Cognition, Technology & Work (2018) 20:489–504
https://doi.org/10.1007/s10111-018-0479-x

ORIGINAL ARTICLE

Do older programmers perform as well as young ones? Exploring the intermediate effects of stress and programming experience

Ned Kock[1] · **Murad Moqbel**[2] · **Yusun Jung**[3] · **Thant Syn**[3]

Received: 29 August 2017 / Accepted: 25 March 2018 / Published online: 16 April 2018

Abstract
There is a widespread perception that older adults are underperformers when compared with younger adults in tasks that involve intense use of technology, such as computer programming. Building on schema theory, we developed a research model that contradicts this perception. To provide an initial test of the model, we conducted a computer programming experiment involving 140 student participants majoring in technology-related areas with ages ranging from 19 to 54 years. The participants were asked to develop, under some time pressure, a simple software application. The results of our analyses suggest that age was positively associated with programming experience and perceived stress, that programming experience was positively associated with programming performance, and that perceived stress was negatively associated with programming performance. A moderating effect analysis suggests that as programming experience increased, the association between perceived stress and programming performance weakened; going from strongly negative toward neutral. This happened even as age was controlled for. When taken together, these results suggest that the widespread perception that older adults are underperformers is unwarranted. With enough programming experience, older programmers generally perform no better or worse than young ones.

Keywords Age · Computer programming · Laboratory experiment · Structural equation modeling · Factor-based PLS

1 Introduction

The perception that older individuals are less adept at performing activities relying on information technology (IT) in both personal and professional contexts is widespread (Czaja 1995; Kraft 2012; Perry et al. 2003) and is often referred to by the terms "ageism" and "digital ageism" (Garstka et al. 2004; Magsamen-Conrad et al. 2015; Oh et al. 2016; Vauclair et al. 2016). Employment practices that are influenced by this perception are frequently considered illegal. This is codified in a variety of laws in many countries, such as the Age Discrimination in Employment Act in the USA.

While the perceptions that underlie digital ageism are widespread, there is little if any theoretical or empirical research directly addressing digital ageism's most fundamental assumption—the existence of a negative link between age- and IT-related task performance. We aim to fill this gap by developing a research model and providing a preliminary test of this model in the context of computer programming. We try to answer a specific research question: do older programmers perform as well as young ones? From a theoretical perspective, as it will be seen, we find reasons

✉ Ned Kock
nedkock@gmail.com
http://www.tamiu.edu/~nedkock

Murad Moqbel
muradmoqbel@gmail.com

Yusun Jung
yusun.jung@tamiu.edu

Thant Syn
thant.syn@tamiu.edu

[1] Division of International Business and Technology Studies, Texas A&M International University, 5201 University Boulevard, Laredo, TX 78041, USA

[2] Management Information Systems Department, University of Oklahoma, Adams Hall Room 305, 307 West Brooks, Norman, OK 73019-4007, USA

[3] Division of International Business and Technology Studies, Texas A&M International University, 5201 University Boulevard, Laredo, TX 78045, USA

to believe that older programmers may in fact perform better than their younger counterparts.

To provide an initial test of the model, we collected data through a software development experiment involving 140 participants, who were graduate and undergraduate students majoring in IT-related areas from a midsized university in the southwestern region of the USA. The participants were asked to individually develop, under some time pressure, a simple software application. The data collected were used to test a path model with latent variables, for which we employed factor-based partial least squares structural equation modeling (Kock 2015a, b).

Consistently with the research model we developed, our analyses suggest that older individuals tend to have more programming experience than younger ones and that programming experience plays a key role in how well programmers perform under stress. The key role played by programming experience is that of a significant moderator of the relationship between stress and programming performance. We also found that older individuals experience more stress while performing a programming task under time pressure than younger individuals, with the end result being a negative and significant total association between a programmer's age and his or her programming performance. However, in line with our theoretical expectations, our analyses suggest that the effect of stress and thus the indirect effect of age become insignificant at high levels of programming experience. In other words, given enough programming experience, age has no appreciable effect on programming performance.

2 Theoretical orientation: the age-experience framework

Our theoretical orientation is summarized through a framework that we propose here and refer to as the age-experience (AE) framework. This framework builds on schema theory (Gardner 1985; Rumelhart 1978; Sorensen and Stanton 2015) and is based on a fundamental idea: as individuals age they go through a variety of life experiences through which they acquire mental schemas (Gardner 1985; Tse et al. 2007) that enable them to successfully handle similar experiences in the future. Mental schemas are knowledge structures formed, often involuntarily, with the goal of understanding and enacting behaviors (Bartlett 1932, 1958; Cossette and Audet 1992; Gioia and Manz 1985). Figure 1 provides a schematic view of the key elements of the AE framework.

The AE framework posits that older individuals tend to possess both general and specific task-related mental schemas (Gioia and Manz 1985; Tse et al. 2007) that influence their performance in both general and specific ways. General schemas are acquired over time through generic

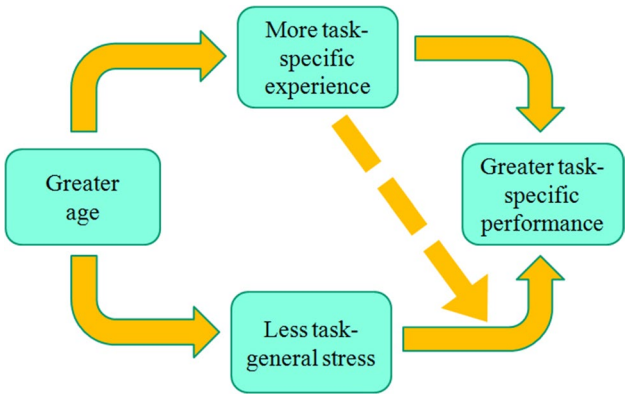

Fig. 1 Age-experience (AE) framework

problem-solving experiences, whereas specific schemas are acquired through the solution of problems in the context of specific tasks. General schemas create a level of general "preparedness" that has a stress-reduction effect (Bartlett 1958; Cohen et al. 2015; Gardner 1985), thus indirectly influencing performance in a positive way for tasks in general. Task-specific schemas lead to task-specific "preparedness," which also positively influences task performance in specific task contexts directly and indirectly; the latter by moderating the effect of stress on task performance (Gardner 1985; Lord and Maher 1990).

