasks better [17, 92, 93, 100].	• *"I want to solve it with pure JavaScript, but it seems that most of the answers here are actually written using jQuery?"* • *"I'm using Python 2.7 at the moment, which is fairly old, does this example also use this version?"*	• Detecting information about languages, frameworks, and their versions mentioned in information snippets with a predefined yet easily extensible list of detectors.
Trustworthiness	Source credibility and diversity	• Source credibility affects trustworthiness of information [35, 39, 43, 87, 118]. • Sources similar to what a consumer usually uses are more likely to be deemed credible [89, 108].	• *"If it's from Stack Overflow, I'm usually fine with it. But if it's from some random blog posts written by some random guy, I would think twice."* • *"I wonder if all of these just came from the official documentation or there're also other developer forums."*	• Visualizing the distribution of information snippets across different domains (websites). • Alerting consumers of potential untrusted domains.
	Information up-to-dateness	• Information currency affects its perceived credibility [15, 26, 87].	• *"Is this speed comparison [between React, Angular, and Vue] up-to-date now that Angular 9 was just released?"*	• Extracting and surfacing the last updated time of information snippets.
	Information popularity	• People apply *the endorsement heuristic* to evaluate credibility [87]. • People seek social proof when evaluating credibility [108].	• *"If there're a lot of other devs [who] also think this is a good idea, then I'm much more comfortable to use it."*	• Extracting and surfacing signals showing information popularity, such as the up-vote count of an answer on Stack Overflow.
	Information consistency	• People apply *the consistency heuristic* to evaluate credibility [87]. • People seek more than one source to verify information [86].	• *"It claims PyTorch is much easier to learn than Tensorflow, but I wonder if there're people suggesting otherwise."*	• Alerting consumers if there are conflicting (both positive and negative) ratings in any of the table cells.
	Author credibility	• The author's level of expertise affects information trustworthiness [35, 65, 108]. • Disclosing patterns of past performance helps people evaluate trustworthiness [65, 113, 116].	• *"Does the table author know what he's doing?"* • *"Is the author saying all the nice things about Caffe because he has lots of experience with it or because he's biased?"*	• Surfacing credibility and bias signals from the table author's Github profile, such as their primary programming language, number of stars on their repositories, and affiliation.
Thoroughness	Research process and effort	• External representations handed off should indicate prior investigative process and insights [100, 101, 131], how much work had been done, and how mature the representation was [109, 111].	• *"How much effort was put into making this decision?"* • *"What did the author focus on?"*	• Keeping track of and visualizing the author's activities on an interactive timeline view, including search queries, pages visited, duration of stay on the pages, information snippets collected, etc.
	Alternatives or competitors	• Knowledge and sensemaking results should indicate their coverage and scope [35, 87].	• *"I heard anecdotally that Svelte gives you much better performance than all these big (JavaScript) frameworks [React, Angular, and Vue]. I should take a look at that before I decide."*	• Finding and surfacing commonly searched-for alternatives mentioned in Google autocomplete suggestions.
	Usable artifacts	• Developers need help finding and reusing code examples [27, 97, 102].	• *"Which option was chosen in the end?"* • *"[Are there] any code snippets that I can immediately plug into mine and test?"*	• Extracting and surfacing code examples from information snippets.

Table 1. A framework summarizing the three major facets (column 1) when evaluating the appropriateness to reuse knowledge, including people's specific information needs (column 2), selected evidence from prior work (column 3), sample quotes from our formative study interviews (column 4), and features we devised to support the information needs in the subsequent Strata system (column 5).

2 RELATED WORK

2.1 Information and Knowledge Reuse

As formulated by Davenport et al. in 1996 [34] and Markus in 2001[85], knowledge processes are often categorized by whether they involve *knowledge creation* (e.g., research and development of new products and services, or writing books or articles) or *knowledge reuse* (e.g., reapplying existing components and best practices to solve common problems). While there is much research into the significance and difficulties of knowledge creation and innovation [34, 51, 53, 63, 68, 95], the effective reuse of knowledge has been shown to be a more frequent strategy and concern to individuals and organizations [34, 36, 85, 96, 98, 129, 130].

Many systems have been developed to support the multiple stages of information and knowledge reuse as mapped out by Markus [85]: *capturing and documenting knowledge*, *packaging and distributing knowledge*, and *reusing knowledge*. Among them, some systems support capturing, organizing, and keeping track of information in the first place (e.g., [23, 52, 77, 78, 83, 120]), some aim to deliver and surface existing knowledge directly to a user without the need of complex matching and frequent context switches (e.g., [27, 30, 102]), and others facilitate the digesting and understanding of knowledge (e.g., [80, 83, 116]). However, having a literal understanding of a knowledge artifact does not by itself imply reuse — a major barrier to that knowledge actually being useful is the consumer does not know whether it is *appropriate* to use it or not [85, 125].

Prior research provides insights into various properties that people look for in order to evaluate the appropriateness for reuse, such as source credibility [35, 39, 43, 87, 108, 118], information currency (or up-to-dateness) [15, 26, 87], information popularity [87, 108], goals and purposes (what the author wanted to achieve) [101, 111], etc. However, much research such as the above focuses on specific issues about the general credibility of web content, while knowledge artifacts previously collected and synthesized by an author require many more types of judgements beyond credibility in order for a consumer to decide its appropriateness for reuse. To the best of our knowledge, there remains no systematic models or frameworks for understanding the factors that affect the judgements of the reuse of previously created knowledge artifacts. Such a framework could be helpful for driving research studying and augmenting reuse across a variety of domains and forms. Here we take a step towards such a framework, starting with knowledge artifacts in the form of comparison tables, which are widely used, and in the domain of programming, where knowledge reuse happens frequently [27, 50, 54, 57, 67, 70, 83, 102, 115, 115]. In the following sections, we discuss three of the most relevant threads of research as they relate to judgements of knowledge reuse.

2.2 Evaluating Online Information Credibility

2.2.1 Models and Heuristics for Evaluating Online Information Credibility. One of the most researched facets of knowledge reuse is evaluating online information credibility [44, 87, 108, 124] (or "trustworthiness" [118]), which focuses on facets of authenticity, reliability, and trustworthiness of a given piece of content online, ranging from e-commerce transactions to online discussions and collaborations [65, 116, 117]. Prior work has employed bottom-up approaches like surveys and contextual inquiries and reported various factors that influence credibility assessment, including but not limited to: domain name and URL, presence of date stamp showing information is current, author identification and indication of his or her expertise, citations to scientific data or references, and user ratings and reviews [15, 26, 39, 43, 48, 86, 87, 89, 113, 118, 124].

In addition, models and heuristics for credibility assessment have also been proposed, for example, the *checklist model*, which guides users through a checklist of critical factors during assessment [87], and the *contextual model*, which emphasizes the use of external information to establish credibility

[86], such as promoting peer-reviewed resources and seeking corroborating or conflicting evidence. A summary by Metzger et al. [89] suggests that users routinely invoke *cognitive heuristics* to evaluate the credibility of information and sources online, such as the *reputation heuristic* (checking if the source of the information has good reputation and credentials), and the *expectancy violation heuristic* (checking if a website or its content conforms to their original expectations).

However, in reality, it has repeatedly been shown that people are often underprepared and have trouble determining how to evaluate the credibility of online information [18, 86, 88, 106], which is often deemed to be too much work [86, 109], having a high possibility of missing important details [87, 89], and eventually leading to abandonment, mistrust or misuse [79, 80, 87] of the information. This reflects a significant gap between research and reality: while prior work provides insights into the various factors affecting online information credibility and ways people reason about them, people need tool support that systematically helps with credibility assessment and information reuse. We address this gap by providing a prototype system that (1) automatically extracts appropriateness signals (including those related to credibility) from the original knowledge content when possible; and (2) processes and presents them to the consumer of the knowledge in a hierarchical visualization that directly addresses their information needs during the evaluation of the appropriateness to reuse.

2.2.2 Interventions and Support for Evaluating Collaboratively-built Knowledge Content. Collaborative knowledge building, exemplified by the Wikipedia project [6] and Stack Overflow [5], has become highly popular in many domains, and its mutable nature that virtually *anyone can edit anything* has invited considerable research into helping users evaluate the trustworthiness of its content. For example, the revision histories [116, 122, 127, 128], review processes [123], and the external references [44, 45] of an article can be modeled and visualized to help improve transparency and the evaluation of its trustworthiness. In addition, an author's past performance, such as their editing history on Wikipedia or previously answered questions on Stack Overflow, can be mined [14, 113] and surfaced [116] to help knowledge consumers determine the author's reputation, expertise, and other accountability metrics. Encouragingly, Kittur et al. [65] showed that surfacing trust-relevant information from Wikipedia articles had a dramatic impact on users' perceived trustworthiness of those articles, holding constant the content itself.

However, despite the overwhelming importance and increasing research effort, being considered trustworthy is often not the sufficient condition for reuse, nor is trustworthiness always the first facet that users evaluate — research has shown that people often have trouble understanding a piece of information when it is taken out of its original context [83, 85] and figuring out if it is indeed relevant to their own situation [25, 105, 109] before they start to think about trustworthiness and credibility. In addition, they also wonder about how much effort has been put into creating a piece of knowledge and does it cover everything that they are interested in [85, 100, 109, 111, 131] before they can give a final verdict on reusing it or not. Therefore, we draw from and build upon these prior works, where we iterated to identify, extract, and surface not only the important elements of trustworthiness but also context and thoroughness to help consumers make a more comprehensive assessment of the appropriateness of reusing knowledge, exemplified by decisions and their rationale in programming.

2.3 Sensemaking Handoff

Much research has explored the activity of *sensemaking handoff*, during which one individual must continue the sensemaking work where another has left off. It frequently happens in asynchronous collaborations [41, 100, 101, 131], shift changes [99], etc., during which the current sensemaker (consumer) needs to make sense of and evaluate the appropriateness of reusing the results generated

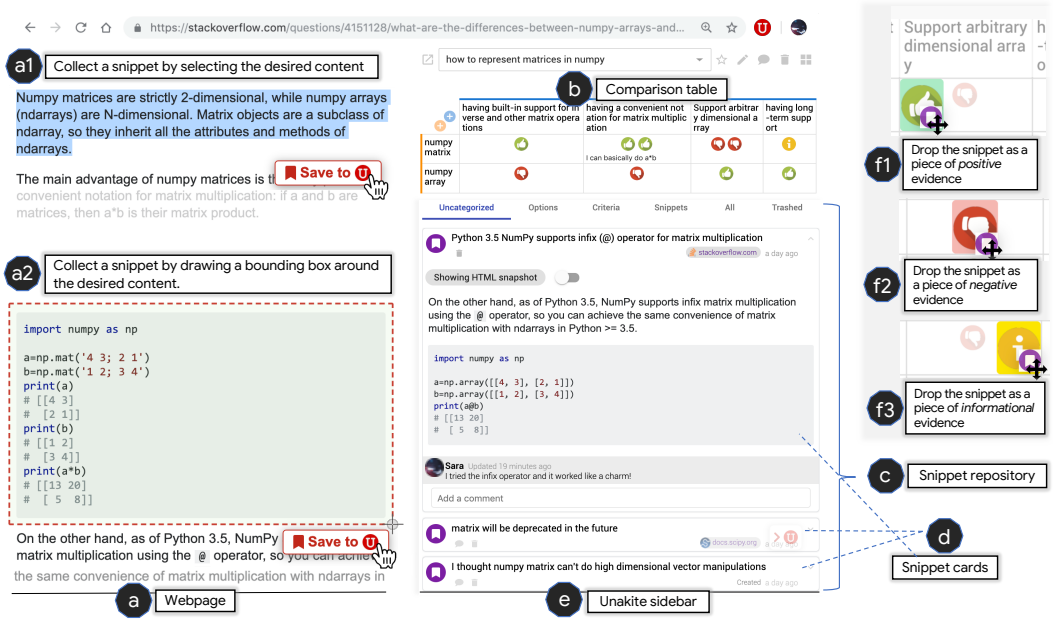


Fig. 2. Unakite's user interfaces. With Unakite, a developer collects a snippet by selecting the desired content (a1) or by drawing a bounding box around the desired content (while holding the `Option` / `Alt` key) (a2) and clicking the "Save to U" button. The collected snippet will show up under the "Uncategorized" tab in the snippet repository (c) as a snippet card (d) inside the Unakite sidebar (e), which shows the current task at the top ("how to represent matrices in numpy"). The developer can drag the snippet and drop it in one of the cells in the comparison table near the top (b), and mark whether it is positive (green thumbs-up) or negative (red thumbs-down) or just informational (yellow "i"). (f1-f3) show the details of the three parts of each cell in the table where the snippet can be dropped. For full details, see [83].

by a previous sensemaker (author) [85, 109]. Various metadata and properties parallel to the main artifacts of sensemaking have been proposed that would help the people with this process, such as the awareness of the previous sensemaking process [37, 100] (e.g., search queries and visited web pages), the level of expertise of the author [85, 111], and the context of the original sensemaking problem [85].

However, it is both time and effort intensive for an author to keep track of their rationale and processes with little immediate payoff, which is also often for the benefit of others rather than themselves [83]. Even in situations where authors have the explicit wish to help, they are often uncertain of what metadata and properties to provide and how those can be instantiated using concrete signals that would be valuable to the consumers in evaluating the reusability of their sensemaking results [109]. We address these barriers in the context of reusing decisions in programming by iteratively developing a framework that summarizes the major facets that consumers care about during the evaluation of appropriateness to reuse along with the corresponding detailed information signals, and a set of technical approaches that can automatically extract, compute, and visualize them when possible. We integrated these into our Unakite system [83] that helps authors organize and record their decisions for reuse, saving them the burden of coming up with the appropriate signals to keep track of as well as potential extra effort needed to accurately obtain them.

2.4 Knowledge Reuse in Programming

The practice of knowledge reuse has been particularly relevant in the software industry [50]. Code reuse, in particular, has become a hugely successful paradigm in the development of new software products and services in both the commercial and open source sector. Developers frequently use well-maintained functional code modules from code-sharing platforms such as GitHub [1] and npm [3], enjoying the benefits of significantly reduced workload, improved productivity, enhanced software performance, stability and security, and more time for innovation [46, 47, 50, 60, 85, 90, 94, 115].

