THE DEVELOPMENT OF A LOW-OVERHEAD ASSESSMENT METHOD FOR IRISH SOFTWARE SMES

FERGAL MC CAFFERY
LERO- THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE,
UNIVERSITY OF LIMERICK
Fergal.mcCaffery@dkit.ie

GERRY COLEMAN
DUNDALE INSTITUTE OF TECHNOLOGY,
gerry.coleman@dkit.ie

ABSTRACT
This paper describes the development of ‘Adept’, a low-overhead method of software process appraisal specifically targeted at Irish software small-to-medium-sized enterprises (SMEs). The method explicitly focuses on organisations that have little or no experience of software process improvement (SPI) programmes. Historically, it has been difficult for software SMEs to find the resources, in both time and money, necessary to engage properly in SPI. To alleviate this, we have created a low-overhead and relatively non-invasive solution to support SMEs in establishing process improvement initiatives. The paper initially describes how Adept was developed and then illustrates how the method is currently being extended to include an on-line tool that may be used by the appraised organization to perform follow-on self-assessments.

Keywords: Software Process Improvement (SPI), Self-Assessment, Management, SMEs, Software Process Assessment Method.

INTRODUCTION
The software industry in the Republic of Ireland is a key component of the national economy. According to Enterprise Ireland (the Republic’s economic development agency for indigenous business) at the end of 2004, Irish-owned software businesses comprised over 750 companies employing almost 12,000 people [6]. The great majority of these Irish-owned software firms are Small to medium-sized enterprises (SMEs) where it is calculated that only 1.9% of these firms employ more than 100 people whilst more than 60% of the total employ 10 or fewer [5].

During the late nineties the SPIRE (Software Process Improvement in Regions of Europe) programme applied the SPICE model to a variety of SMEs [13]. SPIRE case studies reported positive experiences within Irish SMEs who liked the ability to choose SPICE process areas that were directly related to their business goals. However, research indicates that only a small percentage of Irish indigenous software development companies have implemented formal SPI assessment methods [11]. One such study emphasises that indigenous Irish software companies are reluctant to engage in formal SPI assessments because of the high cost and resources involved [4].

The picture of the software industry in Northern Ireland is similar. The findings of a survey, carried out by the Centre for Software Process Technologies (CSPT) showed that 71% of the software companies in Northern Ireland are indigenous with two-thirds of those employing fewer than 20 people [11]. The CSPT survey captured information from fifty-six software development organisations within Northern Ireland. These organisations together employ approximately 80% of the total software engineers in the region. Significantly, 46% felt that formal
SPA methods were too cumbersome and expensive, and favoured a less costly (both in terms of employee resource and finance) method, such as Class C CMMI® [10] appraisal methods.

In terms of ensuring the competitiveness and success of the indigenous software industry of the entire island of Ireland, both EI and Momentum (the Northern Ireland ICT federation) are currently engaging member companies in SPI initiatives. The Adept assessment method [9] was developed as a result of a meeting between the authors and EI in relation to how a culture of SPI could be instilled into Irish software development organisations. Initially, therefore, it was important to understand the current state of software practice within these companies and from this an assessment method was developed that could diagnose any weaknesses in a company’s software process. This diagnosis would then form the foundation from which to base a path for continuous SPI that would make the company more effective in fulfilling its business goals.

EXISTING SOFTWARE PROCESS ASSESSMENT METHODS

In order to provide guidance to our development of an assessment method that would be suitable for the Irish software industry we looked at some preliminary results from the Irish market [4] and performed a literature review of software process assessment methods used within other regions. We discovered that such assessment methods are generally based upon one of two process models: (i) ISO/IEC 15504 [7] and (ii) CMMI® [3].

We decided that the Adept method should be designed to adhere to 8 of the 10 criteria outlined by Anacleto et al. [1], for the development of lightweight assessment methods. Therefore the following criteria are adhered to by the Adept method: low cost, detailed description of the assessment process, guidance for process selection, detailed definition of the assessment model, support for identification of risks and improvement suggestions, conformity with ISO/IEC 15504, no specific software engineering knowledge required from companies’ representatives, and tool support is provided. The exceptions being that no support is provided for high-level process modelling and the method is not made publicly available.