As it will be seen in the next section, the AE framework can be easily adapted to the specific contexts of IT-related work and computer programming (Duschl et al. 2015; Khan et al. 2011), and in ways that find support from past research findings. In any population made up exclusively of professional programmers, it is commonsense to expect that age and programming experience would be positively and strongly correlated with one another. If we consider a population of individuals who are not necessarily professional programmers, but who are either IT professionals or closely involved in IT-related activities, we will likely find a positive association between age and programming experience (Bailey and Mitchell 2006; Huang 2015), although not as strong as that in a population including only professional programmers. While not all IT professionals are professional programmers, most IT professionals are expected to have some experience with programming to be effective at their jobs (Bailey and Mitchell 2006; Johnson 2015).

3 Research model and hypotheses

In this section, we start by showing our research model. This is a path model that follows directly from the AE framework. We then put forth several hypotheses, developed individually building on relevant past theoretical and empirical research.

The hypotheses are not shown directly in the path model because several of them involve indirect and total effects that cannot be easily indicated in the model. Figure 2 shows the path model in question, which embodies our application of the AE framework for the specific task of computer programming.

While our model refers specifically to computer programming, and given our previous discussion regarding IT employment, arguably it applies more broadly to individuals who are either IT professionals or closely involved in IT-related activities. That is, the unit of analysis to which the model refers is an individual who performs IT-related activities in an organization. Even though the model can be tested with students majoring in IT-related areas, its main focus is on professionals who perform IT-related activities on a daily basis. These professionals are expected to have past experience with programming to be effective at their jobs (Bailey and Mitchell 2006; Johnson 2015), which may involve the evaluation of computer programs and their purchase to support mission-critical organizational processes.

The model contains one main independent variable, namely *programmers' age*, and one main dependent variable, which is *programming performance*. The association between the main independent and dependent variables is hypothesized to be fully mediated in the model by *programming experience* and *perceived stress*, which is why the direct relationship at the top of the model is indicated as zero (0). Other associations are indicated as either positive (+) or negative (−). A moderating effect of *programming experience* on the relationship between *perceived stress* and *programming performance* is included in the model. Two control variables, namely *GPA* and *Sex* (*M*/*F*), are also included in the model. The hypotheses arising from this model are discussed in the subsections below. A discussion of how the variables were measured, and of the need for the control variables, is provided later as part of an elaboration of the research method used.

3.1 Age, programming experience, and stress

Individuals who perform IT-related activities are expected to have programming experience to be effective at their jobs (Bailey and Mitchell 2006; Dönmez et al. 2016; Johnson 2015; Kraft 2012), even if they are not professional programmers. Programming experience in this case is the equivalent to task-specific experience in the AE framework and is thus expected to be directly affected by age among those whose work involve IT-related activities. The older the professional, the greater is the amount of programming experience he or she possesses, an expectation that has been strongly supported by past empirical research (Kraft 2012).

The AE framework also posits that as one gets older so do general life experiences that enable better coping with stress, including stress that is related to technology use and development (Maier et al. 2015; Soror et al. 2015). Dyck and Smither's (1994) study contrasted individuals 55 years of age and over with younger adults, whom the researchers classified as 30 years of age and under. Among the bases for comparison were levels of stress when using computers for complex tasks, including computer programming and computer experience. They found that older adults experienced less stress regardless of past computer experience, as predicted by the AE framework. This suggests that age may have an effect on stress in IT-related tasks, including programming, which could be independent from that of experience with the tasks in question. Thus, we hypothesize that:

H1: Age has a positive direct association with programming experience.

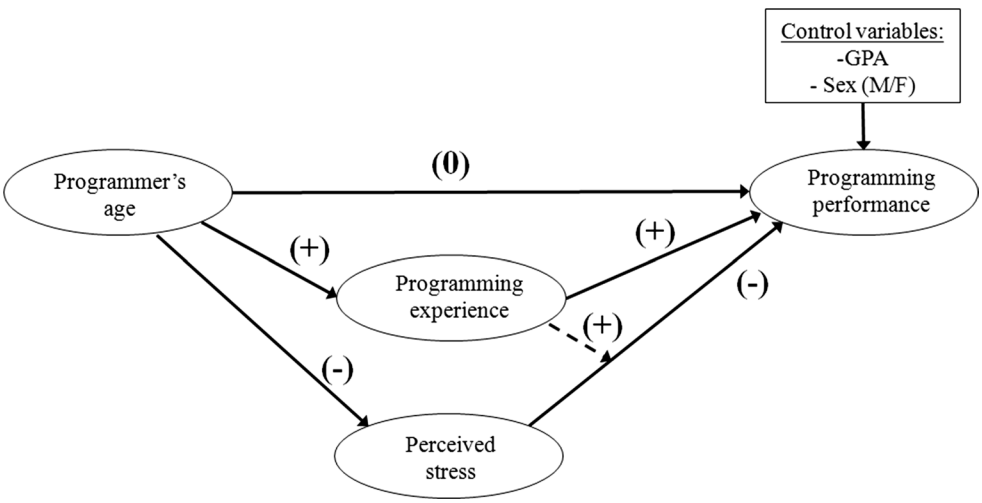

Fig. 2 Research model

H2: Age has a negative direct association with perceived stress.

3.2 Stress, programming experience, and performance

The AE framework incorporates the expectation that increased task-specific experience and decreased task-general stress both lead to increased task-specific performance. From an IT perspective, this would lead to the predictions that more programming experience and less stress contribute to better programming performance. Past studies of performance by individuals with and without prior programming experience provide strong support for the expectation that more programming experience leads to better programming performance (Hagan and Markham 2000; Byrne and Lyons 2001). The link between stress and performance is also well established based on past research in IT-related activities in general (Brosnan 1998; Gilroy and Desai 1986), as well as in the more specific context of computer programming (Beckers et al. 2006; Potosky 2002). The two hypotheses below follow from this discussion.

H3: Programming experience has a positive direct association with programming performance.

H4: Perceived stress has a negative direct association with programming performance.