Despite the fact that software code is the most obvious target for reuse [50, 90, 115], knowledge reuse in programming may go well beyond code, as stated by Barns and Bollinger [19]: "The defining characteristic of good reuse is not the reuse of software *per se*, but the reuse of human problem-solving." Indeed, developers on community Q&A websites like Stack Overflow [5] share not only code examples [27, 102] but also decision making strategies, design rationale such as alternative options, criteria or constraints that should be met, and the resulting trade-offs [57, 83]. Furthermore, questions about design rationale are widely cited by developers as some of the hardest to answer [70, 71, 114]. Tools like Unakite [83] can greatly reduce the costs to keep track of and later understand such rationale knowledge, with the hope that such knowledge can ultimately be better reused rather than be obtained from scratch requiring duplicated research effort [50, 81]. In the current work, we further advance this research thread by developing features and affordances enabling developers to evaluate the context, trustworthiness, and thoroughness of previously-made decisions, which is arguably one of the missing links between understanding and reuse.

3 BACKGROUND AND FORMATIVE INVESTIGATIONS

In this work, we explore augmenting knowledge reuse judgements in the context of programming, specifically in using the Unakite [82, 83] system. We first explain the design and usage of Unakite to provide a background for our research, and then describe a formative study investigating developers' issues and information needs for knowledge reuse when using Unakite.

3.1 The Unakite System

As mentioned, Unakite addresses both the need of initial developers to synthesize online information and recognize the trade-offs in programming decisions and the need of subsequent developers to be able to understand the rationale behind those decisions. Unakite, as a Chrome extension, enables the initial developers to easily collect any content from any web page as *snippets* (pieces of information, Figure 2-d) into the snippet repository (Figure 2-c) by either selecting or dragging out a bounding box around the desired content using the cursor (Figure 2-a1,2). To help with organization, developers can use drag-and-drop to move their collected snippets into a comparison table (Figure 2-b) with *options* (as row headers), *criteria* (as column headers), and *evidence* ("thumbs-up" or positive, "thumbs-down" or negative, and "informational" ("i") ratings that spread across the rest of the table cells) that illustrates the trade-offs among various solutions. All the interaction techniques involved are designed to be natural and lightweight without taxing users with much cognitive load [63, 64] while they are searching and exploring for potential solutions to their programming problems.

The resulting organizational structure is automatically saved by the system and can be accessed through a web application (with the snippet repository on the left and the comparison table on the right) with a unique URL, which can be used as stand-alone documentation of the design rationale or be integrated with code through comments. As opposed to having to speculate about the correctness and legitimacy of a decision [66], subsequent developers who have access to these comparison tables will be able to understand the context of the decision space: what options

and alternatives were considered, what criteria or constraints should be met, what the resulting trade-offs were, and what was deemed to be the most important and why.

Although Unakite has been shown through lab studies [83] to help the initial developer in making a programming decision, it displays few of the signals suggested by the research discussed above on trust and sensemaking handoff that could help consumers of the table decide whether it is appropriate for them to reuse it. For example, the initial table creator may or may not have been thorough in their research; may or may not have the same context and environment; or may or may not care about the same goals as the consumer. Although we use Unakite as a specific context, there are many similar examples of developers creating comparison tables in code documentation, blogs, and Stack Overflow [7, 8], which are typically even sparser in terms of signals for reuse appropriateness, with no supporting interactivity or drill-downs possible.

3.2 Formative Interviews

To characterize the prevalence and types of issues developers have with knowledge reuse, specifically with reuse of programming decisions, we conducted semi-structured interviews with 15 developers (5 female, 10 male). Participants were recruited through mailing lists, social media postings, and word-of-mouth. To capture a variety of processes, we chose 8 professional developers, 3 doctoral students, and 4 master students. While we do not claim that this sample is representative of all developers, the interviews informed and motivated the development of the subsequent framework (Table 1) and the design of the Strata system.

We began by asking participants about their experiences in reusing someone else's decisions when programming and how frequently would that situation occur in their work. We then explored how they manage these situations and their information needs, in particular, what questions do they have when evaluating the appropriateness to reuse and how answers to those questions may affect their final verdicts on reusability. In addition to eliciting facts on their past experiences, we also presented them with a set of decision tables in the running Unakite application (which were directly adapted from real tables online, e.g., [107]) as well as the corresponding background situational context, and asked them to judge if they could reuse these tables in those given situations. We asked them to speak about any questions they had and perform any inquiry they wanted to answer those questions (e.g., checking the sources, searching for evidence online, etc.). Finally, we wrapped up with questions probing their experience with explaining their design rationale to others, and whether and how do they convince others that their decisions are appropriate to be reused.

Interviews were conducted either in person or remotely by the first author and lasted 30 minutes. They were audio-recorded and then transcribed. In addition, screenshots of participants' computers were taken for later analysis when applicable. Then, the first author went through the transcriptions and coded them via an open coding approach [31], which included multiple iterations of discussions with the research team. Our key findings are presented below.

3.3 Preliminary Results

3.3.1 Decision reuse is frequent in programming. All participants were able to recall and describe experiences of evaluating and reusing someone else's decisions. One of the scenarios where reuse frequently happens is during code refactoring, makeovers, and takeovers, where developers are required to re-evaluate decisions made by some other developers or teams for reusability. For example, P8, a professional full-stack developer, said: *"just last month we were taking over another team's project, and the first thing we did was to re-evaluate if it still makes sense to continue building with Ruby on Rails, or it's time to do a whole re-write with React or Angular."* Another frequent reuse scenario is during project startups, where developers actively look for existing decisions on choosing architecture, frameworks, libraries, algorithms, and APIs to reuse, such as *"picking*

the right cryptographic algorithm to encrypt passwords" (P12) and *"choosing the best optimization method to train neural networks"* (P6).

3.3.2 Developers need guidance and tool support when evaluating whether to reuse someone else's decision. Although decision reuse happens frequently, participants' strategies to evaluate the appropriateness of reuse were not without troubles. 10 out of 15 said that they usually have some ideas of what types of evidence to look out for, such as the credibility of the sources, code examples, and library version mismatches, etc., but were not confident that those were sufficient. For example, P5 said: *"I feel the obligation to do more validations other than confirming they [performance metrics for different deep learning frameworks] are from official docs, but I'm not sure what else to look at or where I can find extra information."* In addition, participants reported often having to manually look for evidence of reusability (7/15), such as following the URLs previous developers left in code comments to the original web pages to validate information correctness, and pinging and asking the original author about what alternatives were considered back then. However, sometimes the sources of evidence were not possible to find since none of the sensemaking processes were consciously kept track of during the original author's decision making process other than the result.

4 FRAMEWORK

Data from the formative study suggested that developers would benefit from support in evaluating the appropriateness of reusing decisions. For example, there are many indicators that could be beneficial to surface to help users make these judgements, ranging from the expertise of the author to the quantity and legitimacy of the sources used. Although there has been little prior work characterizing the most important factors for decision reuse specifically by developers, as listed above there has been significant work discussing frameworks and measurements relevant to evaluating and reusing knowledge, such as online information credibility judgement [86–89], asynchronous collaboration [92, 101], and sensemaking handoff [41, 109–111]. From these research papers, we extracted properties and signals that would be important and relevant to decision reuse for developers.

By coding and synthesizing the aforementioned prior work as well as the formative study results through affinity diagramming, we identified three major clusters, that we call *facets*, when evaluating the appropriateness for reuse in programming: the original author's decision making *context*, and the *trustworthiness* and *thoroughness* of the resulting decision. We used these as a guide in developing an integrated framework, shown in Table 1, consisting of the three identified facets (column 1), specific information needs of developers with regard to each facet (column 2), selected evidence for the importance of these information needs as well as possible solutions to address them from prior work (column 3), and sample quotes from our formative interviews (column 4). These insights together inspired the features for our subsequent Strata system (column 5). We now discuss the framework in detail, along with the support from the prior work and the formative interviews. The design of Strata follows in section 5.

4.1 Context

Although in prior work the importance of understanding the trustworthiness of information often outshines everything else when evaluating the appropriateness to reuse [65, 85], we were surprised to find out that, at least in the domain of programming decision reuse, developers often ask questions about the *context* of a previously-made decision before they proceed to assess trustworthiness (9/15). Cited reasons include that one needs to know *"how relevant it is to what I am doing"* (P5) first, and if the context of the original decision does not align very well with the problem at hand, one

would often stop the evaluation process and move on to look for new solutions. For example, if a developer is working in Java, solutions that only work in JavaScript may not be worth investigating.

4.1.1 Goals of the original decision. When evaluating context, most (12/15) participants asked questions about the goals and purposes of the author of the decision in order to compare those with their own. For example, *"this looks like it's trying to pick a speech recognition API, but what I want is actually text to speech,"* (P14) and *"people say they want to do one thing, but after taking a closer look, they really are doing this other thing, which often makes me a tad frustrated"* (P7). Indeed, prior research suggests that the goals of decisions are often treated as "self-evident" given the results, and therefore are often not kept track of by the authors [70, 71]. On the other hand, goal mismatch does not always prevent developers from further evaluating a decision; instead, it can become a *"learning opportunity"* for them to *"know more about a new technology or design pattern"* (P11).

Furthermore, when asked about their experience of making decisions, participants reported that their goals may very well evolve with their exploration process rather than remaining fixed from the beginning (7/15). For example, *"I started out trying to choose a framework to build a mobile app for both Android and iOS, but later I stumbled upon this progressive web app thing that totally fulfills all of my requirements, so I ended up trying to learn more about that, and sort of abandoned the mobile app route that I was originally planning to take"* (P3). This motivated us to develop features (e.g., keeping track of all of the search queries used) to capture not only an author's original goal but also the evolving nature of that goal, so that later knowledge consumers could have a better grasp of how the author's goal changed throughout a decision making process.

4.1.2 Explanation or contextualization of information. One of the frustrations that participants reported having is that they often have trouble understanding the meaning of some of the criteria and evidence used in online decision tables (8/15). For example, *"what does this 'very efficient' mean, is it 'memory' or 'time' efficient?"* (P10). In some other circumstances, they suspect that evidence may not hold true when external constraints or requirements change: *"is it [a sorting algorithm] 'fast' only when there're a few hundred data points or also when there are millions of data points"* (P1). Indeed, prior work suggests that clarity and informativeness of information have a significant impact on how well it is understood [43, 118], and presenting information along with its original context (recontextualization) is considered a good way to help people understand its meaning and the conditions in which it is correct or accurate [42, 83, 85].

In addition, it was also suggested by participants that it is not always easy to recontextualize information, especially when the context is not available (6/15). Unakite partially addressed this by allowing users to create a snippet out of a large block of information in its original HTML format as well as automatically recording the corresponding source URL for later retracing [83]. In Strata, we build on that by introducing the concept of a *context snapshot*, which, at capture time, automatically keeps track of the *surroundings* of an information snippet in addition to the snippet content itself and its source URL. When consumers are reviewing a snippet, they will be able to benefit from the possible explanations such as code examples and performance metrics contained in the surroundings that would otherwise be missing from the snippet content.

4.1.3 Situational awareness. An essential part of context is the situation in which the information will be reused. In programming, this corresponds to the languages, libraries, and platforms being used, which are often referred to as *dependencies*, and participants reported checking if a given decision shares the same language or library usage as to what they have to work with (8/15). For example, P7 asked *"I want to solve it with pure JavaScript, but it seems that most of the answers here are actually written using jQuery."* Furthermore, version mismatch has been a frequent issue for reuse in programming. With the continuous rise of the open source software development model [50]

and the increasing number of frameworks, libraries, languages, and patterns [2, 4, 9], version and dependency mismatches and errors can cause troubles from missing features to breaking dependent downstream applications [10]. Indeed, participants reported checking for versions before they commit to adopting a certain solution (6/15). For example, *"I'm using Python 2.7 at the moment, which is fairly old; does this example also use this version, or is it using Python 3.5?"* These inspired us to try to automatically detect the language, library, platform, and version information whenever possible when an author collects information online, and surface this to the consumer to directly address their information needs.

4.2 Trustworthiness

As mentioned, information trustworthiness or credibility is often used as a surrogate for verifying information correctness [55], and is one of the most reported and researched facets during the evaluation of the appropriateness to reuse knowledge across many domains [85, 87]. Our interview data shows that it plays a crucial role in the domain of reusing decisions in programming as well.

4.2.1 Source credibility and diversity. As suggested by prior work, source credibility has a significant impact on the trustworthiness of information [35, 39, 43, 87, 118]. Not surprisingly, all participants in our study reported this same belief — they are more inclined towards trusting information from sources that are official (e.g., API documentation websites) or with a very good reputation within the community (e.g., Stack Overflow), and are more likely to reject information from sources that they have little experience with, echoing the *reputation heuristic* and the *expectancy violation heuristic* [89, 108] that people generally use to assess trustworthiness. For example, P12 said: *"if it's from Stack Overflow, I'm usually fine with it. But if it's from some random blog posts written by some random guy, I would probably think twice."*

It is worth noting that in addition to credibility, source diversity also plays a role in trustworthiness, according to 7 of the 15 participants. They thought that the more diverse the sources used are, the more likely that the evidence in the table has been *"peer reviewed"* or *"confirmed by a bunch of other devs"*, and *"seeing essentially the same thing independently said on a couple of different sites and forums"* gives them *"peace of mind"*. We believe that source diversity also works in concert with information popularity and consistency, which we will discuss in detail in the upcoming sections. This motivated us to provide source domain information as a direct signal for each of the information snippets collected as well as a visualization of how all the collected snippets are distributed across the different domains, enabling users to easily assess source credibility and diversity.

