

How Software Users Recommend Tools to Each Other

Chris Brown, Justin Middleton, Esha Sharma, and Emerson Murphy-Hill

Department of Computer Science

North Carolina State University

Raleigh, NC

Email: {dcbrow10, jamiddl2, esharma2}@ncsu.edu, emerson@csc.ncsu.edu

Abstract—To help users gain awareness of tools and features available in applications, recommender systems can automatically suggest useful tools. Such systems aim to present recommendations just like users would recommend tools to one another, but little is known about the nature of these user-to-user recommendations. This paper explores user-to-user recommendations through a study of 13 pairs of software users performing data analysis tasks. We found that users were more likely to adopt tools when they were receptive to the recommendation, but did not find the recommendations were any more likely to be effective when they contained other characteristics such as politeness, persuasiveness, or referred to observable tools. These findings suggest that, for example, automated systems should avoid recommending obscure and unfamiliar tools, but making recommendations politely is not a critical design goal.

Index Terms—Peer Interaction; Tool Recommendation; Tool Discovery

I. INTRODUCTION

To make users more efficient and effective, software applications contain numerous *tools*, or features that accomplish tasks. Examples of such tools include FIND in Chrome, FLASH FILL in Excel, and OPEN RESOURCE in Eclipse. Unfortunately, many tools are underutilized because users are not even aware such features exist [13]. The awareness problem becomes more acute over time as applications become increasingly “bloated” with new tools [25].

Applications have attempted to increase awareness through various means. One well-known method is through documentation, such as software help documents, which allow users to search for features and discover tools to help accomplish tasks. But documentation requires users to recognize a tool might exist to make their work more efficient, and then search through the documentation to discover that tool. Web searches for tools likewise suffers from these problems. As Fischer and colleagues argue, such *passive help systems* are ineffective and inefficient for users [8].

As another way to solve the awareness problem, practitioners and researchers have created *active help systems* in the form of recommender systems to suggest new tools to users. One of the most popular examples is Microsoft Clippy, which recommended new tools in Microsoft Office products, such as Word. But users found Clippy so undesirable that Microsoft heralded its later removal from Office as a step

forward [27]. Researchers have suggested that Clippy was ineffective because users found it interruptive [10], impolite [46], and annoying [18].

In contrast to the inadequacy of system-to-user recommendations, prior work has shown that user-to-user tool recommendations are highly effective. When Murphy-Hill and Murphy compared seven different ways users discover new tools, users determined that the most effective way that they learned about new tools was through *peer interaction*, the process of users discovering tools from their peers while completing normal work activities [32], [31]. Such peer interaction can come in two forms, peer observation and peer recommendation. *Peer observation* is when a user sees a colleague using a tool that he is unaware of. *Peer recommendation* occurs when a user sees a colleague accomplish a task inefficiently or without a tool, and recommends a tool.

In this research, our goal is to improve tool discovery by better understanding peer interactions so that future recommender systems can better approximate user-to-user recommendations. Fischer suggests recommendation systems should help and guide users in completing tasks “similar to a knowledgeable colleague or assistant” [8], yet there is limited research on how such user-to-user recommendations take place in practice. Further study in this area is needed because problems with system-to-user tools, such as Clippy, arise from the fact they don’t simulate user-to-user interactions. To examine this, we sought to answer the following research question:

RQ: What characteristics of peer interactions make recommendations effective?

To answer this question, we conducted a study where we asked 13 pairs of users to complete data analysis tasks, observing how they recommended tools to one another and how often suggestions were used or ignored. The main contribution of this paper is the first study, to our knowledge, to characterize how software users make tool recommendations.

II. BACKGROUND

Our study builds on prior research examining the tool adoption problem, technical approaches for improving tool adoption, and how information workers learn from peers.

Previous work has explored the tool adoption problem in a variety of areas. Researchers have examined end-users’ lack of tool adoption in software such as AutoCAD [13]

and Microsoft Word 97 [25]. In software engineering, researchers have examined developer barriers to using debugging [5], refactoring [29], security [48], static analysis [17], and research-off-the-shelf (ROTS) [43] tools. The results of our study aim to reduce these barriers to tool adoption by improving understanding of what makes peer-to-peer tool discovery effective.

Toolsmiths and researchers have developed and evaluated various system-to-user technical solutions for improving tool adoption in software. OWL logs activities by Word users to recommend tools and improve organization-wide learning [23]. ToolBox collects history from networked workstations to recommend Unix commands [24]. Spyglass observes actions and suggests commands to help users navigate code more efficiently [45]. Coronado proposes recommendations to aid code search based on user queries [11]. Researchers have also proposed using various techniques and algorithms to automatically recommend tools, such as collaborative filtering [30] and Information Foraging Theory [34]. Our work aims to improve these and future technical approaches by examining peer interactions and providing implications for integrating qualities that make user-to-user recommendations effective into recommendation systems.

Learning from peers, sometimes called over-the-shoulder learning [44], has been shown to be effective in increasing knowledge. For instance, Cockburn and Williams studied pair programming, where two programmers develop software together on the same computer. They found that the practice improved learning as well as many more aspects of a project for an organization [6]. Peer interaction, a form of over-the-shoulder-learning where users learn about tools from coworkers, has been shown to be effective but infrequent in interviews and diary studies [31], [32]. This research introduced the concept of peer interactions and was essential to our project. While shown to be effective, prior research has not shown under what conditions peer interactions are effective; our paper fills this gap.

