

Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

TEAM PROGRAMMING INFLUENCING FACTORS: A FIELD SURVEY

KYUNGSUB STEVE CHOI

COMPUTER INFORMATION SYSTEMS DEPARTMENT
MANHATTAN COLLEGE, RIVERDALE, NY 10471-4098

kyungsub.choi@manhattan.edu

ABSTRACT

A field survey was carried out on a group of professional programmers to locate influencing factors on team programming or pair programming productivity. From the collected responses, a variety of factors was presented. The most popular factor was “personality” and the second was “cognitive/programming style,” followed by “communication,” “programming skill,” “ability to work with others,” and other factors. There were altogether some twenty factors. The results were further broken down into a number of sub-groups: ten or more years of programming experience, less than ten years of programming experience, twelve months or more of pair programming experience, less than twelve months of pair programming experience, management, and non-management. Overall, the experienced and management groups placed more importance on human factors such as “ability to work with others,” whereas the less-experienced and non-management groups gave more weight to technical factors such as programming skills. The results connote the underlying psychosocial interactions present in the team programming environment. Overall, human factors such as “personality” were perceived to be more significant than technical factors on pair programming productivity.

Keywords: team: team programming, pair programming, extreme programming

MOTIVE

As an alternate option in software and system development, agile software development (www.agilealliance.org) [3] offers an array of unconventional tactics and tools in dealing with today’s volatile and demanding business ecosystem. Among these unconventional techniques, pair programming (PP) features an unorthodox programming style of two programmers coding a program together in real-time while sharing one monitor and keyboard [10]. The two programmers are cognitively engaged to each other, collaborating in code designing, coding, and testing. The unprecedented high levels of code efficiency, code quality, and programmer satisfaction have been well publicized [4],[10].

However, studies that focused on the programmers’ cognition, personal attributes, and the interaction between the pair in PP context have been very limited. Since programming is, after all, a human activity and a very demanding cognitive exercise, it is hard to ignore the importance of the psychosocial interaction between the programmers [9]. Hypothetically, the team programming experience and its output are perceived to reflect many underlying psychosocial interactions. To further this interest, an open field survey on a group of professional programmers is set forth.

SURVEY PREPARATION

A large pool of professional pair programmers was targeted. Searching for the widest variety of careers,

positions, experiences and industries, the Internet was used to conduct the survey. A message thread requesting participation was posted. Most online groups, under the extreme programming (XP) topic category, were named either by their geographic locations or XP applications. Some typical group names were “Chicago-agile-dev” for Agile Development group in Chicago, “xptoronto” for an XP interest group in Toronto, Canada, and “xp-at-school” for XP in classroom usage.

After some career profile questions, the main portion of the survey consisted of a list of possible PP productivity maximization success factors (or impact factors) and provisions for any additional factors that the participants might have wanted to add. The given factors were gender, programming skill, cognitive/programming style, personality, familiarity, fluency in English (communication), and pair protocol. The survey was posted on various XP online forums of portal websites and monitored daily.

Here are the brief explanations of given factors;

Personality - Pair programming is not only a programming activity, but a social activity as well [5], [11]. The general consensus is that a typical programmer is a “detached” personality type [9]. In PP where collaboration is essential, it would be interesting how such “detached” personalities interact.

Cognitive/Programming Style - Programming is perceived to reflect an individual’s unique problem solving approach [2], [6]. Given this subjective factor, the collaboration of two different or similar cognitive styles of individuals in PP context is an interesting question. However, “cognitive/programming style” is not equivalent or even similar to “personality” in this survey.

Familiarity - This addresses the comfort level that is generated by two programmers. It can be defined broadly as two programmers who have worked together for many years, and have similar backgrounds.

Fluency in English/Communication - The face-to-face communication between the pair programmers may be the single most important factor for successful collaboration.”

“Fluency in English” was chosen because the questionnaire was written in English and it was anticipated that mostly English-speaking professionals in English-speaking countries would take the survey. More important is how communication, in whatever language, is perceived to impact PP productivity.

Pair Protocol - This refers to a set of structured or semi-structured guidelines for pair programmers to use during PP. It is a list of action items that the pair adheres to besides the continuous designing, coding and reviewing. One example is how often the pair should switch being the “driver.” A few previous empirical XP studies [7], [8] suggest the need for such guidelines.

