Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools

Brittany Johnson
George Mason University
Virginia, USA
johnsonb@gmu.edu

Christian Bird
Microsoft Research
Washington, USA
cbird@microsoft.com

Denae Ford
Microsoft Research
Washington, USA
denae@microsoft.com

Nicole Forsgren
Microsoft Research
Washington, USA
niforsgr@microsoft.com

Thomas Zimmermann
Microsoft Research
Washington, USA
tzimmer@microsoft.com

***Abstract*—The day to day of a software engineer involves a variety of tasks. While many of these tasks are collaborative and completed as such, it is not always possible or feasible to engage with other engineers for task completion. Software tools, such as code generators and static analysis tools, aim to fill this gap by providing additional support for developers to effectively complete their tasks. With a steady stream of new tools that emerging to support software engineers, including a new breed of tools that rely on artificial intelligence, there are important questions we should aim to answer regarding the trust engineers can, and should, put into their software tools and what it means to build a trustworthy tool. In this paper, we present findings from an industry interview study conducted with 18 engineers across and external to the Microsoft organization. Based on these interviews, we introduce the PICSE (pronounced "pixie") framework for trust in software tools to provide preliminary insights into factors that influence engineer trust in their software tools. We also discuss how the PICSE framework can be considered and applied in practice for designing and developing trustworthy software tools.**

***Index Terms*—trust, software tools, artificial intelligence, framework**

I. Introduction

Engineers rely on tools to support the completion of their day to day tasks, as evidenced by the rapid and consistent increase in available tooling. In fact, the software engineering research community has long encouraged and celebrated new techniques that can help engineers solve new problems or old problems better than before (hence the emergence of tracks such as New Ideas and Tool Demos).

Despite the enthusiasm for creating and disseminating new tools, we still struggle with building bridges between new tools and the engineers they are intended to support. Research suggests that many tools may go unnoticed and unused in practice [1], [2].

As we continue to struggle with tool adoption and use in practice, the tool landscape continues to evolve with technology. With the advent of huge amounts of data and increasingly powerful artificial intelligence (AI) models, new types of software tools are being created that rely on AI for decision-making and recommendations [3], [4]. Most notably is GitHub Copilot [5], an AI-assisted software tool that uses code models to generate code snippets and subprograms that engineers can adapt and integrate into their codebases.

There have been numerous efforts aimed at both improving the techniques and models that power software tools (both AI-assisted and traditional) and exploring what tasks they can support [6]–[9]. However, there is a dearth of understanding about how to build and deploy these tools such that they will be adopted and then effectively used beyond adoption. We know from prior work that developers only use tool that they trust [10], however, we know much less about how trust is formed and what factors effect its evolution over time in the context of software tools.

To help fill this gap, we conducted a qualitative investigation to better understand the key components of trust formation and evolution when adopting and using software tools. We interviewed 18 engineers engineers across and external to the Microsoft organization to answer the research question *"What factors influence engineers' trust in software tools?"*. Our findings identified important factors, along with concrete examples, and serve as guides for those seeking to foster trust around their tools.

Based on these findings, we introduce the **PICSE** framework which organizes factors into five high-level categories: ***P**ersonal*, ***I**nteraction*, ***C**ontrol*, ***S**ystem*, and ***E**xpectations*. While many of the factors in the PICSE framework hold for the majority of tools, we do find some differences in trust dynamics between AI-powered software tools and so-called "traditional" software tools. Finally, as AI-assisted tools have often been compared to artificial team members (*e.g.,* Copilot, is so named in an effort to anthropomorphize the AI into "your pair programmer") we compare and contrast engineer-engineer trust dynamics with engineer-tool dynamics.

The main contributions of this paper are as follows:

- We contribute a conceptual framework, called the PICSE framework, that outlines factors that impact the formation and evolution of trust in software tools (Section III).
- We outline guidance on considering and applying the PICSE framework in practice to increase tool trustworthiness (Section V and Section VI).

II. Methodology

A. Research Question

We designed our study to answer the question: *What factors influence engineers' trust in software tools?* For the purpose

TABLE I
PARTICIPANT DEMOGRAPHICS

	Role	Gender	Race
P1	Software Engineer II	Male	White
P2	Software Engineer	Male	Asian
P3	Software Engineer	Male	Asian
P4	Senior Software Engineer	Male	Asian
P5	Principal Software Engineer	Male	White
P6	Principal SE Lead	Male	Asian
P7	Software Dev Engineer Lead	Male	White
P8	Software Engineer II	Male	White
P9	Senior Software Engineer	Woman	White
P10	Systems Engineer	Male	Black
P11	Software Engineer II	Woman	White
P12	Software Engineer	Woman	White
P13	Software Engineer/Product Owner	Male	White
P14	Senior Security Engineer	Male	Asian
P15	Software Engineer	Woman	Asian
P16	Software Engineer	Woman	Asian
P17	Software Engineer	Woman	White
P18	Software Engineer II	Woman	Black

of this study, we define a software tool as any technology that supports software engineering. For example, this can include an integrated development environment (IDE), an issue tracker, or even Notepad.

B. Interviewees

We recruited software engineers internal and external to Microsoft. To recruit internal engineers, we reached out to individual engineers using Teams. We found engineers to contact on tool distribution lists and in relevant Teams groups. We also got help recruiting interviewees from other engineers who advertised our study in their respective circles. To recruit external engineers, we posted advertisements for our study on Twitter and LinkedIn.

Each potential interviewee was required to provide consent and sign up for an available interview slot via a pre-interview survey. We also used this survey to collect basic demographic information from interviewees. Our recruitment efforts yielded a total of 19 interviewees. We conducted the first interview as a pilot run to test our interview script and catch any issues we may have missed. The results reported in this paper are from the 18 interviews that followed.