Several projects have explored integrating aspects of user-to-user recommendations into recommender systems. Various approaches have been used to simulate recommendations from peers, including: *continuous social screencasting* to facilitate peer interactions remotely through recorded videos of coworkers [28]; *idea gardening* to collect ideas from end-users to recommend solutions for overcoming barriers and completing tasks [4], the *personification* of suggestions in Gidget, a code debugging game, to improve learning for novice programmers [21]; gamifying tool adoption [41]; and crowdsourcing recommendations [12]. Our research aims to provide new implications for improving recommendation effectiveness in automated recommender systems by conducting a novel study to characterize peer interactions.

III. METHODOLOGY

We designed a user study to identify what makes peer interactions an effective mode of tool discovery.

TABLE I
STUDENT PARTICIPANTS

Participant	Gender	Major
S1	Male	Industrial Engineering ♦
S2	Male	Computer Science ♦
S3	Male	Computer Science ♦
S4	Male	Computer Science ♦
S5	Male	Computer Science ♦
S6	Male	Computer Science ♦
S7	Male	Computer Science ♦
S8	Male	Computer Science ♦
S9	Male	Industrial Engineering ♦
S10	Female	Computer Science ♦
S11	Female	Biochemistry ♦
S12	Female	Biochemistry ◊
S13	Female	Computer Science ♦
S14	Male	Computer Science ♦

♦ Graduate Student ◊ Undergraduate Student

TABLE II
LAS PARTICIPANTS

Participant	Gender	Position
L1	Female	Researcher 
L2	Female	Researcher 
L3	Male	Program Manager  
L4	Female	Director of Operations  
L5	Male	Researcher, Analyst
L6	Male	Computer Engineer
L7	Female	Researcher
L8	Male	Language Analyst
L9	Female	Engineer  
L10	Female	Engineer  
L11	Female	Intel Analyst, Researcher 
L12	Male	Systems Researcher 

 Participant had previous relationship with partner

 Participant previously completed computer-based work with partner

A. Participants

Our study was divided into two phases, each with a different sample population and procedure. The participants in the first phase of our study were seven pairs of students from North Carolina State University. Demographic information for subjects in Phase 1 can be found in Table I.

In the second phase, participants consisted of six pairs of professional knowledge workers from the Laboratory for Analytic Sciences (LAS) at NC State.¹ LAS participants had a variety of educational backgrounds and occupations. Table II presents details for pairs in Phase 2, including their role at LAS, if they had a personal relationship, and if they previously performed computer-based work together (excluding emails). In total we had 26 participants form 13 groups in our study.

B. Study Design

We used data from the Titanic shipwreck used in the Kaggle machine learning data science competition to develop tasks for our study.² The data was provided to participants for the tasks in two separate comma-separated values files, TRAIN.CSV and TEST.CSV. Preliminary tasks had participants examine data in

¹<https://ncsu-las.org/>

²<https://www.kaggle.com/c/titanic>

TRAIN.CSV, which contained Titanic passengers’ identification number, whether or not they survived, seat class, name, sex, age, number of siblings and spouses on board, number of parents and children on board, ticket number, ticket fare, cabin, and the port they embarked from. One example of a task we asked participants to complete is to find a pattern between passengers’ age, gender, and the number of siblings and spouses accompanying them. The tasks and datasets used for the study are available with our supplementary materials [42]. The last preliminary task asked participants to rank the factors of survival. In the final task, the participants used TEST.CSV which was similar to TRAIN.CSV but had a different set of passengers and no information on survival. We asked pairs to predict whether eight passengers survived based on their findings from the preliminary tasks.

The first phase of the study required participants to complete six preliminary tasks and the final task. For the second phase, we removed two preliminary study tasks to give us more time to debrief participants after the tasks were completed. The data analysis tasks had the following characteristics:

- allow participants to use a variety of tools, with none of them being the “right” tool;
- not require any specific domain knowledge to allow a variety of software users to participate;
- allow participants to come to some solution quickly, but be able to come up with better solutions using more advanced tools, if they know them; and
- enable a simulation of a typical knowledge work situation where pairs of software users work together at the same computer [31].

So that we could observe any recommendations that might occur, we required each pair to work together on the same computer. One researcher moderated to answer questions about the data and tasks. For each pair, we collected audio and screen recordings while participants worked on the study tasks. We provided participants with a laptop, an external mouse and keyboard, paper, and writing utensils for taking notes.

Participants were allowed to choose any software for completing the tasks. However, we disallowed use of online resources since solutions to the problem can be found online. We did allow them to download software they needed at the beginning of the session. Participants used a Windows 10 machine to complete the study with several data analysis programs installed, including Microsoft Excel 2016 [7], JMP Pro 12 [16], MySQL Workbench 6.3 [33], Python 2.7 [36], PyCharm [35], R (command line and GUI) [37], and RStudio [39]. Participants were allowed to request additional programs to use before the study if they were free and publicly available. Periscope³ was the only software requested that we were not able to provide because it is not free.

