Programming Skill, Knowledge and Working Memory Among Software Developers from an Investment Theory Perspective

Gunnar Rye Bergersen

Simula Research Laboratory

University of Oslo

Jan-Eric Gustafsson

University of Gothenburg

Abstract

This study investigates the role of working memory and experience in the development of programming knowledge and programming skill. An instrument for assessing programming skill where skill is inferred from programming performance was administered along with tests of working memory and programming knowledge. We hired 65 professional software developers from nine companies in eight European countries to participate in a two-day study. Results indicate that the effect of working memory and experience on programming skill is mediated through programming knowledge. Programming knowledge was further found to explain individual differences in programming skill to a large extent. The overall findings support Cattell's investment theory. This work contributes to research on individual differences in semi-realistic work settings where professionals are used as subjects.

PROGRAMMING SKILL, KNOWLEDGE AND WORKING MEMORY AMONG SOFTWARE DEVELOPERS FROM AN INVESTMENT THEORY PERSPECTIVE

Software systems are a cornerstone of modern society. Software production is a highly competitive and globalized industry that is focused on producing high quality software at low cost. One important way to stay competitive is to recruit and retain highly productive software developers. Although companies often use different methods for quantifying competence when recruiting developers, tests of cognitive abilities are frequently employed.

Generally, cognitive abilities can be organized into a hierarchical structure (Gustafsson, 2002). At the apex, general mental ability (*g*) exerts influence over all lower factors. At the next stratum are a handful of broad abilities such as crystallized *g* (*Gc*, acquired knowledge) and fluid *g* (*Gf*, novel and abstract problem solving), while the lowest stratum specifies a large number of narrow abilities. Gustafsson (1984) demonstrated that *Gf* and *g* are perfectly correlated, and Valentin Kvist and Gustafsson (2008) showed that this perfect relation holds true only for homogeneous populations in which the individuals have had reasonably similar opportunities to acquire the knowledge and skills tested.

According to Cattell's investment theory (1971/1987), the acquisition of knowledge and skills and the formation of developed abilities (i.e., *Gc*) are influenced by *Gf*, effort, motivation and interest, and also by previous levels of *Gc* (Valentin Kvist and Gustafsson, 2008). Because *Gf* is involved in all new learning, this ability is identical with *g*. *Gf* (and *g*) also has a wider breadth of influence than other factors of intelligence, but it does not necessarily exert a strong influence on performance on any single task (Humphreys, 1962; Coan, 1964; Gustafsson, 2002; Valentin Kvist & Gustafsson, 2008).

Based on the investment theory, as well as later extensions to this theory (see Ackerman, 2000), we expect that the influence of *Gf* and experience on skills as well as job performance to be mediated through knowledge. That a mediating relationship exists is supported by theories of skilled behaviour and skill acquisition (Neves & Anderson, 1981; Anderson, Conrad, & Corbett, 1989), in which knowledge is a key component. For example, according to the three phases of skill acquisition proposed by Fitts and Posner (1967), declarative knowledge is central to the first cognitive phase where an individual "tries to 'understand' the task and what it demands" (p. 11). Anderson (1987) also states that knowledge initially "comes in declarative form and is used by weak methods to generate solutions [which] ... form new productions ... [and a] key step is the knowledge compilation process, which produces the domain specific skill" (p. 187). Further, knowledge is also a central component of adult intelligence (Ackerman, 2000) and can therefore be an important factor in the acquisition of new knowledge and skills (Ackerman, 2007).

The investment theory also is consonant with earlier reported results on job performance. For example, in a meta-analysis by Schmidt, Hunter & Outerbridge (1986), the strongest determinant of job sample performance was job knowledge (.74), much higher than the direct path from experience (.08) or general mental ability (.04). Further, Schmidt (2002) states that individuals with high general cognitive ability acquire more job knowledge and acquire it faster, which leads to "higher levels of job performance" (p. 201). Also, in another meta-analysis, experience and job performance were found to be positively correlated, even though this relationship diminished over time (McDaniel, Schmidt & Hunter, 1988).

Constructs close to the apex of the hierarchical model can be described as having high referent generality, and constructs that are highly specific to a limited situation have low referent generality (see, e.g., Coan, 1964). For example, programming skill in a single programming

language can be regarded as a narrow construct with low referent generality. It is well suited for predicting the outcomes of an individual for a specific programming language, but there might be limited transfer of knowledge and skills to, for example, other kinds of programming languages. When assessing constructs with low referent generality, it is therefore important to also assess constructs with high referent generality (Gustafsson, 2002).