Our final set of interviewees is listed in Table I. Our final sample included 12 Microsoft employees and 6 non-Microsoft employees. Eleven of our interviewees identified as male and 7 who identified as female. Most of our interviewees were located in the United States, with the exception of one who was located in the United Kingdom and one located in Canada. Two interviewees self-identified as Black or African American, 7 self-identified as Asians, and the remaining self-identified as White. P12 identified as a White Woman of Hispanic or Latino descent. Most engineers in our sample are software engineers, but we also interviewed security, systems, and lead engineers. Interviewees' years of professional software development experience range from 1 year to 15 years and of the 18 interviewees included in our final data set, six consider themselves to also be open source developers.

C. Data Collection

To answer our research question, we conducted semi-structured interviews. The interview was divided into three main portions: *background*, *trust in other engineers*, *trust in software tools*. The interview guide can be found online in the supplemental materials [11].

We designed the background portion of the interview to gather insights on the day to day tasks and interactions for each interviewee. This also helped set the stage for the questions to follow on their trust in the individuals and tools they interact with. We asked questions such as "What project or team do you currently work on?" and "What are the typical activities or tasks you typically work on in your current role?".

We designed the second portion of our interview to gain insights into our interviewees' thoughts about trust when developing and maintaining software with other engineers. We first asked them to define trust when it comes to collaborators on software projects. To determine if there are nuances to trust in different contexts, we asked a series a follow-up questions on trust when coding with and learning from other engineers.

The third portion of our interview shifted the dialogue from humans to tools. We again asked interviewees to first define trust, but this time in the context of tools they use to develop and maintain software. The follow-up questions in this portion mapped explicitly to tasks developers could use tools for: writing code, generating tests, finding bugs, and fixing bugs. We concluded the third portion of the interview with an opportunity for interviewees to share any ideas they had for creating or improving software tools that aide in the completion of their day to day tasks. Following the third portion of the interview, we opened the floor for interviewees to share any additional insights or thoughts they had on trust in software tools, or specifically AI-assisted software tools.

Each interview lasted approximately 1 hour and took place on Teams. We recorded both audio and the screen for the duration of each session. Following each session, we compensated interviewees with a $50 Amazon.com gift card. Each interview video was transcribed using a transcription company.

D. Data Analysis

To analyze our data, we conducted a mixed qualitative coding approach led by the first author. Given we conducted this research with a partially distributed research team, all discussions regarding data analysis occurred virtually.

We followed a combination deductive and inductive coding approach, which means that we first created an initial codebook for coding our interviews (deductive) [12]. We derived the categories and code for the initial codebook based on the questions we asked our interviewees and the information we intended to collect from those questions. For example, the first category in our codebook was "Background" given we designed the first portion of our interview to collect background information. Codes in this category include *background-project* (what project or team the interviewee works on), *background-daytoday* (typical tasks or activities

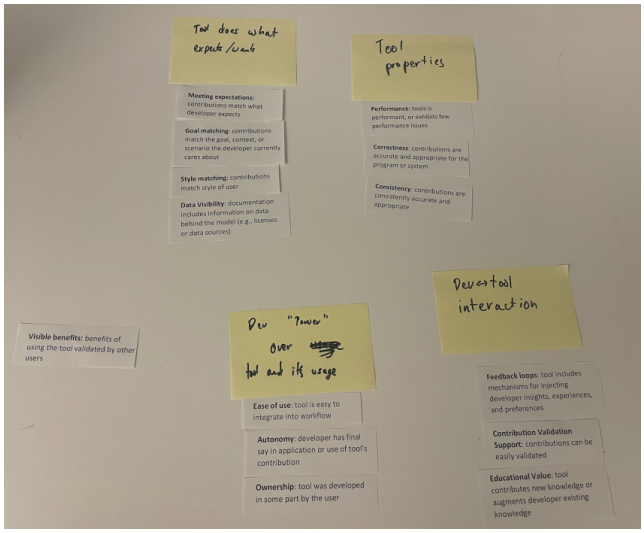

Fig. 1. Snapshot of the open card sort conducted to elicit final set of categories for the PICSE framework.

interviewees complete in their current role), and *background-tooluse* (tools interviewees use in their day to day). Our initial codebook included fourteen codes across the six categories (background, human trust, tool trust, AI tools, tool ideas, and other).

After deriving our initial codebook, we began coding our transcripts using ATLAS.ti [13] As we came across data that suggested the need for a new code (e.g., a commonly used sub- or follow-up question), we added new codes to our codebook (inductive) [12]. Once the first author finished coding all 18 interviews and refining the codebook, we conducted two validation tasks to improve the trustworthiness of our findings. The first validation task followed guidelines by Lincoln & Guba [14]. We invited an outside auditor to review our methodology and developing codebook. Our external auditor is an empirical researcher that is external to Microsoft, an expert in qualitative work, and has published qualitative work in top tier venues. They have knowledge of the space in which we are collecting data (human factors and software tools).

We provided the auditor with samples of raw, coded data along with the coding process and codebook used. We gave instructions to confirm that the initial codebook was in fact driven by the interview script, the emergent codes were properly documented, and that the inferences from the data made sense. We discussed and revised codes based on this person's feedback, but there were no disagreements in our coding and categorizing the data. Under their advice regarding our code and category descriptions, we updated code and category descriptions, as well as corresponding examples when necessary. Our final codebook, which includes 31 codes across six categories, can be found in the supplemental materials [11].

Once we determined our final set of codes, the final step in our data analysis was to conduct a thematic analysis of the resulting codes and categories [15] to determine the high level themes that eventually form the PICSE framework. This process was also led by the first author, who started by created code groups in ATLAS.ti in order to narrow data analysis to quotations that map to trust in software tools.

In the first iteration through the relevant code groups, the first author went through each of the quotations and documented the higher order themes as they emerged across quotations. After going through all the quotations that mapped to trust in tools, the first author made a second pass through the emergent themes to identify any potential overlap across themes. We iteratively discussed as a group each round of themes until we determined a final set of unique themes from our findings (the factors in our framework).