Demographics of Respondents

In total, forty-four responses were received through email or as a response messages. Due to confidentiality concerns, most responses came via email. Two responses were posted as messages. Judging by the names of the groups and email addresses, the participants seemed to be from many different parts of the world, from nearby Canada to Europe. Chart 1 provides the participants’ career profiles. The participants were mostly software engineers and developers, but a noticeable number of people working in software management also participated. The complete list of job titles was Programmer, Programmer Analyst, Vice President of IT, Chief Technical Officer, Software Developer, Consultant, Java Architect, IT Analyst, Software Engineer, Assistant Professor, Software Architect, Project Engineer, Product Developer, System Architect/ Designer, Developer, Senior Technical Manager, Team Partner, Vice President of Engineering, XP Coach, Director of Software Engineering, Director of Consulting, and Technical Consulting. In Chart 1, jobs with similar capacities are grouped together. For example, Software Architect, Project Engineer, and Programmer are placed under the software engineers and developers category. A wealth of different industries and fields were represented (Chart 2). The most common industry was software, followed by finance and consulting. The other industries were telecommunication, healthcare, automotive, education, publishing, trading, and insurance.

The most interesting attribute is probably the length of programming experience. The survey asked the respondents for the length of experiences for both programming experience in general and for pair programming experience (Chart 3).

Chart 3 shows that the respondents are very experienced in programming, with an average of fifteen years of programming and a maximum of 44.5 years. With standard deviation being 10.7 years, the programming years range from five to twenty-five years. Pair programming experience was also well established with an average of twenty-one months. The fact that the respondents had professional programming experience gave substantial validity to the survey results. The typical participant could be depicted as a software developer who works at a software development organization and has nearly two years of PP and fifteen years of professional programming experience. The gender distribution was predominantly male as only five of forty-four participants were female. Many indicated interest in the survey’s outcome, which indicates the relevance of the survey.