Consistently with the AE framework, task-specific schema development has been found to moderate the relationship between stress and performance in IT-related contexts (Elias et al. 2012; Gilroy and Desai 1986; González et al. 2012). In the milieu of computer programming, this would mean that the level of schema development related to this specific task would reduce the negative effect of stress on programming performance. That is, more experienced programmers would perform better under stress than less experienced ones, an expectation that has been supported by past empirical research (Beckers et al. 2006). Therefore, we hypothesize that:

H5: Programming experience positively moderates the direct association between perceived stress and programming performance.

3.3 Indirect and total effects of age on performance

Implicit in the AE framework is the belief that its main dependent variable, namely age, has indirect and positive effects on task-specific performance via decreased task-general stress and increased task-specific experience. Past research has shown that stress is a likely mediator of the possible negative relationship between age and performance in IT-related tasks (Caplan and Schooler 1990; Kraft 2012). The same is true regarding the mediating effect of experience in the context of IT-related tasks (Beckers et al. 2006; Dollinger 1995; Hasan 2003; Morrell et al. 2000), as well as in the more specific context of computer programming (Bergin and Reilly 2005; Gilroy and Desai 1986; Potosky 2002). Therefore, we hypothesize that:

H6: Age has a negative indirect association with programming performance via perceived stress.

H7: Age has a positive indirect association with programming performance via programming experience.

Age-induced decreases in task-general stress and increases in task-specific experience are fundamental elements in the AE framework, which are posited to play a significant role in the improvement of task-specific performance. In the context of IT-related tasks, including computer programming, the overall effect would be a positive and significant total effect of age on task-related performance. Past research has indeed shown that older adults often display specific behaviors and capabilities that make them perform better in IT-related tasks than younger adults (Dibiase and Kidwai 2010). Moreover, it follows from the AE framework that this relationship between age and task performance is fully mediated by task-specific experience and stress. The foregoing discussion leads to our two final hypotheses:

H8: The total association between age and programming performance is fully mediated by programming experience and perceived stress.

H9: The total association between age and programming performance is positive.

Testing of the above hypotheses allows us to answer our principal research question: do older programmers perform as well as young ones? As we can see, the AE framework provides the basis on which we can conclude that programmers who are older, and thus more experienced, may perform better than younger programmers. The test of the hypothesis referring to the total association between age and programming performance is central to answering our principal research question, whereas the tests of the other hypotheses should clarify the inner mechanisms influencing the total association.

4 Research method

The data for this study have been collected through a software development experiment involving 158 student participants; from whom 140 valid sets of results and responses were obtained, for an 88.6% response rate. This was slightly above our minimum sample size estimate obtained through a statistical power analysis (Kock 2016; Martin 2007; Rosenthal and Rosnow 2007), which called for a sample size equal to or greater than 126 to ensure that any direct effect coefficient in our model found to be statistically significant at the $P < 0.05$ level was associated with a power greater than 80%. We also ensured that each direct effect coefficient yielded a practically relevant *f*-squared effect size coefficient of more than 0.02 (Cohen 1988).

The participants were graduate and undergraduate students from a midsized university in the southwestern region of the USA. All students were management information systems' majors, i.e., all students were majoring in IT-related areas. Extra credit was provided for participation. Institutional review board approval was achieved prior to conducting the experiment. Informed consent was obtained from each participant prior to their participation in the experiment.

All students received instruction on how to develop basic computer programs with Microsoft Visual Basic prior to participating in the experiment. The experimental task lasted approximately 1 h. In it, students were asked to individually develop, under some time pressure, a simple software application to help the director of a PhD program in a school of business make decisions regarding admissions of student applicants into the program. The software development was in Visual Basic. A description of the experimental task is provided in Appendix 1.

After carrying out the software development task, the participants were asked to complete a questionnaire. The questions and question-statements used are available in Appendix 2. The questionnaire contained demographic questions, as well as question-statements related to perceived stress and a rubric for the scoring of programming performance. See Appendix 3 for details on the rubric for programming performance scoring. Both the experimental task and the construct measurement instrument were developed based on previously published materials (Akerstedt and Gillberg 1990; Burgess 2005; Cohen et al. 1983; Ramalingam and Wiedenbeck 1998).

The participants' grade point average (GPA) ranged from 1.75 to 4.00, with a mean of 3.09 and a standard deviation of 0.50. Age ranged from 19 to 54 years, with a mean of 24 and a standard deviation of 6.4 years. Programming experience ranged from 0 to 5 years, with a mean of 0.49 years and a standard deviation of 0.85 years. Approximately 62% of the participants were females. These and other descriptive statistics are listed in Table 1. The two normality tests for which results are shown at the bottom of the table are the classic Jarque–Bera test (Jarque and Bera 1980; Bera and Jarque 1981) and Gel and Gastwirth's (2008) robust modification of this test. These tests suggest various deviations from normality in the data.

The data analysis method employed in this study was factor-based partial least squares structural equation modeling, a variation of the classic partial least squares technique (Haenlein and Kaplan 2004; Kock 2014). This variation yields robust estimates with small samples, does not require that the data be normally distributed, and deals with factors as opposed to composites—thus accounting for measurement error (Kock 2015a; Kock and Mayfield 2015). The software WarpPLS, version 5.0 implements this variation (Kock 2015b; Kock and Chatelain-Jardón 2016) and therefore was used. The structural equation modeling analysis was preceded by a confirmatory factor analysis, through which the measurement instrument was validated (Kline 1998; Kock 2014; Thompson 2004).

The following variables were included as control variables, with respect to programming performance, in the structural equation modeling analysis: grade point average (GPA) and biological sex (male or female). GPA was included because academic achievement is likely to be correlated with performance in many academic tasks (Catherine and Wheeler 1994; Gnambs 2015). Biological sex was included because of the common finding in past research that males generally find programming easier than females (Rubio et al. 2015). The use of these control variables essentially means that the results of the analysis, with respect to the model's main dependent variable, hold regardless of GPA and biological sex.