REQUIREMENTS FOR THE ADEPT ASSESSMENT METHOD

From investigating other assessment methods, and the experiences gained by one of the authors with using the EPA method, Adept was designed to adhere to 8 of the 10 criteria outlined by Anacleto et al. [1], the exceptions being, Support for high-level process modelling and Public availability. However, Enterprise Ireland (representing the Irish SMEs), requested that Adept also take into account the following factors:

- Improvement is more important than certification and a rating is not required;
- Minimal preparation time should be required by the company;
- The assessment should be performed over a short period of time;
- Companies should be able to select assessment in process areas that are most relevant to their business goals;
- Whilst the assessment will be based upon both the CMMI® and ISO/IEC 15504 models, the SPI models should be invisible to the SMEs that are being assessed.

THE ADEPT METHOD

Though based largely on the structure of the EPA, Adept also supports the following:

- The development of a SPI path, based upon a company’s business goals, from the findings report that is produced as a result of the assessment;
- The inclusion of a stage that involves revisiting the company after a period of 3 months and re-assesses the company’s SPI path;
- A generic approach. It does not highlight either CMMI® or the ISO/IEC:15504 SPI models but rather refers to general SPI, thus focusing on improvement rather than certification;
- The incorporation of Agile methodologies as possible SPI recommendations;
- Reduced resource requirements. For example, Stage 1 (Develop Appraisal Schedule) and stage 3 (Conduct site Briefing) of the EPA method, though both covered in Adept, require only half the time to perform.
Selecting an SPI Model

The main aim of the Adept method is to encourage SPI based upon the generic SPI principles that are shared by both CMMI® and ISO/IEC:15504. So to progress Adept we had to decide whether to:

a. Develop a completely new model that would contain new process areas based upon input from both the CMMI® and the ISO/IEC:15504 models;

b. Base the Adept method upon relevant process areas from the CMMI® model and include input from the ISO/IEC:15504 model;

c. Base the Adept method upon relevant process areas from the ISO/IEC:15504 model and include input from the CMMI® model.

As Irish software development companies have greater awareness of the CMMI® model [11], option (b) was preferred to option (c). Additionally, option (a) was ruled out due to the effort that would be involved in developing both a new assessment model and an assessment method. Therefore Adept consists of an assessment component for each CMMI® process area that is deemed applicable for Irish SMEs. However, even though each assessment component adopts a CMMI® process area name, it will provide equal coverage of both the CMMI® and ISO/IEC 15504 models by containing questions that relate both models.

What Process Areas should the Adept Method Assess?

The next important key decision in the development of Adept was to decide what process areas are most applicable to the Irish software SMEs. Based on previous research into software processes with Irish SMEs [2,4,12] and the involvement in EPA, we decided to investigate the potential benefits to software SMEs of each of the process areas within the CMMI® model.

What CMMI Maturity Level 2 Processes should be Included?

Maturity level 2 of the CMMI consists of 7 process areas. Upon investigation, six of the seven process areas were selected as they constitute the engineering management basis of an organisation and the foundation upon which an efficient software company is based [14]. On this basis, Adept includes assessment components for each of the following CMMI® level 2 process areas: Requirements Management; Configuration Management; Project Planning; Project Monitoring & Control; Measurement & Analysis; Process & Product Quality Assurance. We omitted the seventh CMMI® process area at maturity level 2 (Supplier Agreement Management), as previous research [14] indicated that it would not be as beneficial, as other process areas, to Irish SMEs. Therefore, Adept does not attempt to provide any form of rating.

What Higher Level CMMI Processes should be Included?

Maturity level 3 of the CMMI consists of 14 process areas. Upon investigation [4,10,14], six of the fourteen process areas, Requirements Development; Technical Solution; Product Integration; Verification; Validation; Risk Management were deemed applicable for Irish software SMEs and an assessment component for each was included. The process areas listed at CMMI® maturity levels 4 and 5 would be of less benefit to companies that have little or no experience in SPI and therefore an assessment component for these was not required. Therefore, in total, Adept will enable an assessment to be performed in 12 process areas.

Should assessment in certain process areas be given priority?

While all 12 process areas may be assessed using Adept, four will be mandatory - Requirements Management; Configuration Management; Project Planning; Project Monitoring & Control. These process areas are critical to the success of any software development company. The choice of mandatory process areas was based upon the overlap of three factors. Firstly, priority was given to the process areas that are deemed to be the foundation of the CMMI® model. Second, priority was assigned to the process areas in which SMEs would gain most benefit [14]. Thirdly, research in Ireland has shown that these specific processes are seen as important by software SME managers [2,4,12].

How many process areas should be assessed within a single Adept appraisal?