To form the high level categories of the PICSE framework, the second author conducted an open card sort on the final set of unique themes. Figure 1 depicts the initial themes and categories. The second author shared the initial categories with the research team for discussion. This discussion led to the renaming and clarification of some of the themes and categories, resulting in the final categories and themes in Figure 2.

III. Findings

Our interviews gathered insights on how engineers think about trust in the context of software development. Figure 2 summarizes the various factors that engineers may consider when determining initial trust and working to build, or re-build, trust before and during use.

Based on the interviews we conducted with software engineers, the following categories emerged to form the **PICSE** framework of trust in software tools:

- ***P**ersonal*: internal, external, and social factors that impact trust
- ***I**nteraction*: various aspects of engineer's engagement with the tool that impact trust
- ***C**ontrol*: factors that impact trust as it pertains to engineers' power and control over the tool and its usage
- ***S**ystem*: properties of the tool that impact trust
- ***E**xpectations*: meeting engineers' expectations that they have built impacts trust

Below we outline each of the categories in the PICSE framework and factors within them. It is important to note that each factor can be impacted by (or have impact on) different users and stakeholder; no one person can make sure all components are considered. We discuss ways in which practitioners can attempt to make use of the PICSE framework in Section V.

A. Personal Factors

***P**ersonal* factors in the **PICSE** framework represent the intrinsic, extrinsic, and social aspects of tool adoption and use that impact trust. Based on our findings, this includes **community**, **source reputation**, and **clear advantages**.

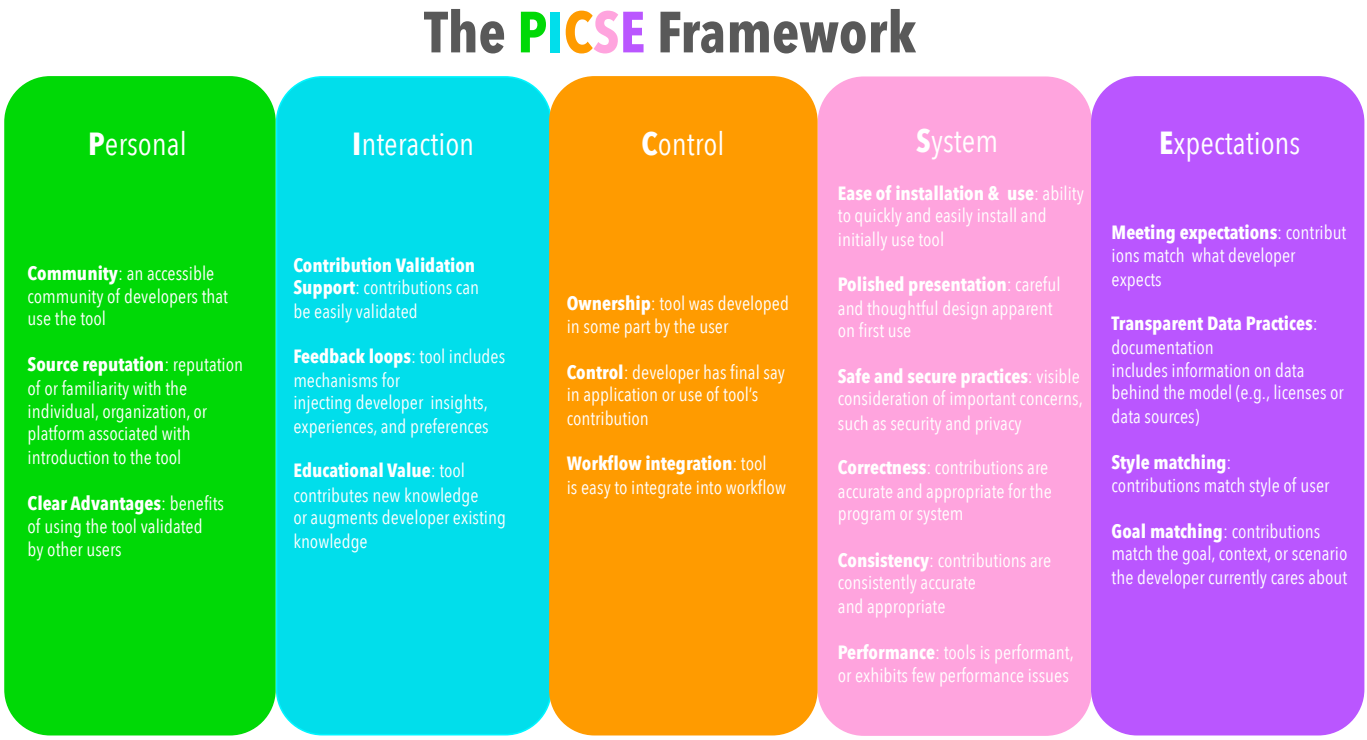

Fig. 2. The PICSE framework for trust in software tools

Community We found that one aspect of a tool that can impact trust is the community of users (or lack there) behind a given tool.

For some, like P5 who polls collaborators and co-workers about tools they should trust, the community of users should be in their own circles. For others, like P7, community can be more broad and a part of that trust is knowing the practice is a preferred practice by others.

> *"That's probably recommended because over the community that's how it's preferable. Then you're leaning towards more into the more community-wide practices."* (P7)

P17 compared tool adoption and use to that of social media platforms, stating *"Even if I trust the brand, nobody else is on there... I wouldn't download the app, the social media. If there is no network, why would I use it?"*

Having a community of users publicly available also provides current and potential users with a way to easily ascertain use cases, success stories, failures, and other relevant information regarding the tool. According to P3, *"if a lot of people think it's a good idea, then I would probably follow and assume it may be a good idea...that I should try it out.* He goes on to explain the value of community around tools, noting that he especially relies on the discussions happening around *"sketchy"* or less than favorable things regarding the tool.

Source Reputation Another way engineers may form and build trust is through the reputation of (or familiarity with) the individual, organization, or platform associated with their introduction to the tool.