4.2.2 Information up-to-dateness. There was a consensus among the participants that in order to make a correct decision, the evidence used must be up-to-date (11/15). Indeed, prior work also suggests that information currency is another crucial element contributing to its credibility, with the intuition that the older a piece of information is, the more obsolete it gets, which implies a lower level of trustworthiness [15, 26, 87]. This is especially true in today's software development world, where languages and libraries are constantly being updated and older versions are quickly rendered obsolete by newer versions. For example, P6 was keen to stay on top of the state of the art of the JavaScript frontend framework competition: *"Is this speed comparison [between React, Angular, and Vue] up-to-date now that Angular 9 was just released?"* However, the above heuristic can be taken with a grain of salt by some participants, citing reasons that software that was updated a long time ago does not necessarily mean that it is obsolete. As P4 put it, *"the last release of Haskell was like 10 years ago, but it's still the latest version, and I still use it all the time in my work."* Nevertheless, we elect to provide users with direct access to at least the last updated timestamp information of each snippet that the author collected in an effort to help consumers assess up-to-dateness

faster. In addition, the separate information about versions, as mentioned above, allows users to use whichever property is most relevant.

4.2.3 Information popularity. Echoing what has been reported in prior work that people seek social proof when evaluating information credibility [87, 108], participants (8/15) said that the popularity of information also plays an important role in its trustworthiness, with the general rule suggesting that the more people that stand behind a solution, the more trustworthy it is. For example, P9 said: *"if there're a lot of other devs [who] also think this is a better idea, then I'm much more comfortable to use it."* This is similar to the *endorsement heuristic* [89], which suggests that people are inclined to perceive information and sources as credible if others do so too. This inspired us to directly present consumers with popularity signals (such as an answer's up-vote number on Stack Overflow, or the number of claps of an article on Medium.com) from where snippets are collected.

Also included in the endorsement heuristic is that people sometimes follow others' endorsements without much scrutiny of the site content or source itself [89]. However, some of our study participants suggest quite the opposite (7/15) — they often put much more emphasis on source credibility over the popularity of specific information snippets from that source. For example, *"in retrospect, if an answer is taken from Stack Overflow, I don't really care about its up-vote number or if it's the officially accepted one, I'll just trust it and use it"* (P3), or *"I don't really look at how many people clapped over a Medium article, the fact that it's from Medium.com is usually good enough for me"* (P8). Though seemingly inconsistent with prior work, we do not claim that this is typical in the domain of programming — one possible explanation is that websites like Stack Overflow by default rank the most up-voted posts at the very top with the specific intention to present the most popular information to readers.

4.2.4 Information consistency. In addition to source credibility, diversity, up-to-dateness, and popularity, a few participants (5/15) suggested that having more corroborating evidence implies that a piece of information is more trustworthy. For example, P6 said: *"This [deep learning library comparison chart] claims that PyTorch is much easier to learn than Tensorflow, but I wonder if there're people suggesting otherwise? I kind of want to see at least one other expert that has experience with both and also says PyTorch is better."* Prior research has also found that people will apply the *consistency heuristic* to evaluate credibility, validating information by checking different websites to make sure that the information was consistent [86, 89]. Meanwhile, consistency also implies the converse — having contradicting evidence will undermine the trustworthiness of an existing piece of information.

4.2.5 Author credibility. Prior work has shown that the author's level of expertise impacts the credibility of information [35, 108]. This is especially significant in the domain of programming, where there is a substantial difference between novice and expert developers in their experience and ability to evaluate code and libraries [22]. For example, when shown with a comparison table on the topic of choosing a deep learning framework, P11 asked: *"Does the author know what he's doing? I'd rather take advice from someone who's an expert rather than some random undergrad."* However, participants (4/15) also reported that there is no easy way to tell the level of expertise of a table author or if that expertise matches with the topic of the table in the current Unakite system.

Another factor that impacts the credibility of an author is if he or she is biased, possibly due to his or her affiliation or personal preferences — for instance, P12 asked: *"is the author saying all the nice things about Caffe [a deep learning framework] because he has lots of experience with it or because he's biased?"* However, one participant also acknowledged that sometimes these "biases" may not be as negative as it sounds — it could be an indication that an author is highly experienced with one particular option and therefore gives favorable evidence for it. To address the above concerns,

prior research suggests that disclosing patterns of an author's past performance may be a good indication of his or her expertise as well as possible biases [65, 113, 116]. This motivated us to at least allow the author to provide a link to his or her GitHub profile, and Strata will automatically compute and show relevant expertise metrics (contribution activities, most proficient programming languages, etc.) and affiliation information to the consumer.

4.3 Thoroughness

Another important facet when evaluating the appropriateness to reuse knowledge is thoroughness, which deals with the process and the amount of effort used when creating the knowledge, its coverage and scope, as well as any usable artifacts discovered or produced in the process.

4.3.1 Research process and effort. Prior work in sensemaking handoff recommends that when knowledge is handed-off from the author to the consumer, it should let the consumer be aware of the prior investigative process and insights [100, 101, 131], such as how much work has been done, and how mature the knowledge representation is [109, 111]. We also found relevant evidence from the interviews: three participants recalled similar experiences where they learned that the previous decision makers spent little time on exploring the decision space, and therefore the results were *"too immature to be picked up and reused"* and *"missing obvious criteria that you should definitely not leave out"*, and they ended up choosing to ignore those previous decisions and started from scratch to conduct their own research instead. This motivated us to automatically keep track of some of the authors' actions as they create tables using Unakite, such as the search queries used, the pages visited, the duration of their stay on each page and each query, etc. We then use these data to compute key statistics as well as timelines and visualize them to the consumers to help them better understand the author's research and exploration process.

P9 also envisioned that having a holistic understanding of the author's process would give her the ability to parse out the author's intention and focus (which may shift throughout the process, as discussed earlier), and therefore provide hints about what she needs to focus on next if she were to reuse this table as the basis for her own decision.

4.3.2 Alternatives or competitors. In addition to the process and effort, prior research recommends that knowledge and sensemaking results should also make apparent their coverage and scope [35, 87], for example, what alternatives have been considered, since not all options will necessarily appear in a Unakite table (especially when the author thinks one does not fit his or her particular needs and is therefore not worth further investigation). However, this does not necessarily imply that the option is inferior for the consumer. In our study, a few of our participants (6/15) were also interested in knowing what would those alternatives (or competitors) be and how they compare with the existing options before they could know if it is appropriate to reuse a table. For example, *"I heard anecdotally that Svelte gives you much better performance than all these big (JavaScript) frameworks [React, Angular, and Vue]. I should take a look at that before I decide. Or maybe there's again something else?"* (P14). This motivated us to take advantage of the Google Autocomplete API to automatically obtain commonly searched-for alternatives to the options that are already in the table, and present these alternatives to the consumers.

4.3.3 Usable artifacts. Lastly, participants (10/15) stressed the need for code examples and other usable artifacts from a decision, just as prior work reported that developers need help finding and reusing code examples [27, 28, 97, 102]. For example, P2 directly asked for code examples and the author's chosen option when presented with a decision table on various Java AST parsers: *"[are there] any code snippets that I can immediately plug into mine and test? Or if you can tell me which is the one that the author used, I'll just try that one first."* A few (3/15) participants also

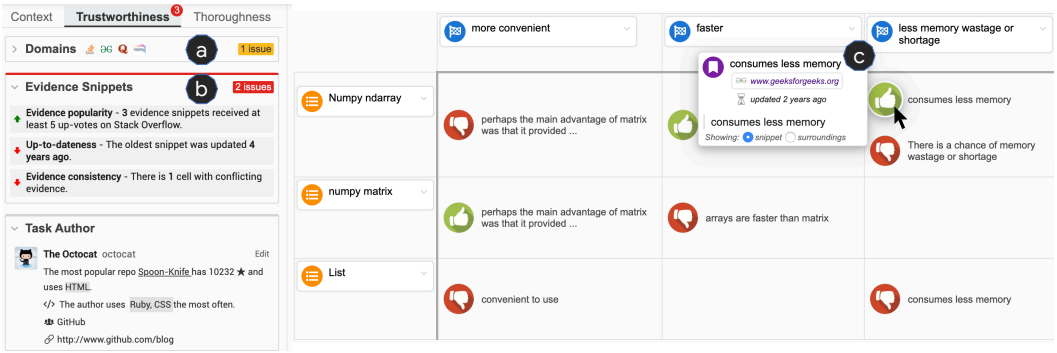


Fig. 3. On Strata startup, none of the groups are activated to keep the Unakite table on the right clean and concise. Groups can also be collapsed to keep the sidebar interface clean (such as (a)). Mousing over each snippet in the table will only show the exact content that an author captured by default (c), the same as the original Unakite system, rather than the automatically captured *context snapshots*. Only after a user activates some groups in the Strata sidebar (by clicking on their titles) will the corresponding additional metadata appear on the snippets in the table, as shown in Figure 1.

suggested that quickly trying out code examples to see if they work or not supersedes almost all other information needs. However, we do not claim this is typical, and later follow-up exchanges with these participants revealed that a vast majority of their current work is low-level detailed implementation, where making sure the code works is of paramount importance. Nevertheless, we implemented techniques to automatically extract code blocks from various snippets and present them to consumers. In addition, we also detect authors' copy events in the browser, and use those as the basis for a heuristic to tell which option the author chose for the decision.

4.4 Summary

We found that when evaluating the appropriateness to reuse a piece of knowledge, one should not only assess its trustworthiness (as the majority of the prior research has focused on), but also check for its context and thoroughness. However, no previous system has made significant attempts to address developers' specific information needs with regard to all three of these facets, or to extract appropriateness properties from the original content and present them to the consumer of the knowledge to facilitate reuse. In addition, this process should not put much burden on either the author or the consumer [83, 119] by requiring them to manually locate those appropriateness properties, suggesting the need for largely automatic mechanisms.

5 STRATA DESIGN AND IMPLEMENTATION

Based on the findings in our interviews and the framework, we built a prototype system called Strata to visualize properties and signals of the appropriateness to reuse for the consumers of a decision.

5.1 Core Design Process and Rationale

We first consulted the interview data and brainstormed the various signals and properties that would theoretically address each of the information need listed in Table 1, column 2. Some information needs can be directly addressed by obvious signals, such as surfacing the domain names of the source web pages to consumers so that they know where the information in the table were collected from and if those sources are credible. For information needs that would require explicit effort from

the table author to provide, such as the goal of a decision, we also consulted prior literature as well as brainstormed about potential indirect signals that can be used by consumers to infer those needs. For example, search queries are useful for inferring task goals and contexts of an author [23, 92, 100, 111].

In order to obtain these signals, we then built tracking techniques to automatically keep track of the author's activities in the browser while searching and browsing during the creation of a Unakite table. Many of these tracking and extraction techniques use heuristics that are based on the current design of websites that developers most often use, such as extracting the number of up-votes for an answer on a Stack Overflow page. These are meant as a proof-of-concept, and more elaborate and crowd-sourced extraction techniques could be added in the future.

We then set out to design a visualization that presents the consumers with these signals and properties. During our exploration of the design space, we struggled with a fundamental tension between consumers' awareness of all the signals and consumers' limited attention bandwidth. In our initial prototypes, we placed all the signals (approximately 15) in a scrollable vertical list to the left of the original Unakite table. Users would also be able to hide a signal if it was not relevant. We hoped to make the users aware of all the signals that Strata can provide and give them complete freedom to explore them as they wish. Another rationale for this design was that users would be able to use a combination of signals to fulfill a single information need, for example, both the search queries and the pages visited will help indicate the author's research process and effort, as evidenced by the formative interviews. However, by implementing and testing these design probes with a convenience sample of 8 developers, we realized that having *"everything all at once"* can be overwhelming to the consumers, and they would prefer to just examine one facet at a time and tune out the "noise" (signals that are irrelevant to the facet currently being examined). In addition, we found that there was a disconnect between the signals we showed in the list on the left and the actual content in the table on the right, causing consumers the additional mental burden of trying to match them up. Showing the signals in context along with the various information snippets in the table seemed to be a much better design to address this problem.

These findings guided us towards a hierarchical visualization design of Strata's consumer-facing user interface: to structure these properties and guide the consumers through their evaluation process, we designed Strata as a sidebar to a Unakite table. Strata's sidebar contains three tabbed overview panels for the three facets in the aforementioned framework (Figure 1-a). Each overview panel provides multiple *groups* (e.g., Figure 1-b,c,d) of appropriateness properties to directly address consumers' information needs as summarized in the framework. In addition, by activating one or more of the groups (by clicking on their titles in the sidebar), consumers will be able to view additional information specific to each snippet in the table. For example, Figure 3 shows a state where none of the groups are activated. After activating the Domains group and Evidence Snippets group, consumers will be able to see for each snippet: where it originated (Figure 1-g1), how popular it is (Figure 1-g2,3), and how old it is (Figure 1-g4). This is designed to provide consumers with a high-level overview of each of the facets of reuse as well as the ability to dive into the parts of interest, as recommended by Shneiderman [112]. It is also inspired by the *lens* interaction [24, 30] where the same table content is addressed from three different perspectives.

Like Unakite, Strata consists of an extension to the Chrome Web browser and a web application. Strata's Chrome extension implements the aforementioned new tracking techniques on top of the Unakite Chrome extension. The Strata web application is implemented in HTML, JavaScript, and CSS, using the React JavaScript library [40] as the primary frontend UI development framework and Google's Firebase on the Google Cloud for data management and synchronization as well as user authentication.

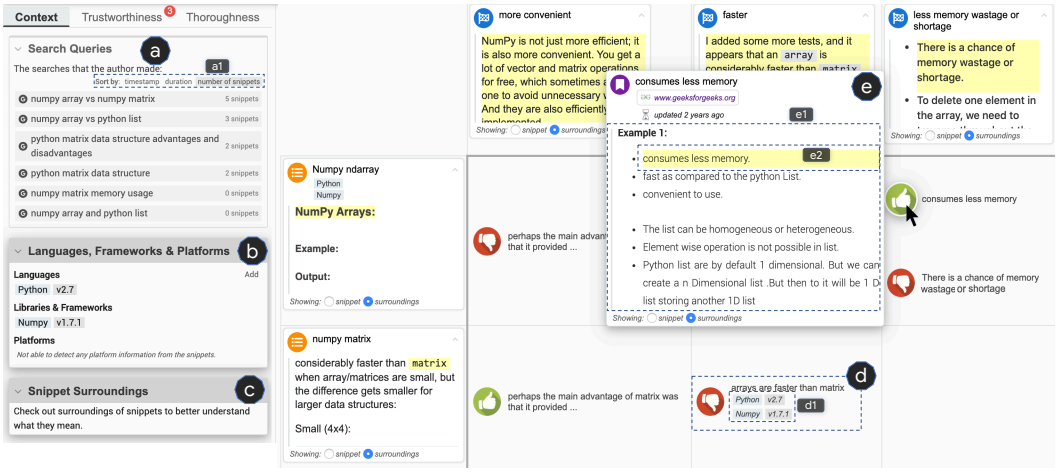


Fig. 4. Strata's *Context* panel. Consumers are able to check the search queries (a) that the author used to understand his or her goal, examine the languages, frameworks, platforms, and their versions of the snippets (b, d1), and view the surroundings of a snippet through the automatically captured context snapshots (e1).