Working memory is a construct that has a close relationship to *Gf* and *g* and can further be regarded as a construct with high referent generality. Although this relationship is complex (Ackerman, Beier, & Boyle, 2002), several researchers have reported a large degree of overlap between working memory and *g* (Ackerman et al., 2002; Colom, Rebollo, Palacious, Juan-Espinosa, & Kyllonen, 2004; Unsworth, Heitz, Schrock, & Engle, 2005). Working memory also has a substantial overlap with perceptual speed (Kyllonen & Stephens, 1990; Ackerman et al., 2002). In addition to these overlaps, limitations of working memory, which are revealed in theories of skilled behaviour, are an important source of error in skilled performance (Anderson, 1987). We maintain, therefore, that working memory is a suitable high referent generality construct for the purpose of assessing programming skills both from a *Gf* perspective and from a procedural learning perspective (Kyllonen & Stephens, 1990).

Computer programming is sometimes described as one of several archetypes of complex cognitive behaviour; overall, programming requires the programmer to have a high level of declarative knowledge as well as much practice to perform well. Working memory has previously been found to be a good predictor of programming skill acquisition (Shute, 1991), as has experience (Arisholm & Sjøberg, 2004). It has also been noted that performance on tasks operating under skill constraints can be good predictors of other tasks as well. For example, for programming in the LISP programming language, Anderson (1987) states that “the best predictor

of individual subject differences in errors on problems that involved one LISP concept was number of errors on other problems that involved different concepts" (p. 203). Further, amount of errors was also correlated with amount of programming experience (see Anderson et al., 1989, for further details).

Skill as a latent construct is inferred from observed performance that varies in both time and accuracy (or quality) (Fitts & Posner, 1967; Anderson, 1987). Together with knowledge and motivation, skill is one of three direct antecedents of performance (Campbell, McCloy, Oppler, & Sager, 1993). Skill is sometimes also referred to as procedural knowledge, and its acquisition consists of three overlapping phases. During each phase, different abilities (as antecedents) are hypothesized to exert different levels of influence on the acquisition of skill (Ackerman, 1988): General mental ability is predominant during the first cognitive phase, perceptual speed during the second associative phase, and psychomotor ability during the final and autonomous phase. Kyllonen and Woltz (1989) refer to the first phase as the "knowledge acquisition phase," while phase two and three are skill acquisition and skill refinement, respectively.

Much research on the prediction of job performance has relied on a construct of general mental ability indexed by test batteries which measure a mixture of *Gf* and *Gc* (Valentin Kvist & Gustafsson, 2008). To distinguish this conception of general mental ability from the apex factor *g*, we will refer to it as *G*. A comprehensive meta-analysis by Schmidt & Hunter (1998) showed *G* to be a strong predictor of job performance, but they also showed that the predictive validity of work sample tests exceeds that of *G*. For the field computer programming, this would not come as surprise: if expert performance in programming develops in the same manner as expertise in other fields (see, e.g., Ericsson & Charness, 1994), an employer should certainly not choose an

individual with high *G* and no programming experience over an individual with average *G* and 10 years of programming experience.

Although research has been conducted on group differences in the *acquisition* of programming skills, individual differences in already *acquired* skills have not been adequately quantified. One reason for the lack of research on the quantification of skills is that the construct validity of programming performance is currently unresolved (Hannay, Arisholm, Engvik, & Sjøberg, in press); indeed, the construct validity of work samples in general seems unresolved (Campbell, 1990). Programming usually operates under a time-quality trade-off, making any single score assigned to a solution an aggregate trade-off where the relative weight of the quality of the solutions becomes important with respect to the time used to obtain that solution. In addition, to what degree observed programming performance on one or several programming tasks can be generalized to other tasks, systems, people or countries is uncertain.

The purpose of this study is to investigate the relationship between programming skill and its main antecedents, using Cattell's investment theory (1971/1987) as a conceptual framework. The study extends previous research in two ways. First, work samples for a specific programming job have been substituted by a latent variable representing programming skill. In this study, inferences regarding individual programmers' standing on the skill construct are inferred from programming tasks performance operating under skilled constraints (i.e., a high quality solution obtained with little time spent indicates a high level of skill). Second, the instrument we propose is unidimensional; task performance for all the tasks in the instrument are good predictors of each other when task difficulty is accounted for, even though all tasks are very different from each other.

We predict that, in line with skilled behaviour theory, investment theory, and previous research on work samples, programming knowledge is the main causal antecedent of programming skill. Further, programming knowledge is expected to mediate the relationship between programming skill and the causal variables of working memory and experience.

