Eximia journal

Eximia Journal

(ISSN 2784-0735)

Vol. 5

2022

Eximia Journal
Vol. 5, 289-300, July, 2022
ISSN: 2784-0735
www.eximiajournal.com

Best Practices for Minimizing Rejection and Optimizing Acceptance Among Learners Seeking Online Assistance: A Qualitative Study

Don Deever, Steven Grubaugh, Greg Levitt, Joe Maderick, Gabe Gonzales

University of Nevada

ddeever@unr.edu, steven.grubaugh@unlv.edu, greg.levitt@unlv.edu, joe.maderick@gmail.com, ggonzalez.edu@gmail.com

Abstract. Observing an online community of computer programmers from an insider's perspective, a team of ethnographic researchers were able to carefully monitor the manner in which international newcomers, just getting into the arena of computer programming, were able to successfully, and not-as-successfully, garner assistance from highly experienced programmers. From the qualitative transcripts of the communications between these newcomers and their professional counterparts, it was observed that there seems to exist implied unwritten protocols among software developers and new students of technology, which when followed either knowingly or intuitively, seemed to lead to better quality assistance and more fluent explanations from helpers. This study describes and defines the top ten best practices that were identified and explains the potential significance of emulating those protocols in a wide variety of online communities to assist novices to become more proficient in many academic and non-academic fields.

Keywords. Online help, online assistance, online newcomers, online mentoring, communication skills, online protocols, technology help, online communities, online novices

Introduction

"Protocol" might be defined by this study as the official set of procedures that have been adopted by a society, governing how the different members in that society should conduct themselves in private, and defining what is considered to be acceptable behavior amongst two or more members, as well as what is appropriate when operating in public learning environments (Hillman, Schudy & Temkin, 2022). To successfully partake in a lived experience among a culture that is foreign to the observer, it is vital that a newcomer to that culture learn to honor the most cherished protocols of that culture in an effort to avoid accidentally violating their trust (Dittrich, Floyd & Klischewski, 2002).

There is, in fact, an observable culture among computer programmers, as both learners and experienced helpers, and among such a technologically advanced society, protocols are just as important for operating in that culture as they are in any unique ethnic society (Wellman & Gulia, 2018). This means that newcomers to the world of computer programming must quickly learn to navigate a virtual maze of unwritten protocols, which for some proves to be a harrowing

gauntlet of failures, and for others proves to be a labyrinth of self-fulfillment (Liu, 2017). Virtually all fields of expertise talk about their "learning curve," and meeting the demand for online help, especially throughout the COVID crisis (Schijns, 2021), but it might be argued that few learning curves could be perceived as being more steep with technology newcomers having to overcome barriers such as learning a new language filled with unfamiliar symbols, a deep immersion into mathematical algorithms and a completely unforgiving environment of spelling where even a single comma out of place or one mistyped letter can produce disastrous results (Bosch & D'Mello, 2017).

According to Bloch & Antaki (2019), newcomers to the programming field can encounter enormous and nearly insurmountable challenges including a solely left-brain logic that requires if-then-else thinking and rejects illogical or abstract thought, regardless of how clever such notions may be. The researchers in this study, having immersed themselves in the world of computer programming for many years, have concluded that the best way for a newcomer to become established and develop a good working skill in the field is with the help of other, significantly more experienced programmers. This study is unique in that it explores a gap in the literature where the protocols of how beginners successfully asked software development questions of experts, doing so by observing the attitude in which questions were asked rather than how questions were phrased (Wellman & Gulia, 2018); Bosch et al, 2017; Treude, Barzilay, & Storey, 2011). In this sense, this study is truly ethnographic in nature since it focuses on how to act within the software developer culture in order to receive the most beneficial responses from its members (Anderson, Huttenlocher, Kleinberg, & Leskovec, 2012; Vasilescu, Filkov, & Serebrenik, 2013).