C. Debriefing

We debriefed participants after both phases of our study. In the first phase, participants were emailed a survey that

asked about demographic information to gather participants’ major and classification. It also asked about recommendations that occurred during the study. However, we found that participants’ recollections were poor and their descriptions were brief, or no responses were provided at all.

Consequently, for the second phase we switched to an interview and questionnaire at the end of the session. The interview was semi-structured and focused on one effective and ineffective interaction. During each session, the researcher who moderated took note of any possible occurrences of peer interaction and selected two to discuss during the post-interview. The questionnaire gathered information from participants on previous years of experience with software used to complete the tasks and relationship between pairs. To examine the relationship, we asked partners to provide the number of years they’ve known each other and the nature of their relationship (Professional, Personal, Academic, or None), to list software programs regularly used by the other participant in the pair, and to describe past computer-based work they completed together. We excluded “exchanging emails” as computer-based work in our responses.

D. Data Analysis

We analyzed tool recommendations by coding the data we collected. Two independent researchers reviewed and coded screen recordings of each session to find and verify instances of tool recommendations.

We coded screen recordings by:

- determining when a recommendation took place;
- determining if the recommendation was effective, and
- evaluating whether recommendations were polite, persuasive, under time pressure, or made about observable tools.

A list of the data we collected can be found with our study materials [42]. In the remainder of this section, we describe our coding criteria for each of these, which we arrived at iteratively.

1) *Recognizing Recommendations*: We first explain what we mean by tool recommendation. To identify tool recommendations between peers, we created a model for recommendations:



Each node in this model represents a step the users take in the process of tool recommendation. When explaining this model, we use terms borrowed from pair programming to describe the roles for each user, referring to the person operating the computer at the keyboard and mouse as the “driver” and the person working with the driver as the “navigator” [6]. A recommendation can come from either role within our model. We next explain each step in detail.

In the **task analysis** step, users analyze the task and define their strategy to accomplish the task. This step is based on Kieras’ GOMS model [1], where both the driver and navigator mentally divide the planned task into Goals, Operators, Methods, and Selection rules. Methods are a series of operators that

³<https://www.periscopedata.com/>

are used to accomplish specified goals, and selection rules are applied when more than one method exists. Many applications have multiple methods for users to complete a task and reach a goal using different tools.

For example, two users working together and using the same Excel software, independently plan different methods for achieving the shared goal of calculating average grades for a course. Adam, a novice user, plans the following operators in his method: select an empty cell, navigate Excel’s menus to choose the “AVERAGE” function, then enter A1, A2, A3, . . . A15 as individual parameters in the Function Arguments pop-up box. Meanwhile, Zach is an expert Excel user and plans his operators as: select an empty cell, then type “=AVERAGE(A1:A15)” in it.

In the **task execution** step, the driver executes their method, leading to the navigator noticing a mismatch between his method and the driver’s method. This occurs differently for peer recommendation and peer observation. During peer recommendation, the navigator observes the driver completing a task in an inefficient way. For instance, suppose Adam is driving and Zach observes Adam navigating to the “Formulas” menu to find the AVERAGE function. Zach knows that it is more efficient to type the function in a cell directly. During peer observation, the driver executes their method, which is unfamiliar to the navigator. In our example, if we reverse the roles, Adam observes Zach driving and typing text into a cell. Adam notices that AVERAGE was calculated by Zach without using the menu, a method he was previously unaware of.

The **dialogue** step consists of a discussion between peers after observing the discrepancy between the driver’s and navigator’s methods. While dialogue is optional in peer interaction [32], observing dialog was crucial to our study because without it we cannot be sure that a recommendation took place.

During peer recommendation, the navigator observes the driver and then proposes a new method [31]. In this case, the driver discovers a new tool from the navigator. Adam and Zach demonstrate this if Zach observes Adam using the Excel menu, and Zach interrupts to say “Just type AVERAGE into the cell.” For peer observation, the dialogue occurs when the navigator inquires about the driver’s actions after observing an unfamiliar method. For example, when Adam observes Zach calculate the mean by typing a function a cell, he asks “What did you just do?”. This leads to Zach explaining his method.

2) *Determining Effectiveness*: To determine whether a recommendation was effective, we identified tasks for which the recommendee could use the tool after the recommendation. We identified such tasks as those that recommendees performed where the tool could be used to efficiently complete the task. We used the following criteria for categorizing effectiveness:

- **Effective**: Recommendee uses the tool after the recommendation.
- **Unknown**: There were no opportunities to use the tool later in the study after it was recommended.
- **Ineffective**: Recommendee ignores the recommended tool after the recommendation.

3) *Determining Characteristics*: We analyzed the Dialogue between participants to determine if the way a recommendation is made plays a role in its effectiveness. Based on prior work, we selected five characteristics of the recommendation: politeness, persuasiveness, receptiveness, time pressure, and tool observability. We measured the first three characteristics using a valence scale:

- +1 Participant obeyed a specific characteristic
- 0 Participant neither obeyed nor violated a specific characteristic
- 1 Participant violated a specific characteristic

This scale was used to categorize recommendations by politeness (polite, neutral, impolite), persuasiveness (persuasive, unpersuasive), and receptiveness (receptive, neutral, unreceptive). Persuasiveness did not include a neutral category because, as we will show shortly, omitting any of its criteria was a violation of the definition. We measured time pressure and tool observability using a binary (time pressure or no time pressure, observable or non-observable) scale.