In an attempt to reduce the cost and time associated with the assessment, on-site interviewing should be restricted to one day. As such, we decided to limit an Adept assessment to six process areas as this is as many as can reasonably be covered within one day [14]. So, in addition to being assessed in the four mandatory process areas, companies will also be able to choose two of the other process areas. Based upon previous research into the
applicability of process areas to software SMEs [14], companies will be advised against initially selecting either the Measurement & Analysis or Process & Product Quality Assurance process areas unless they are directly linked to their business goals.

The Stages of the Adept Method

Adept is divided into eight stages. The appraisal team consists of two assessors who conduct the appraisal between them.

Stage 1 (Develop Appraisal Schedule and Receive Site Briefing) is a preliminary meeting between the appraisal team and the software company wishing to undergo an SPI assessment. This stage consists of two parts. The first part involves establishing the logistics, selecting the most applicable process areas and determining the schedule of the appraisal. The second part is used by the appraised organisation to explain elements of the company structures to the appraisal team, who learn a little about the company’s history, the company’s business objectives and about the types of ongoing projects, along with the lifecycle stage that each project has reached. This meeting involves 2 assessors and at least one representative from the company. This meeting lasts approximately two hours. Therefore 4 person-hours of assessor time and at least 2 person-hours of company time are normally required for this stage.

During stage 2 (Conduct Overview Briefing) the lead assessor provides an overview of the method for members of the appraised organisation who will be involved in subsequent stages. This session is used to remove any concerns that individuals may have and to establish codes of conduct and confidentiality. This overview session involves 2 assessors and on average 7 company staff (the number of company staff involved depends upon the size of the company). The overview typically lasts 1 hour. Therefore 2 person-hours of assessor time and 7 person-hours of company time are required for this stage.

Stage 3 (Analyse Software Documentation) provides a brief insight into project documentation. Normally the following documents will be requested: a typical project plan, a typical project progress report, a typical approved requirements statement and any documentation relating to the company policy on configuration management. The primary source of data for Adept is through a series of process area interviews conducted during stage 4. The brief consideration of some sample documents during stage 3 is used mainly to craft further questions for stage 4. This stage will involve 2 assessors and usually 1 member of personnel from the appraised organisation. Typically, this stage will involve the company member dedicating 1 hour to retrieving the requested documents. The 2 assessors performing the appraisal will each then analyse this data for approximately 3 hours. Therefore 6 person-hours of assessor time and 1 person-hour of company time are required for this stage.

The main part of Adept is stage 4 (Conduct Process Area Interviews). In this stage 6 interviews take place with key staff members from the appraised organisation. Each interview is scheduled to last approximately 1 hour. However, based upon the experiences with the EPA in Northern Ireland, interviews for the process areas of project planning and project monitoring and control typically require 1.5 hours. Therefore 7 hours is required to complete the 6 process area interviews. Each interview involves two assessors, and at least one representative from the company (on average 3 staff are involved) is present for each process area interview making 14 person-hours of assessor time and 21 person-hours of company time typical for this stage. The schedule of the process area interviews should be carefully designed to follow the natural sequence of the software development lifecycle. This will assist the assessors in painting an accurate profile of the software development practices adopted by a company and responses from process area interviewees, earlier in the day, may be cross-referenced in later interviews. During each of the process area interviews, one of the assessors invokes responses from the interviewees using a combination of pre-defined and follow-up questions while the other assessor makes notes. One of the assessors use a tool which enables him/her to record an initial judgement about the responses by judging them against a discrete set of values – Red (not practiced), Amber (partially practiced), Yellow (largely practiced) and Green (fully practiced). In this way, the opinions of the questioner and not just the note-taker will also be recorded for subsequent review.

Stage 5 (Generate Appraisal Results and Create the Findings Report) is very much a collaborative exercise between the two assessors. The findings report will consist of a list of strengths, issues and suggested actions for each of the process areas evaluated. Global observations covering all process areas are also covered and the initial judgements recorded in the Excel tool are revised. The findings report is then developed through a review of the interview notes and the scores produced by the Adept tool for each of the 6 assessed process areas. The findings report takes the format of a Microsoft PowerPoint presentation. This stage involves 2 assessors collaborating together for six hours meaning a total 12 person-hours of assessor time is required for both these tasks.
Stage 6 (Deliver the Findings Report) involves presenting the findings report to the staff in the appraised organisation that participated in the interviews. This presentation involves the assessors and typically 7 company staff (this depends upon the number of the appraisal participants). The briefing normally lasts 1 hour. Therefore 2 person-hours of assessor time and 7 person-hours of company time are required for this stage.