Purpose

As with virtually all major software development languages, this particular programming language resource on ResearchGate, a for-profit European social networking site where researchers can post and respond to queries, share papers, and exchange ideas, offers a variety of question-and-answer forums to accelerate the learning curves of new users. The current research focuses on a portion of the forum that is accessible to the general public. In fact, one of the authors of this study has been immersed in this elite digital technologist culture for nine years and is therefore accustomed to observing how different new members are treated in response to requests for assistance (Guo, 2017). Using an inside-out and outside-in perspective on responses to beginner's requests for assistance, the team involved in this qualitative research project was able to provide plausible explanations for the varying degrees of quality of assistance received by "askers," identified in this qualitative analysis as new learners of coding and technology, who were new to this programming language as in Stack Overflow (Schijns, 2021; Thomas, & Hassan, 2014; Vasilescu, Filkov, & Serebrenik, 2013; Anderson et al, 2012). In this regard, the focus of this study was on what the askers did when posing questions to effectively attract help, while the actions of the mentors or expert "helpers" were examined primarily from the perspective of how they reacted to the askers (Foong, Dow, Bailey & Gerber, 2017; Treude, Barzilay, & Storey, 2011).

A combination of extensive online experience with this particular community and the observations of twenty-two subjects allowed for the thematic generation of the ten suggested protocols to be used when asking for highly technical help online, many of which have implications in other online situations outside of the computer programming/software development field, for instance help lines in medicine, education, etc. (Hodgson, 2021; Stephenson, (Ed., 2018). Such insights could be applied to virtual classrooms, aiding in the

establishment of a knowledge base of the best-practice protocols for requesting online assistance in order to promote more advantageous responses across many fields (Schijns, 2021; Liu, 2017; Schueller, Tomasino, & Mohr, 2017). Such insights may also be applicable to online learning websites, where students are becoming increasingly reliant on expert advice, particularly in light of the ongoing COVID disruptions. Overall, it is hoped that this study will provide valuable insights that will assist students of all ages in understanding how to approach the process of requesting assistance from online experts through a variety of help forum genres.

This research focuses more on our observations of other relative newcomers to the software development industry, although it is related to the digital researchers' own experiences conversing with elite digital natives in the public forum of computer programmers specializing in one particular programming language (whose name has been omitted for privacy reasons). Therefore, he and other members of the research team have endeavored to interpret the manner in which askers behaved positively within this culture in order to elicit the most goodwill from the help givers or expert mentors.

The internal researcher's own requests for assistance on the software developer forum are essential to this study in order to provide an expert interpretation of how his own successful communications relate to what the team observed in the help-seeking attempts of the study's subjects (Deever, Grubaugh, Levitt, Maderick & Gonzales, 2019). Through the lens of his personal experience, the team was able to interpret and offer insights that, if heeded, may result in improved reactions and outcomes for new learners of coding and technology when seeking assistance in understanding and utilizing computer language for the purpose of learning to develop software (Stephenson, (Ed., 2018; Perrow, 2017). This study aims to identify the unwritten protocols that appear to lead to the most successful communications between new software developers seeking assistance from experienced computer programmers (Schueller et al, 2017). Moreover, it is the intention of this study that the insights presented here will enable new learners to confidently enter the creative realm of software development and other realms of online help, as well as have greater success in locating effective mentors who can help them build a solid foundation of knowledge and skills.

Methods

The main research question in this ethnographic study was simply, “What unspoken protocols, if any, if followed by new students of software development who were asking for help, seemed to produce the best responses from the mentoring programmers who attempted to answer their questions (Perrow, 2017). Considering the research question, this study sought mainly to look at the interaction between two major categories of subjects, which have been labeled as “askers” and “helpers.” Askers represent a single group of subjects who were new at learning to use the particular programming and development language that this research focused upon for this study. The reason askers are considered to be a single category is because of all cases observed in this study, every asker used the portion of the forum titled, “Getting Started with Development Software - Complete Beginners.”