We now discuss how the different features in Strata support the three facets listed in the previous framework, and how they are implemented.

5.2 Context

5.2.1 Capturing goals with search queries. First of all, Strata automatically keeps track of authors' search queries used in Unakite tasks as well as the duration of time they spent on each and the number of information snippets they collected. The duration information is approximated by comparing the timestamp when the next query is issued to that of the current one. It also automatically leaves out any idle time (i.e., time where there is no activities detected in the browser, by monitoring mouse movements, keyboard input, etc.) that are longer than a certain threshold to make the duration approximation more accurate. The idle threshold was empirically tuned to be 8 seconds based on data obtained through pilot testing, and can be flexibly adjusted in the future. For consumers, Strata visualizes these search queries as a list (Figure 4-a) to help consumers understand the goals of the task author. They can use the sorting mechanisms at the top (Figure 4-a1) to sort the search queries by chronological order, by duration, or by the number of information snippets yielded from each (which is the default sorting order, where ties are broken by ascending chronological order).

There are several advantages of using search queries as a representation of an author's goals. First, they are direct translations of what an author thinks and intends to do to satisfy their information need [104] — for example, issuing the query "numpy matrix vs list" implies that the author would like to find out the differences between the two options. Second, unlike the original Unakite where an author sets the single task goal (as the name of a task) at the beginning, keeping track of all of the search queries (in temporal order) captures not only the author's original goal (which usually is the first query based on pilot study data) but also the evolving nature of the goal (as identified in the formative interviews). Third, the number of snippets yielded from each query serves as an approximation of an author's effort spent on that particular part of the task, which informs consumers of the author's focus throughout the decision making process.

5.2.2 Contextualizing information with automatic context snapshots. To help consumers contextualize and understand the meanings of options, criteria, and evidence in Unakite (identified as one of participants' frustrations), Strata introduces the idea of automatically keeping a snapshot of the surroundings of a piece of content called *context snapshot* (inspired by [58]) as an author collects information snippets. Strata uses Unakite's *snapshot* feature, where website content can be captured and preserved with its original styling, including the rich, interactive multimedia objects supported by HTML. The bounds of the surroundings are by default defined as the main content (Strata automatically tries to exclude any advertisements and other forms of injected content on a website) in the visible area of a web page in the browser window. In addition, due to the popularity and importance of Stack Overflow in the domain of programming, we specifically optimized this feature to include not only the particular answer block an author collects information from but also the original question block regardless of whether they are within the bounds, which provides consumers with extra context information. Similar optimizations for other popular developer sites, such as the official documentation, could be added in the future. On the consumer side, by clicking on the title of the *Snippet Surroundings* group (Figure 4-c) in the Strata sidebar, consumers will be able to view and scroll through the surroundings for each snippet (Figure 4-e1), with the content that the author specifically collected highlighted in yellow (Figure 4-e2).

This feature offers several benefits to both the authors and the consumers. The surrounding of a snippet is highly likely to include explicit explanations (such as screenshots, code examples, and execution results) that can help consumers understand exactly what a snippet means. For example, the *Python Lists VS Numpy Arrays* article [12] where a criterion snippet "more efficient" was scooped from, also gives examples of how the two data structures allocate memory blocks under the hood, suggesting that the author actually meant "more **memory** efficient" rather than "more **time** efficient". Unlike in Unakite, where an author needs to specifically include that entire paragraph when creating a snippet and then manually change the title of the snippet into "more memory efficient" (which may disrupt the workflow), Strata will automatically capture that helpful paragraph into the snippet's context snapshot. During the evaluation of context, consumers will be able to directly view a snippet in its surroundings through its context snapshot without frequent switches to the corresponding original web page to find where the content where the snippet was taken from (which is exactly what participants reported doing in the formative study interviews).

5.2.3 Detecting languages, frameworks, and their versions. Strata tries to automatically detect the languages, frameworks, platforms, and their versions used in the snippets to directly address consumers' information needs. To ground this feature, we picked the top 10 of each of the most popular languages, frameworks, and platforms from the 2020 Stack Overflow developer survey [13] and built *detectors* for them. The detectors for a language (or a framework, platform, etc.) is implemented as a set of manually devised keywords (e.g., language statements, special variables, file extensions, etc.) that can uniquely identify the usage or presence of that language. For example, "`es7`", "`console.log`", "`setTimeout`", etc. can be used to identify *JavaScript*, and "`useState`", "`componentDidMount`", "`findDOMNode`", etc. and be used to identify the *React* library. Keywords that can cause ambiguities are specifically avoided, such as "`$`" (the dollar sign) is simultaneously a way to refer to variables in *PHP* and a shortcut for *jQuery*. Strata then automatically tries to find these detectors through optimized string matching in a snippet upon its collection. If there is no hit within the snippet content, Strata will make a second attempt with the content of the snippet's parent web page. Subsequently, Strata uses regular expressions to find version numbers in the vicinity of detected languages, frameworks, and platforms (e.g., "`Angular` **`9`**", "`Python` **`3.5`**", "`React` **`16.13.1`**", etc.) or in the web page's URL (e.g., Java SDK version numbers are encoded in the URL of its official documentation website). In an informal evaluation using materials containing

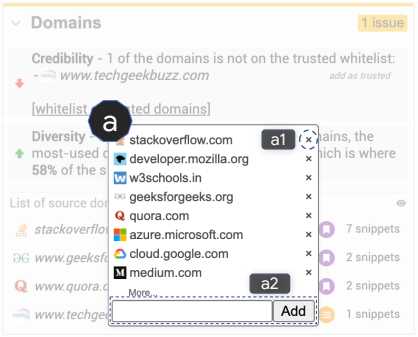


Fig. 5. The *trusted domains whitelist*. Consumers are able to remove a certain domain from the whitelist (a1) or add new ones (a2).

only the currently supported languages, this mechanism was able to successfully extract language information 100% of the time and correctly identify the version information 96% of the time. In the future, one might imagine Strata pulling detectors from open-source detector repositories built, verified, and maintained by the community, which can improve their quality, precision, and recall, or at the very least, letting authors add or correct wrongly detected versions. On the consumer side, this detected information is then presented directly on the corresponding snippet cards in the table (Figure 4-d) as well as aggregated in the *Languages, Frameworks, and Platforms* group (Figure 4-b).

Directly surfacing these language, framework, and platform version entries to consumers will help them quickly understand the technologies used in the task as well as the specific versions each snippet uses at a glance, to support comparing those with their own situation. For example, one developer would be easily able to figure out that the example code collected by the other developer uses Python 2.7 and therefore does not match with his or her own environment, which uses Python 3.5.

5.3 Trustworthiness

To help consumers evaluate the trustworthiness of a table, Strata provides visualizations of various properties that directly address their information needs listed in the framework (e.g., source credibility, information popularity, etc.). Prior work has suggested that surfacing issues or problems that could cause distrust is an effective way to alert and guide users' attention during credibility evaluations [87]. Therefore, in addition to visualizing the trustworthiness properties, we remind users of potential issues that could negatively impact a table's trustworthiness by marking them with a red downward arrow (Figure 1-b2,c3,c4). The count of the number of issues is shown in a colored badge on the top-right corner of the Trustworthiness panel (Figure 1-a1), with one issue having a yellow color, and more than one issue having a red color (these user-adjustable levels were empirically determined). Future development will explore more sophisticated weighting of the issues beyond counting them equally.

5.3.1 Visualizing source credibility and diversity. As shown in Figure 1-b, Strata visualizes the distribution of the snippets across different domains (websites) (Figure 1-b5), giving consumers a high-level overview of the provenance of the information in the table. In addition, each snippet in the table is also marked with its domain (Figure 1-g1), giving consumers a detailed understanding of where each snippet originated.

Strata also alerts consumers of potential untrusted domains by checking the presence of each domain on a user-defined *trusted domains whitelist*, and flags the ones that are not on the list. For example, a consumer will be able to immediately notice that one of the websites that the author used to collect evidence, `techgeekbuzz.com`, is not on his or her own trusted domains whitelist (Figure 1-b2). Currently, the default whitelist was generated by mining and aggregating the websites that 5 full-stack developers (who work for different technology companies and routinely use a variety of languages and technology stacks) visited from their browsing history. We then had them each annotate the websites as either "credible" or "not credible", and removed the ones that they did not all agree upon. This resulted in 25 domains that are considered "credible", including community Q&A sites like `stackoverflow.com`, official documentation sites like `angular.io`, and blog sites like `medium.com`. Domains that sometimes contain non-objective and low-quality information are rejected, such as `reddit.com`. We by no means claim this is complete nor that it applies to everybody — instead, it serves as a starting point and the consumers are able to add and remove items themselves (Figure 5-a1,a2). They can also use the "add as trusted" button (Figure 1-b3) to add a flagged website to the whitelist so that any future information originating from that website will not be considered as an issue. In the future, one can imagine taking advantage of a larger consumer base and automatically marking websites as trusted if a majority of the consumers have it on their whitelist. We also expect to periodically update the default whitelist over time, as new programming technologies are created and become popular in the future.

To help with the evaluation of source *diversity*, Strata also alerts consumers when there is only limited sources used to construct a table. Currently, Strata considers that there is an issue in terms of source diversity if all of the information comes from one single source (reported by participants in the formative studies as the worst scenario). If that is the case, the green upward arrow for source diversity in Figure 1-b4 will become a red downward arrow, reminding consumers that it is an issue. However, this threshold can be set by individual consumers, which would then apply to all future table evaluations they perform. Similar to source credibility issues, this can also be resolved or dismissed by individual consumers if they do not think it is problematic.

5.3.2 Examining evidence trustworthiness. Consumers will be able to get information about the popularity, up-to-dateness, and the consistencies of the evidence by activating the *Evidence Snippets* group (Figure 1-c).

Each snippet in the table will be marked with signals showing its popularity depending on the websites and pages that it originates from. For example, if a snippet is collected from a Stack Overflow answer post, Strata will automatically extract and show the up-vote number of that post (Figure 1-g2) as well as if that answer is the officially accepted answer (Figure 1-g3). If a snippet is collected from a Medium.com article, Strata will show the number of claps that article had at the time of collection. We designed this feature to closely fit developers' current ways of evaluating popularity, as reported in the formative studies. Strata will also display an alert in the Evidence Snippets group if some of the snippets in the table have particularly low popularity, such as down-votes on Stack Overflow. As with the other kinds of detectors, we envision these being augmented over time based on where developers are mostly getting their information from.

Unlike the original Unakite, which only showed *when* information was collected (reported as *"not exactly helpful"* by participants in the formative interviews), each snippet in the table will be marked by Strata with the timestamp of when its parent webpage (or answer post if it is from Stack Overflow) was last updated (Figure 1-g4). Strata uses a combination of techniques to extract the last updated timestamp information, including using regular expressions to look for date strings in website source code and taking advantage of the JavaScript `document.lastModified` variable (only when the website is static). This serves as a direct measurement of the age of information,

and gives consumers an idea of how old the information is. Our study participants also mentioned that they often had trouble quickly locating when articles or blogs are updated online as these timestamps are often displayed in less salient font styles or not visible at all. In addition, Strata will flag snippets that are older than 3 years as a potential issue in the Evidence Snippets group (Figure 1-c3), which, similar to other issues, can be manually adjusted or dismissed by the consumer.

Finally, Strata provides initial support for information consistency by informing consumers if there are corroborating or conflicting evidence snippets in a table cell (e.g., there are simultaneously both thumbs-up and thumbs-down ratings for "`numpy ndarray`" causing "less memory wastage or shortage") (Figure 1-c4). The culprit table cells with conflicting evidence will be highlighted by mousing over the issue in the Evidence Snippets group, addressing concerns from participants in the formative studies about how such contradictions could be overlooked once a table gets larger with more evidence ratings.

5.3.3 Surfacing properties about author credibility. Strata provides consumers with help in evaluating author credibility by allowing authors to manually provide information about themselves. In the current implementation, a table author can input a link to their GitHub profile, and Strata will automatically present the author's name, numbers of stars on the most popular code repositories he or she owns, most used programming languages, affiliation, and a link to his or her GitHub profile page in the *Task Author* group (Figure 1-d). We opted to let authors voluntarily provide this information in order to give them the option to protect their privacy and identity. In the future, we will work on mechanisms to automatically perform author modeling in a privacy-preserving way — one idea is to analyze the topics of Stack Overflow questions and coding forums that an author frequently visits to infer his or her expertise. We will also provide an option for authors to provide certain information to consumers anonymously.

5.4 Thoroughness

5.4.1 Understanding the research process. In order to provide consumers with a clear understanding of an author's research and exploration process, Strata automatically keeps track of several of the author's activities in the background — in addition to the search query tracking discussed earlier, Strata also automatically records the web pages visited, as well as the time spent, progress made (approximated by tracking the percentage of a page that has been scrolled into the visible browser window using JavaScript's `window.onscroll` event), and the number of information snippets collected on each of the web pages.

With these activity data, Strata computes the duration of time the author spent working on a task, the length of time since the task was last updated by the author, and the numbers of options, criteria, and evidence snippets that the author collected (Figure 6-a1).