We present these characteristics in the following paragraphs. Each characteristic explanation consists of a table containing the definition, as well as positive and negative examples of each criteria. The quotes in the examples use participant identifiers from our study, where *L* is the prefix indicating professional participants and the *S* prefix represents a student. The criteria list for each characteristic we used can be found here [42].

a) *Politeness*: We hypothesize that recommendation politeness improves tool adoption. Previous research supports this hypothesis; Whitworth suggests that the reason Microsoft Clippy made recommendations unsuccessfully was because it was impolite [46]. When studying user-to-user recommendations, Murphy-Hill and colleagues found that respect and trust were important for the effectiveness of peer interactions [31]. To measure politeness, as shown in Table III, we use Leech’s six maxims for politeness: Tact, Generosity, Approbation, Modesty, Agreement, and Sympathy [22].

b) *Persuasiveness*: We hypothesize that persuasive recommendations are more effective than unpersuasive ones. This hypothesis is supported by Fogg’s argument that persuasiveness is important in convincing users to adopt desired behavior through software [9]. The criteria for persuasiveness recommendations are presented in Table IV. We use the three features of persuasive messages described by Shen and Bigsby: Content, Structure, and Style [40].

c) *Receptiveness*: We hypothesize that receptiveness of the recommendee improves the success of tool discovery. Prior work suggests receptivity is important; the second step of Fogg’s best practices for designing persuasive technology is to “Choose a receptive audience” [9]. Fogg provides two considerations for what makes a receptive audience: demonstrating a desire to adopt the target behavior and familiarity with the technology [9]. We used “Demonstrate Desire” and “Familiarity” to categorize recommendations, and define these criteria in Table V.

TABLE III
DEFINITION OF POLITENESS CRITERIA AND EXAMPLES FROM THE USER STUDY

Politeness Criteria		
Tact	Definition Polite Impolite	Minimize cost and maximize benefit to peer “We can do all of it together, just sort by level.” - S9 “We can do a histogram...which is always sort of a pain in the butt to do in Excel.” - L14
Generosity	Definition Polite Impolite	Minimize benefit and maximize cost to self “CONCATENATE you can do. I can do this for you, very easily.” - S10 “Maybe you should write a python script for this.” - L6
Approbation	Definition Polite Impolite	Minimize dispraise and maximize praise of peer “I’m not as good at the Excel stuff as you are.” - L5 “This[partner’s suggestion] is useless.” - S14
Modesty	Definition Polite Impolite	Minimize praise and maximize dispraise of self “From whatever limited knowledge of data analysis I have, I think you need to create a linear regression model...” - S14 “I’m very good at Paint.” - S10
Agreement	Definition Polite Impolite	Minimize disagreement and maximize agreement between peers “Do you want to use Python?” - S8 “No, no, no...Don’t you want it comma separated? That’s what I’m doing.” - S14
Sympathy	Definition Polite Impolite	Minimize antipathy and maximize sympathy between peers “We can try JMP...” [“I haven’t done anything in JMP.”] “Neither have I!” - L14 “It doesn’t matter how you do it.” - L16

TABLE IV
DEFINITION OF PERSUASIVENESS CRITERIA AND EXAMPLES FROM THE USER STUDY

Persuasiveness Criteria		
Content	Definition Persuasive Unpersuasive	Recommender provides credible sources to verify use of the tool “Go here, go to Data. Highlight that...Data, Sort, and it lets you pick two.” - L8 “Let’s try to text filter, right?” - S5
Structure	Definition Persuasive Unpersuasive	Messages are organized by climax-anticlimax order of arguments and conclusion explicitness “I know that SUMIF is a type of function that allows you to combine the capabilities of SUM over a range with a condition that needs to be met.” - S3 “There’s a thing on Excel where you can do that, where you can say if it is this value, include, if it is not, exclude...Yeah, IF.” - S11
Style	Definition Persuasive Unpersuasive	Messages should avoid hedging, hesitating, questioning intonations, and powerless language “Control-Shift-End” - S1 “I guess we’re going to have to use some math calculations here, or a pivot table.” - L9

TABLE V
DEFINITION OF RECEPTIVENESS CRITERIA AND EXAMPLES FROM THE USER STUDY

Receptiveness Criteria		
Demonstrate Desire	Definition Receptive Unreceptive	User showed interest in discovering, using, or learning more information about the suggested tool “That was cool, how [the column] just populated.” - S4 [“So you want to use R for it?”] “No, no, no...” - S14
Familiarity	Definition Receptive Unreceptive	User explicitly expresses familiarity with the environment “Control shift...how do I select it completely?” - S2 “I’ve never done anything in JMP.” - L10

d) *Time Pressure*: Based on prior work, we hypothesize time pressure will negatively impact the effectiveness of recommendations. Andrews and Smith assert time constraints affect decision-making in marketing by stifling creativity, reducing exploratory thinking, and forcing a dependence on familiar approaches [2]. Additionally, Murphy-Hill and colleague’s peer interaction study identified time pressure as a barrier to peer interactions through external factors such as project deadlines [31]. In our study we did not strictly enforce time limits for completing tasks, but recommended each pair spend approximately 7–8 minutes on each one. We measured time pressure using statements mentioning time

from participants or the researcher who moderated before or during a recommendation. If we determined a statement was made regarding time, then we categorized the recommendation as being under time pressure. For example, during one study L13 was driving and noted “I think we have like, four minutes left”, ignoring L14’s recommendation to use the IF function in Excel. Time pressure caused the driver to continue using her own methods and limited exploratory thinking for completing the task.

e) *Tool Observability*: We hypothesize that recommendations of observable tools, tools that the recommendee can easily perceive during use, are more effective than imper-

ceptible ones. This hypothesis follows from Murphy-Hill and colleagues’ suggestion that recommendation systems should have noticeable causes and effects [31]. We examined this by analyzing the tools recommended between participants in our study, categorizing them into two different types of tools: observable and non-observable.