When it came to the helpers, however, their group was divided into three separate categories that included “power helpers,” “moderate helpers,” and “regular users.” The distinction between these three groups of subjects has been based on the number of postings they have made in the forum. When it comes to the top tier of helpers, they are highly experienced software developers who have made anywhere from 3,000 to 8,000 postings to help others. The more moderate helpers are those who have made from 1,000 to 2,999 postings, and finally, the regular helpers are those people who have made less than 1,000 postings.

Eximia Journal
Vol. 5, 289-300, July, 2022
ISSN: 2784-0735
www.eximiajournal.com

In keeping with ethnographic traditions, it was a goal of this research team that a minimum of twenty subjects were to be observed. The actual numbers turned out to include twenty-two separate individuals classified as askers, while the total number of the group labeled as helpers equaled exactly twenty separate individuals. The reason for using the term “individuals” is due to the observation that three of the askers initiated two different requests for help, while at least half of the helpers made dozens of responses each. The time period chosen for observing these subjects was conducted over a recent holiday period which is not being specified in order to maintain participant anonymity

Results

This study's raw data consisted of sixty-eight pages of transcripts from a ResearchGate developer forum that were analyzed using content analysis and manual coding. The pages were compiled from conversations between newcomers to the world of software development and “good Samaritans” with vastly superior knowledge and experience with the programming language in question.

During the observed fifteen-day period, there were a total of twenty-five questions or pleas for assistance from novice programmers seeking assistance in creating a properly running computer function. This request for assistance resulted in a total of 140 messages or "threads" between askers and responders, which equates to an average of 5.6 messages per response. In other words, there were approximately six exchanges of dialogue for each question. In total, the number of threads per request for assistance ranged from a maximum of eighteen messages to a minimum of two messages, with seven and three messages per request being the most frequent on four occasions. Five messages, four messages, and two messages, which all occurred on three separate occasions, accounted for the second-most frequent number of responses. All other exchanges per request numbers (18, 13, 11, 10, 9, and 6) occurred only once. It was determined that, out of the 140 messages exchanged between question-askers and helpers, an equal number of exchanges occurred between question-askers and helpers, totaling 70 responses each.

In addition to recording the number of exchanges between the observed subjects, the various types of topics that were brought up were also recorded. These included how-to questions regarding the following software development topics: Apps Creation = 2, Databases = 3, Deletion = 1, Digital Photos = 1, Games = 2, Getting Started = 1, Internet = 1, Key Controls = 2, Menus = 1, Multimedia = 2, Naming Objects = 1, Printing = 3, Shells = 1, Software Upgrades = 1, Spreadsheet = 2, and Tutorials = 1. Printing and databases were the two most popular topics, with three discussions each. The second-most popular topics were games, key controls, spreadsheets, and multimedia, with each involving two discussions.

The frequency with which a question-asker received "useful" responses, i.e., answers that were understandable, applicable, and resolved the issue they were experiencing, was also noted as an important result. This category of data was divided into four types of observed outcomes: "Useful," "Moderately Useful," "Self-answered," and "Not Useful" "Joy," "Moderate Joy," "Self-answered," or "No Joy" meant that no usable answer was forthcoming, the answers provided were inadequate, or the questioner withdrew and did not interact further with those who offered assistance. According to the useful to not useful joy-to-no-joy data outcomes, fourteen times out of fifteen when a requester sought assistance, the response was deemed "Useful" (Joy on the evaluation scale). When it came to moderate or partial happiness (Satisfaction), There were six instances where responses of "Moderately Useful" were provided and received. There were two occurrences in the category of self-answered questions (questions

from users who figured out the answer to their own question). Finally, only three requesters discovered responses that were Not Useful.

Implications

By interpreting the results from the perspective of an insider, it was clear that more than half of the time (56%) when a request for assistance was made, the requester received information and responses that they regarded Useful. They were completely happy with the response, and the issue was resolved. This insight should be reassuring for anyone delivering or attempting to receive knowledge from technology veterans and learning how to become a software developer. The results of the research indicate that technology inquirers or new students are twice as likely to receive substantive (Useful) responses. As a result of knowing that there is a greater than 50 percent possibility that their problems would be resolved by kind members of the community, they may face any obstacles with confidence. In the case of those who found Moderately Useful responses to be a source of joy or at least a partial solution to their problem, this research team identified several common mistakes made by the askers, such as not responding to requests from helpers for more information, not providing ample information to receive adequate help, not being more forthright when an answer was not working for them, not fully admitting when they did not understand the advice received, and not acknowledging when they did not understand the advice received Additionally, it must be understood that not every request for assistance may be fulfilled. Every programming language has its own restrictions or faults that occasionally prohibit the successful completion of a function.