Method

Participants

Sixty-five professional software developers were hired from nine companies located in eight Central/Eastern European countries: Belarus, Czech Republic, Italy, Lithuania, Moldova, Norway, Poland and Russia. Each participant received their base salary for participation, but they were not compensated beyond this. From each company we requested voluntary participation, and all companies and subjects were guaranteed anonymity. Subjects were free to terminate participation at any time without loss of compensation. Voluntary participation was difficult to guarantee, however; negotiations were conducted with company project leaders, not the subjects themselves. All developers were required to be proficient in English and to have at least six months of consecutive programming experience in the Java programming language prior to study participation.

The mean age of participants was 28 years (SD = 5.66, range = 21–53) with an average professional working experience of 5.3 years (SD = 4.94, range 0.3–30). Of this group, 63.1% had a master's degree, 33.8% a bachelor's degree, and 3.1% a high school degree; 10.8 % were female.

Tests administered

All materials and instructions were in English, which is the de facto business language of software developers working in the global IT industry. Some minor language problems occurred during the study for a few of the subjects.

Java programming skill

As a proxy for a job sample test of programming performance, we used an instrument for assessing programming skill (Bergersen, to be submitted). The instrument contains 12 programming tasks (items) in the Java programming language, which has become one of the most common programming languages during the last decade. All tasks require either implementation or modification of programming code and the duration of each task is between 10 and 50 minutes. Between 5 and 10 minutes was also allowed for reading the task instructions before code was downloaded.

Performance on each task was scored as an aggregate of both the time required to implement a correct solution and the quality of the solution. Starting from incorrect tasks (or tasks submitted too late), increasingly higher (ordinal) scores were assigned to more correct solutions. Further, even higher scores were given to tasks of acceptable quality, but with less time used. The instrument consists of both automatically and semi-automatically scored tasks, as some programming quality aspects can only be evaluated by humans. The scoring scheme for these kinds of tasks used objective criteria (such as, “is functionality x present for attribute y?” 0 = no, 1 = partially, 2 = yes). Some items were testlets, implying that task requirements were to be solved in consecutive steps until time ran out.

For subject scaling, task scores were fitted to the polytomous Rasch model (Andrich, 1978). This is a generalization of the Rasch model (1960) where ordinal scores are transformed to interval scale measurements, provided certain conditions are met.

Programming knowledge

A 30-item multiple choice knowledge test of Java programming was purchased from a large international (anonymous) test vendor. Items were selected from a large pool of existing items that are in current use for assessing software professionals globally. All items were specifically selected to cover the same content domain that was present in the programming skill instrument. The knowledge test imposes a three-minute time limit per item, but is not speeded. The test takes approximately one hour to complete and is scored according to the number of correct items within each item's time limit.

Working memory

Three 20-minute tests of working memory (Operation Span, Symmetry Span, and Reading Span) were acquired from Unsworth et al. (2005). Each test requires the memorization of letters or locations while simultaneously being distracted by having to solve simple math problems, determining whether a figure is symmetrical, or determining whether an English sentence is correctly formulated or not. For the distracting task, each subject was informed that accuracy must be kept over 85%, to limit memorization of letters or locations. However, for Reading Span, we informed each subject that accuracy below 85% would be acceptable as none of the subjects were native English speakers. Each test was scored according to the number of perfectly recalled sets.

Procedure

The programming skill test was administered at each company's office location during two full work days (16 hours). The first author was present during the study to answer questions, but the test was administered online through a specially built experiment support environment (Arisholm, Sjøberg, Carelius, & Linsjørn, 2002). Subjects were free to use whatever software development tools they normally used in their job. All subjects were given instructions and a practice task before the instrument was administered. All items were administered in random item order and the subjects were only allowed to take breaks between tasks. The working memory tests were, further, administered in these individual breaks between tasks during the two days, mainly during the second day. The working memory tests were, however, not available for the first four companies visited, implying fewer subjects (N=29). The knowledge test was administered online, 1 to 4 months after the programming skill test, on a sub sample of the subjects (92.3%, N = 60). Missing developers in this sub sample had either changed employer, did not want to participate, or were currently involved in other projects.

Statistical analysis and models tested

The hypothesized causal relationship between working memory, experience, programming knowledge, and programming skill is shown in Figure 1 (Model 1). Working memory and experience are hypothesized to affect programming knowledge. However, according to investment theory, direct paths from working memory and experience to skill should not be present as the causal effect of these two constructs should be mediated through knowledge. Model 2, also shown in Figure 1, is a competing model where paths from working memory and experience lead to skill instead and where the path from skill to knowledge is reversed in direction.