Stage 7 (Develop a SPI Path with the Company) involves collaborating with staff from the appraised company to develop a roadmap that will provide guidance to the appraised company in relation to practices that will provide the greatest benefit in terms of the company’s business goals. This stage involves 2 assessors and one member of the appraised organisation working together for 4 hours, thus requiring 8 person-hours of assessor time and 4 person-hours of company time.

Stage 8 (Re-assess the SPI Path and Produce a Final Report) involves revisiting the appraised company approximately 3 months after the completion of stage 7 and reviewing progress against the SPI path that was developed in stage 7. The outcome of this stage will be an updated SPI path and a final report detailing the progress that has been accomplished along with additional recommendations. This stage will involve the 2 assessors and one member of the appraised organisation working together for 3 hours. Additionally the 2 assessors will dedicate a further 2 hours to producing a final report in relation to the assessment. Therefore 10 person-hours of assessor time and 6 person-hours of company time will normally be required for this stage. This stage is crucial, as it provides feedback and assistance to the appraised company after a period of time, and assists in compiling research material in terms of SPI experiences.

Table 1 provides a summary of the effort required to complete each stage of the Adept method. Overall, Adept requires approximately 56 person-hours of assessor time and 47 person-hours of the appraised organisation’s time. Ideally stages 1 to 7 of the appraisal process are completed over two elapsed weeks, with stage 8 happening approximately 3 months later.

### Table 1: Effort Involved in Conducting an Adept Assessment

<table>
<thead>
<tr>
<th>Stage</th>
<th>Assessment team (person-hours)</th>
<th>Appraised Organisation (person-hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>7&lt;sup&gt;1&lt;/sup&gt; (estimate)</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>2&lt;sup&gt;1&lt;/sup&gt; (estimate)</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>7&lt;sup&gt;3&lt;/sup&gt; (estimate)</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td><strong>Total Effort</strong></td>
<td><strong>56</strong></td>
<td><strong>47</strong></td>
</tr>
</tbody>
</table>

### EXTENDING ADEPT TO ENCOURAGE CONTINUOUS SPI

Adept assessments have recently been performed in several Irish SMEs. Companies particularly liked the fact that no preparation time was required by them prior to the assessment. Some companies mentioned that they would like to be able to engage in continuous process appraisal and that it would therefore be useful if they could measure their progress periodically. Therefore in an attempt to encourage continuous improvement among SMEs we have developed an on-line appraisal tool to enable companies to perform self-assessment of process areas that have previously been assessed using the Adept method.

This tool enables SMEs to self-assess their processes to determine if the capabilities of their software processes have improved as a result of implementing the actions suggested by the Adept assessment team during the initial assessment. The tool is only available to organisations that have previously had their processes assessed using Adept. As the tool is web-based it is easily distributed to companies and eligible companies may access the tool upon demand. This facility enables company personnel to perform a process self-assessment in an informal, flexible manner with minimal preparation time. Using the tool, performing the self-appraisal, for all 6 process areas that were originally appraised in the on-site assessment,

1. Assuming 7 staff for 1 hours.
2. Assuming average of 3 staff per process area interview.
3. Assuming 7 staff for 1 hours.
requires no external assessor time and only a total estimated time of 6 hours from a company employee who attended the initial on-site Adept assessment.

Whilst a company may access the full range of questions that were used in the Adept method by the appraisal team, the company may only access those process areas in which they were previously appraised. The questions within each process area have multiple-choice answers. These range of answers are expressed in terms that do not require an understanding of SPI models (see figure 1). However, a facility is also provided that permits users to enter a free text answer if they feel that none of the options fully equates to their required response. This response is then automatically emailed to a member of the appraisal team who will make a judgement and provide feedback. The tool also enables an appraisal to be performed on one process area, in isolation from other process areas. A process appraisal does not have to be completed in a single session; it may be partially saved and then continued in later sessions.

Records of each appraisal are stored and used to monitor SPI over time. This information is also accessible to the assessment team so that empirical SPI information may be compiled.

**Features of the Adept self-assessment tool**

The Adept Self-Assessment tool, which adheres to the Adept reference model structure, consists of a five main parts: User Accounts; Process Area Management; Question Management; Perform Process Appraisal; Appraisal Assistant.

The first 3 sections: User Accounts, Process Area Management and Question Management are only visible to the assessment team. The only sections that may be accessed by company personnel are Perform Process Appraisal and Appraisal Assistant. The User Accounts section provides management features for: user registration; checking that the user is authorized; resetting user passwords.
passwords; adding user accounts; deleting existing user accounts; amending user accounts; and exiting the appraisal.