Moreover, these extracted ten communication protocols can be fairly easily emulated by beginners wanting to successfully immerse themselves for the first time into the high-tech world of learning to become software developers as well as other digital pursuits. It was observed by this research team that paying strict adherence to these ten forthcoming communication protocols can significantly improve one's chances of successfully receiving valuable answers to the programming how-to questions that they seek.

Fortunately for newcomers entering a field of study, there appear to be protocols to be followed when seeking assistance from computer programming experts during what can be a difficult initiation into any field. When these "top ten protocols" are used thoughtfully, as depicted in this ethnographic study, newcomers appear to significantly increase their chances of not only receiving helpful assistance, but also receiving such assistance in a manner that can be best comprehended. Using the study's rendered themes as a basis, ten protocols were derived and generalized from this qualitative research; the protocols, which are listed below, should be helpful to anyone seeking assistance from online support sources.

Protocol 1: Exhibit Humility

It was observed that a beginning "asker" does best when exhibiting humility to the point of self- deprecation. This may call for admitting that their question is most likely a very basic one, or even admitting that the question is probably stupid, or using an emoticon to signify that one is embarrassed about having to ask. Other means of establishing sufficient humility in this culture is to refer to oneself as a "newbie," or otherwise openly admit one's total ignorance of software development in general. Along these lines, the asker is taking responsibility for not being able to figure out what they need to do, rather than blaming the tools or accompanying documentation of the computer programming language. Examples of humility could be seen in statements that started with the words, "I am new to [this programming language] and have only

been working with it for the past few months." "Hi all, as you might guess, I am new to [this programming language]." "I'm a stone cold beginner to coding entirely, so I'll probably be posting a few dumb questions... :lol:" Simply admitting that one's question is very basic usually had the same positive effect as being deferential in other ways. An example of that idea was found in these words, "HI. Very basic question (I hope)."

Protocol 2: Disclose an Honest Effort

Would-be askers would do well to remember that power helpers were not born power helpers but established their rank in this culture through thousands of hours of effort and experimentation and studying programming languages on their own. In this respect, an asker (help seeker) should always remember the old maxim, "God helps those who help themselves." This means that one is most successful in receiving help when first going out of their way to mention how much they have tried to help themselves. This could include mentioning that they spent the past hour searching the forum for a similar or related question but to no avail, and that they did not neglect to consult the programming language documentation or refer to the dictionary of codes that typically accompanies most computer languages. An honest effort that shows that one has made an honest effort to help themselves would also include referring to having searched through all available sample codes and/or sample programs that are free for one to dissect in order to discover how they work, or to mention a careful look at any of the available training tutorials that exist online for the programming language. Basically, what one is trying to establish here is that they put in the time and made a valiant effort, but because of their ignorance of the topic, their only recourse at this point was to turn to the forum for help. A good example of showing that one tried, as well as exhibiting some humility at the same time, could be found in this statement, "I have searched this forum and did not find a relevant post. If I missed it please let me know." In the case of the asker who made the aforementioned statement, they basically put it out there that they would humbly receive directions to where to find the instructions rather than have a potential helper have to explain it to them. That act of generosity on the part of the asker resulted in a generosity of spirit on the parts of the helpers. In this second protocol, there also seems to be an aspect of helpers feeling that a particular new user has suffered enough, such as when seeing words such as these, "I have been banging my head all day yesterday trying several commands to shut down a media player..."