The most prominent aspect of this factor among engineers in our study was the individual that they learned about the tool from. This factor can be impacted by users seeking or acquiring insights regarding the tool from others they know. Our findings suggest that for some engineers it is intentional that they seek the input of reputable individuals when looking for or considering using a tool. For others, they do not necessarily seek out reputable sources for recommendations, but their familiarity with and the reputation of the individual(s) they associate with the tool *"carries weight"*.

While most of our interviewees were in agreement that the reputation of or familiarity with an individual is an influential factor, our data suggests engineers may be split on how much weight other source-related attributes carry, such as brand name or company. For some engineers, like P17, the company or brand name behind the tool has a definite impact on trust.

> *"We would still use GitHub because it's such a massive brand, everybody uses GitHub."* (P17)

For others, like P9, while a tool coming from a *"reputable company"* can impact trust, it may not weigh in as much as others given the fact that *"there's so many companies that do put out good tools."*

Clear Advantages According to our interviews, the ability to clearly see the potential benefits that come with using a

given tool has an impact on trust. For most interviewees in our study, determining the advantages or disadvantages of using a tool involved gauging benefits claimed by other users. Some like P5 search for *"anything online which says that for this application, this tool is good."* Others rely on personal or professional contacts for this information. When discussing how he became an avid VS Code user, P10 noted that when some co-workers gave a presentation on the tool *"it was a combination of [...] seeing how powerful it was and how easy it was."*

For some engineers in our study, it is not enough to hear what others' experiences are like. They need to see the benefits themselves. P7 described his thoughts on this matter using the analogy of autonomous vehicles:

> *"But as more and more people use it, and while I'm in that car and AI Is doing the right thing, I'll see, it actually stopped the right car. It actually identified that someone crossing the road and all those small nitpick details. Then that trust will build up and I can rely on AI okay."* (P7)

B. Interaction Factors

***I**nteraction* factors pertain to considerations engineers make regarding the kind of support and outcomes they expect from their interactions with the tool. The factors from our study that fall into this category include **contribution validation support**, **feedback loops**, and **educational value**.

Contribution Validation Support Engineers in our study have increased trust in a tool that supports quick and easy validation of the tool's contributions or recommendations. This factor speaks to the provision of mechanisms for confirming aspects of a contribution such as its correctness, fit, or quality. For P3, this is especially important because the process of validating tool contributions can *"create a lot of annoyances"* and comes with a time cost. P8 was especially particular about this factor when it comes to AI-assisted tools that produce outcomes he *"wouldn't have come to naturally."*

> *"When that's the case, I think there's even more expectations around being able to validate that it's actually valid and correct." (P8)*

Our interviewees highlighted that an important part of providing contribution validation support is providing rationale for the contributions or recommendations being made. For engineers like P12, tools are like human collaborators and should be able to explain contributions made:

> *"It'd be great if they could explain to me the rationale behind its change, because I think just reflecting on what I've been saying, it sounds to me like I'm treating these things as if another person wrote them, and as I said before, when I'm working with someone else, it matters to me if they tell me why. I think with these tools, it would also matter to me if it explained why it made that change. That would help me gain more trust in that system."* (P12)

Feedback Loops Engineers want to feel like the tools they use are taking their preferences and needs into consideration. For AI-assisted and traditional tools, our findings suggest that it builds trust when tools have mechanisms for injecting developer insights, experiences, and preferences. P3 used VS Code as an example of a tool that successfully integrates one form of feedback loop:

> *"The other thing is the level of care towards the user. For example, I see that VS Code is pretty responsive to what people want and they try to create something that everyone enjoys. I think that helps because they really show that they care about your user experience."* (P3)

Educational Value Engineers find value in tools that add value. More specifically, our findings suggest trust increases when tools make contributions that either the engineer themselves would not have thought of or improves upon their own solution. In fact, some engineers (like P12) feel that *"if it's telling you to fix something and that's it...giving no other information...I'm not really going to pay attention or trust it."*

The most common form of educational value that emerged from our study was a tool making a contribution that the engineer themselves may not have thought of themselves, or as P8 put it *"wouldn't have come to naturally."* This is a less intentional form of learning; some engineers in our study explicitly look to tools as learning aids.

> *"Copilot is one of the things I'm going to try and make time to play with because I feel like it will help me learn Python quicker, and write better quality code than I can immediately with Python."* (P9)

C. Control Factors

***C**ontrol* factors are considerations tool users make regarding their ability to make the tool experience what they want and need it to be. The factors in this category include **ownership**, **autonomy**, and **workflow integration**.

Ownership We find that that engineers may have increased trust when they have some ownership over the tool that is being used. This factor was most prevalent amongst tool developers, like P8. They elaborated further, stating:

> *"As a developer, do I trust it or not? Especially if it's using something that I own versus moving to something that somebody else owns."* (P8)

Autonomy Another factor we find can have an influence on trust is the extent to which engineers feel they have autonomy over the integration of contributions. This factor

takes *contribution validation support* a step further by including mechanisms that ensure the engineer *"take[s] the final decision"* (P15).

According to P17, the most important attribute that helps with feeling *"comfortable"* with a tool that automatically contributes to your code base is if one can *"actually see the code that's been written"* and be able to *"at least review it and see what's happening."*

Workflow Integration This factor covers matters that pertain to how well the tool fits into the user's existing workflow. This factor is related to **ease of installation & use** in that it speaks to the ability to easily integrate into a workflow. However, workflow integration is much more user-dependent than ease of installation & use. In our study, engineers may trust tools more that fit into the platforms and processes they are already using.

> *"For something I'm using every day and that I really want to rely on, I want to have that built in and also something that's as easy as possible, as I mentioned, to get into my workflow. I don't want to spend a ton of time."* (P13)

D. System Factors

***S**ystem* factors in the PICSE framework outline considerations tool users may make regarding properties that a tool does, or does not, possess when determining trust. Based on our findings, this includes **ease of installation and use**, **polished presentation**, **safe and secure practices**, **correctness**, **consistency**, and **performance**.