In addition, Strata visualizes the activity information on a timeline view (Figure 6-b), which provides an integrated chronological representation of the author's entire research and exploration process during a task. The timeline view is organized with two levels of hierarchies: first by the search queries, and then by the pages that are visited during a particular search. The timeline view is color-coded by different shades of a violet color, with increasing intensity indicating the chronological order (a lighter violet means older). The same color scheme is also applied to the background of the table cells (Figure 6-c) when the *Research Process* group is activated. The timeline view is also interactive, mousing over a search query or a page will highlight its corresponding information snippets in the table, together with the colored background, giving consumers an understanding of how the table was constructed chronologically.

5.4.2 Suggesting alternatives. Another way for Strata to help with the thoroughness evaluation is to provide consumers with *commonly searched for alternatives* to each option (Figure 6-f1,f2,f3). For

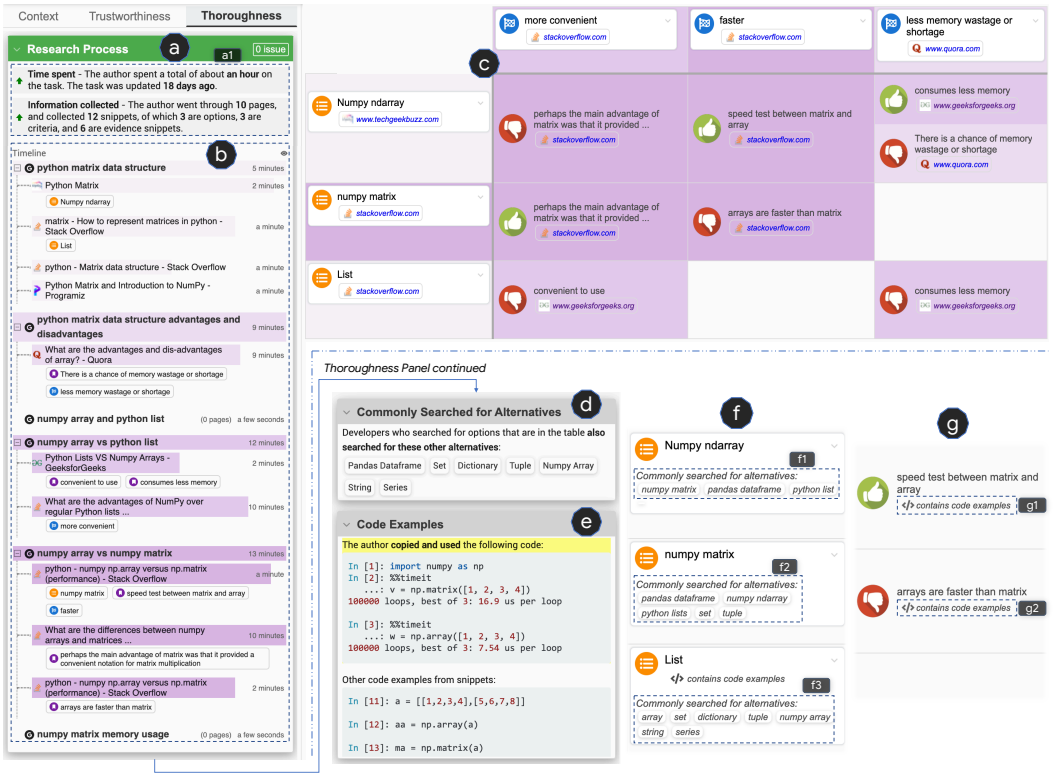


Fig. 6. Strata's *Thoroughness* panel. Consumers are able to understand the author's research process (a) with the help of the timeline view (b) (a lighter violet means older chronologically), check commonly searched for alternatives to the existing options (d, f1, f2, f3), and check the code examples in the snippets (e).

every option in the table, Strata will automatically obtain the potential alternatives to that option by making Google search queries in the form of "[option_name] vs" or "[option_name] versus" and obtaining a list of top 10 auto-complete candidates using the Google Autocomplete API. This will then be transformed into the *alternatives list* for the corresponding option by extracting and cleaning the part after "vs" or "versus" for each auto-complete candidate, followed by aggregating and removing duplicates. The results are presented in the *Commonly Searched for Alternatives* group (Figure 6-d). These alternative lists are generated on the spot every time a table is being reviewed, making sure that Strata always presents the latest information.

This approach offers several benefits to the consumers of the table. First, it offers insights into the popularity of the existing options in the table — if an option (such as "React") appears in all other options' alternatives lists (such as for "Angular" and "Vue"), it suggests that this option has a high popularity. Second, it provides consumers with an understanding of the coverage of the author's research process as well as guidance on potential new opportunities to explore next — if an item (such as "pandas dataframe" in Figure 6-d) frequently appears in the existing options' alternatives lists (and therefore will rank higher in the aggregated list in the Commonly Searched for Alternatives group), it suggests that this item might have been overlooked by the author initially, or it might not have been available back when the table was made, and the consumers can focus their investigative effort on it next before deciding whether to reuse this table. This feature could

help authors as well, offering real-time reminders of the coverage of their research process and possible new options to consider as they are making decisions.

5.4.3 Presenting usable artifacts. Finally, Strata automatically detects and extracts any code examples included in the collected snippets and presents them in the *Code Examples* group under the Thoroughness panel (Figure 6-e). This provides consumers the opportunity to directly examine and try out any code examples involved first without diving deeper into the table. In addition, when the Code Examples group is activated, a *"contains code examples"* badge (Figure 6-g1,g2) will appear on snippets that contain code examples, helping consumers quickly locate potential code examples for a particular option or criterion in the table.

6 EVALUATION

We conducted a lab study to evaluate the effectiveness of the framework and the prototype Strata system in helping developers evaluate the appropriateness of reusing decisions.

6.1 Experiment Design

6.1.1 Participants. We recruited 20 participants (13 male, 7 female) aged 22-37 ($\mu = 26.95$, $\sigma = 3.81$) years old through emails and social media. The participants were required to be 18 or older, fluent in English, and experienced in programming. Participants on average had 8.3 ($\sigma = 3.3$) years of programming experience, with 11 of them currently working or having worked as a professional developer and the rest having programming experience in universities.

6.1.2 Procedure. Participants were presented with 3 tasks in random order. The topics of the tasks were: *(a) choosing a python data structure to represent matrix-like data* (referred to as *Python* from here on), *(b) choosing a deep learning framework to build neural networks* (referred to as *Deep* from here on), and *(c) choosing a cloud computing service to build a video-streaming application* (referred to as *Cloud* from here on). For each task, participants were told what to pretend their background and context was, and they needed to read a table and answer questions about: (1) how much do they think the table is relevant to their given background and context; (2) how much do they trust the content of the table; and (3) to what extent do they think the research effort put into making the table is thorough. Participants were required to list out specific reasons to justify their evaluations.

The study was a between-subjects design, where participants were randomly assigned to either the Strata condition or the Unakite (control) condition. In the Strata condition, participants had full access to all the Strata features described above (along with the table produced by Unakite), while in the Unakite condition, these new features were turned off, so the participants saw only the table, and snippets in the table only showed their titles, contents, timestamps of collection, and links to their original web pages. We imposed a 10-minute limit per task to keep participants from getting caught up in one of the tasks. However, participants were instructed to inform the researcher when they thought they had finished the task or felt like they could make no further progress.

We chose Unakite as the control condition as opposed to raw (and textual) comparison tables online to make sure both conditions had a similar user interface to work with. It also makes the comparison between conditions more realistic — since the original Unakite is already keeping track of where snippets are collected, participants in the Unakite condition would have the ability to go back to the source to examine the appropriateness signals (such as up-vote numbers, last-updated timestamp, etc.) if they wanted to.

Each study session started by obtaining the proper consent and having the participant fill out a demographic survey. Participants in the Unakite condition were given a 10-minute tutorial showcasing the various features of the Unakite web application as well as a practice task on the topic of "choosing a JavaScript frontend framework" before starting. Those in the Strata condition

	Time	$n_{\textbf{Total}}$	$n_{\textbf{Valid}}$ for Context	$n_{\textbf{Valid}}$ for Trustworthiness	$n_{\textbf{Valid}}$ for Thoroughness	$n_{\textbf{Valid}}$	$n_{\textbf{High Quality}}$	Precision	Recall
Unakite	484.2 (37.8)*	5.20 (0.92)*	1.50 (0.53)	1.30 (0.48)*	1.20 (0.42)*	4.00 (0.67)*	2.90 (0.57)*	55.7% (4.9%)*	24.2% (4.7%)*
Strata	328.2 (48.1)*	7.90 (1.91)*	1.50 (0.53)	3.20 (0.79)*	2.70 (0.82)*	7.40 (1.51)*	7.10 (1.45)*	90.1% (6.8%)*	59.2% (12.1%)*

(a) Python ($n_{\textbf{Ref. High Quality}} = 12$)

	Time	$n_{\textbf{Total}}$	$n_{\textbf{Valid}}$ for Context	$n_{\textbf{Valid}}$ for Trustworthiness	$n_{\textbf{Valid}}$ for Thoroughness	$n_{\textbf{Valid}}$	$n_{\textbf{High Quality}}$	Precision	Recall
Unakite	393.4 (50.9)*	5.70 (1.06)*	1.70 (0.48)	1.60 (0.70)*	1.40 (0.52)*	4.70 (0.82)*	3.20 (0.92)*	56.1% (12.4%)*	29.1% (8.3%)*
Strata	276.2 (68.3)*	7.80 (1.87)*	1.70 (0.67)	3.00 (1.15)*	2.60 (0.70)*	7.30 (1.83)*	6.90 (1.97)*	88.1% (9.7%)*	64.5% (17.4%)*

(b) Deep ($n_{\textbf{Ref. High Quality}} = 11$)

	Time	$n_{\textbf{Total}}$	$n_{\textbf{Valid}}$ for Context	$n_{\textbf{Valid}}$ for Trustworthiness	$n_{\textbf{Valid}}$ for Thoroughness	$n_{\textbf{Valid}}$	$n_{\textbf{High Quality}}$	Precision	Recall
Unakite	420.4 (58.9)*	6.20 (1.03)*	1.40 (0.51)*	1.90 (0.74)*	1.50 (0.53)*	4.80 (1.14)*	3.60 (0.97)*	58.5% (15.2%)*	30.0% (8.1%)*
Strata	271.8 (35.3)*	9.60 (2.37)*	2.60 (0.84)*	3.80 (0.92)*	2.60 (0.70)*	9.00 (2.00)*	7.90 (1.45)*	83.8% (8.5%)*	65.8% (12.1%)*

(c) Cloud ($n_{\textbf{Ref. High Quality}} = 12$)

Table 2. Lab study results. The numbers of gold standard high quality reasons for each task, $n_{\textbf{Ref. High Quality}}$, are listed in their respective captions. We report the mean and standard deviation for: (1) the **time** in seconds taken to finish a task; (2) the total number of reasons participants came up with, $n_{\textbf{Total}}$; (3) the number of valid reasons, $n_{\textbf{Valid}}$; (4) the number of high quality reasons, $n_{\textbf{High Quality}}$; (5) the precision of high quality reasons, calculated as $n_{\textbf{High Quality}}/n_{\textbf{Total}}$; (6) as well as the recall of high quality reasons, calculated as $n_{\textbf{High Quality}}/n_{\textbf{Ref. High Quality}}$. Statistically significant differences ($p < 0.05$) through t-tests are marked with an *.

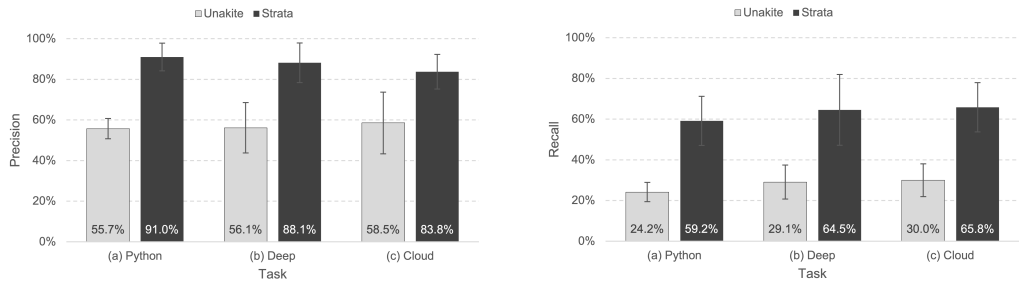
were given a same-length tutorial as well as the same practice task but in Strata instead. At the end of the study, the participant was invited to fill out a questionnaire focusing on the experience of using either Strata or Unakite. We asked questions on the usability of the system they used in their respective conditions, the usefulness of such tables generated by the system, their opinions of the different features of the system, their willingness to author tables using the system to keep track of their decisions, their concerns about privacy if they were to author tables, as well as their familiarity with the topic of the three tasks used in the study. Finally, we ended the session with an informal interview on any additional thoughts they had about the system they used. Each study session took about 60 minutes per participant and was done remotely using the Zoom video-conferencing application. All participants were compensated $15 for their time.

6.2 Quantitative Results

All participants were able to complete all of the tasks in both conditions, and none of them went over the pre-imposed time limit.

The results show that the participants in the Strata condition took significantly *less time* to finish compared to the Unakite condition for all three tasks, as shown in Table 2. Across all three tasks, the average time for completion was reduced by 32.5% when using Strata (Mean = 292.1 seconds, σ = 56.9 seconds) compared to using Unakite (Mean = 432.7 seconds, σ = 61.8 seconds), which is also statistically significantly ($p < 0.05$). Thus, using Strata did help participants evaluate the appropriateness for reuse faster.

To assess the *quality* of the reasons that participants came up with, before the study, two professional developers who are not affiliated with the research each generated a list of *high quality* reasons for all three tables independently. After resolving conflicts through discussions between the two developers, we produced a list of high-quality reasons for each table as the "gold standard". We then calculated and report in Table 2 the numbers of high quality reasons participants identified that are on the "gold standard" list, as well as the precision (calculated as $n_{\textbf{High Quality}}/n_{\textbf{Total}}$) and



(a) Precision of high quality reasons ($n_{\textbf{High Quality}}/n_{\textbf{Total}}$.)