--- Insert Figure 1 about here ---

In both models, programming skill was represented as a latent variable with a single indicator, following the procedure described by Jöreskog and Sörbom (as cited in Mathieu, Tannenbaum, & Salas, 1992):

> [T]he path from a latent variable to its corresponding observed variable (lambda) is equal to the square root of the reliability of the observed score. In addition, the associated amount of random error variance (theta) is equal to one minus the reliability of the observed score times the variance of the observed score. (p. 837)

We used person separation index (PSI) as a reliability estimate (see e.g., Streiner, 1995), because PSI can be calculated with missing data. However, results are virtually identical to those obtained when using Cronbach's alpha (α).

Programming knowledge was represented as a latent variable by two indicators with equal loadings and variances. The 30 items were divided randomly into two parcels, each with the same average difficulty (items were delivered by the test provider in increasing order of difficulty based on data from their existing item bank). Working memory was represented by the three working memory tests, and neither factor loadings nor error variances were constrained.

Programming experience was measured by two indicators, both of which are log transformed. The first indicator, eLNLOC, is the subject's estimate of how many lines of code (LOC) that person has written in the Java programming language in total (during education and professional career). This indicator is, in part, a productivity indicator, as people perceived as more productive individuals would be expected to have written more code during their career.

The other indicator, eLNTotJProg, is the total number of months the individual has programmed in Java, both during education (if applicable) and professionally.

The models shown in Figure 1 were estimated with Amos 16.0, using maximum likelihood estimation for missing data which is implemented in this program.

Results

Descriptive statistics and reliability estimates

Descriptive results are shown in Table 1. The Java skill instrument was sufficiently reliable to assess individual differences ($\alpha = .85$ and PSI = .86). As has already been mentioned, results on the WMC test were only available for approximately half the subjects, while for the other variables there is little missing data. The programming skills instrument is unidimensional according to the independent paired t-test for unidimensionality (Smith, 2002), a procedure that is also implemented in the software used to carry out the Rasch analysis, RUMM2020 (Andrich, Sheridan, & Luo, 2006). The programming knowledge test had acceptable reliability as well ($\alpha = 0.81$), but was not unidimensional. We did not calculate reliability of the working memory tests, but these have previously been reported to have acceptable levels of reliability (Unsworth et al., 2005). Also, although ceiling effects are present for all the working memory tests, the mean and SD of our population is highly comparable to results reported elsewhere (Unsworth, Redick, Heitz, Broadway, & Engle, 2009). All variables used in the analysis, except programming knowledge, can be regarded as normally distributed by the one-sample Kolmogorov-Smirnov Test (1KS).

--- Insert Table 1 about here ---

Model 1—The investment theory model

Model 1, as shown in Figure 2, had a close model fit (χ^2[19] = 14.7, n.s., RMSEA = 0.000, LO90 = 0.000, HI90 = 0.081). As shown by the wide confidence interval for RMSEA, the power to reject a poor-fitting model is relatively low, but the confidence interval is almost entirely within the range that indicates a well-fitting model (RMSEA < 0.08). We also tested adding direct paths from working memory and experience to skill; the loadings were -.03, n.s. and -.08, n.s., respectively, with a slightly reduced model fit (χ^2[17] = 14.4, n.s., RMSEA = 0.000, LO90 = 0.000, HI90 = 0.096). We therefore chose to keep Model 1 as presented.

--- Insert Figure 2 about here ---

Model 2—The competing model

In Model 2, as shown in Figure 2, the relations from working memory and experience to knowledge only changed in a minor way. However, model fit decreased (χ^2[19] = 19.1, n.s., RMSEA = 0.007, LO90 = 0.000, HI90 = 0.110); χ^2 increased by 4.4 units for the same degrees of freedom and the upper limit of the confidence interval of RMSEA fell above an acceptable value.

Discussion

The good fit of Model 1 and the worse fit of Model 2 compared to Model 1 support Cattell's investment theory: the effect of working memory capacity and experience on programming skill is mediated through programming knowledge, which in turn accounts for a large degree of variance in programming skill. Furthermore, as is elaborated in theories of skill acquisition (Ackerman, 2000; Anderson, 1982; Fitts & Posner, 1967), there are good reasons why skill is mediated by knowledge rather than the other way around.