Protocol 3: Admit Confusion

It is vital to sound adequately confused and in need of direction, which can be a difficult tone of voice for many newbies to take because beginners in any field often want to sound as if they know more than they do at first, but this team observed that if an asker sounded as if it might be possible to answer their own question or they sounded as if they were on the verge of answering it, power helpers seemed to stand back and let fate take its course. To further enhance the need of a request, it behooves beginners to genuinely plead for help and to make it clear that they will not be able to accomplish what they are trying to do without receiving help. Helpers love to be needed, and the more they are needed, so long as that need does not become habit, the more they are willing to assist. A good example of a statement from an asker that fits into both the protocol of demonstrating that they made a valiant effort before seeking help and that they are not likely to find the solution they need without help, can be seen in this statement, "After wasting a LOT of index cards testing this I am hoping someone more experienced with this software may be able to shed some light on this issue." Other examples of demonstrating need included, "No matter what I can't seem to print the full height of the card." "Anyone got

Eximia Journal
Vol. 5, 289-300, July, 2022
ISSN: 2784-0735
www.eximiajournal.com

any ideas where I could be going wrong." "I didn't know how to solve issue :cry:" "I am struggling with very basic problems that keep me from even creating the simplest stack." "I seem to be having some trouble with naming groups. I was attempting to make it work in a wider script but have tried it independently on a stack by itself and can't get it to work for the life of me." Certainly, the ultimate example from the transcript data began with the words, "I may have an emergency on my hands... Hoping someone knows what I can do to recover my file without having to do a few dozen hours of work to attempt to reproduce it. :cry: :cry: :cry:"

The opposite example of this protocol occurred when an asker began with the words, "I have an awesome idea..." That message was followed later with the statement, "I have partial success," and then the asker continued documenting their progress so well, it seemed likely that they would not need help, so it took from February 15 to December 22, more than ten months later before they did actually receive some help. This was a very unusual occurrence, since most responses take place within twenty- four hours and seldom exceed more than seventy-two hours, but it does serve as a good example of the hazards of appearing to be too independent.

Protocol 4: Ask Only One or Two Questions at a Time

Power users who do not mind answering a single question, do not seem to like it if two questions are asked at the beginning. From the personal experiences of the inside researcher, he found that he needed to transform multiple-part questions into a single question, even if other parts of that questions were equally vital to know. In that case, when the first single part of the question was answered, it was observed that one could always expand upon it with the next portion of the question, or humbly ask the forum whether they should post the second part of their question as a completely new and different thread. An example of a proper single question would be the following, "What is the correct procedure?" Also, a good example of adding a question once the original question has been asked would be, "Yes, thats it. Thank you. And one last question. How do I clear the whole datagrid?" "Next stupid question - unrelated but I didn't think it warranted its own topic." "Newest issue is that I'm not having any luck sending ctrl-c to the process."

Protocol 5: Provide the Context(s) Which Frame the Problem

It is important for the asker to provide proper information up front about the model of the operating system they are using, and the version number of the computer programming language being used, prior to a power helper having to be annoyed by asking for it. An example of this action would be the following, "I now have version 8.0.0-dp-12 | build 13012." "Is it possible to create a menubar application for mac?" Those who receive the best help, as witnessed in this study, are those who make it easiest for power helpers to assist them. Providing proper information can also mean letting potential helpers know what one can and cannot do, which is a thoughtful act, so they know what not to waste their time on explaining, as well as allows them to know where one really does need the most help. An example of that type of thoughtfulness on the part of an asker can be seen in these words, "I know how to write entries on the datagrid but i don't know how to add one, one at a time. If anyone has the time, plz write a simple example."

Protocol 6: Ask Follow-up Questions

To receive adequate joy from an answer, it helps to assume that the advice received will likely be inadequate at first and therefore the asker must be ready to initiate some further prompting. Quite often computer programmers will throw out advice that takes a greater degree

of knowledge and proficiency to utilize than the original topic being asked. This means that the asker needs to be aware that they will most likely be required to ask for specific examples in code from would-be helpers. (Kim & Ko, 2017). There is an aspect here and potential danger of causing helpers to feel as if they are doing one's work for them when asking for specific examples of code. One way of getting around that uncomfortable situation is to ask helpers if they know of any sample programs whose available code could help them to understand what they mean and provide a workable example at the same time. In that case, instead of dredging their memories for examples, they will usually gladly provide a new example for the asker of just what they need for their own program.