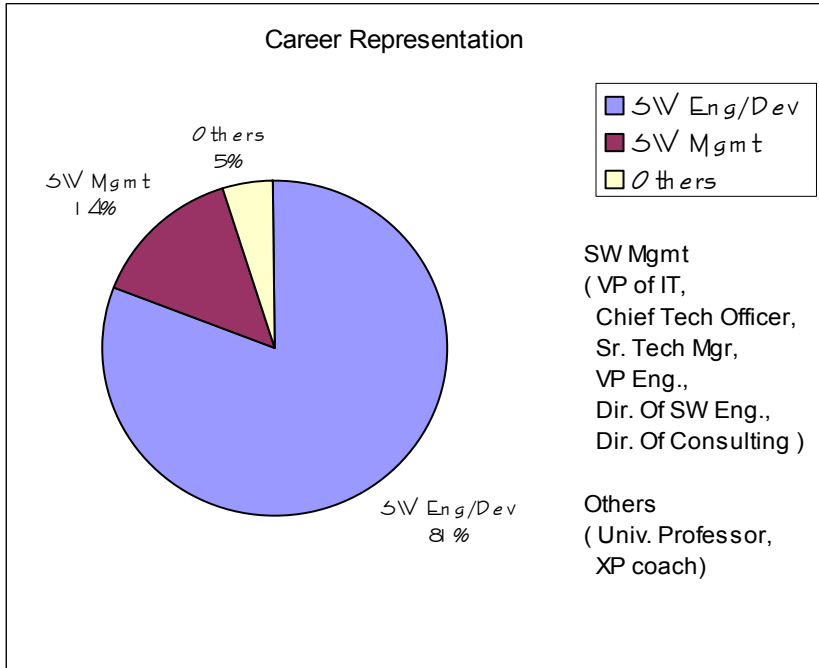


Chart 1 Career representation

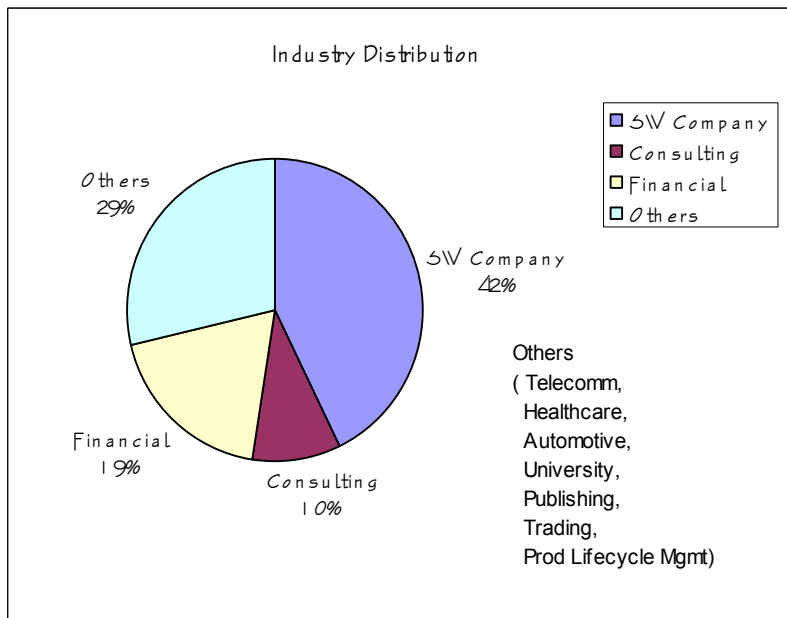


Chart 2 Demographic of Industry Distribution

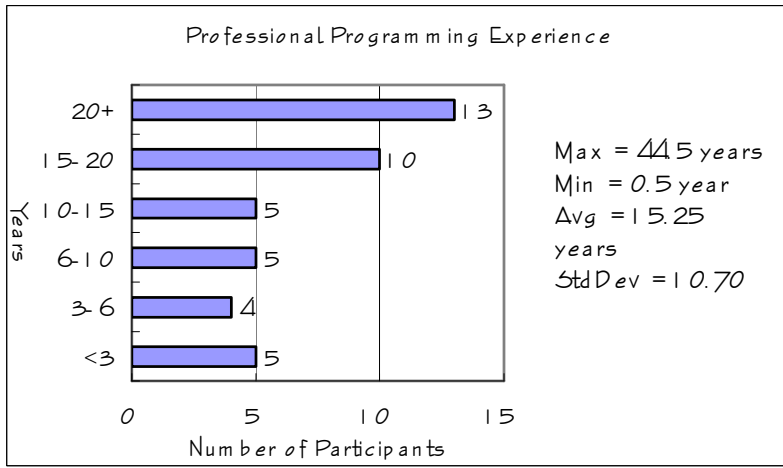


Chart 3: Professional Programming Experience.

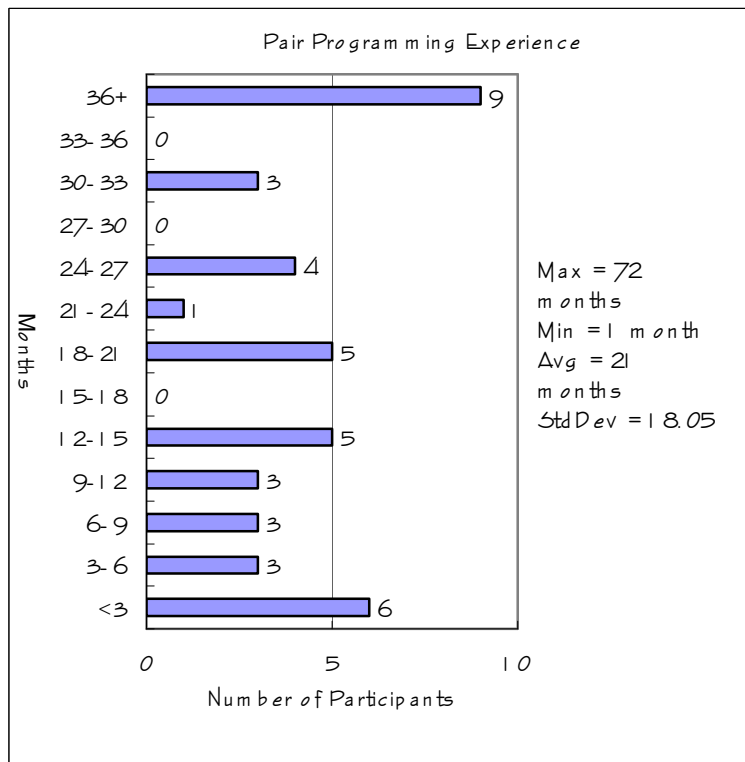


Chart 4: Pair Programming Experience

SURVEY RESULT ANALYSIS

There were some difficulties in decoding the submitted data. For example, a participant used his own ranking system by assigning “High Impact,” “Medium Impact,” and “Low Impact” to the factors instead of assigning algebraic numbers one through seven or higher. A couple of responses ranked “gender” factors with “99” and “1000” to exhibit strong disagreement. Assigning the same ranking number to two or more factors was observed and in many cases, some factors were not ranked at all.

Along with the ranking, the new factors from the participants were of great interest. A blank provision was provided where the participants could add any relevant factors. The new factors were technical knowledge, business domain knowledge, keyboard switch, ability to work with others, willingness to communicate, personal hygiene, working environment, coding conventions/platform, time constraint, availability of snacks, open to new ideas, experience, common expectation, tiredness, and desire to learn.

Chart 5 illustrates the ranking summary for each factor. Seven factors (A to G) were provided in the survey and fifteen factors (H to V) were the factors added

by the respondents. Each factor also shows the number of votes it received. The first, second, and third place votes indicate the varied perceptions of importance for each factor. For example, cognitive/programming style (C) received eight first place votes, eleven second place votes, and six third place votes.