Ease of Installation and Use It helps if a tool is easy to install and set up inorder to quickly begin use. As pointed out by P12 *"a lot of it's [the] setup."* This includes having easily accessible and useful documentation around getting started with the tool, as engineers like P9 *"hate reading 10 pages of documentation"* and having to put in *"so much work to understand how to use [the tool]."* When it comes to complexity in tool setup, P7 summarized the sentiments of many of our interviewees:

> *"If it's complex, I'll probably won't spend too much time. If it's simple enough, installable, and easy to adopt. Then it also gives what you are looking for."*

Polished Presentation When it comes to trust building, our findings suggest a little polish can go a long way. Engineers in our study valued a tool that looks like the developers paid attention to detail when building the resulting tool. As stated by P4, it helps to see that the tool creators *"went the extra mile and did a little bit more than strictly necessary"* and that a little extra polish gives a good first impression. P10 elaborated on the importance of polish further, noting that if engineers can *"see stuff that looks broken...or any little visual inconsistencies"* he may think the tool is poorly made and thereby less trustworthy.

Safe and Secure Practices Another factor that can impact trust is the ability for users to see if and how tool creators made important considerations that impact trust, such as security and safety-related concerns like privacy, in the design, implementation, and documenting of their tool. P15 summarized this factor best:

> *"Second thing is what public information do they have detailing the technical architecture? How are they trying to influence others by saying that they are trustworthy? What evidence do they have?"* (P15)

Most of our interviewees felt strongly about the importance of considerations around safety and security in software tools. P18, for example, noted that she "really appreciate tools that have clear privacy policies" and "invest in trust and safety." P4 used a medical analogy to explain his stance:

> *"It's like if there's a somebody who has invented a brand new brain chip, are you going to install that in your head? Well, maybe not the first version. You let some people take it and then you figure out, okay, is it safe? Then you start using it. Because using the wrong tool can do some damage, right?"* (P4)

Correctness Another factor that can impact engineers' trust in a given tool is the accuracy of the contributions made. According to P12, engineers want *"alerts that are accurate, that are actually valid"*, a part of which according to P4 is setting the right expectations regarding tool use. By providing *"right recommendations,"* (P7) tools can easily build trust with time. A part of building trust over time is being consistent, the next factor in the ***S**ystem* category.

Consistency For engineers in our study, consistency with respect to both tool features and contributions can have an impact on overall trust in a given tool. For most interviewees in our study, they are looking for consistency with respect to the tool's functionality. This includes facets such as consistency in the code it contributes, how quickly it makes contributions or suggestions, the issues it reports, and overall in the anticipated outcomes.

> *"If I'm accustomed to or I have been programmed to do things a certain way, I expect that it will turn out the same every time and then the trust really, like for most services it's like that already [...] You submit something, you expect that it's going to work."* (P11)

Another aspect to consistency that emerged from our findings is consistent maintenance of the tool. This includes things like consistency in the features available, how those features work, and relevant packages or technologies it supports. For P9, this meant *"not breaking things that are working already"*

and making sure *"to support new technology as it comes out."*

> *"Anything that is not getting updates is suspicious. [...] Getting to the more technical, software that is maintained consistently, that is actually supported versus something that someone built in 2003 and packaged for download. Naturally, when it comes to the security folks, anything that is not receiving updates is suspicious.*" (P15)

Performance Finally, and possibly obviously, performance emerged as a relevant consideration with respect to trust in software tools. Our findings suggest that trust may be higher in tools that are performant and lower in tools that exhibit performance issues. P7 summed up the sentiments of several of our interviewees, stating *", it should be performant and reliable. If it's taking too much time and making computer slower and all those things, then you lose that trust, because it's not worth it."*

E. Expectations Factors

***E**xpectation* factors represent tool users' considerations regarding expectations they have built from their own experiences and would like tools to consider. The factors in this category include **transparent data practices**, **style matching**, **goal matching**, and of course **meeting expectations**.

Transparent Data Practices According to engineers in our study, trust is increased when there is visibility into the data behind the model. This includes licenses, data sources, and guarantees, such as legality, regarding data being used. For engineers in our study, this boiled down to where the data is coming from and how the data users contribute will be used.

> *"Let's say that you're using some software tool. Do I trust that this is not selling my data to some third party versus do I trust that it's not going to give me bogus information or it's not going to break my [code]."* (P4)

Style Matching We find that when building trust, it is also important to provide contributions or suggestions that match the coding style that their project is using. This factor was much less prevalent in our data than others. But for some engineers, like P10, they expect *"reasonable"* contributions that *"follow the same style as any other code in the file."*

Goal Matching This factor conveys the importance of making contributions that map to the goals of the engineer using the tool at the time they are using it. Of course goals vary by task; as does the way goal matching can be implemented.

> *"To me, it's a tool [...] that are tuned to my context. That can mean a number of different things. It can mean it's only relevant to what I'm working on right now versus the whole system. Or it can mean maybe something like prioritization, it's showing me the most critical things first. It's not wasting my time, essentially."* (P12)

Engineers in our study realize that goal matching may not always be easy, or even feasible, to achieve. We find that the current landscape of AI-assisted debugging tools may not lend themselves well to goal matching. This is because it is not obvious if and how the tests generated, bugs found, and fixes suggested would match with what they wanted to accomplish (or the scenarios they care about). This was especially the case for the idea of AI-assisted test generation, which P2 noted *"can never know what scenarios I care about."*

Meeting Expectations As implied by the emergence of the ***E**xpectations* category, engineers develop expectations regarding the tools they have and will use. This factor represents the setting expectations and then meeting set expectations. Generally, according to P11, *"you break trust when the outcome is not as expected,"* so it is important to adequately communicate about the tool to help engineers set appropriate expectations.

> *"That's certainly something when thinking about the design or how to just give verbiage that describes how the tool will work. You want to be cognizant of making sure that you're very accurate with those expectations."* (P8)

Put it plainly, and in the words of P4, any given tool *"should really be good at one thing."* Our findings suggest tools should be explicit and upfront about what the tool can and cannot do in order to build trust.