We now turn to how the findings can be interpreted in more specific terms. Our research reveals at least three important similarities between the meta-analysis by Schmidt et al. (1986) and this investigation, bearing in mind that Schmidt et al. reported on different types of jobs as well as different job-related constructs. First, knowledge was found in both studies to be the most important antecedent of skill/job sample performance, although we found the strength of this relationship to be larger in our study (.86 compared to .74). The greater significance of knowledge to skill performance in our study can probably be attributed to the content of the knowledge test being specifically chosen to mirror the required knowledge for successful completion of the programming tasks in the programming skill instrument. Another similarity is that the direct effect on skill/job sample performance from working memory/*G* and experience, which was small in Schmidt et al. and insignificant and close to zero in our study. Finally, Schmidt et al. reported a path between *G* and job knowledge of .65 for civilian data when excluding experience, while the path from working memory to knowledge in our study is .77 when experience is excluded.

The most obvious difference between our results and those of Schmidt et al., is the high correlation we observed between working memory and experience; *G* and experience were not correlated in their study. We cannot rule out that this result is an artefact due to the low N of the tests on working memory. However, when we inspected other experience variables such as number of lines of code written in other programming languages (besides Java), a similar and significant positive correlation with working memory was found as well. One plausible reason for this may be that programmers with a low working memory capacity might not continue to program over many years due to the continuous learning required to stay updated on the programming language they use for development. Another reason could be that LOC is partially

a productivity indicator; from the result in this study, we would expect developers with high levels of working memory to be more productive.

Further, in the Schmidt et al. (1986) meta-analysis, the impact on knowledge from experience was .57, which is somewhat larger than for *G* (.46). In our study, working memory explained most of the variance, although a lot of this variance was shared with experience. However, when we omitted working memory from Model 1, the path from experience to knowledge increased to .58, indicating a similar result.

One obvious reason for these differences is that Schmidt et al. used number of months on the present job for the experience variable, while we used both the number of months programming in Java (irrespective of job changes) and number of lines of code written in Java. As programming should be seen as a high complexity job, it would be expected that the effect of experience on knowledge would be less than for low complexity jobs (McDaniel et al., 1988). The reason for this is that initial experience with programming is often obtained through the educational system rather than on the job (in our subject sample, 96.9% had a bachelor's degree or higher).

Unidimensionality was a requirement for the programming skill instrument. However, to what degree an instrument is considered unidimensional or not is a relative matter; the instrument might not have the precision to distinguish between highly correlated dimensions that constitute separate constructs. We also had an extended version of the skill instrument that contained several programming tasks on Java debugging (finding errors in program code). These tasks were shorter and more akin to sudden insight problems (Smith & Kounios, 1996). Overall, the results for the two versions of the instrument were almost identical ($r > .99$), which is unsurprising given that they have 12 programming tasks in common. However, the least

unidimensional version of the instrument displayed worse model fit as well as slightly, but not significantly, higher correlation with both working memory and knowledge. It may well be that the more unidimensional version of the instrument is better at capturing the idiosyncrasies of programming skill as a construct, when inspecting the correlations with programming knowledge, experience and working memory.

Other questions remain. We do not know to what degree the strictly unidimensional version of the instrument does a better job of predicting programming performance "on the job." Also, we still do not know if the skill inferred from programming performance and attested by a unidimensional instrument can partially resolve the poorly understood construct validity of work samples (see Campbell, 1990). However, this study does meet Campbell, Gasser, and Oswald's criticism (1996) that using the dependent variable in SEM models as a rating of overall job performance is inadequate; we used instead a unidimensional measure of programming skill.

The main limitation of this study is low power, and in particular the number of observations for working memory. Due to the low power, the confidence intervals for RMSEA were wide. Another possible concern was the parcelling procedure used for the knowledge test. As a post hoc analysis, we investigated several alternative parcelling procedures for knowledge, but found only insubstantial differences for regression weights. However, RMSEA estimates were somewhat more affected, both for better and for worse, by different combinations of items in each parcel, but well within the reported 90% confidence limits.

Clearly, there are several loose ends requiring further work. In addition to including more subjects, we want to include a purer *g* measure, such as Ravens. Also, it could have been interesting to devise new tests of working memory capacity for programmers that use programming concepts as memorization or distracting elements. These tests would allow

experienced software developers to better use their expected superior chunking capacity (see Miller, 1956) for programming concepts. Such data could provide additional information on the structure of the abilities, knowledge and skill required by programmers to achieve superior job performance.

Our overall results support Cattell's investment theory. The results are also similar to those reported by Schmidt et al. (1986), albeit with some differences in operationalizations. Although the power of this study is low, the results show close fit to an investment theory model framed within a high-realism setting of software developers from multiple countries.