In ranking the factors, the respondents first only had the seven given factors before them, and then they were allowed to add any new factors. A respondent’s new factors were not seen by any other respondent, and the final list of factors was different for each respondent. This process may have affected the rankings, which might be different if the final twenty-two factors had been given. Also, this may explain why the added factors received a relatively low number of top votes.

Interestingly, the participants added many psychosocial factors such as “open to new ideas”, “common expectation,” “desire to learn,” “ability to work with others,” and “willingness to communication.” They outnumbered explicit factors such as “coding convention” and “working environment.” The legend refers to how many first, second, and third place votes a factor received. Rankings below third place were treated the same as all other factors as the ranking lost its value.

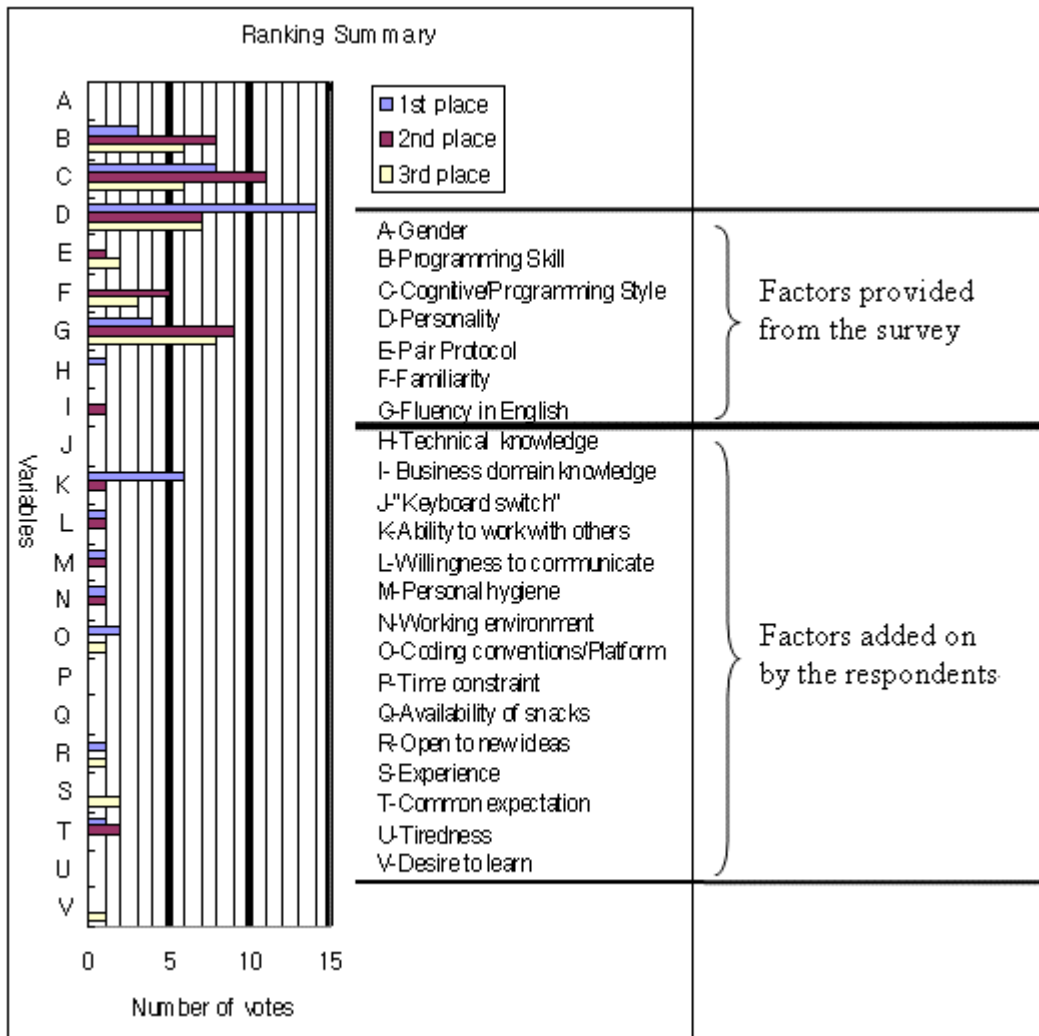


Chart 5 Ranking summary

“Personality” (D) was chosen as the most influencing factor, receiving fourteen first place votes, seven second place votes and seven third place votes. This is followed by “cognitive/programming style” (C), “fluency in English/Communication” (G), and “programming skill” (B). K, “Ability to work with others” (K) is an interesting appended factor which received six first place votes and one second place vote. In the larger picture, a close relationship between “personality” and “ability to work with others” can be assumed. There are a total twelve factors that received at least one first place vote, twelve factors that received at least one second place vote, and ten factors that received

at least one third place vote. There were five factors that did not receive any top place vote.