IV. Threats to Validity

External. We conducted our interviews by selecting developers across Microsoft and working in industry, including those with experience in open source and small startups. We asked interviewees to refer others we should talk to, and endeavored to diversify the pool of developers in our study. While we continued to interview and code until saturation was reached, the extent to which our findings generalize across settings may be limited and warrant some future research.

Findings reported in this paper are based on a qualitative study conducted with a convenience sample of 18 engineers, mostly located in the US. We report our findings in the form of aggregated, emergent categories and that factors that our data suggests are relevant. While quantitative insights provide useful information, prior work has cautioned against using quantitative methods on qualitative data [16]. Therefore, we center the discussion in this paper on the qualitative insights gathered rather than frequency of their occurrence.

The goal of our research was the identify and categorize factors that contribute to trust. As with most qualitative studies, we endeavoured to identify as many unique factors

as we could from our data. While the potential for overlap is difficult to completely avoid, we conducted our data analysis to reduce overlap and produce a set of unique factors that, while inter-operable in practice, each contribute something unique to the framework.

Internal. We conducted most of our interviews virtually, which lends itself to a variety of scenarios that could effect the validity of our data and findings. To reduce the potential for issues during our interviews, we used a familiar and commonly used platform and informed participants up front of a contingency plan if the call is disconnected. We did not encounter any major technical issues that would effect the integrity of our data.

Construct. The goal of our study was to better understand how developers think about trust in the context of completing software engineering tasks. However, because we did not have developers complete any actual tasks during the interview, we were only able to collect data on thoughts based on remembered experiences. We mitigated this threat by beginning the interview with background questions that helped participants center their responses in relevant experiences.

As outlined above, we used Suply and Atlas.ti to support the analysis of our data. Subply provides automated transcription support, which could affect the integrity of the data. To reduce the potential for any issues with our data, we manually read through each transcript to ensure it matched the audio files.

Atlas.ti provides a collaborative environment for coding qualitative data. Using Atlas.ti still requires the knowledge and rigor of qualitative data analysis, but makes organizing and collaborating with the data easier which does reduce some of the effort. Both of these tools improve the ability to conduct qualitative research without compromising the integrity or rigor of the research.

V. Applying the PICSE framework in Practice

The PICSE framework provides insights into considerations engineers make when determining if and to what extent they trust a tool. There are some things that tool developers can leverage to improve trustworthiness, while other may be less in the control of the tool creator and more in control of the engineer, context, or task in which the tool would be used. In this section, we outline considerations that can be made to increase tool trustworthiness in practice. To do so, we introduce Aisha, an engineer who leads a team that maintains a suite of software engineering tools.

Much of trust building relies on usage of and interaction with the tool. So what can tool developers do to signal their tool can be trusted? Before engineers adopt and begin using a tool, findings from our study suggest that factors from the ***S**ystem* and ***P**ersonal* categories in the PICSE framework can affect their initial trust in a tool.Our findings reflect that, **Community** and **Source Reputation** are ***P**ersonal* factors that influence trust before adoption and **Ease of Installation and Use**, **Polished Presentation**, and **Safe and Secure Practices** are ***S**ystem* factors that impact trust before use.

Let us imagine that Aisha's team is developing a new tool separate from the existing tool suite her team maintains. Aisha has been made aware of the PICSE framework and wants to be intentional about building a trustworthy tool, but where can she start?

Building trust through **source reputation** would involve introducing the tool via a trusted individual, organization, or platform. Unless Aisha or her team knows each of the tool's potential users, it is difficult to directly have much impact on this factor. However, it is feasible to aim for building accessible **community** around the tool. In fact, our findings suggest it is possible to aim for community and gain benefits of source reputation as well when using platforms like GitHub. GitHub is a platform for building community around software development and is a recognizable and reputable brand. Because of this, having a tool on GitHub can help reduce some of the concerns engineers may have when considering a tool to adopt. In general, it is expected that the factors in the PICSE framework can, and likely must, overlap to build trust in practice.

***S**ystem* factors give tool creators more tangible ways to impact user trust. When Aisha's team is designing and developing their new tool, our findings suggest they can have a positive impact on potential user trust by following **safe and secure practices** and making those practices visible. This includes things like privacy considerations, especially when developing AI-assisted tools.

Another aspect of the PICSE framework Aisha's team can explicitly consider is the **ease of installation and use**. According to engineers in our study, this factor speaks to the complexity and steps involved in setting up the tool for initial use. More complexity means higher cost, which we already know can affect adoption and use [2].

In some cases, factors can overlap across categories of the framework to impact user trust. As it pertains to ease of installation, our findings suggests that one place engineers may look to determine **ease of installation and use** is the **community of users** around the tool. So while working towards ease of installation and use is beneficial, it further helps to have a community of users that potential users can look to for these insights.

Our findings suggest that another way Aisha's team can work to further build initial trust is by working towards a **polished presentation** for the tool. This would start with the design of the tool, making considerations such as the aesthetics, flow, and usability of the tool. Also important, of course, is that the implementation reflects careful thought and consideration in the design phase of the tool.

While not mentioned explicitly by engineers in our study, some of the other factors in the PICSE framework that tool creators can consider for increasing the trustworthiness of their tools include **contribution validation support**, **feedback loops**, **correctness**, **consistency**, **performance**, and **transparent data practices** (which relates closely to safe and secure

practices, but specifically in the context of AI-assisted tool development).

As implied by the diversity and volume of factors in the PICSE framework, trust is built, broken, and re-built beyond initial adoption and use. The factors discussed regarding pre-adoption trust building would apply beyond adoption, but there are additional factors that can only be assessed upon use. Less obvious are some of the possible distinctions between factors as they pertain to trust between different entities, which we discuss next.

VI. Differences in Trust

The focus of our study was on engineers' trust in their software tools. While the PICSE framework is meant to be applicable to any kind of tool, our findings suggest the potential for differences in to what extent certain factors weigh in on the process of trust building.