Table 1 Descriptive statistics

	GPA	Age	Sex (M/F)	Exp
Mean	3.091	24.478	0.619	0.492
SD	0.479	6.344	0.486	0.839
Min	1.750	19.000	0.000	0.000
Max	4.000	54.000	1.000	5.000
Median	3.091	23.000	1.000	0.000
Mode	3.091	21.000	1.000	0.000
Skewness	−0.193	2.474	−0.491	2.278
Excess kurtosis	−0.274	7.201	−1.752	6.309
Normal-JB	Yes	No	No	No
Normal-RJB	Yes	No	No	No

Exp, programming experience; Sex (M/F), M:0/F:1, SD, standard deviation; Normal-JB, Jarque–Bera test of normality; Normal-RJB, robust Jarque–Bera test of normality

5 Measurement instrument validation

The main goal of measurement instrument validation is to ensure that there is congruence among instrument designers and respondents, as well as among respondents as a group, in their understanding of the questions and question-statements in the instrument with respect to the underlying constructs that they are supposed to measure (Kline 1998; Kock and Lynn 2012; Kock and Mayfield 2015). Congruence among instrument designers and respondents regarding underlying constructs refers to convergent validity, with lack of confusion across constructs referring specifically to discriminant validity. Congruence among respondents as a group refers to reliability, i.e., all respondents understand the instrument questions and question-statements in the same way. The absence of model-wide collinearity refers to measurement discrimination among constructs, i.e., different constructs measure different "things."

Loadings, weights, cross-loadings, cross-weights, and indicator effect sizes were calculated, primarily for convergent validity tests (see Appendix 4 for the coefficients mentioned in this section). *P* values were calculated for loadings and weights via resampling; all loadings and weights were found to be significant at the $P < 0.001$ level. Also, all loadings were significantly greater than 0.5, ranging from 0.639 to 0.970, and all cross-loadings were lower than 0.5. All indicator effect sizes were greater than Cohen's (1988) threshold of 0.02. These results, when taken together, suggest that the measurement instrument passes a convergent validity test in the context of this study (Hair et al. 2009; Kock 2014).

R-squared, composite reliability, Cronbach's alpha, average variance extracted, and *Q*-squared coefficients were calculated, chiefly for reliability and predictive validity assessment. All composite reliability and Cronbach's alpha coefficients were greater than 0.7, suggesting that the measurement instrument has acceptable reliability in the context of this study (Fornell and Larcker 1981; Nunnaly 1978; Nunnally and Bernstein 1994). All average variances extracted were greater than 0.5, suggesting acceptable convergent validity (Fornell and Larcker 1981), consistently with the loadings mentioned earlier. *Q*-squared coefficients were close in value to *R*-squared coefficients and are all greater than 0, suggesting acceptable predictive validity (Geisser 1974; Kock 2015b; Stone 1974).

Latent variable correlations and square roots of average variances extracted were calculated mainly for discriminant validity assessment. For each latent variable, the square root of the corresponding average variance extracted was found to be greater than any of the correlations involving the latent variable in question. Therefore, it can be concluded that the measurement instrument has acceptable discriminant validity in the context of this study (Fornell and Larcker 1981; Kock and Lynn 2012; Schumacker and Lomax 2004).

In addition to multi-indicator latent variables, the model also contains variables measured through single indicators (e.g., age), which tend to increase the model's full collinearity (Hair et al. 2009; Kock and Lynn 2012). Single-indicator variables tend to increase the model's full collinearity because the collinearity-minimization weight assignment scheme in partial least squares methods cannot be applied to them (Kock and Mayfield 2015; Lohmöller 1989).

A full collinearity test takes all variables in a model into consideration (Kock and Lynn 2012). This test has also been shown to be an effective and conservative alternative for the identification of common method bias (Dermentzi et al. 2016; Kock 2015c). Therefore, a full collinearity test was conducted through the calculation of variance inflation factors for all variables in the model. All variance inflation factors were found to be lower than 3.3, suggesting no model-wide collinearity or common method bias (Kock 2015c; Kock and Lynn 2012).

In summary, the measurement instrument used in this study passes a comprehensive number of fairly stringent tests. The results of these tests suggest that the measurement instrument presents acceptable convergent and discriminant validity, as well as reliability. They also suggest that the measurement instrument presents acceptable predictive validity. Finally, the results of the tests suggest that the measurement instrument used is free from model-wide collinearity and common method bias.

6 Results

Table 2 lists the hypothesized links, the corresponding effect types (e.g., direct, moderating, etc.), the related coefficients of association, and their statistical significance levels. The term "Exp" refers to programming experience and "Perf" to programming performance. Effect types shown are

Table 2 Hypothesized links and coefficients of association

Hypothesized links	Effect type	Coefficient	Significance
Age → Exp	Direct	0.211	$P < 0.01$
Age → Stress	Direct	0.239	$P < 0.01$
Exp → Perf	Direct	0.262	$P < 0.001$
Stress → Perf	Direct	−0.443	$P < 0.001$
Exp → (Stress → Perf)	Moderating	0.242	$P < 0.01$
Age → Stress → Perf	Indirect	−0.106	$P < 0.05$
Age → Exp → Perf	Indirect	0.035	Not significant
Age → Perf	Direct	−0.111	Not significant
Age → → Perf	Total	−0.162	$P < 0.05$

Exp programming experience, *Perf* programming performance

"direct," referring to single direct links in the model (e.g., Age → Exp); "moderating," referring to the moderating link in the model where one variable affects the relationship between other two variables (i.e., Exp → (Stress → Perf)); "indirect," referring to effects for paths of multiple connected links in the model (e.g., Age → Stress → Perf); and "total," referring to the total effect of all paths connecting two variables (i.e., Age →→ Perf).

Two direct links departing from the variable Age make up the left side of our structural model. The coefficient of association for the direct link Age → Exp ($\beta = 0.211$) was found to be statistically significant at the $P < 0.01$ level. This supports hypothesis H1, which is that a programmer's age has a positive direct association with programming experience. However, the coefficient of association for the direct link Age → Stress ($\beta = 0.239$), which was found to be statistically significant at the $P < 0.01$ level, does not support hypothesis H2, which is that a programmer's age has a *negative* direct association with perceived stress. Support for H2 would have required a statistically significant, but *negative*, coefficient of association.