The popular factors that most professionals chose as the PP productivity influencing factors are mostly psychosocial factors. The overall picture of the ranking summary suggests that the issue of teamwork, involving trust, willingness, and open-mindedness, was extremely important to the participants. Rather than any controllable physical factor, the psychosocial factors involved in the effort needed to reach a goal were deemed most critical. This effort is manifested through factors such as “open-minded,” “willingness to communication,”

“ability to work with others,” “common expectation,” “desire to learn,” and “personality.”

The top choice of “personality” reinforces the view that “you have to be nice to your playmate” [10]. Every programmer has his or her own programming style and habits, and compromising and reconciling personal inclinations within PP environment is a constant challenge. The “gender” factor consistently has been voted as the least influencing factor. Surprisingly, “personal hygiene” received one second place vote and one third place vote. Working so closely for long periods of time, a certain level of courtesy is understandably important.

In a diverse participant pool such as this one, further data breakdown and analysis may uncover more hidden information. The data was broken down into two sectors of “more experienced” and “less experienced.” What constitutes “more” and “less” difficult to determine precisely as these terms are very subjective. But, for the purposes of this research and based on the profile, a group of ten or more years of programming experience was defined as “more experienced” and less than ten years as “less experienced”. Similarly, experience with twelve or more months in PP was labeled as “more experienced” and less than twelve months in PP as “less experienced.” Furthermore, four additional groupings were done through the following categories: > 10 years Prog., < 10 years Prog., > 12 mon PP, < 12 mon PP. These four additional groups were formed to observe any salencies from the association of PP and general programming experience in a relation to the PP productivity maximization.

The group with ten or more programming years placed “personality” as the most influencing factor followed by “ability to work with others,” “cognitive/programming style” and “fluency in English/communication.” The group with less than ten years experience chose “personality,” “cognitive/programming style,” “fluency in English/communication,” “ability to work with others,” and “programming skill” as significant.

In both groups, the factors “personality,” “programming skill,” “cognitive/programming style,” and “fluency in English/communication” were the most popular choices with “personality” receiving the most votes for first place. In contrast to the “less experienced” group, the “more experienced” group valued “personality” and “ability to work with others” the most.

The “more experienced” group also had more added factors and many of them received first, second or third place votes. “Ability to work with others” had five first place votes. A possible explanation of this observation is that the “more experienced” group has deeper understanding and more insights about PP, arising from the greater experience and maturity.

In the group with twelve months or more of PP experience, “personality” and “cognitive/programming style” were almost equal in perceived importance. The distribution of the votes is in parity where no one factor really stands out from the others. This group added new factors such as “willingness to communicate,” “working environment,” and “coding conventions/platform.”

In the group with less than twelve months of PP experience, the factor “personality” stood out with the most first place votes. Except for “ability to work with others,” there were no third place and higher votes for the rest of the factors. This is in contrast to the “ ≥ 12 mon. PP” group where the votes were spread out among all factors. A cautious explanation for “personality” garnering the most first place votes in the “<12 mon. PP” group may be the early learning curve of PP. The spontaneous and demanding nature PP requires one to quickly adjust to and synchronize with the partner. Dealing with a partner’s incompatible personality can be a challenge in the early stages of learning PP.

Both groups more or less replicated the summary results. Interestingly, the “more experienced” professionals valued “fluency in English/communication” over “programming skill” while the opposite is true with the “less experienced” professionals.

Table 1 displays the average ranking of each factor in its respective group. For example, for “programming skill,” the average rank value is 3.90 in the group with ten or more years of professional programming experience. Under the “responses” heading, 28 indicates the number of people in the group who have ten or more years of programming experience. “Differences” indicates the difference between the two average rank values (i.e. $-0.18 = 3.90 - 4.08$). Only six factors were considered because all factors after “fluency in English/communication” did not carry enough data points to be considered. The “gender” factor, was omitted because it consistently ranked the lowest in all conditions.