There is also the possibility that some advice, no matter how well-intentioned, could be more than inadequate in that it could completely fail to work. In that case, the helpers need to be nicely told that their advice was a bust, such as in this example from the transcripts of this study, "Thanks for your effort, but 'mobileControlDo' did not work." When doing so, experience has shown that attempting to accept the blame for it goes over best and gets the most helpful responses in return. This would include statements such as these from the transcript, "I must be doing something wrong," or "I must not know the correct way to use your advice," and then show them how one is using it and explain what happened or did not happen when they made the attempt.

Moreover, a help-seeking asker should always be ready to request clarification on advice. There should be no limiting shame in admitting when one does not understand what a helper said because that situation is a common one. Helpers are going to be answering back in a difficult language and are generally going to assume that one knows more than less, which is typically a wrong assumption.

Protocol 7: Demonstrate Your Own Knowledge and Accept the Limitations of Your Knowledge

A common mistake to avoid is to blame the computer programming language for one's difficulties. It is much better to do as one of the subjects did when repeatedly putting the blame for their problem squarely on their own shoulders, "The issue I am having with it is that I am unable to determine which section of the code defines..." It is likely that the power helpers are great critics of that programming and development language too, and they feel it is their language to judge but not the right of someone who is new. In other words, they treat the programming and development language as if it were a sibling, and even if that sibling is annoying on a regular basis, only family has the right to criticize family.

Conversely, the act of being impressed with the programming language goes a long way toward receiving maximum beneficial help. Despite its flaws, and all programming languages have plenty of idiosyncrasies and bugs, power helpers are proud of it or they would not be hanging around on that forum. It is like the old cliche, "You can catch a lot more flies with honey than with vinegar." If power helpers think that one is already impressed with their favorite programming language, they want them to be even more impressed by demonstrating to them that it can do what they are wanting it to do. A good example of a user exercising this important protocol in this study was the following: "[This programming language] seems to be well adapted for my needs, but I am having a hard time getting this project started."

Protocol 8: Ask for Illustrative Examples

It is wise for an asker to share a sample of the code they are using that is causing them problems. This not only alerts helpers to the exact nature of the problem but also prompts them

to write more and better code for the asker, which might not only solve the initial glitch but greatly improve the program they are writing. Related to this thought is the idea that the forum offers formatting tools that highlight if one's words on their posting consist of programming language or dialog, and it is greatly to the advantage of the asker to use those formatting tools for maximum responses. After all, what the inside researcher of this study observed over nine years is that it is almost impossible for a power helper to resist looking at the code of someone new to programming, or indeed, to forgo viewing any pure code that is placed on the forum. Besides, it is only when one offers a sample of their own faulty code that they can ask such questions as the following query that was seen in this study, "Any ideas on which part of the code I need to change...?"

To add to this thought, one also wants to be able to describe to the power helpers what they hope their code will accomplish, as well as provide an adequate description of what it is doing instead. Basically, this allows a power helper to either visually recognize the problem or to cut & paste that faulty code, put it into a trial program, and test it first-hand themselves. Without providing them with faulty code, they would have to create the same function on their own, which requires much more time and work on their part.

Protocol 9: Use the "Challenge Effect" When Appropriate

This protocol is referred to as the "challenge effect" and is based on the observation that every show-off likes a challenge, and make no mistake, power helpers can be show-offs. There are many ways that newbies can challenge power helpers. A few of those ways can be seen in these lines from the transcript, "It's like, if the Android player is initiated from the phone itself there is no way that [this programming language] can shut it down. Tell me it's not true!" Other ways of posing a challenge can be seen in these selected portions of the transcript, "Is it possible to...?" or "Is there a way to...?" In one rare instance, a frustrated new user enticed power helpers to rush in to defend the reputation of their cherished programming language when she questioned, "I am not sure if I should invest in this tool or better search for another a[t] this point." Because the user was legitimately frustrated, while using the newest beta version of the software that was not quite stable yet, friendly advice swiftly flowed in to set her on the correct path with the right version of the software. When it comes to a challenge, quite often it is as simple as asking a question like, "Any ideas where I could be going wrong?"