Two additional direct links are important in the assessment of downstream causal effects in the model that make up indirect effects. The coefficients of association for the direct links Exp → Perf and Stress → Perf (respectively, $\beta = 0.262$ and $\beta = -0.443$) were both found to be statistically significant at the $P < 0.001$ level. These results, respectively, support hypothesis H3, which is that programming experience has a positive direct association with programming performance, and hypothesis H4, which is that perceived stress has a negative direct association with programming performance.

Our model contains one single moderating link. The coefficient of association for the moderating link Exp → (Stress → Perf) was found to be positive ($\beta = 0.242$) and statistically significant at the $P < 0.01$ level. This means that, as values of Exp increase (i.e., more programming experience), the coefficients of association for the link Stress → Perf tend to increase in value, going from negative to neutral. This provides support for hypothesis H5, which is that programming experience positively moderates the direct association between perceived stress and programming performance.

Two indirect links allow us to assess the effects of intervening variables. The coefficient of association for the indirect link Age → Stress → Perf ($\beta = -0.106$) was found to be statistically significant at the $P < 0.05$ level. This supports hypothesis H6, which is that a programmer's age has a negative indirect association with programming performance via perceived stress. However, the coefficient of association for the indirect link Age → Exp → Perf ($\beta = 0.035$), found to be statistically nonsignificant, does not support hypothesis H7, which is that a programmer's age has a positive indirect association with programming performance via programming experience. Support for H7 would have required statistical significance, i.e., a statistically significant positive coefficient of association.

One final direct link, combined with a total effect link, allows us to assess our last two hypotheses. The coefficient of association for the direct link Age → Perf ($\beta = -0.111$) was found to be statistically nonsignificant, whereas the coefficient for the total effect link Age →→ Perf ($\beta = -0.162$) was found to be statistically significant at the $P < 0.05$ level. These results provide support for hypothesis H8, which is that the total association between a programmer's age and programming performance is fully mediated, not by programming experience but by perceived stress. However, these results do not provide support for hypothesis H9, which is that the total association between a programmer's age and programming performance is positive.

Table 3 provides a summary of model-data fit indices that are often used in this type of analysis (Chan et al. 2015; Kock 2015b; Jaradat and Faqih 2014). Four fit indices are shown, namely the average path coefficient (APC), average *R*-squared (ARS), average full collinearity variance inflation factor (AFVIF), and the Tenenhaus goodness-of-fit index (GoF). These fit indices allow us to assess the extent to which the hypothesized model fits the data. The indices act in concert with APC and ARS unveiling problems with the structural model (relationships among linked variables) and the AFVIF and GoF being useful in the identification of problems with the measurement model (relationships among latent variables and indicators).

The APC was found to be statistically significant ($\beta = 0.196$, $P < 0.001$), and so was the ARS ($\beta = 0.156$, $P < 0.001$). The AFVIF was found to be 1.318, which is well below the threshold of 3.3, indicative of absence of model-wide collinearity. The GoF was calculated at 0.380, which is above the 0.360 for a large fit. Overall, these fit indices suggest good model-data congruence, when considered together, and give us confidence that the hypothesis-testing results are not significantly distorted by model misspecification bias. Table 4 summarizes the support, or lack of support, for the hypotheses.

Table 3 Model fit indices

Fit index	Value	Significance or acceptance level
APC	0.196	$P < 0.001$
ARS	0.156	$P < 0.001$
AFVIF	1.318	Acceptable if ≤ 5, ideally ≤ 3.3
GoF	0.380	Small ≥ 0.1, medium ≥ 0.25, large ≥ 0.36

APC, average path coefficient; ARS, average *R*-squared; AFVIF, average full collinearity variance inflation factor; GoF, Tenenhaus goodness-of-fit index

Table 4 Support for the hypotheses based on the results

Hypothesis	Supported?
H1: Age has a positive direct association with programming experience.	Yes
H2: Age has a negative direct association with perceived stress.	No
H3: Programming experience has a positive direct association with programming performance.	Yes
H4: Perceived stress has a negative direct association with programming performance.	Yes
H5: Programming experience positively moderates the direct association between perceived stress and programming performance.	Yes
H6: Age has a negative indirect association with programming performance via perceived stress.	Yes
H7: Age has a positive indirect association with programming performance via programming experience.	No
H8: The total association between age and programming performance is fully mediated by programming experience and perceived stress.	Yes
H9: The total association between age and programming performance is positive.	No

Three of the nine hypotheses were not supported by the data. Unexpectedly, we found that age was significantly and positively associated with perceived stress, i.e., older individuals experienced more stress while programming than younger ones. Also unexpectedly, we found that age had a nonsignificant indirect association with programming performance via programming experience, i.e., programming experience was not a significant mediator of the relationship between age and programming performance. A third unexpected finding was that the total association between age and programming experience turned out to be negative and significant.

Figure 3 contains a plot of the total negative effect of age on programming performance. This plot is meant to give a practical idea to the reader of the magnitude of this total effect. It suggests that each 10-year increment in age is on average associated with a 28.6% decrease in programming performance, calculated as:

$$((0.21 + 1.15)/(54 - 19) * 10)/(0.21 + 1.15) = 0.286.$$

It is noteworthy that we did find a positive and significant moderating effect of programming experience on the direct association between perceived stress and programming performance. This moderating effect was positive and significant after we controlled for the direct effect of age on programming performance. That is, the more programming experience one has, the less perceived stress affects programming performance, regardless of age. Figure 4 provides a visual representation of this significant moderating effect. The graph shows two best-fitting lines, referring to individuals with low and high programming experience, for the associations between perceived stress and programming performance. The data were segmented into two sub-datasets of the same size, for grouping of participants according to low and high programming experience.