Observable refers to tools that are visible through a user interface. Examples of observable tools recommended by participants during our study include complete applications such as PyCharm and R, in addition to features such as Sort, Recommended Charts, Text to Columns, and pivot tables in Excel. Non-observable tools are features that do not have a user interface, such as keyboard shortcuts. Examples we observed include Control-Space in PyCharm for code completion, dragging the corner of a cell to automatically copy a formula in Excel, Control-V to paste, and Control-S to save.

4) *Resolving Coding Disagreements*: Occasionally, the two researchers disagreed about the codes they applied. We calculated our interrater agreement for politeness ($\kappa = 0.50$), persuasiveness ($\kappa = 0.28$), and receptiveness ($\kappa = 0.51$) using Cohen’s Kappa. According to Landis’ measurement of observer agreement, which has been used by other studies in this field, our agreement for politeness and persuasiveness had a moderate strength of agreement while persuasiveness had a fair strength of agreement [19]. To resolve disagreements, the two researchers watched the video of the instance in question together, explained the reasoning behind their individual rating, debated the reasoning behind their decision, and came to an agreement after the discussion.

IV. RESULTS

We observed 142 recommendations in our study. We categorized 71 as effective, 35 as ineffective, and 36 as unknown. Each pair averaged approximately 11 recommendations with a maximum of 26 and minimum of 3. The first phase of the study contributed 99 recommendations with 48 effective, 22 ineffective, and 29 unknown between students. The second phase added 43 recommendations, consisting of 23 effective, 13 ineffective, and 7 unknown between data analysts. For statistical analysis, we used Wilcoxon rank sum test to evaluate ordinal data and Pearson’s chi-squared test for categorical data. All tests were calculated with an alpha level of $\alpha = .05$ and odds ratios (OR) were used to measure effect size.

A. Characteristics

We categorized each recommendation based on politeness, persuasiveness, receptiveness, time pressure, and tool observability. Figure 1 presents the classifications of peer interactions for each of the characteristics we observed. Table VI displays the rate of effectiveness for the peer interaction characteristics.

1) *Politeness*: We found that polite recommendations were not significantly more likely to be adopted than impolite recommendations (Wilcoxon, $p = 0.4936$), but polite interactions did have higher odds of effectiveness ($OR = 0.6786$). Fischer claims that systems should make recommendations like a trusted peer [8]. Recommender systems are more like a

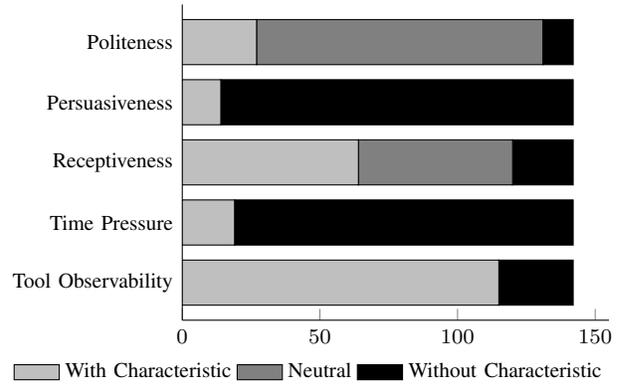


Fig. 1. Breakdown of 142 recommendations for each characteristic

stranger than a peer and while politeness may not be necessary in interactions between peers, it may be more important for system-to-user recommendations [46]. In fact, we categorized most interactions as neutral because many participants suggested tools without explicitly obeying our politeness criteria.

2) *Persuasiveness*: We actually found that unpersuasive recommendations were 1.5 times as likely to be adopted ($OR = 1.4722$), though this difference was not statistically significant (Wilcoxon, $p = 0.4556$, $OR = 1.4722$). This stands in contrast to Fogg’s argument that persuasiveness is important to convince users to adopt a desired behavior [9]. Participants in our study were rarely persuasive during interactions according to our criteria; there were only 14 persuasive recommendations in total.

3) *Receptiveness*: We found that receptive recommendations were significantly more effective than unreceptive ones (Wilcoxon, $p = 0.0002$, $OR = 0.2840$). Receptive interactions had the highest rate of effectiveness out of all the characteristics we studied, with approximately 61% of recommendations classified as receptive also categorized as effective. Our results confirm Fogg’s suggestion that a recommender systems should choose a receptive audience [9].

4) *Time Pressure*: Recommendations without time pressure were more than twice as likely to be effective ($OR = 2.2857$), however time pressure did not play a significant role in effectiveness (Pearson, $p = 0.1470$). The high odds of effectiveness for recommendations without time pressure align with Murphy-Hill’s claim that time constraints hinder peer interaction [31]. We categorized seven interactions as being under time pressure.