Protocol 10: Show Genuine Appreciation

Power helpers have long memories so it certainly behooves the asker who knows they will likely be seeking more help in the future to come across as genuinely appreciative of the help they received. There are several ways of accomplishing the art of appreciation in such a culture. An appropriate way observed was to use the advice that was given and then report back on one's success with a note of thanks. Another excellent way of showing thanks is to remember that when one asks for help and receives an adequate response, for years to come others may refer to that thread to seek answers to similar problems they are experiencing. Therefore, an excellent way to demonstrate one's appreciation for the help they received is to provide a full example of how they finally put together and used the code that now works for them. Providing a concrete example helps to turn one's original request for help into a reference for others to use and seems to cause the power helpers to feel proud about having contributed a beneficial amount of documentation that furthers the interest of their cause. Some good examples of appreciative responses in this study included the following: "Perfect! Thanks." "Perfect. Works as desired. Thanks guys!" "Help appreciated." "Worked beautifully, thanks!" "Thank you for

your response." "Hi, thanks for replying." It also appeared to go over well when the thanks were presented with humility such as in this case, "Really? All I was missing was 'of this card'? Damn." "ah. didn't think binfile would have worked that easily. thanks a lot."

Certainly, one of the most successful ways observed of saying thanks was to not only give thanks but to also praise the programming language, as in this example, "Anyway I am back on track and able to really get going with this promising tool. Thanks also for welcoming, I expect to need some more help or advice in the future. Happy coding to you all." It even pays to be appreciative of information one did not ask for, as in the following example, "haha I did mean what Jacque said, but thank you all anyway :D . Its cool to learn more stuff then I need hehe."

It was interesting to note that out of twenty-five requests, only seven askers did not directly express "thanks" afterward. This means that helpers know that there is a very great chance that their efforts will be appreciated, which most likely continues to motivate them to help. As can be seen, there are overwhelming odds that an asker will be appreciative of the assistance they receive from helpers.

These ten included communication protocols generated from the study can significantly improve one's chances of finding valuable answers to programming how-to questions that they seek. These protocols can be fairly easy to emulate by beginners wanting to successfully immerse themselves into the high-tech world of learning to become software developers.

Conclusions

Perhaps the most important potential ramifications of this study are the high rate of happiness experienced by question-askers and the wealth of helpful advice received by the vast majority of question-askers as a result of the ten key communication protocols found in this study. As demonstrated by this study, novice software engineers were able to obtain an abundance of free assistance by requesting it and presenting their request in a way that boosted the quality of the responses they received. In fourteen out of twenty-five requests for assistance, novice software developers were provided with actual code to make their programs function, demonstrating the quality of the guidance provided.

It should also be noted that there were no instances in which the helpers appeared unmotivated or acted as if the requester was asking too much of them. These responding experts were eager to assist and never gave the idea that they were being exploited. Moreover, it was found that the helpers provided the same number of responses as the requesters, which was an impressive indication of the forum participants' kindness and desire to devote as much time as necessary to assisting others.

Given that there were no instances of rudeness on the part of the helpers, these study results should allay any concerns that a new software developer and other technology learners may have about asking for assistance to get started, to overcome a learning obstacle, or to clarify some challenge or quirk of computer programming. Even among a hierarchy of digital natives as elite competent software developers, it proved to be a remarkably secure place for newcomers. It is the hope of this research team that such knowledge will encourage many more people to venture into the digital arena, knowing that if they do so armed with knowledge of the ten communication protocols identified in this study, they will knowingly ask questions based on the protocols revealed in this research and thus prepared, receive the best answers they need to grow and become knowledgeable help-seekers in every request for assistance using digital help services.