Acknowledgements

This research was supported by the FORNY programme at the research council of Norway. It is based on the first author's ongoing PhD thesis at Simula Research Laboratory with Dag Sjøberg, Erik Arisholm and Tore Dybå as supervisors. We thank Steinar Haugen for his programming assistance and Jo Hannay for his useful insights. We also thank the project managers, developers and companies involved in this study, as well as the test vendor company for allowing us to use their tests.

References

Ackerman, P. L. (1988). Determinants of individual differences during skill acquisition: Cognitive abilities and information processing. *Journal of Experimental Psychology: General, 117*(3), 288–318.

Ackerman, P. L. (2000). Domain-specific knowledge as the "Dark Matter" of adult intelligence: Gf/Gc, personality and interest correlates. *Journal of Gerontology, 55B*(2), P69-P84.

Ackerman, P. L. (2007). New developments in understanding skilled performance. *Current Directions in Psychological Science, 16*(5), 235-239.

Ackerman, P. L., Beier, M. E., & Boyle, M. O. (2002). Individual differences in working memory within a nomological network of cognitive and perceptual speed abilities. *Journal of Experimental Psychology: General, 131*(4), 567-589.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*(2), 192-210.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP Tutor. *Cognitive Science, 13*, 467-505.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika 43*(4): 561-573.

Andrich, D., Sheridan, B., & Luo, G. (2006). RUMM2020 [computer software]. Perth: RUMM Laboratory.

Arisholm, E. & Sjøberg, D. I. K. (2004). Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. *IEEE Transactions on Software Engineering, 30*(8), 521-534.

Arisholm, E., Sjøberg, D. I. K., Carelius, G. J., & Lindsjørn, Y. (2002). A web-based support environment for software engineering experiments. *Nordic Journal of Computing, 9*(3), 231-247.

Bergersen, G. R. (to be submitted in 2010). *Assessing programming skill*. PhD diss., University of Oslo.

Campbell, J. P. (1990). Modeling the performance prediction problem in industrial and organizational psychology. In M. D. Dunnette & L. M. Hough (Eds.), *Handbook of industrial and organizational psychology* (2nd ed.; Vol 1, pp. 687-732). Palo Alto: Consulting Psychology Press.

Campbell, J. P., Gasser, M. B., & Oswald, F. L. (1996). The substantive nature of job performance variability. In K. R. Murphy (Ed.), *Individual differences and behavior in organizations* (pp. 258-299). San Francisco: Jossey-Bass Inc.

Campbell, J. P., McCloy, R. A., Oppler, S. H., & Sager, C. E. (1993). A Theory of Performance. In N. Schmitt and W. C. Borman (Eds.) *Personnel selection in organizations* (pp. 35-70). San Francisco: Jossey-Bass.

Cattell, R. B. (1971/1987). *Abilities: Their structure, growth, and action*. Boston: Houghton-Mifflin. (Revised and reprinted in *Intelligence: Its structure, growth, and action*. New York: North-Holland).

Coan, R. B. (1964). Facts, factors and artifacts: The quest for psychological meaning. *Psychological Review, 71*, 123-140.

Colom, R., Rebollo, I., Palacios, A., Juan-Espinosa, M., & Kyllonen, P. (2004). Working memory is (almost) perfectly predicted by g. *Intelligence, 32*, 277-296.

Ericsson, K. A. & Charness, N. (1994). Expert performance: Its structure and acquisition. *American Psychologist, 49*(8), 725-747.

Fitts, P. M. & Posner, M. I. (1967). *Human Performance*. Belmont, California: Brooks/Cole Publishing Company.

Gustafsson, J. -E. (1984). A unifying model for the structure of intellectual abilities. *Intelligence, 8*, 179-203.

Gustafsson, J. -E. (2002). Measurement from a hierarchical point of view. In H. I. Braun, D. N. Jackson, & D. E. Wiley (Eds.), *The Role of Constructs in Psychological and Educational Measurement* (pp. 73-95). London: Lawrence Erlbaum Associates, Publishers.

Hannay, J. E., Arisholm, E., Engvik, H., & Sjøberg, D. I. K. (in press). Effects of personality on pair programming. *IEEE Transactions on Software Engineering*.

Humphreys, L. G. (1962). The organization of human abilities. *American Psychologist, 17*(7), 475-483.

Kyllonen, P. C. & Stephens, D. L. (1990). Cognitive abilities as determinants of success in acquiring logic skill. *Learning and Individual Differences, 2*(2), 129-160.

Kyllonen, P. C., & Woltz, D. J. (1989). Role of cognitive factors in the acquisition of cognitive skill. In R. Kanfer, P. L. Ackerman, & R. Cudeck (Eds.), *Abilities, motivation, and methodology: The Minneapolis symposium on learning and individual differences* (pp. 239-280). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Mathieu, J. E., Tannenbaum, S. I., & Salas, E. (1992). Influences of individual and situational characteristics on measures of training effectiveness. *The Academy of Management Journal, 35*(4), 828-847.