(b) Recall of high quality reasons ($n_{\textbf{High Quality}}/n_{\textbf{Ref. High Quality}}$.)

Fig. 7. Precisions and recalls of high quality answers in all three tasks. All results are statistically significant under t-tests ($p < 0.05$).

recall (calculated as $n_{\textbf{High Quality}}/n_{\textbf{Ref. High Quality}}$) of high-quality reasons (where $n_{\textbf{Total}}$ is the total number of reasons they generated, and $n_{\textbf{Ref. High Quality}}$ is the number of "gold standard" high-quality reasons for each task). By plotting the precisions and recalls in Figure 7, we can see that participants in the Strata condition achieved higher precision in all three tasks, that is, they gave a higher percentage of high-quality reasons in their responses compared to the Unakite condition. Participants in the Strata condition also achieved higher recall in all three tasks, that is, they were able to find more high-quality reasons compared to the Unakite condition. Thus, using Strata did help participants improve the quality of their evaluations compared to using Unakite.

In case participants came up with valid answers we had not thought of, after the study, we asked the same two developers as above to rate each reason that participants gave as either *valid* or *not valid* blind to the conditions. Valid reasons are considered as the ones that are specific and correct according to the content of the table. After resolving conflicts through discussions between the two developers, we filtered out the reasons that are considered *invalid*, and presented the resulting numbers of valid reasons in Table 2 (the numbers of invalid reasons were negligible and were therefore not included in the table). Across all three tasks, the average total number of valid reasons ($n_{\textbf{Valid}}$) increased by 75.6% when using Strata (Mean = 7.90, σ = 1.90) compared to using Unakite (Mean = 4.50, σ = 0.94), which is also statistically significant ($p < 0.05$). Thus, using Strata appeared to help participants come up with more valid evaluations for appropriateness for reuse compared to Unakite alone.

In the survey, participants reported (in 7-point Likert scales) that they thought the interactions with Strata were understandable and clear (Mean = 6.20, Median = 6.00, 95% CIs = [5.75, 6.46]), they enjoyed Strata's features (Mean = 6.00, Median = 6.00, 95% CIs = [5.45, 6.72]), and would recommend Strata to friends and colleagues (Mean = 6.10, Median = 6.00, 95% CIs = [5.65, 6.35]).

6.3 Qualitative Results

6.3.1 Usability and usefulness of Strata's features. Overall, participants appreciated the increased transparency and efficiency afforded by various Strata features and highlighted the values of the appropriateness properties that we visualize, arguing that *"it helps me understand how a table was made step by step"* (P10), *"lets me know what the author searched for, so if I don't understand something, I can search again. And more importantly, I can sort of know what the author didn't look for, and sometimes that'll become what I can do next"* (P4), *"[the automatic context snapshot feature] saves me lots of time that I would otherwise spend going to the source web pages and making sense of things, which could be a rabbit hole sometimes"* (P15), and *"[allows me to] see on a high-level where stuff*

comes from and if there's any source that is potentially questionable" (P13). In addition, P8 reflected that Strata *"serve(d) as a guidance for things that I should pay attention to,"* which underlines the value of our framework, and reminded some participants of appropriateness properties that they would otherwise overlook, such as *"I never really thought about what the author(s) looked for or not, but now I think it's actually quite important, especially if they miss obvious things that an expert would never miss,"* (P6) and *"I realize that I'm more of a grab-and-go kinda person and I don't usually remember to check how many up-votes a Stack Overflow answer gets or when it was last updated"* (P17).

6.3.2 Authoring tables. Participants were also excited about authoring tables with Strata running, as it will automatically extract and produce the sidebar on the left and the various signals in the table. They mentioned that such *"honest signals enhanced"* (P10) tables would be particularly useful in situations such as code reviews (P6: *"going through the three main aspects is like going through our usual quality checklist, which makes sure that we're not missing anything"*) and project takeovers (P13: *"if my previous browsing sessions are captured by this, then I won't need to make myself available again and again if somebody else suddenly has a question that only I know the answer to, since I made it in the first place—this table thing will almost be self-explanatory"*).

6.3.3 Privacy concerns. Some participants shared their privacy concerns from an author's perspective, mentioning that certain types of metadata that could reveal their personal preferences and idiosyncrasies (e.g., the code that they used, the snippet surroundings, and their search queries) should be kept private until they felt comfortable sharing. Indeed, prior work has pointed out that there may be negative effects of surfacing certain types of information [38]. These findings identified new research opportunities for (1) intelligent mechanisms that can automatically screen for and block out information that should be kept private (e.g., similar to [74] or [61]) and (2) mixed-initiative and interactive mechanisms [56] that collaborate with users to only preserve the information that they are comfortable sharing (e.g., similar to [75]) without compromising the usability and effectiveness of the system.

7 DISCUSSION

Prior research on web credibility stressed the importance of trustworthiness measurement during the evaluation of the appropriateness to reuse a previously created knowledge artifact [55]. However, as we found from literature on sensemaking handoff and our formative study, evaluating the appropriateness of reuse is much more than simply verifying the trustworthiness [55], especially since the artifacts are often an author's collection and synthesis of different individual pieces of information from different sources and reflect the author's opinion about the trade-offs among multiple valid options [83]. As a result, in addition to understanding whether the content is trustworthy, consumers also need to understand if the original problem context when the author created the artifact matches with the consumer's [55, 85], and if the author's research process was thorough [37, 100]. One of the contributions that we make in this work is a framework (Table 1) that summarizes the aforementioned three major facets, serving as a checklist that guides consumers through their evaluation processes. Strata, which is an instantiation of the framework, improves consumers' abilities to evaluate these facets compared to using Unakite alone, as evidenced by both the quantitative (i.e., number of valid reasons given by the participants in terms of each facet) and qualitative results (e.g., participants' comments on Strata reminding them of double checking appropriateness properties that they would otherwise overlook).

Although prior work on trust and sensemaking handoff offers insights into the various aspects and properties that are important for evaluating the appropriateness of reuse, it remained costly and difficult for not only the author who was creating the knowledge to also keep track of those

signals and save them somewhere (since it is extra work without immediate benefit), but also for the consumer who was interpreting the knowledge to deduce and speculate about those signals. Through our research, we learned that an reasonable number of appropriateness signals can automatically be captured at authoring time as well as processed and visualized to the consumers subsequently to help with the reuse evaluation, and thereby reduce the cost for people to build on each other's knowledge artifacts.

8 FUTURE WORK

One participant (P4) in the evaluation study said *"I can imagine myself having this table page open as I collect stuff so I can check how well I'm doing as I go"*, which suggests that Strata not only can help consumers but also provides value for authors at collection time — authors can use Strata features to help them know how well their decisions will be judged, how thorough they have been, whether they are using up-to-date materials, if there have been any version mismatches, etc. In the future, we would like to investigate how to integrate these Strata visualization features into authors' workflows to help them "proofread" their decision making processes in real time.

Currently, Strata has settings that consumers can tune based on their personal preferences, such as the trusted domain whitelist. Future work is needed to investigate mechanisms that can enable consumers to also personalize an existing table, such as adding, editing, and removing certain elements, effectively creating new versions of that table without overriding the original author's version. In addition, it would also be an interesting challenge to aggregate the changes in different consumers' versions and propagate them back to the original author as constructive feedback.

Finally, our approach may have potential implications for other situations and domains involving user-generated content (beyond comparison tables), in which knowledge consumers need to evaluate the relevance and trustworthiness of that content. For example, the context, trustworthiness, and thoroughness facets could provide generative inspirations for helping users evaluate how knowledge artifacts were constructed, such as in Wikipedia (e.g., which sources were considered for an article, properties of the contributors, and coverage of key topics mined from similar articles), Q&A sites like Stack Overflow where many people collaborate and edit questions and answers together; curation platforms such as Pinterest, or thousands of other wiki systems. Generalizing how to augment knowledge reuse for situations beyond decision making in programming is an interesting and potentially fruitful area for future investigation, including exploring which information needs identified in this paper may not be as relevant and which additional needs become important. On the one hand, such an endeavor could unlock cycles of knowledge reuse in which people can quickly make good judgements about which information to aggregate and accumulate, which then become useful signals for making future judgements easier as well. On the other hand, the various signals and properties that are automatically surfaced could raise consumers' awareness of the potential existence of mis-information online [121] and provide readily available evidence to combat it.

9 LIMITATIONS AND RISKS

There are certain types of information that Strata is not able to automatically obtain and visualize. One set of limitations results from Strata working in the browser, so it cannot monitor activities which happen in the authors' code editors or IDEs, command line interfaces, and relevant discussions with friends and colleagues (communicated either verbally or electronically through chat applications like Slack). Further development of extensions in these different environments as well as research into how to coordinate the collection and organization of this information would be needed in order to provide consumers with a more complete picture of an authors' working context beyond the browser. However, even in situations where Strata cannot automatically calculate a

signal, we believe that the three major facets still alert consumers that these are important aspects to be considered. Also, to the extent that consumers come up with their own measurements and ways to fulfill their information needs, they are perfectly welcome to do so, such as testing if a piece of sample code returns the desired result by running it in a terminal, which the current Strata does not automatically do.

Some of the features in Strata are currently implemented based on heuristics, such as the bounds of the automatic context snapshots and the threshold beyond which information is considered out-of-date. These heuristics are based on our preliminary piloting through limited iterations, and may not apply universally to every situation. Further development can make these features more universally applicable and more adaptive to different situations so that users will be able to rely more on the judgements that Strata automatically generates.

The current design of Strata is intended for use cases where people collaborate and communicate their knowledge artifacts with each other in good faith; for example, software engineers sharing design rationale within a team. However, for Strata to be used at scale with potentially malicious actors, such as in situations where some authors might try to increase the trustworthiness and thoroughness scores by manipulating the different metrics that it uses and displays, additional signals as well as mitigation techniques might be needed to combat such gaming behaviors. One approach would be to aggregate multiple knowledge artifacts with similar context (options, criteria, and goals in the case of Unakite comparison tables) together and detect and filter out anomalous components, inspired by mechanisms like "down-voting" that community Q&A sites (e.g., Stack Overflow) use to guard against incorrect and malicious answers at scale. Further, some of the information, like the context, seems difficult and pointless to distort.

One of the concerns that repeated during our iterative design process is that each surfaced appropriateness property ultimately competes for user attention and takes time for the reader to process [65], which could result in the overall user interface being overwhelming. The current solution we employed, inspired by prior work in recursive summarization and sensemaking [129, 130], takes a hierarchical approach that presents users with an overview and the ability to dive into specific details, letting them take the initiative of exploring parts relevant to their own interests. Future research is needed to untangle the relative importance of the various factors and how they can be alternatively represented. One idea is to gather large amounts of usage data from a field deployment and develop statistical or machine learning-based models that can predict importance metrics given various input parameters.

Finally, our lab evaluation contains several limitations. Given the short amount of training time participants had, some may not have been able to get fully acquainted with the various features that Strata offers. The tasks used in the study may not be what participants encounter in their daily work, and participants may not have the necessary context and sufficient agency as they do in real life. We mitigated these risks by asking participants to complete a practice task simulating what they would need to do in the study to help them get familiarized with Strata as well as the flow and cadence of the tasks. To improve realism, all three tasks used in the study were based on actual questions asked by real developers online, and the tables used in the study were adapted by the first author from real comparison tables we found online. For each task, we also provided participants with some background information and context to get them prepared. In the future, we would like to further address these limitations by conducting a long-term larger-scale field study, where developers will have both sufficient familiarity with Strata through repeated usage and motivation to reuse decisions that are relevant to their own work.

10 CONCLUSION

Appropriate reuse of previously created knowledge requires judging its relevance, trustworthiness, and thoroughness in relation to an individual's goals and context. In this work, we synthesized a framework for such reuse judgements in the domain of programming through analysis of prior research on sensemaking and trust as well as new needs-finding interviews with developers. In addition, we developed a prototype system called Strata that automatically captures and visualizes some of the signals described in the framework that would facilitate subsequent knowledge consumers' reuse decisions, which proved to be effective and useful in a user study.

Unakite and Strata together point to the importance of having tool support that helps people more efficiently organize and manage information as they find it in a way that could also be beneficial to others, and therefore bootstrapping the virtuous cycle of people being able to build on each other's sensemaking results, fostering efficient collaboration and knowledge reuse.

ACKNOWLEDGMENTS

This research was supported in part by NSF grants CCF-1814826 and FW-HTF-RL-1928631, Google, Bosch, the Office of Naval Research, and the CMU Center for Knowledge Acceleration. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. We would like to thank our study participants for their kind participation and our anonymous reviewers for their insightful feedback. We are grateful to Amber Horvath, Toby Jia-Jun Li, Haojian Jin, Joseph Chee Chang, Nathan Hahn, Zheng Yao, Yiyi Wang, Tianying Chen, Haitian Sun, Jiachen Wang, and Jinlei Chen for their valuable feedback and constant support, especially during the COVID-19 pandemic.