Moreover, these ten communication protocols can be fairly easily emulated by beginners wanting to successfully immerse themselves for the first time into the high-tech world of learning to become software developers. It was observed by this research team that adhering to these ten included communication protocols can significantly improve one's chances of successfully receiving valuable answers to the programming how-to questions that they seek.

References

[1] Anderson, A., Huttenlocher, D., Kleinberg, J., &Leskovec, J. (2012). Discovering value from community activity on focused question answering sites: A case study of stack overflow. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, 850-858.

[2] Bloch, S., & Antaki, C. (2019). The pivot point between problem presentation and advice in a health helpline service. Applied Linguistics, 40(4), 699-716.

[3] Bosch, N., & D'Mello, S. (2017). The affective experience of novice computer programmers. International Journal of Artificial Intelligence in Education, 27(1), 181-206.

[4] Dittrich, Y., Floyd, C., &Klischewski, R. (2002). Social Thinking--Software Practice. Mit Press.

[5] Foong, E., Dow, S. P., Bailey, B. P., & Gerber, E. M. (2017). Online feedback exchange: A framework for understanding the socio-psychological factors. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 4454-4467). ACM

[6] Deever, D.A., Grubaugh, S., Levitt, G., Maderick, J. & Gonzales, G. (2019). A Qualitative Study of Ethnographic Variables Affecting Exchange of Knowledge Among Online Help Seekers and Elite Digital Natives with Recommendations for Getting the Best Technology Assistance Possible. Journal of Education and Human Development, 8 (3), 1-12.

[7] Guo, P. J. (2017, May). Older adults learning computer programming: motivations, frustrations, and design opportunities. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 7070-7083). ACM

[8] Hillman, D, Schudy, R., Temkin, A., (2022). Winning Online Instruction: a Q & A for Higher Education Faculty. (First edition). Routledge.

[9] Kim, A. S., & Ko, A. J. (2017, March). A pedagogical analysis of online coding tutorials. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 321-326). ACM

[10] Liu. (2017). Relationship between the factors influencing online help-seeking and self-regulated learning among Taiwanese preservice teachers. Computers in Human Behavior, 72, 38–45. https://doi.org/10.1016/j.chb.2017.02.034

[11] Morrison, P., & Murphy-Hill, E. (2013, May). Is programming knowledge related to age? an exploration of stack overflow. In Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on, 69-72. IEEE.

[12] Palmer, E. K. (2013). Use of LiveCode in CS0. Journal of Computing Sciences in Colleges, 29(1), 94-94.

[13] Perrow, M. (2017). Strengthening the Conversation in Blended and Face-to Face Courses: Connecting Online and In- Person Learning with Crossover Protocols. College Teaching, 65(3), 97-105.

[14] Schijns, J. M. C. (2021). Measuring service quality at an online university: using PLS-SEM with archival data. 27(2), 161–185. https://doi.org/10.1007/s11233-021-09071-7

[15] Schueller, S. M., Tomasino, K. N., & Mohr, D. C. (2017). Integrating human support into behavioral intervention technologies: the efficiency model of support. Clinical Psychology: Science and Practice, 24(1), 27-45.
[16] Stephenson, J. (Ed.). (2018). Teaching & learning online: new pedagogies for new technologies. Routledge
Treude, Barzilay, &Storey (2011). How do programmers ask and answer questions on the web?: NIER track. In Proceedings of the ACM/IEEE International Conference on Software Engineering, ICSE '11, 804-807.
[17] Treude, Barzilay, & Storey (2011). How do programmers ask and answer questions on the web?: NIER track. In Proceedings of the ACM/IEEE International Conference on Software Engineering, ICSE '11, 804-807.
[18] Vasilescu, B., Filkov, V., &Serebrenik, A. (2013). StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. In Social Computing (SocialCom), 2013 International Conference on, 188-195. IEEE.
[19] Wellman, B., &Gulia, M. (2018). Net-surfers don't ride alone: Virtual communities as communities. In Networks in the global village (pp. 331-366). Routledge.