Table 1: Survey Result Statistic Figures

Groups		Prog. Skill	Cognitive/ Prog. Style	Personality	Pair Protocol	Familiarity	Fluency in English (Comm.)	Resp.
Professional prog. Exp.	≥ 10 yrs.	3.90	3.43	2.92	6.18	5.20	3.28	28
	< 10 yrs.	4.08	3.29	2.21	6.08	5.15	4.43	14
Differences		-0.18	0.14	0.71	0.10	0.05	-1.15*	
Pair Programm ing Exp.	≥ 12 mon.	4.50	3.74	3.13	6.55	5.58	3.38	27
	< 12 mon.	2.92	2.79	1.86	5.46	4.50	4.20	15
Differences		1.58**	0.95	1.27*	1.09	1.08	-0.82	
Management		3.00	3.13	2.14	5.00	4.63	2.75	9
Non-management		4.26	3.45	2.77	6.38	5.33	3.94	33
Differences		-1.26	-0.32	-0.63	-1.38*	-0.70	-1.19	

*Significant at $\alpha = 0.1$ level **: significant at $\alpha = 0.05$ level
(non-parametric test= Mann-Whitney difference test)

In the PP experience group comparison, almost all average rank differences are one full rank different. “Programming skill” shows the largest difference (1.58), which explains the difference of perspective between the two groups. A possible explanation is that, for the “more experienced” professionals, programming skill is no longer an obstacle, whereas the “less experienced” professionals are still developing their skills. The second largest difference is “personality” (0.71). As is the case for “programming skill,” the ability to deal with different personalities is a bigger challenge for the “less experienced” professionals than for the “more experienced” ones.

A comparison of “management” versus “non-management” was done to observe any differences in perspectives. “Management” was defined purely by job titles such as Vice President, Senior Director, and Chief Technical Officer. “Non-management” was defined by job titles such as software engineer, software developer, consultant, and team partner. However, all the participants in these two categories indicated that they had either practiced or currently practice PP. In the management versus non-management sectors, a one to three head ratio of management to non-management was shown. “Pair protocol” had the largest difference, followed by “programming skill” and “communication”. For “pair protocol,” one can suppose that management prefers a mode of “control and maintain” with a set of clear pair conducting protocols, but with the rankings of 5.00 and 6.38, it is hard to argue for its importance. Through these ranking comparisons one can ascertain differences in

perspective and that the differences mainly stem from the group’s characteristic.

COMMENTS BY SURVEY RESPONDENTS

Besides rank, another rich source of information was the participants’ comments. Many illustrate frontline PP experiences of programmers with subjective, vivid accounts. The comment section was provided so that the participants could add to their variable choices and also express personal views about PP and the survey in general. The comments were very informative and insightful. The personal accounts of PP experience were of immeasurable value. The comments were sorted into a few common themes: the humanistic nature in PP, PP satisfaction, and concerns for PP. All excerpts are directly quoted from the submitted comments.

As expected, many comments emphasized the human dimension of the pair interaction. Psychosocial factors such as “open to new ideas,” “desire to learn,” “ability to work with others,” and “willingness to communicate” were echoed throughout the comments.

The key parameter is to have an open mind. I have found from experience that working with beginners can be as enriching as working with experienced persons. Listening to the other person and

respecting him/her is the only two ingredients for success.

Pair programming takes patience and a desire to make it work on the part of both parties. The faster more facile one must realize that he/she will make it farther with the help of the weaker programmer, and be able to see the advantage of that, "courage."

Pairs have to understand each other and communicate well. They have to be able to "get along" with each other, but they don't have to be "buddies" or have the same culture.

I think the most important thing to do with the productivity of the pairs would have to be each pairs [sic] willingness and ability to compromise and grasp concepts that the other member of the pair is espousing. Else, there is no point.

I have worked with some real persons who think they know everything and pairing means "do it my way". I have worked with some real great people where I have learned a lot from them and I know they have learned from me.

Pair that listens to one another and are willing to try new ideas seems to achieve more. Explaining things to one another (without being patronizing) when one member of the pair doesn't understand a concept, is also helpful in

increasing knowledge for the team.

If the team has respect for each other, and enjoy working with each other they will do well. Protocols, Gender and Skills will be completely eclipsed by a willingness to share ideas, and a desire to have conversations in voice and code at the same time.

The participants may have expressed their views differently, but they are collectively voicing the need for teamwork and openness. On the other hand, the reason why so many professionals focused on these concerns may be that it is difficult to achieve the ideals of teamwork and openness.

There were also a number of comments that were convinced of the effectiveness of XP and PP. Most mentioned code quality and the knowledge transfer as the common benefits. Participants praised the reduction of time spent in tedious code debugging, as well as the opportunity for knowledge transfer between the senior programmers and junior programmers without having to set aside additional training time.

The one thing that needs to be present is an open-mindedness and a willingness to try PP the first time. After that, I think anyone who gives it an honest try can see the real value in it, and truly believes that it is better (i.e. more productive, more fun, more rewarding, and results in higher quality designs) than 2 people working separately.