REFERENCES

[1] [n.d.]. Build software better, together - Github. https://github.com
[2] [n.d.]. Getting started with machine learning. https://github.com/collections/machine-learning
[3] [n.d.]. npm | build amazing things. https://www.npmjs.com/ Library Catalog: www.npmjs.com.
[4] [n.d.]. Programming languages: A list of programming languages that are actively developed on GitHub. https://github.com/collections/programming-languages
[5] [n.d.]. Stack Overflow - Where Developers Learn, Share, & Build Careers. https://stackoverflow.com/
[6] [n.d.]. Wikipedia. https://www.wikipedia.org/
[7] 2009. PUT vs. POST in REST. https://stackoverflow.com/a/32524385
[8] 2009. Which equals operator (== vs ===) should be used in JavaScript comparisons? https://stackoverflow.com/a/26923895
[9] 2019. Front-end JavaScript frameworks. https://github.com/collections/front-end-javascript-frameworks
[10] 2020. "exports" config · Issue #20 · then/is-promise. https://github.com/then/is-promise/issues/20 Library Catalog: github.com.
[11] 2020. pip - The Python Package Installer — pip 20.1 documentation. https://pip.pypa.io/en/stable/
[12] 2020. Python Lists VS Numpy Arrays. https://www.geeksforgeeks.org/python-lists-vs-numpy-arrays/ Library Catalog: www.geeksforgeeks.org Section: Python.
[13] 2020. Stack Overflow Developer Survey 2020. https://insights.stackoverflow.com/survey/2020/
[14] B. Thomas Adler and Luca de Alfaro. 2007. A content-driven reputation system for the wikipedia. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*. Association for Computing Machinery, Banff, Alberta, Canada, 261–270. https://doi.org/10.1145/1242572.1242608
[15] Janet E. Alexander and Marsha A. Tate. 1999. *Web Wisdom; How to Evaluate and Create Information Quality on the Webb* (1st ed.). L. Erlbaum Associates Inc., USA.
[16] Rana Alkadhi, Teodora Lata, Emitza Guzmany, and Bernd Bruegge. 2017. Rationale in Development Chat Messages: An Exploratory Study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 436–446. https://doi.org/10.1109/MSR.2017.43
[17] Saleema Amershi and Meredith Ringel Morris. 2008. CoSearch: A System for Co-located Collaborative Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1647–1656. https://doi.org/10.1145/1357054.1357311

[18] Jonathan Howard Amsbary and Larry Powell. 2003. Factors influencing evaluations of web site information. *Psychological Reports* 93, 1 (Aug. 2003), 191–198. https://doi.org/10.2466/pr0.2003.93.1.191

[19] B.H. Barns and T.B. Bollinger. 1991. Making reuse cost-effective. *IEEE Software* 8, 1 (Jan. 1991), 13–24. https://doi.org/10.1109/52.62928 Conference Name: IEEE Software.

[20] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2007. An engineering design knowledge reuse methodology using process modelling. *Research in Engineering Design* 18, 1 (May 2007), 37–48. https://doi.org/10.1007/s00163-007-0028-8

[21] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2008. A framework to integrate design knowledge reuse and requirements management in engineering design. *Robotics and Computer-Integrated Manufacturing* 24, 4 (Aug. 2008), 585–593. https://doi.org/10.1016/j.rcim.2007.07.010

[22] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the Fourth international Workshop on Computing Education Research (ICER '08)*. Association for Computing Machinery, Sydney, Australia, 3–14. https://doi.org/10.1145/1404520.1404522

[23] Krishna Bharat. 2000. SearchPad: explicit capture of search context to support Web search. *Computer Networks* 33, 1 (June 2000), 493–501. https://doi.org/10.1016/S1389-1286(00)00047-5

[24] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 73–80. https://doi.org/10.1145/166117.166126

[25] Pia Borlund. 2003. The concept of relevance in IR. *Journal of the American Society for Information Science and Technology* 54, 10 (2003), 913–925. https://doi.org/10.1002/asi.10286 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.10286.

[26] D. Scott Brandt. 1996. Evaluating Information on the Internet. *Computers in Libraries* 16, 5 (1996), 44–46.

[27] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric Programming: Integrating Web Search into the Development Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 513–522. https://doi.org/10.1145/1753326.1753402

[28] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1589–1598. https://doi.org/10.1145/1518701.1518944 event-place: Boston, MA, USA.

[29] Joseph Chee Chang, Nathan Hahn, and Aniket Kittur. 2020. Mesh: Scaffolding Comparison Tables for Online Decision Making. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 391–405. https://doi.org/10.1145/3379337.3415865

[30] Joseph Chee Chang, Nathan Hahn, Adam Perer, and Aniket Kittur. 2019. SearchLens: composing and capturing complex user interests for exploratory search. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI '19)*. Association for Computing Machinery, Marina del Ray, California, 498–509. https://doi.org/10.1145/3301275.3302321

[31] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. SAGE. Google-Books-ID: 2ThdBAAAQBAJ.

[32] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. 2017. Codeon: On-Demand Software Development Assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6220–6231. https://doi.org/10.1145/3025453.3025972

[33] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In *Perspectives on socially shared cognition*. American Psychological Association, Washington, DC, US, 127–149. https://doi.org/10.1037/10096-006

[34] Thomas H. Davenport, Sirkka L. Jarvenpaa, and Michael C. Beers. 1996. Improving Knowledge Work Processes. *Sloan management review* 37, 4 (1996), 53–65. https://dialnet.unirioja.es/servlet/articulo?codigo=2514140 Publisher: MIT press Section: Sloan management review.

[35] Peter Denning, Jim Horning, David Parnas, and Lauren Weinstein. 2005. Wikipedia risks. *Commun. ACM* 48, 12 (Dec. 2005), 152. https://doi.org/10.1145/1101779.1101804

[36] Nancy M. Dixon. 2000. *Common Knowledge: How Companies Thrive by Sharing What They Know*. Harvard Business School Press, USA.

[37] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work (CSCW '92)*. Association for Computing Machinery, Toronto, Ontario, Canada, 107–114. https://doi.org/10.1145/143457.143468

[38] Thomas Erickson and Wendy A. Kellogg. 2000. Social translucence: an approach to designing systems that support social processes. *ACM Transactions on Computer-Human Interaction* 7, 1 (March 2000), 59–83. https://doi.org/10.1145/344949.345004

[39] Gunther Eysenbach and Christian Köhler. 2002. How do consumers search for and appraise health information on the world wide web? Qualitative study using focus groups, usability tests, and in-depth interviews. *BMJ (Clinical research ed.)* 324, 7337 (March 2002), 573–577. https://doi.org/10.1136/bmj.324.7337.573

[40] Facebook. 2018. React - A JavaScript library for building user interfaces. https://reactjs.org/

[41] Kristie Fisher, Scott Counts, and Aniket Kittur. 2012. Distributed Sensemaking: Improving Sensemaking by Leveraging the Efforts of Previous Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 247–256. https://doi.org/10.1145/2207676.2207711

[42] Andrew J. Flanagin and Miriam J. Metzger. 2000. Perceptions of Internet Information Credibility. *Journalism & Mass Communication Quarterly* 77, 3 (Sept. 2000), 515–540. https://doi.org/10.1177/107769900007700304 Publisher: SAGE Publications Inc.

[43] B. J. Fogg. 2002. Persuasive technology: using computers to change what we think and do. *Ubiquity* 2002, December (Dec. 2002), 5:2. https://doi.org/10.1145/764008.763957

[44] B. J. Fogg and Hsiang Tseng. 1999. The elements of computer credibility. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, 80–87. https://doi.org/10.1145/302979.303001

[45] Adam Fourney and Meredith Ringel Morris. 2013. Enhancing Technical Q&A Forums with CiteHistory. In *Seventh International AAAI Conference on Weblogs and Social Media*. https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6082

[46] William Frakes and Carol Terry. 1996. Software reuse: metrics and models. *Comput. Surveys* 28, 2 (June 1996), 415–435. https://doi.org/10.1145/234528.234531

[47] W B Frakes and B A Nejmeh. 1986. Software reuse through information retrieval. *ACM SIGIR Forum* 21, 1-2 (Sept. 1986), 30–36. https://doi.org/10.1145/24634.24636

[48] John W. Fritch and Robert L. Cromwell. 2001. Evaluating Internet resources: Identity, affiliation, and cognitive authority in a networked world. *Journal of the American Society for Information Science and Technology* 52, 6 (2001), 499–507. https://doi.org/10.1002/asi.1081 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.1081.

[49] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. 2012. Comparative Evaluation of Javascript Frameworks. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12 Companion)*. ACM, New York, NY, USA, 513–514. https://doi.org/10.1145/2187980.2188103

[50] Stefan Haefliger, Georg von Krogh, and Sebastian Spaeth. 2007. Code Reuse in Open Source Software. *Management Science* 54, 1 (Nov. 2007), 180–193. https://doi.org/10.1287/mnsc.1070.0748 Publisher: INFORMS.

[51] Nathan Hahn, Joseph Chang, Ji Eun Kim, and Aniket Kittur. 2016. The Knowledge Accelerator: Big Picture Thinking in Small Pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2258–2270. https://doi.org/10.1145/2858036.2858364

[52] Nathan Hahn, Joseph Chee Chang, and Aniket Kittur. 2018. Bento Browser: Complex Mobile Search Without Tabs. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Montreal QC, Canada, 251:1–251:12. https://doi.org/10.1145/3173574.3173825

[53] Udo Hahn and Ulrich Reimer. 1999. Knowledge-based text summarization: Salience and generalization operators for knowledge base abstraction. *Advances in automatic text summarization* (1999), 215–232. Publisher: MIT Press, Cambridge, Mass.

[54] Tom-Michael Hesse, Veronika Lerche, Marcus Seiler, Konstantin Knoess, and Barbara Paech. 2016. Documented decision-making strategies and decision knowledge in open source projects: An empirical study on Firefox issue reports. *Information and Software Technology* 79 (Nov. 2016), 36–51. https://doi.org/10.1016/j.infsof.2016.06.003

[55] Johan F. Hoorn and Teunis D. van Wijngaarden. 2010. Web Intelligence for the Assessment of Information Quality: Credibility, Correctness, and Readability. *Web Intelligence and Intelligent Agents* (March 2010). https://doi.org/10.5772/8372 Publisher: IntechOpen.

[56] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030

[57] Jane Hsieh, Michael Xieyang Liu, Brad A. Myers, and Aniket Kittur. 2018. An Exploratory Study of Web Foraging to Understand and Support Programming Decisions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 305–306. https://doi.org/10.1109/VLHCC.2018.8506517 ISSN: 1943-6092.

[58] Donghan Hu and Sang Won Lee. 2020. ScreenTrack: Using a Visual History of a Computer Screen to Retrieve Documents and Web Pages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, Honolulu, HI, USA, 1–13. https://doi.org/10.1145/3313831.3376753

[59] Robert F. Hurley and G. Tomas M. Hult. 1998. Innovation, Market Orientation, and Organizational Learning: An Integration and Empirical Examination. *Journal of Marketing* 62, 3 (July 1998), 42–54. https://doi.org/10.1177/002224299806200303 Publisher: SAGE Publications Inc.

[60] Haojian Jin, Swarun Kumar, and Jason Hong. 2020. Providing architectural support for building privacy-sensitive smart home applications. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers (UbiComp-ISWC '20)*. Association for Computing Machinery, New York, NY, USA, 212–217. https://doi.org/10.1145/3410530.3414328

[61] Haojian Jin, Minyi Liu, Kevan Dodhia, Yuanchun Li, Gaurav Srivastava, Matthew Fredrikson, Yuvraj Agarwal, and Jason I. Hong. 2018. Why Are They Collecting My Data? Inferring the Purposes of Network Traffic in Mobile Apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (Dec. 2018), 173:1–173:27. https://doi.org/10.1145/3287051

[62] Haojian Jin, Tetsuya Sakai, and Koji Yatani. 2014. ReviewCollage: a mobile interface for direct comparison using online reviews. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (MobileHCI '14)*. Association for Computing Machinery, New York, NY, USA, 349–358. https://doi.org/10.1145/2628363.2628373

[63] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, and Michael Bove. 2014. Standing on the Schemas of Giants: Socially Augmented Information Foraging. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 999–1010. https://doi.org/10.1145/2531602.2531644

[64] Aniket Kittur, Andrew M. Peters, Abdigani Diriye, Trupti Telang, and Michael R. Bove. 2013. Costs and Benefits of Structured Information Foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2989–2998. https://doi.org/10.1145/2470654.2481415

[65] Aniket Kittur, Bongwon Suh, and Ed H. Chi. 2008. Can you ever trust a wiki? impacting perceived trustworthiness in wikipedia. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work (CSCW '08)*. Association for Computing Machinery, San Diego, CA, USA, 477–480. https://doi.org/10.1145/1460563.1460639

[66] Amy J. Ko, Robert DeLine, and Gina Venolia. 2007. Information Needs in Collocated Software Development Teams. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 344–353.

[67] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC '04)*. IEEE Computer Society, Washington, DC, USA, 199–206. https://doi.org/10.1109/VLHCC.2004.47

[68] Professor of Management and Director at the Institute of Management Georg Von Krogh, Georg von Krogh, Associate Professor in the Faculty of Social Sciences and the Graduate School of International Corporate Strategy Kazuo Ichijo, Kazuo Ichijo, Ikujiro Nonaka, and Professor of Graduate School of International Corporate Strategy at Hitotsubashi University and the Xerox Distinguished Professor in Knowledge at Hass School of Business Ikujiro Nonaka. 2000. *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. Oxford University Press, USA. Google-Books-ID: JVESDAAAQBAJ.

[69] Charles W. Krueger. 1992. Software reuse. *Comput. Surveys* 24, 2 (June 1992), 131–183. https://doi.org/10.1145/130844.130856

[70] Thomas D. LaToza and Brad A. Myers. 2010. Hard-to-answer Questions About Code. In *Evaluation and Usability of Programming Languages and Tools (PLATEAU '10)*. ACM, New York, NY, USA, 8:1–8:6. https://doi.org/10.1145/1937117.1937125

[71] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining Mental Models: A Study of Developer Work Habits. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 492–501. https://doi.org/10.1145/1134285.1134355

[72] John Lawrence, Jonas Malmsten, Andrey Rybka, Daniel Sabol, and Ken Triplin. 2017. Comparing TensorFlow Deep Learning Performance Using CPUs, GPUs, Local PCs and Cloud. *Publications and Research* (May 2017). https://academicworks.cuny.edu/bx_pubs/50

[73] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, Denver, Colorado, USA, 6038–6049. https://doi.org/10.1145/3025453.3025483

[74] Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, and Brad A. Myers. 2020. Privacy-Preserving Script Sharing in GUI-based Programming-by-Demonstration Systems. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (May 2020), 060:1–060:23. https://doi.org/10.1145/3392869

[75] Toby Jia-Jun Li, Jingya Chen, Haijun Xia, Tom M. Mitchell, and Brad A. Myers. 2020. Multi-Modal Repairs of Conversational Breakdowns in Task-Oriented Dialogs. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 1094–1107. https://doi.org/10.1145/3379337.3415820

[76] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Wanling Ding, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Natural Language Instructions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 105–114. https://doi.org/10.1109/VLHCC.2018.8506506 ISSN: 1943-6106.