5) *Tool Observability*: We found that observable tools were recommended no more effectively than non-observable tools (Pearson, $p = 0.4928$), but recommendations consisting of non-observable tools were twice as effective compared to those referring to observable ones ($OR = 2.4060$). This finding contrasts with Murphy-Hill and colleagues’ suggestion that effective recommender systems should make tools’ causes and effects visible [32]. Observable features were more recommended in our study, with 115 recommendations compared to only 27 for non-observable tools.

TABLE VI
RATE OF EFFECTIVENESS FOR RESULTS

	Effective		Ineffective		Unknown	
	<i>n</i>	%	<i>n</i>	%	<i>n</i>	%
<i>Politeness</i>						
Polite	14	52%	5	19%	8	30%
Neutral	52	50%	27	26%	25	24%
Impolite	5	45%	3	27%	3	27%
<i>Persuasiveness</i>						
Persuasive	5	36%	4	29%	5	36%
Unpersuasive	66	52%	31	24%	31	24%
<i>Receptiveness*</i>						
Receptive	39	61%	9	14%	16	25%
Neutral	27	48%	14	25%	15	27%
Unreceptive	5	23%	12	55%	5	23%
<i>Time Pressure</i>						
Yes	7	37%	7	37%	5	26%
No	64	52%	28	23%	31	25%
<i>Tool Observability</i>						
Observable	57	50%	30	26%	28	24%
Non-Observable	14	52%	5	19%	8	30%
<i>Recommendation Type</i>						
Peer Observation	16	30%	5	9%	32	60%
Peer Recommendation	55	62%	30	34%	4	5%

B. Other Findings

Beyond the five characteristics we studied as part of our research question, several other interesting findings emerged.

First, we found that peer recommendations ($n = 89$) occurred more often than peer observations ($n = 53$). Although peer recommendations had higher odds of effectiveness, the difference was not statistically significant (Pearson, $p = 0.3163$, $OR = 0.5729$).

Second, while most subjects responded to questioning about effectiveness by alluding to characteristics we studied, they rarely used them when making recommendations. When we asked participants why they suggested a certain tool, 69% used “I” statements noting their own knowledge and experience. S7 embodies this attitude by stating he suggested using Find in Excel because “This was a better way to solve the problem at hand and I have used it in similar situations”. When asked about the phrasing of recommendations, 74% mentioned using language that was shorter and easier for themselves. S2 demonstrates this by noting he made his recommendation to use Control-Shift-End in Excel in the “simplest way I could phrase it”. This suggests that most of the time recommenders were not implementing characteristics such as politeness and persuasiveness when offering suggestions to their partner.

Third, some partners had previous interactions before participating in our study. We examined this to see if it impacted effectiveness. In Phase 2, four pairs noted having a personal relationship with their partner while two had previously done computer-based work together. We found that relationship (Pearson, $p = 0.0781$, $OR = 0.2400$) and prior collaboration (Pearson, $p = 0.828$, $OR = 0.3214$) had higher odds of effectiveness but did not significantly impact our results.

V. DISCUSSION

This section presents our observations, implications for recommender systems, threats to the validity, and future work.

A. Observations

Our results were unable to show politeness, persuasiveness, time pressure, and tool observability significantly impacted tool recommendations. Furthermore, contrary to our original hypotheses, we found that unpersuasive recommendations and non-observable tools had higher odds of effectiveness. Alternate theories may explain these findings.

The ad hoc nature of interactions in our study could have limited persuasiveness between peers. Restricting internet use also played a role, with participants often unsure about tool usage and wanting to look up information online. Many subjects violated the persuasiveness criteria by questioning their recommendation and using weak language. The criteria expected detailed recommendations, but participants were often brief and concise. This complies with previous research by Wood and colleagues who found that the length of messages has little impact on persuasiveness [47].

We expected observable tools to be more effectively recommended than non-observable features, but this was not the case in our results. Keyboard shortcuts are non-observable, and previous research points out they are more efficient than observable tools such as menus [20] and are also more beneficial for cognitive learning and recall [3].

We also distinguished peer observations and recommendations from “expected recommendations”. Expected peer interactions are instances of tool learning that occur when the driver actively seeks help by asking for a recommendation, expecting to discover a new tool to complete the task. These

help-seeking recommendations rarely occurred in our study, as we only found 9 occurrences compared to 142 instances of proactive help-giving suggestions between peers. This motivates the need for active recommender systems because users rarely seek help from new tools in their work and automatic recommendations can increase tool discovery and adoption.

B. Implications

Our results indicate receptiveness significantly impacts the outcome of peer interactions. Receptiveness is difficult to implement since it depends on how recommendees respond to suggestions, which recommenders cannot control. Our receptiveness criteria concern users' desire and familiarity, and incorporating these into automated recommendation systems can help improve the tool discovery problem.

a) Demonstrate Desire: We captured demonstrating desire by looking for instances where recommendees explicitly expressed eagerness to use a recommended tool. During a study, L11 was driving when L12 suggested using multi-level sort in Excel. L11 was unfamiliar with that functionality, but demonstrated a desire by responding "Oh! Add level! Yes, awesome!". The recommendation was then adopted for the remaining tasks after L11 expressed interest in using the tool.