A. Traditional vs. AI-assisted Tools

One goal of this work is to better understand differences that may exist between trust in and use of traditional software development tools versus AI-assisted tools. Our findings suggest that there are in fact nuances to how engineers think about trust in AI-assisted tools, some of which are motivated by unique challenges to developing AI-assisted software tools.

According to engineers in our study, it might be more difficult to develop a trustworthy AI-assisted tool in comparison to traditional software tools. One reason for this is that engineers view AI as "fundamentally closed source," or less compatible with open source than traditional tools. While it is possible for the implementation of an AI-assisted tool to be made open source, the underlying model is much more difficult to make open source.

AI-assisted and traditional tools are both affected by the factors outlined in the PICSE framework. However, our findings suggest that some factors may be more important with respect to AI-assisted tools than they are when it comes to trust in traditional software tools, such as safe and secure practices and **contribution validation support**. Furthermore, while expectations may be initially low for any tool, engineers' expectations are higher for the growth and evolution of AI-assisted tools. They expect AI-assisted tools to be smarter and therefore improve with time in comparison to traditional tools.

Our findings also suggest that developers may trust AI-assisted tools more than they trust traditional tools when it comes to certain tasks. One common comparison was between the AI-assisted tool Copilot and the traditional tool IntelliSense. Because Copilot is not aware of the user's codebase but Intellisense is, engineers may be more likely to trust tools like Copilot for "boilerplate things" that are not necessarily specific to the current project or domain and use IntelliSense for more project-specific tasks.

Related to this is the fact that some tools may be especially ill-suited to AI-assistance in the eyes of engineers. In particular, our findings suggest debugging tools may be more difficult to make useful, and thereby trustworthy, for engineers. This is where factors such as **goal matching** and **control** become especially important.

B. Tools vs. Collaborators

One connection that has been made and discussed with the increase in availability and use of AI-assisted tools is whether these tools are comparable to human collaborators [17]. Inspired by this line of work, our study included questions about trust in collaborators. In collecting this data, we gathered some interesting insights on similarities and differences between how engineers think about trust in collaborators and trust in tools.

For most of our interviewees, trust in collaborators and trust in their software tools required similar interactions. Though we never explicitly asked in our interviews, a handful of our interviewees explicitly compared and contrasted collaborators and tools when it comes to trust building in software development. More specifically, interviewees often made comparisons (or contrasts) between humans and tooling backed by AI.

Most of our interviewees felt that contributions made by AI-assisted tools are quite comparable to human contributions. As with their collaborators, engineers expect AI-assisted tools to get better with time. They make similar considerations when evaluating the contributions made by both humans and AI-assisted tools, such as **consistency** and **educational value**.

But for a handful of other interviewees, given the nature of AI, there is at least one important distinction between human collaborators and AI-assisted tools. That is the fact that additional vigilance is required when reviewing and integrating contributions made by AI-assisted tools.

VII. Related Work

While ours is one of the first studies focusing on trust in the context of software engineering tools, research has been done that examines trust in the context of AI systems, professional teams, and software development. While none of these studies examine trust in the context of software engineering tools, they all explore trust in one way or another and most have findings that support one or more factors in the PICSE framework.

A. Trust and AI

More than 20 years ago, Fogg and Tseng proposed that "computer credibility" would become increasingly important and offered perspectives in an effort to promote further research. They posit that credibility comprises two key components: trustworthiness and expertise. Their findings of types of credibility map clearly to factors we uncovered in interviews. For example, "reputed credibility" describes how much the perceiver believes something because of what third parties have reported, similar to **source reputation**, while "surface credibility" refers to the perceiver's view of the system based on a *simple inspection*, similar to **ease of installation & use** and **polished presentation**.

Omrani [18] recently explored trust in AI based systems in general and found that the sector where AI technology is applied can influence the level of trust in AI and that

"there are certain sectors that are more likely than others to induce trustworthiness in AI." This finding supports the value of investigation into AI tools specific to software engineering.

Kocielnik *et al.* [4] examined the role of expectation management in success of AI tools. They find that expectation setting is critical for adoption of such tools and show how different tool designs such as communicating AI accuracy and providing explanation can increase the trust of users even when actual AI performance is unchanged. Findings from our own interviews confirm that setting and meeting expectations around tools is an important part of trust formation in the software engineering domain.

Gille *et al.* [19], examined trust in AI tools in the context of healthcare and makes an explicit call that "we need to develop and validate measure that aid he buildup of trust in AI. Such measures may [include] guidance for AI designers [...] including development approval, implementation, use and evaluation." The PICSE framework represents provides first steps in this direction for the context of software tools.

Wang and Siau [20] found that AI models based on neural models may suffer more from trust because of their "black box" nature in which only the input (features) and outputs (predictions) are visible to the user and there is little transparency into the inner workings. This may hinder trust from users of tools based on these models.

B. Trust in Professional Teams

Casey [21] examined trust in geographically distributed teams across four independent studies and identified how bespoke software engineering tools were able to develop and in some cases re-establish trust between remote teams, facilitating processes such as configuration management and document exchange and approval.

We refer the interested reader to the work of Rousseau *et al.* [22] who provide an in-depth survey of trust in the context of firms and professional teams from the organizational literature. Similar to our findings in the SE domain, they find that trust is not static and has multiple phases. They also describe multiple definitions and forms of trust, for example characterizing trust as a level of control in some contexts and about positive expectations in others (both aspects of trust in the PICSE model).

C. Trust in Software Development

Smith *et al.* [23] explored in-house software tool building and found that successful tools often take into account factors in the PICSE model including integration into existing processes, reputation of the tool builder and recommender, and existence of a supportive community.

Vaithilingam *et al.* [3] conducted a user study of Copilot where interviewees in their study clarified their mistrust of "opaque suggestions from Copilot" and would only trust it for simple tasks due to difficulty understanding the code, fear of unknown bugs, and failure to match coding style. While trust was not the primary aim of this study, many of these reasons appear in the PICSE model.