XP is great, increases productivity and especially quality, distributes knowledge among team.

We had a mix of skill levels (Lisp novice to very-skilled practitioners). In our

experience the weak got strong and the strong got stronger. We put out the highest code quality and had the highest programmer productivity I have seen in 20 years of software development.

If the time is managed well, with sufficient breaks, pair programming can be very successful. The software product is at a much higher quality than it would have been because of the peer pressure. The timesaving isn't half as long as a single programmer but maybe 75% of what a single programmer would do. The biggest benefit is pairing junior programmers with senior programmer and getting the transfer of knowledge. It doesn't slow the senior down very much, but the junior gets a huge boost in understanding the problem area and programming knowledge. It reduces the time for a new programmer to be a contributor to a project.

Most responses were favorable to XP and PP, but in all fairness these comments were likely from XP proponents in XP favoring discussion groups. The most challenging part of PP may be the introduction of PP, i.e. convincing someone to try it. Breaking stereotypes and dealing with stubborn preferences for traditional programming are truly tough for any manager.

One of the critical issues with pair programming is getting people to actually start doing it. After some time, people will value it and use, but getting them to this point is the problem I'm still struggling

with this step. It might be interesting to also investigate the factors to better facilitate initial adoption of pair programming.

One of the twelve core XP principles is "Forty hour work week" [1]. Many programmers can relate to instances where he or she had to work all night to complete a programming assignment. In PP, this is not usually possible. Therefore the two programmers must be productive and maximize quality time. One by-product of the continuous cross designing, coding, and reviewing is the mental fatigue. The variables such as "time constraint" and "tiredness" reflect this situation.

Since people cannot effectively work in such high-intensity for extended periods without break, it is essential that the human needs of the pair-team individual also be paid "extreme" attention.

Pair programming can be very tiring over extended periods of time. It is very difficult to pair program 8 hours a day 5 days a week. It is very intense. When two people are concentrated working on a solution there are a lot of ideas and discussion that require a lot of mental energy. You program at a faster pace when pairing and don't take breaks when you normally would since you're normally involved programming or discussing something while you pair. You have to force yourself to take regular breaks so that you don't burn out. You can tell when you have been pairing too long because you become impatient with one another, argue more, and feel extremely tired.

It is apparent that PP brings out a high level of effort from the programmers, but it appears that management is unaware of this mental fatigue. Mental fatigue may lead to possible deterioration in code quality and productivity as well as in the pair's relationship. The consensus of the comments is that if PP is not controlled or supported by management properly, the mental fatigue will eventually lead to low morale. This dimension of PP is certainly not unavoidable. Ample attention, support, repletion, and other support are necessary.

In other domains as well as in some Information Systems, the gender variable has been one of the common discussions [7], [8]. However, the participants did not seem to find the factor "gender" significant. The following comment typifies the consensus.

I really don't have any experience pair programming with a female, but I don't know think gender should matter.

The majority of respondents were male, however, and a female professional had a noticeably different point of view.

Note I am female working in a team of males I think that this is sometimes an issue when pairing.

It is not clear how much PP experience with the opposite sex each participant had. Unfortunately, we have no conclusive data at this time to elaborate on the gender issue.

After voting "personality" the number one variable, many participants went on to give more detailed comments in support of their choice.

Personality types are extremely important. Kent Beck is describing, "Watching another person program is like watching grass die in a desert" in Extreme Programming Explained. Both programmers must be "driving".

Personality conflicts seem to be the biggest issue I've encountered during my

experiences, and can really hinder productivity.

The most impact was personality. We had one guy who did not really work well with others did not like others changing HIS code, and did not want "the spec" (user stories) to change! This is typically a self-correcting problem.

The worst thing that can happen is when you alert your partner to a mistake and he starts to defend himself.

Based on the comments and the variable results, PP seems to involve intricate social and psychological interaction within the programming context. The programming aspect plays a small role here, in the sense that the main focus is on the bonding process between the pair, rather than on skill or experience. One hypothesis is that the programmer's high satisfaction and confidence level after experiencing PP do not come from programming "savvy," but rather from a sincere, mutually accommodating relationship between the pair that enhances programming output.

LIMITATIONS OF THE SURVEY

An inherent limitation of a survey is that it presents mere opinions. Furthermore, with only forty-four programmers' opinions, it would be unreasonable to argue that the survey results substantiate the views of the entire professional programming community.

The second limitation of this survey is the initial factor list. Although participants were allowed to add more choices, most of the survey participants made their choices from the given list. This disposition may have diverted the participants from choosing "true" choices that they did not add to the list, for whatever reason. It is a reasonable assumption that, if factors had not been given, the result may have been different.