In summary, older individuals seem to have more programming experience than younger ones, as expected. Also, programming experience appears to play a key role in how

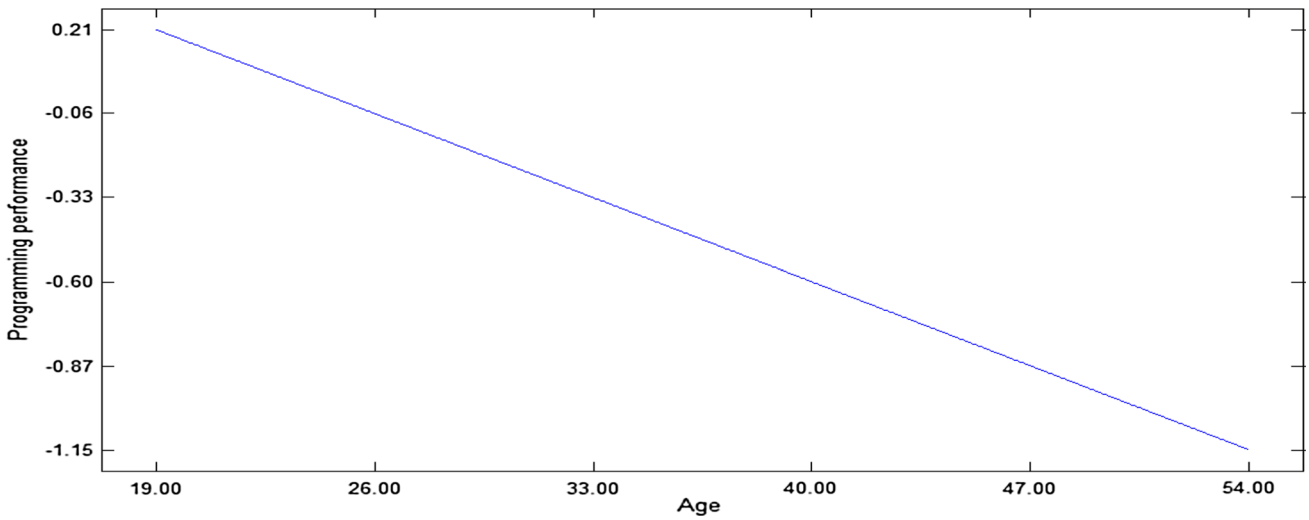

Fig. 3 Total effect of age on programming performance. *Note* Programming performance shown on a standardized scale

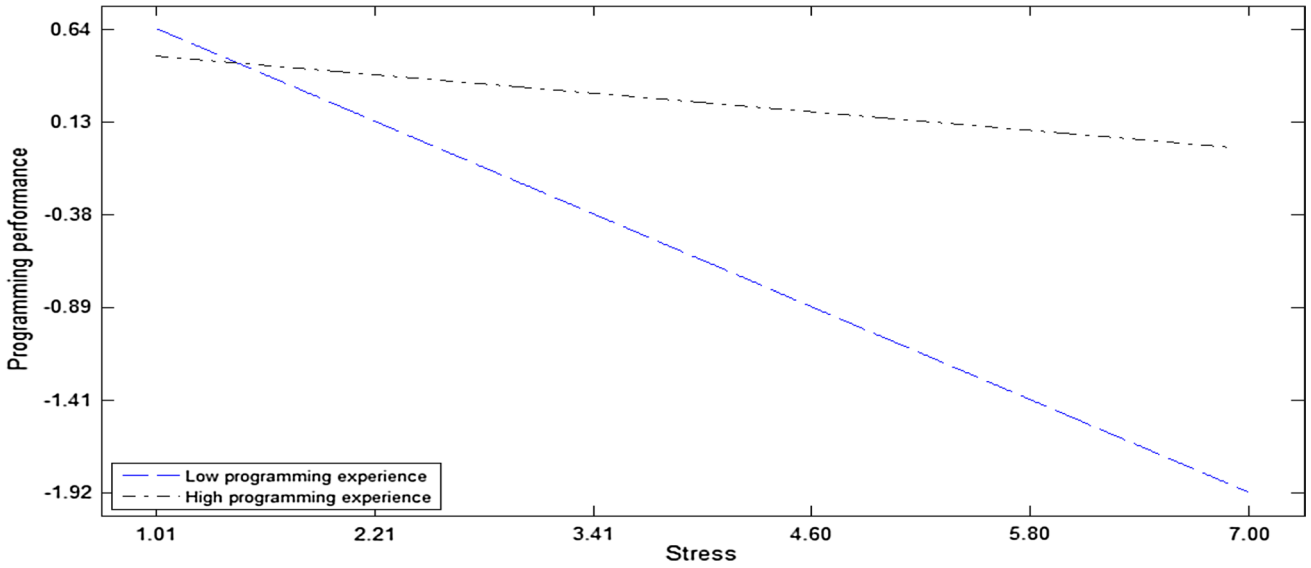

Fig. 4 Moderating effect of programming experience with respect to stress and performance. *Note* Programming performance shown on a standardized scale

well programmers perform under stress. However, this key role of programming experience is played chiefly as a significant moderator of the relationship between stress and programming performance, and not as a significant mediator of the relationship between age and programming performance. Finally, older individuals seem to experience more stress while performing a programming task under time pressure than younger ones.

The end results are: a negative and significant total association between age and programming performance and an important moderating role of programming experience. The latter moderating role can be summarized as follows: given enough programming experience, stress does not seem to matter much regarding programming performance.

Since a programmer's age influences programming performance only via stress, we can also conclude that, given enough programming experience, age does not matter either with respect to programming performance. This seems to happen even though age may appear to strongly influence programming performance, when we look only at age's total association with programming performance and ignore the important moderating effect of programming experience.

6.1 What if the study's participants had been professional programmers?

As we indicated earlier, our study involved student participants and thus provides an initial test of our theoretical model. Note that even though the age of the student participants ranged from 19 to 54 years, programming experience ranged only from 0 to 5 years. Had the participants been professional programmers, we could reasonably expect a much different range of programming experience, perhaps from a few months to more than 30 years. For example, an individual who started working as a programmer at around 20 years of age would have had 34 years of programming experience at age 54. Those extra years of programming experience likely would lead to stronger associations for the links Age → Exp and Exp → Perf. On a bivariate basis, those associations translated to small effect sizes (Cohen 1988) in our investigation with the student participants.

Employing the technique of variation sharing (Kock and Sexton 2017), which builds on the Monte Carlo method (Paxton et al. 2001; Robert and Casella 2013), we simulated the effects that such a wider range of variation in programming experience would have on our study's results. We did so by conservatively strengthening the associations only for the links Age → Exp and Exp → Perf so that they achieved medium effect sizes on a bivariate basis. We restricted ourselves to medium, as opposed to large (Cohen 1988), effect sizes to be conservative. Variation sharing was conducted departing from the original empirical values of the variables Age, Exp, and Perf. That is, as recommended in discussions of the technique (Kock and Sexton 2017), no exogenous variation was introduced into the model.