Widder *et al.* studied trust in autonomous software tools via an ethnographic study at Nasa and found that trust was influenced by transparency, usability, social context, and the organizations processes [24].

Murphy-Hill *et al.* found that developers are more likely to use refactoring tools that they trust, but they did not investigate trust formation or what factors increase or erode trust [10]. Later, Murphy-Hill *et al.* explored how developers find and adopt new tools in software development and found that trust in the recommender of the tool plays a critical role, whether the recommendation comes from a teacher/mentor, discussion forums, tutorials, or even twitter [25].

VIII. Conclusions & Future Work

This paper introduced the PICSE framework for trust in software tools, a collection of factors that speak to considerations engineers make when forming and building trust in their tools. The PICSE framework emerged from 18 interviews conducted with engineers both internal and external to the Microsoft organization on their trust in traditional and AI-assisted software tools. Our findings have implications for how we can work to intentionally develop trustworthy tools in practice and effectively harness the power of artificial intelligence to build AI-assisted tools engineers seek as collaborators. As we continue to strive for this goal, our future work will gain additional insights at a larger scale to validate and expand on this framework such that it can provide useful guidance and metrics for evaluating and improving tool trustworthiness.

Acknowledgements

We thank the engineers who participated our interviews and shared their experiences. We thank Ruijia Cheng, Ruotong Wang, and Eirini Kalliamvakou for the great and insightful discussions about this project. Brittany Johnson conducted this work as a visiting researcher in Microsoft Research's Software Analysis and Intelligence in Engineering Systems Group (http://aka.ms/saintes).

References

[1] J.-M. Favre, J. Estublier, and A. Sanlaville, "Tool adoption issues in a very large software company," in *Proceedings of 3rd International Workshop on Adoption-Centric Software Engineering (ACSE'03), Portland, Oregon, USA*, 2003, pp. 81–89.

[2] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in *Proceedings of the 2013 International Conference on Software Engineering*, San Fransisco, CA, USA, May 2013, pp. 672–681.

[3] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models," in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, pp. 1–7.

[4] R. Kocielnik, S. Amershi, and P. N. Bennett, "Will you accept an imperfect ai? exploring designs for adjusting end-user expectations of ai systems," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–14. [Online]. Available: https://doi.org/10.1145/3290605.3300641

[5] "Github copilot," https://github.com/features/copilot, 2022.

[6] P. Anderson, "The use and limitations of static-analysis tools to improve software quality," *CrossTalk: The Journal of Defense Software Engineering*, vol. 21, no. 6, pp. 18–21, 2008.

[7] W. F. Tichy and S. J. Koerner, "Text to software: developing tools to close the gaps in software engineering," in *proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 379–384.

[8] A. Groce, I. Ahmed, J. Feist, G. Grieco, J. Gesi, M. Meidani, and Q. Chen, "Evaluating and improving static analysis tools via differential mutation analysis," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 207–218.

[9] D. Sobania, M. Briesch, and F. Rothlauf, "Choose your programming copilot: a comparison of the program synthesis performance of github copilot and genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1019–1027.

[10] E. Murphy-Hill, C. Parnin, and A. P. Black, "How we refactor, and how we know it," *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 5–18, 2011.

[11] B. Johnson, C. Bird, D. Ford, N. Forsgren, and T. Zimmermann, "Supplemental material for "Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools"," Jan. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7516966

[12] T. Azungah, "Qualitative research: deductive and inductive approaches to data analysis," *Qualitative research journal*, 2018.

[13] "Atlas.ti," https://atlasti.com/, 2022.

[14] Y. S. Lincoln and E. G. Guba, *Naturalistic inquiry*. Sage, 1985.

[15] A. Castleberry and A. Nolen, "Thematic analysis of qualitative research data: Is it as easy as it sounds?" *Currents in pharmacy teaching and learning*, vol. 10, no. 6, pp. 807–815, 2018.

[16] N. K. Denzin and Y. S. Lincoln, *The Sage handbook of qualitative research*. sage, 2011.

[17] S. Imai, "Is github copilot a substitute for human pair-programming? an empirical study," in *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2022, pp. 319–321.

[18] N. Omrani, G. Rivieccio, U. Fiore, F. Schiavone, and S. G. Agreda, "To trust or not to trust? an assessment of trust in ai-based systems: Concerns, ethics and contexts," *Technological Forecasting and Social Change*, vol. 181, p. 121763, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162522002888

[19] F. Gille, A. Jobin, and M. Ienca, "What we talk about when we talk about trust: Theory of trust for ai in healthcare," *Intelligence-Based Medicine*, vol. 1-2, p. 100001, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666521220300016

[20] W. Wang and K. Siau, "Artificial intelligence, machine learning, automation, robotics, future of work and future of humanity: A review and research agenda," *Journal of Database Management (JDM)*, vol. 30, no. 1, pp. 61–79, 2019.

[21] V. Casey, "Developing trust in virtual software development teams," *Journal of theoretical and applied electronic commerce research*, vol. 5, no. 2, pp. 41–58, 2010.

[22] D. M. Rousseau, S. B. Sitkin, R. S. Burt, and C. Camerer, "Not so different after all: A cross-discipline view of trust," *Academy of management review*, vol. 23, no. 3, pp. 393–404, 1998.

[23] E. K. Smith, C. Bird, and T. Zimmermann, "Build it yourself! homegrown tools in a large software company," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 369–379.

[24] D. G. Widder, L. Dabbish, J. D. Herbsleb, A. Holloway, and S. Davidoff, "Trust in collaborative automation in high stakes software engineering work: A case study at nasa," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–13.

[25] E. Murphy-Hill, D. Y. Lee, G. C. Murphy, and J. McGrenere, "How do users discover new tools in software development and beyond?" *Computer Supported Cooperative Work (CSCW)*, vol. 24, no. 5, pp. 389–422, 2015.