Our results suggest recommending desirable tools can increase tool adoption. History-based recommendation systems suggest software features using past actions to measure desire [30]. Furthermore, Fischer notes systems should not respond to behavior but actively make suggestions while users are working [8], and predicting desire with goal-recognition techniques such as the Lumière Project can anticipate user goals and needs to effectively recommend useful tools [14]. How recommendations are made can also impact desire, and previous research in diffusion of innovations shows message communication can influence distribution and reception [38].

b) Foster Familiarity: Previous research suggests users are more likely to adopt target behaviors they are familiar with it [9]. For example, during an interaction S10 recommended creating a plot in R and persuasively added it takes about two lines of code. However S9 responds saying "I don't know R", and her unfamiliarity led to an ineffective recommendation where a potentially helpful tool was ignored.

According to our results, recommending familiar tools can increase effectiveness. One challenge with this is fostering familiarity while proposing new and unfamiliar tools. History-based systems can incorporate familiarity by making suggestions similar to functionality users already use. For example, users who often utilize GUI features are more likely to adopt observable tools than keyboard shortcuts. Murphy-Hill and colleagues have also explored integrating familiarity in systems by using collaborative filtering to rank tools based on similarity to commands used by colleagues [30].

VI. THREATS TO VALIDITY

One internal threat to validity is the Hawthorne Effect, where participants modify behavior due to knowing they are being studied [26]. To minimize observer expectancy and

altering behavior, we did not inform subjects of our objective until debriefing. Additionally, the data was contained in two comma-separated values files but Microsoft Excel was the default program to open *.csv* files. Most pairs used Excel to complete the tasks, however we allowed groups to choose any software for the study. Likewise, participants had to request software without specific knowledge of the tasks. This may have limited informed decisions about programs, such as one participant (L13) noting during a study she should have requested Tableau.⁴ Still, we specified beforehand the study required data analysis software. Another internal threat is that our valence scoring system treated compliances and violations of criteria equally, but this is not accurate in real-world situations. For instance, polite statements do not necessarily cancel out impolite remarks.

A threat to external validity is that we only observed politeness, persuasiveness, receptiveness, time pressure, and types of tools. Other traits may influence effectiveness, but prior research suggests these impact peer interactions. We also measured short-term recommendation effectiveness and not long-term adoption. Furthermore, we categorized recommendations from explicit statements and did not account for implicit behavior. We minimized these by defining effectiveness, criteria, and scoring systems based on observable behavior. Another threat is that the 13 pairs we studied may not represent all computer users. Finally, different cultures have varying definitions for characteristics. For example, Huang compared Leech's maxims used in our study with Chinese concepts of politeness [15].

VII. FUTURE WORK

Future research includes examining additional characteristics that may impact the effectiveness of user-to-user recommendations such as experience, length, task difficulty, or disruption from work. Studies could also examine how our results translate to long-term tool adoption. Future work can improve generalizability by studying additional populations and tool domains, for example software engineers completing refactoring tasks. Further research could also see how our findings apply to co-located teams or larger groups. Finally, future work will involve developing and evaluating automated recommendation systems that incorporate user receptivity.

VIII. CONCLUSION

This research examines what makes peer interactions effective as a means to make automated recommender systems more effective. Our results suggest that the receptiveness of recommendees has a significant impact on the effectiveness of peer interactions. This suggests that automated recommendation systems should clearly recommend tools early in users' actions based on their desire and knowledge in order to increase tool discoverability and adoption in software.