McDaniel, M. A., Schmidt, F. L., & Hunter, J. E. (1988). Job experience correlates of job performance. *Journal of Applied Psychology, 73*(2), 327-330.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 61*(2), 81-97.

Neves, D. M. & Anderson, J. R. (1981). Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Mahwah, NJ: Lawrence Erlbaum Associates, Publishers.

Rasch, G. (1960). *Probabilistic models for some intelligence and achievement tests*. (Copenhagen: Danish Institute for Educational Research) Expanded edition, with a forward and afterword by B. D. Wright, Chicago: University of Chicago Press, 1980.

Schmidt, F. L. (2002). The role of general cognitive ability and job performance: Why there cannot be a debate. *Human Performance, 15*, 187-210.

Schmidt, F. L. & Hunter, J. E. (1998). The validity and utility of selection methods in personnel psychology: Practical and theoretical implications of 85 years of research findings. *Psychological Bulletin, 124*(2), 262-274.

Schmidt, F. L., Hunter, J. E., & Outerbridge, A. N. (1986). Impact of job experience and ability on job knowledge, work sample performance, and supervisory ratings of job performance. *Journal of Applied Psychology, 71*(3), 432-439.Shute, V. J. (1991). Who is likely to acquire programming skills? *Journal of Educational Computing Research, 7*(1), 1-24.

Smith, E. V. Jr. (2002). Detecting and evaluating the impact of multidimensionality using item fit statistics and principal component analysis of residuals. *Journal of Applied Measurement, 3*(2), 205-231.

Smith, R. W. & Kounios, J. (1996). Sudden insight: All-or-none processing revealed by speed-accuracy decomposition. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(6), 1443-1462.

Streiner, D. L. (1995). *Health measurement scales*. Oxford: Oxford University Press.

Unsworth, N., Heitz, R. P., Schrock, J. C., & Engle, R. W. (2005). An automated version of the operation span task. *Behavior Research Methods, 37*(3), 498-505.

Unsworth, N., Redick, T., Heitz, R. P., Broadway, J. M., & Engle, R. W. (2009). Complex working memory span tasks and higher-order cognition: A latent-variable analysis of the relationship between processing and storage. *Memory, 17*(6), 635-654.

Valentin Kvist, A., & Gustafsson, J. -E. (2008). The relation between fluid intelligence and the general factor as a function of cultural background: A test of Cattell's Investment theory. *Intelligence, 36*, 422-436.

Table 1

Descriptives of all variables used

Variable	Score	N	Min	Max	Mean	SD	1KS
Java programming skill (ProgSkill)	LGT	65	-4.12	1.58	-0.83	1.30	.489
Java programming knowledge (Know)	NC	60	7	29	21.72	4.80	.013
Working Memory (WMC)							
Symmetry Span	NR	28	10	42	25.11	8.36	.979
Operation Span	NR	29	21	75	52.14	15.33	.945
Reading Span	NR	28	25	75	54.25	14.48	.750
Experience							
Total lines of code written in Java	Ln(LOC)	64	6.21	13.82	10.46	1.76	.806
Total Java programming experience	Ln(months)	65	.69	4.87	3.45	0.79	.161

Notes: LGT = logits (0 is defined in the Rasch model as the average difficulty of items), C = number correct, NR = number of perfectly recalled sets

Figure Captions

Figure 1. Hypothesized structure of regression weights for Model 1 (left) and Model 2 (right)

Figure 2. Results for Model 1 (left) and Model 2 (right)

Figure 1:

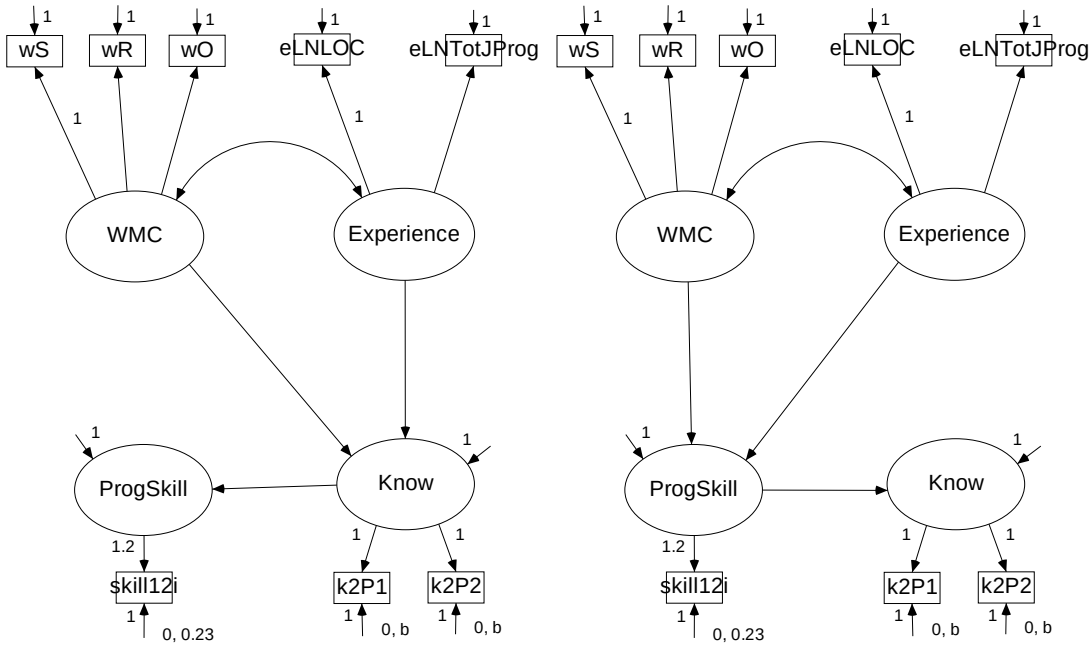
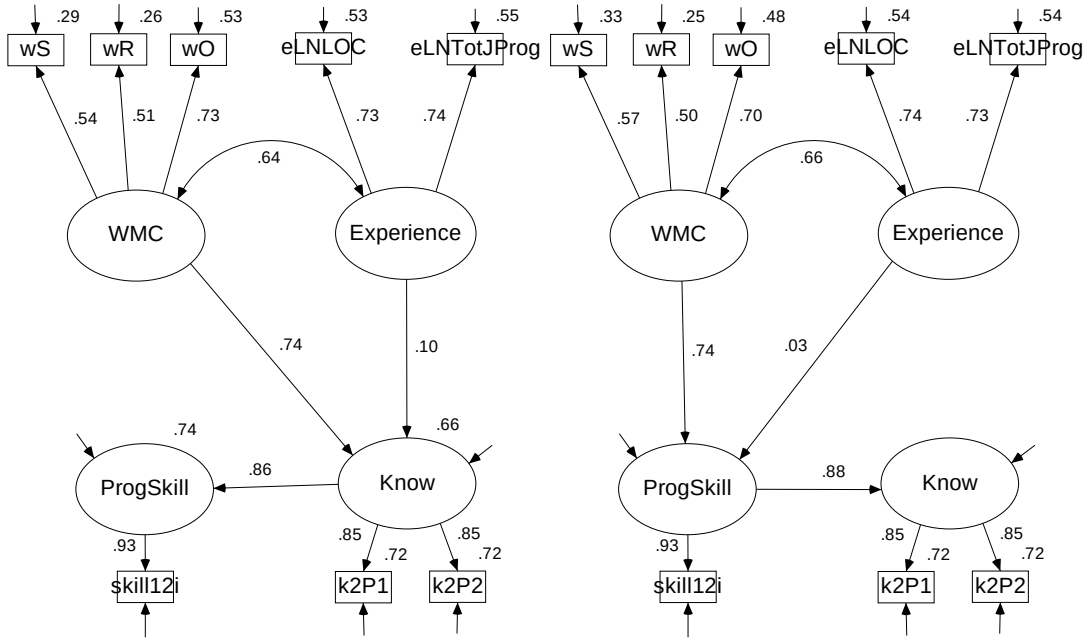

wS
wR
wO
eLNLOC
eLNTotJProg
WMC
Experience
ProgSkill
Know
skill12i
k2P1
k2P2
1
1.2
0, 0.23
0, b

Figure 2:


.29
.26
.53
.53
.55
wS
wR
wO
eLNLOC
eLNTotJProg
.54
.51
.73
.73
.74
.64
WMC
Experience
.74
.10
.74
.66
ProgSkill
.86
Know
.93
.85
.72
.85
.72
skill12i
k2P1
k2P2
.33
.25
.48
.54
.54
wS
wR
wO
eLNLOC
eLNTotJProg
.57
.50
.70
.74
.73
.66
WMC
Experience
.74
.03
ProgSkill
.88
Know
.93
.85
.72
.85
.72
skill12i
k2P1
k2P2