[77] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New Orleans, LA, USA, 577–589. https://doi.org/10.1145/3332165.3347899

[78] Toby Jia-Jun Li and Oriana Riva. 2018. Kite: Building Conversational Bots from Mobile Apps. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. Association for Computing Machinery, Munich, Germany, 96–109. https://doi.org/10.1145/3210240.3210339

[79] Brian Y. Lim and Anind K. Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing (UbiComp '09)*. Association for Computing Machinery, Orlando, Florida, USA, 195–204. https://doi.org/10.1145/1620545.1620576

[80] Brian Y. Lim and Anind K. Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing (UbiComp '10)*. Association for Computing Machinery, Copenhagen, Denmark, 13–22. https://doi.org/10.1145/1864349.1864353

[81] Michael Xieyang Liu, Shaun Burley, Emily Deng, Angelina Zhou, Aniket Kittur, and Brad A. Myers. 2018. Supporting Knowledge Acceleration for Programming from a Sensemaking Perspective. *Sensemaking Workshop at CHI Conference on Human Factors in Computing Systems* (April 2018). https://par.nsf.gov/biblio/10152063-supporting-knowledge-acceleration-programming-from-sensemaking-perspective

[82] Michael Xieyang Liu, Nathan Hahn, Angelina Zhou, Shaun Burley, Emily Deng, Aniket Kittur, and Brad A. Myers. 2018. UNAKITE: Support Developers for Capturing and Persisting Design Rationales When Solving Problems Using Web Resources. *Workshop on Designing Technologies to Support Human Problem Solving at the IEEE Symposium on Visual Languages and Human-Centric Computing* (Oct. 2018). https://par.nsf.gov/biblio/10152060-unakite-support-developers-capturing-persisting-design-rationales-when-solving-problems-using-web-resources

[83] Michael Xieyang Liu, Jane Hsieh, Nathan Hahn, Angelina Zhou, Emily Deng, Shaun Burley, Cynthia Taylor, Aniket Kittur, and Brad A. Myers. 2019. Unakite: Scaffolding Developers' Decision-Making Using the Web. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New Orleans, LA, USA, 67–80. https://doi.org/10.1145/3332165.3347908 event-place: New Orleans, LA, USA.

[84] Ann Majchrzak, Lynne P. Cooper, and Olivia E. Neece. 2004. Knowledge Reuse for Innovation. *Management Science* 50, 2 (Feb. 2004), 174–188. https://doi.org/10.1287/mnsc.1030.0116 Publisher: INFORMS.

[85] Lynne M. Markus. 2001. Toward a Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success. *Journal of Management Information Systems* 18, 1 (May 2001), 57–93. https://doi.org/10.1080/07421222.2001.11045671 Publisher: Routledge _eprint: https://doi.org/10.1080/07421222.2001.11045671.

[86] Marc Meola. 2004. Chucking the Checklist: A Contextual Approach to Teaching Undergraduates Web-Site Evaluation. *portal: Libraries and the Academy* 4, 3 (July 2004), 331–344. https://doi.org/10.1353/pla.2004.0055 Publisher: Johns Hopkins University Press.

[87] Miriam J. Metzger. 2007. Making sense of credibility on the Web: Models for evaluating online information and recommendations for future research. *Journal of the American Society for Information Science and Technology* 58, 13 (2007), 2078–2091. https://doi.org/10.1002/asi.20672

[88] Miriam J. Metzger, Andrew J. Flanagin, Keren Eyal, Daisy R. Lemus, and Robert M. Mccann. 2003. Credibility for the 21st Century: Integrating Perspectives on Source, Message, and Media Credibility in the Contemporary Media Environment. *Annals of the International Communication Association* 27, 1 (Jan. 2003), 293–335. https://doi.org/10.1080/23808985.2003.11679029 Publisher: Routledge _eprint: https://doi.org/10.1080/23808985.2003.11679029.

[89] Miriam J. Metzger, Andrew J. Flanagin, and Ryan B. Medders. 2010. Social and Heuristic Approaches to Credibility Evaluation Online. *Journal of Communication* 60, 3 (2010), 413–439. https://doi.org/10.1111/j.1460-2466.2010.01488.x

[90] Audris Mockus. 2007. Large-Scale Code Reuse in Open Source Software. In *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*. 7–7. https://doi.org/10.1109/FLOSS.2007.10

[91] Dan Morris, Meredith Ringel Morris, and Gina Venolia. 2008. SearchBar: A Search-centric Web History for Task Resumption and Information Re-finding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1207–1216. https://doi.org/10.1145/1357054.1357242

[92] Meredith Ringel Morris and Eric Horvitz. 2007. SearchTogether: An Interface for Collaborative Web Search. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/1294211.1294215

[93] B. Myers, R. Malkin, M. Bett, A. Waibel, B. Bostwick, R.C. Miller, Jie Yang, M. Denecke, E. Seemann, Jie Zhu, Choon Hong Peck, D. Kong, J. Nichols, and B. Scherlis. 2002. Flexi-modal and multi-machine user interfaces. In *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. 343–348. https://doi.org/10.1109/ICMI.2002.1167019

[94] Brad A. Myers, Amy J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. Making End User Development More Natural. In *New Perspectives in End-User Development*, Fabio Paternò

and Volker Wulf (Eds.). Springer International Publishing, Cham, 1–22. https://doi.org/10.1007/978-3-319-60291-2_1

[95] Ikujiro Nonaka, Hirotaka Takeuchi, and Katsuhiro Umemoto. 1996. A theory of organizational knowledge creation. *International Journal of Technology Management* 11, 7-8 (Jan. 1996), 833–845. https://doi.org/10.1504/IJTM.1996.025472 Publisher: Inderscience Publishers.

[96] Carla O'Dell and C. Jackson Grayson. 1998. If Only We Knew What We Know: Identification and Transfer of Internal Best Practices:. *California Management Review* (April 1998). https://doi.org/10.2307/41165948 Publisher: SAGE PublicationsSage CA: Los Angeles, CA.

[97] Stephen Oney and Joel Brandt. 2012. Codelets: Linking Interactive Documentation and Example Code in the Editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2697–2706. https://doi.org/10.1145/2207676.2208664

[98] Margit Osterloh and Bruno S. Frey. 2000. Motivation, Knowledge Transfer, and Organizational Forms. *Organization Science* 11, 5 (Oct. 2000), 538–550. https://doi.org/10.1287/orsc.11.5.538.15204 Publisher: INFORMS.

[99] Emily S. Patterson and David D. Woods. 2001. Shift Changes, Updates, and the On-Call Architecture in Space Shuttle Mission Control. *Computer Supported Cooperative Work* 10, 3-4 (Dec. 2001), 317–346. https://doi.org/10.1023/A:1012705926828

[100] Sharoda A. Paul and Meredith Ringel Morris. 2009. CoSense: Enhancing Sensemaking for Collaborative Web Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1771–1780. https://doi.org/10.1145/1518701.1518974

[101] Sharoda A. Paul and Meredith Ringel Morris. 2011. Sensemaking in Collaborative Web Search. *Human–Computer Interaction* 26, 1-2 (March 2011), 72–122. https://doi.org/10.1080/07370024.2011.559410

[102] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, San Francisco, CA, USA, 1295–1298. https://doi.org/10.1109/ICSE.2013.6606701

[103] N. Rutar, C. B. Almazan, and J. S. Foster. 2004. A comparison of bug finding tools for Java. In *15th International Symposium on Software Reliability Engineering*. 245–256. https://doi.org/10.1109/ISSRE.2004.1

[104] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web (WWW '10)*. Association for Computing Machinery, Raleigh, North Carolina, USA, 841–850. https://doi.org/10.1145/1772690.1772776

[105] Tefko Saracevic. [n.d.]. Relevance reconsidered.

[106] Ann Scholz-Crane. 1998. Evaluating the Future: A Preliminary Study of the Process of How Undergraduate Students Evaluate Web Sources. *RSR: Reference Services Review* 26 (1998), 53–60.

[107] Rever Score. 2017. Why we moved from Angular 2 to Vue.js (and why we didn't choose React). https://medium.com/reverdev/why-we-moved-from-angular-2-to-vue-js-and-why-we-didnt-choose-react-ef807d9f4163 Library Catalog: medium.com.

[108] Mirjam Seckler, Silvia Heinz, Seamus Forde, Alexandre N. Tuch, and Klaus Opwis. 2015. Trust and distrust on the web: User experiences and website characteristics. *Computers in Human Behavior* 45 (April 2015), 39–50. https://doi.org/10.1016/j.chb.2014.11.064

[109] Nikhil Sharma. 2008. Sensemaking handoff: When and how? *Proceedings of the American Society for Information Science and Technology* 45, 1 (Jan. 2008), 1–12. https://doi.org/10.1002/meet.2008.1450450234

[110] Nikhil Sharma. 2011. Role of available and provided resources in sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, Vancouver, BC, Canada, 1807–1816. https://doi.org/10.1145/1978942.1979204

[111] Nikhil Sharma and George Furnas. 2009. Artifact usefulness and usage in sensemaking handoffs. *Proceedings of the American Society for Information Science and Technology* 46 (2009). https://doi.org/10.1002/meet.2009.1450460219

[112] B. Shneiderman. 1996. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*. 336–343. https://doi.org/10.1109/VL.1996.545307 ISSN: 1049-2615.

[113] Ben Shneiderman. 2000. Designing trust into online experiences. *Commun. ACM* 43, 12 (Dec. 2000), 57–59. https://doi.org/10.1145/355112.355124

[114] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2006. Questions Programmers Ask During Software Evolution Tasks. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '06/FSE-14)*. ACM, New York, NY, USA, 23–34. https://doi.org/10.1145/1181775.1181779

[115] Manuel Sojer and Joachim Henkel. 2010. *Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments*. SSRN Scholarly Paper ID 1489789. Social Science Research Network, Rochester, NY. https://papers.ssrn.com/abstract=1489789

[116] Bongwon Suh, Ed H. Chi, Aniket Kittur, and Bryan A. Pendleton. 2008. Lifting the veil: improving accountability and social transparency in Wikipedia with wikidashboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. Association for Computing Machinery, Florence, Italy, 1037–1040. https:

//doi.org/10.1145/1357054.1357214

[117] Yi Yi Thaw, Ahmad Kamil Mahmood, and P. Dhanapal Durai Dominic. 2009. A Study on the Factors That Influence the Consumers Trust on Ecommerce Adoption. *arXiv:0909.1145 [cs]* (Sept. 2009). http://arxiv.org/abs/0909.1145 arXiv: 0909.1145.

[118] Meinald T. Thielsch and Gerrit Hirschfeld. 2019. Facets of Website Content. *Human–Computer Interaction* 34, 4 (July 2019), 279–327. https://doi.org/10.1080/07370024.2017.1421954

[119] Michael L Van De Vanter. 2002. The documentary structure of source code. *Information and Software Technology* 44, 13 (Oct. 2002), 767–782.

[120] Laton Vermette, Parmit Chilana, Michael Terry, Adam Fourney, Ben Lafreniere, and Travis Kerr. 2015. CheatSheet: A Contextual Interactive Memory Aid for Web Applications. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 241–248. http://dl.acm.org/citation.cfm?id=2788890.2788933 event-place: Halifax, Nova Scotia, Canada.

[121] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences* 113, 3 (Jan. 2016), 554–559. https://doi.org/10.1073/pnas.1517441113 Publisher: National Academy of Sciences Section: Physical Sciences.

[122] Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. 2004. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. Association for Computing Machinery, Vienna, Austria, 575–582. https://doi.org/10.1145/985692.985765

[123] Fernanda B. Viégas, Martin Wattenberg, and Matthew M. McKeon. 2007. The Hidden Order of Wikipedia. In *Online Communities and Social Computing (Lecture Notes in Computer Science)*, Douglas Schuler (Ed.). Springer, Berlin, Heidelberg, 445–454. https://doi.org/10.1007/978-3-540-73257-0_49

[124] Ye Diana Wang and Henry H. Emurian. 2005. An overview of online trust: Concepts, elements, and implications. *Computers in Human Behavior* 21, 1 (Jan. 2005), 105–125. https://doi.org/10.1016/j.chb.2003.11.008

[125] Sharon Watson and Kelly Hewett. 2006. A Multi-Theoretical Model of Knowledge Transfer in Organizations: Determinants of Knowledge Contribution and Knowledge Reuse*. *Journal of Management Studies* 43, 2 (2006), 141–173. https://doi.org/10.1111/j.1467-6486.2006.00586.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-6486.2006.00586.x.

[126] Zhen Yue, Shuguang Han, and Daqing He. 2012. An investigation of search processes in collaborative exploratory web search. *Proceedings of the American Society for Information Science and Technology* 49, 1 (2012), 1–4. https://doi.org/10.1002/meet.14504901386

[127] Honglei Zeng, Maher A. Alhossaini, Li Ding, Richard Fikes, and Deborah L. McGuinness. 2006. Computing trust from revision history. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (PST '06)*. Association for Computing Machinery, Markham, Ontario, Canada, 1. https://doi.org/10.1145/1501434.1501445

[128] Honglei Zeng, Maher A. Alhossaini, Richard Fikes, and Deborah L. McGuinness. 2006. Mining Revision History to Assess Trustworthiness of Article Fragments. In *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. 1–10. https://doi.org/10.1109/COLCOM.2006.361890

[129] Amy X. Zhang and Justin Cranshaw. 2018. Making Sense of Group Chat Through Collaborative Tagging and Summarization. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW (Nov. 2018), 196:1–196:27. https://doi.org/10.1145/3274465

[130] Amy X. Zhang, Lea Verou, and David Karger. 2017. Wikum: Bridging Discussion Forums and Wikis Using Recursive Summarization. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 2082–2096. https://doi.org/10.1145/2998181.2998235

[131] Jian Zhao, Michael Glueck, Petra Isenberg, Fanny Chevalier, and Azam Khan. 2018. Supporting Handoff in Asynchronous Collaborative Sensemaking Using Knowledge-Transfer Graphs. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 340–350. https://doi.org/10.1109/TVCG.2017.2745279 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

Received June 2020; revised October 2020; accepted December 2020