The third limitation is the low number of survey respondents. The higher the number, the more creditable the data would be. However, this survey was conducted online and was open to anyone who wanted to participate. With this open arrangement it was very difficult to attain

statistically satisfying data points. A more controlled field survey would have been preferred, but by the same token, it would have limited the diversity of the survey participants and their attributes.

In decoding the survey results, some difficulties were experienced. First, the factors were vague because they were words, not numbers, and the meaning of each factor could be interpreted differently by diverse individuals. For example, “ability to work with others” can be considered similar to “common expectation” and it is the same case with “fluency in English” and “willingness to communicate.” This phenomenon also may have affected how each respondent voted. A remedy to these deviations is to provide definitions for each given factor on the survey to minimize misinterpretation.

Another shortcoming of this survey was the factor ranking. Since the respondent did not have the opportunity to review both given and added factors by other respondents before voting, each respondent voted from a different list of factors. The list was different because each respondent had different added factors. This process very likely had an effect on the voting outcome, as the added factors had a low numbers of votes relative to the given factors.

CONCLUSION

Despite the aforementioned limitations, this survey is a step toward unveiling the underlying influences on team programming. Fascinatingly, the factors were as diverse as the views of the survey participants, from the many human factors to the technical factors. Led by the personality factor, more and more human factors were cited by the professional programmers. This underpins the view that is held by agile alliance group (www.agilealliance.org). Additionally, more senior programmers echoed this sentiment than the novice or junior programmers.

The comments – bold, precise, particular, and vivid - not only gave a view into the real-life application of PP, but also disclosed many insights drawn from personal experience. Each comment allows insights to real pair programming scenes from the programmer’s point of view. Each of these perceived factors may present another dimension to team programming. It is an area to explore and study further. For example, matching different types of personality may render different outcomes. Through optimization and manipulation on these factors, such a study can even reveal the degree of optimization.

Another significant finding of this survey is the difference in the perceived value of each factor. Depending on the individual’s level of professional skills

and programming experience, the perceived value of each factor varied distinctly among the groups. The more experienced professionals preferred human-centric factors whereas newer professionals were inclined toward technical factors. Although the accuracy and validity of these factors must stand up to scientific empirical tests, this study and future studies may eventually lead to factor optimization and maximum PP productivity.

REFERENCES

- [1] Beck, K. *Extreme Programming eXplained: embrace change*, Addison-Wesley, Reading, MA, 2000.
- [2] Cheng, M. M., Luckett, P.F., and Schulz, A. K-D. “The effects of cognitive style diversity on decision-making dyads: An Empirical Analysis in the Context of a complex task.” *Behavioral research in accounting*, volume 15, 2003. pp.39-62.
- [3] Cockburn, A. *Agile Software Development*, Addison-Wesley, Boston, MA, 2001
- [4] Cockburn, A., and Williams, L. *Extreme Programming Examined*, Addison Wesley, Boston, MA, 2001.
- [5] DeMarco, T., and Lister, T. *Peopleware: Productive Projects and Teams*. 2nd ed. Dorset House Publishing. New York, NY, 1999.
- [6] Hough, J. R. and Ogilvie, D.T. “An Empirical Test of Cognitive Style and Strategic Decision Outcomes,” *The Journal of Management Studies*. Volume42, Number 2, 2005, pp. 417-448.
- [7] Kivi, J., Haydon, D., Hayes, J., Schneider, R., and Succi, G. “Extreme programming: university team design experience”, *Electrical and Computer Engineering, 2000 Canadian Conference on*, Halifax, Nova Scotia, Canada, 2000, pp.816-820.
- [8] Muller, M.M. and Tichy, W.F. “Case study: extreme programming in a university environment”, *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, Toronto, Ontario, Canada, 2001. pp. 537-544.

- [9] Weinberg, G. M., *The Psychology of Computer Programming*, Silver anniversary ed. Dorset House Pub. New York, NY, 1998.
- [10] Williams, L, and Kessler, R. R., *Pair programming illuminated*, Addison-Wesley, Boston, MA, 2002.
- [11] Yourdon, E., *Death March*, Prentice-Hall, Upper Saddle River, NJ, 1997.

AUTHOR BIOGRAPHY

Kyungsub S. Choi is Assistant Professor in the Department of Computer Information Systems, School of Business, Manhattan College. His research interests are agile software process paradigms, human-computer interactions, and system quality control. Before turning into academia, he spent over ten professional years in major pharmaceutical companies in various capacities such as computer systems specialist and systems quality manager. He holds BA in Chemistry and Economics from Rutgers College, New Brunswick and BS and MS in Management Information Systems and Ph.D. in Information Systems from New Jersey Institute of Technology.