The new simulated results, summarized in Table 5, should be seen as possible results if our study's participants had been professional programmers. Several of these new results match the results obtained with student participants, from a hypothesis-testing perspective, with a few notable differences. One notable difference is that the coefficient of association for the direct link Age → Stress ($\beta = -0.116$) changed

Table 5 Hypothesized links and revised coefficients of association (professional programmers)

Hypothesized links	Effect type	Coefficient	Significance
Age → Exp	Direct	0.388	$P < 0.001$
Age → Stress	Direct	−0.116	Not significant
Exp → Perf	Direct	0.303	$P < 0.001$
Stress → Perf	Direct	−0.378	$P < 0.001$
Exp → (Stress → Perf)	Moderating	0.351	$P < 0.001$
Age → Stress → Perf	Indirect	0.044	Not significant
Age → Exp → Perf	Indirect	0.118	$P < 0.05$
Age → Perf	Direct	0.250	$P < 0.01$
Age → → Perf	Total	0.412	$P < 0.001$

Exp, programming experience; Perf, programming performance. Results based on simulated data

sign and became statistically nonsignificant, although by a very small margin ($P = 0.081$). This provides some, but limited, support for hypothesis H2, which is that a programmer's age has a *negative* direct association with perceived stress.

Another notable difference is that the coefficient of association for the indirect link Age → Exp → Perf ($\beta = 0.118$) became statistically significant, which provides support for hypothesis H7, which is that a programmer's age has a positive indirect association with programming performance via programming experience. Two other notable differences were that the coefficient of association for the direct link Age → Perf ($\beta = 0.250$) became positive and statistically significant and that the coefficient for the total effect link Age → → Perf ($\beta = 0.412$) also became positive and statistically significant. Figure 5 illustrates this latter result vis-à-vis the corresponding result obtained with student participants.

These new results suggest that the total association between a programmer's age and programming performance is partially mediated, not by perceived stress but by programming experience. They also provide strong support for hypothesis H9, which is that the total association between a programmer's age and programming performance is positive. As we can see, the results of this "what-if" analysis with simulated data provide much stronger support for our theoretical model. They also call for further research with the participation of professional programmers, to complement our investigation with student participants, and further test our theory-based predictions.

7 Discussion

In our initial test of the model, we conducted a software development experiment involving 140 participants. The participants were graduate and undergraduate students majoring in IT-related areas. The setting for the study was a midsized university in the southwestern region of the USA. In the experiment, the participants were asked to individually develop a simple software application during a limited amount of time, which placed them under some stress due to the time pressure. A factor-based partial least squares structural equation modeling analysis was conducted to analyze

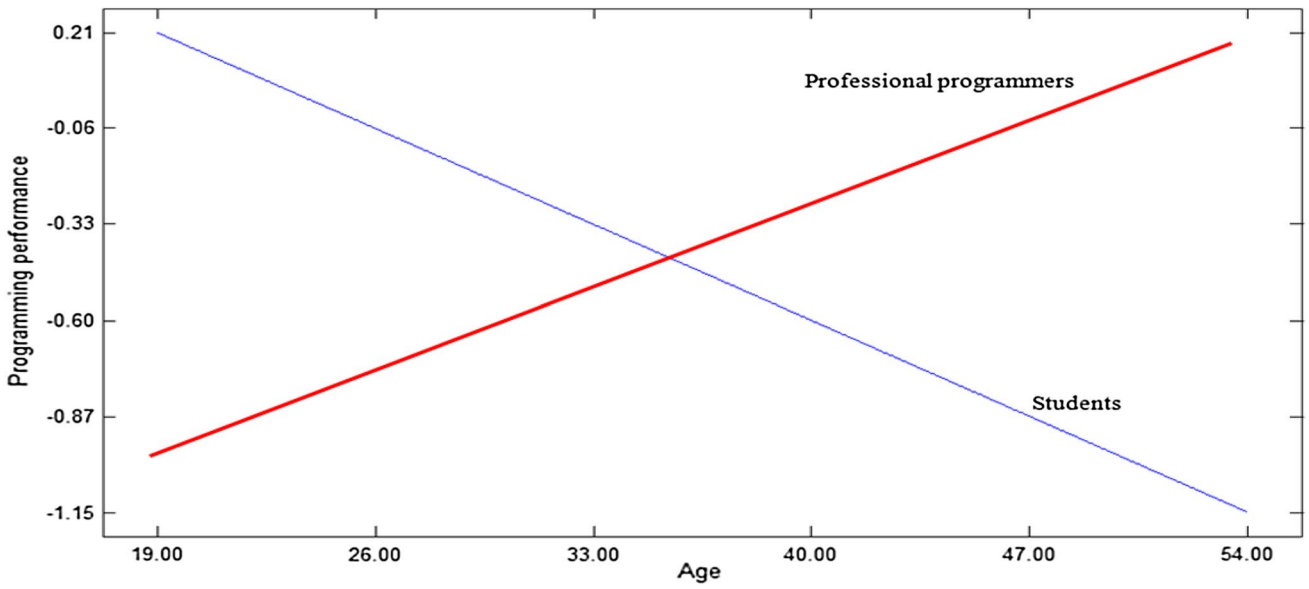

Fig. 5 Total effect of age on programming performance (students vs. professional programmers). *Notes* Programming performance shown on a standardized scale. Downward slope line = students. Upward slope line = professional programmers (simulated data)

the data and test a number of hypotheses related to the theoretical framework we developed.

The results of our analyses with student participants suggest that age is positively and significantly associated with programming experience and perceived stress; the latter finding (regarding stress) going against the framework we developed. The results also suggest that programming experience is positively and significantly associated with programming performance, whereas perceived stress seems to be negatively and significantly associated with programming performance.

Two separate analyses with student participants of mediating effects suggest that while perceived stress is a significant mediator of the association between age and programming performance, programming experience is not a significant mediator of this association. A moderating effect analysis suggests that as programming experience increases, the association between perceived stress and programming performance weakens, going from strong and negative toward neutral.