The Process Area Management section facilitates: the addition of new process areas; the addition of goals/practices within process areas; the amendment of existing process areas; the amendment of existing goals/practices; the deletion of existing process areas and the deletion of existing goals/practices.

The Question Management section enables: the addition of new process area questions to practices; the amendment of existing questions; and the deletion of existing questions.

The Perform Process Appraisal section enables: the user to access each process area that their company has previously been assessed in by the Adept assessment team; a user to perform the self-appraisal of one or more process areas; the assessment results for a process area to be viewed in either textual or graphical format (after the appraisal is completed); the results of the appraisal to be time-stamped; a copy of the results to be automatically stored for future reference by the assessment team; a user to partially answer process area questions in one session, then save their work and resume during a subsequent session; several users to make separate evaluations of process areas, enabling consistency checking to be performed and thereby increasing confidence levels.

CONCLUSIONS

The Irish software industry contains many software SMEs which are driven by entrepreneurs and which often lack a quality culture. Research [4,8] has shown that software SMEs are typically not aware of SPI models or initiatives. In such an environment it is very difficult for software organisations to appreciate the global importance of having effective software processes. Part of the problem is one of education where software development managers fail both to understand how to improve their business, and to appreciate their company’s technical performance with regard to international standards. To combat this requires an appropriate approach that facilitates education and initiates the engagement of software managers in a quality agenda.

Adept has been developed to assess software processes within Irish software SMEs based upon information that has been obtained from four different sources: (a) by reflecting upon the effectiveness of the EPA method to assess software processes in SMEs within Northern Ireland; (b) through investigating the characteristics of other lightweight assessment methods; (c) from the outcome of a meeting between researchers from DkIT, Lero and Enterprise Ireland that discussed how a culture of SPI could be instilled into Irish software SMEs; and (d) through research that has been performed by Lero and DkIT in relation to Irish software SMEs. The method is designed as a low-resource assessment model for SMEs that have very little experience of SPI and can therefore help raise the level of SPI education within Irish software companies.

The method relies heavily on information obtained from interviewing company personnel and performs limited cross-referencing checks (due to the limited time available for data collection and analysis). As a result, this approach depends on the willingness of the company to engage in SPI. It is therefore vital that senior management within the company encourage their employees to answer interview questions in a truthful and supportive manner so that the resultant findings report will provide an accurate reflection of the company’s strengths and weaknesses within each of the appraised process areas. As the findings report will contain a list of recommendations we strongly advise the companies concerned to actively engage and collaborate with the assessors to prioritise these recommendations into an action plan based upon the company’s business goals and aspirations. We have observed that companies disliked appraisal in a complete set of process areas as they felt this may highlight multiple weaknesses which could demoralise staff and severely hinder any SPI effort. Consequently companies have indicated that they preferred to have an appraisal performed using a balanced mixture of well-performed process areas and less efficient process areas.

We are currently performing a series of software process assessments in Irish SMEs using the Adept method. The early assessments indicated that companies liked the idea of using a tool to periodically re-evaluate their processes. We have now developed that tool and look forward to incorporating it into the appraisal process.

REFERENCES


THE DEVELOPMENT OF A LOW-OVERHEAD ASSESSMENT METHOD FOR IRISH SOFTWARE SMES


ACKNOWLEDGEMENTS

This research is supported by the Science Foundation Ireland funded project, Global Software Development in Small to Medium Sized Enterprises (GSD for SMEs) as part of Lero - the Irish Software Engineering Research Centre (http://www.lero.ie).

AUTHOR BIOGRAPHIES

Dr. Fergal Mc Caffery is a Senior Research Fellow in the GSD for SMEs project, which is affiliated with Lero—the Irish Software Engineering Research Centre (www.lero.ie), University of Limerick. He is a member of the programme committee for several leading software engineering journals and conferences. His research focuses on software process improvement models and methods. He received his PhD in Intelligent Adaptive Multi-media Interfaces from the University of Ulster. Contact him at the Regional Development Centre, Dundalk Inst. of Technology, Dundalk, Co. Louth, Ireland.

Dr. Gerry Coleman is a member of the Department of Computing and Mathematics at Dundalk Institute of Technology. His research focuses on software processes in small software companies. He received his PhD from Dublin City University. Contact him at the Dept. of Computing and Mathematics, Dundalk Inst. of Technology, Dundalk, Co. Louth, Ireland.
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, “xpitoronto” 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 be 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 communicate.” 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.
“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 communicate,”
“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 saliencies 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 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.
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 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.

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.

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 avoidable. 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


**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.