⁴<http://www.tableau.com/trial/data-analysis-software>

REFERENCES

- [1] *The Handbook of Task Analysis for Human-Computer Interaction*, chapter GOMS Models for Task Analysis. Lawrence Erlbaum Associates, 2004.
- [2] J. Andrews and D. C. Smith. In search of the marketing imagination: Factors affecting the creativity of marketing programs for mature products. *Journal of Marketing Research*, pages 174–187, 1996.
- [3] C. Appert and S. Zhai. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2289–2298, New York, NY, USA, 2009. ACM.
- [4] J. Cao, I. Kwan, F. Bahmani, M. Burnett, S. D. Fleming, J. Jordahl, A. Horvath, and S. Yang. End-user programmers in trouble: Can the idea garden help them to help themselves? In *2013 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 151–158, Sept 2013.
- [5] J. Cao, K. Rector, T. H. Park, S. D. Fleming, M. Burnett, and S. Wiedenbeck. A debugging perspective on end-user mashup programming. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 149–156, Sept 2010.
- [6] A. Cockburn and L. Williams. Extreme programming examined. chapter The Costs and Benefits of Pair Programming, pages 223–243. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
- [7] Microsoft excel. <https://products.office.com/en-us/excel>.
- [8] G. Fischer, A. Lemke, and T. Schwab. *Active help systems*, pages 115–131. Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.
- [9] B. Fogg. Creating persuasive technologies: An eight-step design process. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 44:1–44:6, New York, NY, USA, 2009. ACM.
- [10] J. L. Franke, J. J. Daniels, and D. C. McFarlane. Recovering context after interruption. In *Proceedings 24th Annual Meeting of the Cognitive Science Society (CogSci 2002)*, pages 310–315, 2002.
- [11] X. Ge, D. Shepherd, K. Damevski, and E. Murphy-Hill. How developers use multi-recommendation system in local code search. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*, pages 69–76. IEEE, 2014.
- [12] M. Gordon and P. J. Guo. Codepourri: Creating visual coding tutorials using a volunteer crowd of learners. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 13–21, Oct 2015.
- [13] T. Grossman, G. Fitzmaurice, and R. Attar. A survey of software learnability: Metrics, methodologies and guidelines. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 649–658, New York, NY, USA, 2009. ACM.
- [14] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 256–265, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- [15] Y. Huang. Politeness principle in cross-culture communication. *English Language Teaching*, 1(1):96, 2008.
- [16] Jmp pro. http://www.jmp.com/en_us/home.html.
- [17] 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*, ICSE '13, pages 672–681, Piscataway, NJ, USA, 2013. IEEE Press.
- [18] J. A. Krisch. Why you hated clippy, that annoying microsoft paperclip, 2016.
- [19] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.
- [20] D. M. Lane, H. A. Napier, S. C. Peres, and A. Sándor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, 18(2):133–144, 2005.
- [21] M. J. Lee and A. J. Ko. Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 109–116, New York, NY, USA, 2011. ACM.
- [22] G. Leech. *Principles of Pragmatics*. Longman linguistics library ; title no. 30. Longman, 1983.
- [23] F. Linton, D. Joy, H. peter Schaefer, and A. Charron. Owl: A recommender system for organization-wide learning, 2000.
- [24] C. Maltzahn. Community help: Discovering tools and locating experts in a dynamic environment. In *Conference Companion on Human Factors in Computing Systems*, CHI '95, pages 260–261, New York, NY, USA, 1995. ACM.
- [25] J. McGrenere and G. Moore. Are we all in the same "bloat"? In *Proceedings of the Graphics Interface 2000 Conference, May 15-17, 2000, Montr'éal, Qu'ebec, Canada*, pages 187–196, May 2000.
- [26] F. Merrett. Reflections on the hawthorne effect. *Educational Psychology*, 26(1):143–146, 2006.
- [27] Microsoft. Farewell clippy: What's happening to the infamous office assistant in office xp. <https://news.microsoft.com/2001/04/11/farewell-clippy-whats-happening-to-the-infamous-office-assistant-in-office-xp>, April 2011.
- [28] E. Murphy-Hill. Continuous social screencasting to facilitate software tool discovery. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 1317–1320, Piscataway, NJ, USA, 2012. IEEE Press.
- [29] E. Murphy-Hill and A. Black. Breaking the barriers to successful refactoring. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 421–430, May 2008.
- [30] E. Murphy-Hill, R. Jiresal, and G. C. Murphy. Improving software developers' fluency by recommending development environment commands. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 42:1–42:11, New York, NY, USA, 2012. ACM.
- [31] 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)*, 24(5):389–422, 2015.
- [32] E. Murphy-Hill and G. C. Murphy. Peer interaction effectively, yet infrequently, enables programmers to discover new tools. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, pages 405–414, New York, NY, USA, 2011. ACM.
- [33] Mysql workbench. <http://www.mysql.com/products/workbench/>.
- [34] D. Piorkowski, S. Fleming, C. Scaffidi, C. Bogart, M. Burnett, B. John, R. Bellamy, and C. Swart. Reactive information foraging: An empirical investigation of theory-based recommender systems for programmers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1471–1480, New York, NY, USA, 2012. ACM.
- [35] Pycharm. <https://www.jetbrains.com/pycharm/>.
- [36] Python. <https://www.python.org/>.
- [37] R. <https://www.r-project.org/>.
- [38] E. M. Rogers. *Diffusion of innovations*. Free Press, New York, NY, 5th edition, 2003.
- [39] Rstudio. <https://www.rstudio.com/>.
- [40] L. Shen and E. Bigsby. The effects of message features: content, structure and style. *The SAGE handbook of persuasion developments in theory and practice*, 2012.
- [41] W. Snipes, A. R. Nair, and E. Murphy-Hill. Experiences gamifying developer adoption of practices and tools. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 105–114, New York, NY, USA, 2014. ACM.
- [42] Study materials. <http://go.ncsu.edu/peer-interaction-study>.
- [43] S. Tilley, S. Huang, and T. Payne. On the challenges of adopting rote software. In *Proceedings of the 3rd International Workshop on Adoption-Centric Software Engineering*, pages 3–6, 2003.
- [44] M. B. Twidale. Over the shoulder learning: supporting brief informal learning. *Computer Supported Cooperative Work*, 14(6):505–547, 2005.
- [45] P. Viriyakattiyaporn and G. C. Murphy. Improving program navigation with an active help system. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '10, pages 27–41, Riverton, NJ, USA, 2010. IBM Corp.
- [46] B. Whitworth. Polite computing. *Behaviour & Information Technology*, 24(5):353–363, 2005.
- [47] W. Wood, C. A. Kallgren, and R. M. Preisler. Access to attitude-relevant information in memory as a determinant of persuasion: The role of message attributes. *Journal of Experimental Social Psychology*, 21(1):73–85, 1985.
- [48] S. Xiao, J. Witschey, and E. Murphy-Hill. Social influences on secure development tool adoption: Why security tools spread. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1095–1106, New York, NY, USA, 2014. ACM.