Let us consider the magnitudes of the indirect effect with student participants of age on programming performance via perceived stress ($\beta = -0.106$), the only significant indirect effect of the two analyzed, and of the moderating effect of programming experience on the association between perceived stress and programming performance ($\beta = 0.242$). These magnitudes suggest that the moderating effect is approximately 2.28 stronger, or 128% greater, than the indirect effect and that these effects control for the direct effect of age. Given this, one can reasonably conclude that, at high levels of programming experience, age does not matter when it comes to programming performance. The reason for this is that, at high levels of programming experience, stress has practically no effect on programming performance; rendering the indirect effect of age on performance via stress insignificant.

Our study with student participants contradicts the AE framework in one key aspect: it suggests that older adults experience more stress than younger ones in a computer programming task. This implies that general mental schemas acquired over time through generic problem-solving experiences do not actually protect older individuals from stress if their task-specific mental schemas are significantly underdeveloped—as in the case of older individuals with low programming experience and a computer programming task.

Past research focusing on stress hormones and their prevalence among individuals of various ages provides some support for this result (Barnes et al. 1982; Whitbourne 2012). However, it is possible that our use of students distorted the results by artificially increasing the proportion of individuals with low programming experience among older participants. Given this, we recommend that our study be replicated with non-student participants, preferably professional programmers. This is a line of research that we intend to pursue in the future. The results of a "what-if" analysis, where we simulated an investigation with professional programmers, support this line of research.

8 Conclusion

There is a widespread perception that older adults are underperformers when compared with younger adults with respect to tasks that involve heavy use of IT (Czaja 1995; Kraft 2012; Perry et al. 2003). One instance of such tasks is computer programming, which also happens to be a key driver of the economy in the USA and other developed countries (Hannah 2014; John 2014). This perception bias against older adults is often referred to by the terms "ageism" and "digital ageism" (Garstka et al. 2004; Magsamen-Conrad et al. 2015; Oh et al. 2016; Vauclair et al. 2016). One of the goals of laws such as the Age Discrimination in Employment Act in the USA is to prevent this type of bias from influencing employment decisions (Neumark 2003, 2009).

Even though the perceptions that underlie digital ageism are commonly held in organizations, as well as society as a whole (Czaja 1995; Kraft 2012; Perry et al. 2003), little if any theoretical and empirical research has been published assessing whether digital ageism has any basis in reality. We tried to fill this gap through the development of a theoretical model and its initial testing in the specific context of computer programming. We conducted a study that enabled us to provide a preliminary answer a fundamental research question: do older programmers perform as well as young ones? The theoretical framework we developed was based on schema theory and provides the conceptual grounding on which one can assume that older adults may in fact perform better than younger one in the task of computer programming.

Our initial test of our theoretical predictions was implemented through a software development experiment involving 140 student participants. We followed this initial test with a simulation to obtain and assess possible results if our study's participants had been professional programmers. In line with our theoretical expectations, this initial test and follow-up simulation study suggest that the effect of stress and thus the indirect effect of age become too weak to be of any practical consequence at high levels of programming experience. Stated differently, we can reasonably conclude based on our analyses that given enough programming experience, age has no relevant effect on programming performance. In fact, our simulation study allows us to conclude that age may be positively associated with programming performance among professional programmers.

From an employment law perspective, our results suggest that laws such as the Age Discrimination in Employment

Act in the USA are generally appropriate in their spirit with respect to IT-related employment involving computer programming tasks (Neumark 2003, 2009). It is sensible to conclude, based on our study, that such laws may prevent unfair discrimination based on stereotypes that have no clear or strong empirical basis. Moreover, age-based discrimination may negatively affect the competitiveness of software development organizations, as well as of other organizations where IT is heavily used to accomplish mission-critical tasks.

While advanced age may be associated with more stress, as past research focusing on stress hormones and their prevalence among individuals of various ages suggests, organizational stress can be reduced via other means such as non-threatening and relaxed work environments (Billings and Moos 1982; Hetherington and Blechman 2014), countering the possible effect of age on stress. Moreover, when we consider the very important role that programming experience plays in influencing programming performance, directly and as a moderator, we can see the wisdom of disregarding age in employment decisions. Our study suggests that, with enough programming experience, older programmers may perform just as well as young ones, if not better.

Appendix 1: Experimental task description

Purpose

The director of the PhD program at a school of business needs to make decisions on whether or not to admit a doctoral program applicant in a timely manner. The director of the PhD program will save time and be more productive if he or she has a stand-alone application at his desktop to make such a decision.

Algorithms

For an applicant to be eligible for admission, he/she must satisfy one of the following sets of conditions:

- Have a GMAT greater than or equal to (≥) 600, a GPA greater than or equal to (≥) 3.5, WE greater than or equal to (≥) 0, and RecLtr greater than or equal to (≥) 80%.
- Have a GMAT greater than or equal to (≥) 500, GPA greater than or equal to (≥) 3.8, WE greater than or equal to (≥) 1, and RecLtr greater than or equal to (≥) 90%.

Legend

- GMAT: Graduate Management Aptitude Test.
- GPA: Grade Point Average.
- WE: Work Experience related to the major.
- RecLtr: Composite rating of the student based on a structured recommendation letter.

Notes

- The application should allow the director of the PhD program to reset all values on the screen to blank so that another calculation can be performed.
- The decision should be run based on the term "Decide," so please include a working button for this. The reset of the values should be designated by the term "Reset," so please include a working button for this term as well. The decision output, once all information is entered into the program and the "Decide" button is clicked should be either, "Admit" or "Not Admit."

Appendix 2: Measurement instrument

The questions and question-statements below were used for data collection, in addition to demographic questions. The questions on perceived stress were answered on a Likert-type scale going from 1 to 7. Programming performance was measured based on a rubric with five dimensions, whereby three researchers independently scored the quality of the software applications developed by the participants.

Perceived stress

Stress1: I felt stressed while completing this task.
Stress2: I felt nervous while completing this task.
Stress3: This task made me feel stressed.
Stress4: Completing this task was stressful.

Programming performance (based on rubric)

Perf1: Completeness of the software application.
Perf2: Correctness of the software application.
Perf3: Extent to which the software application met the requirements.
Perf4: Ease of use of interface.
Perf5: Programming code clarity.

Appendix 3: Programming performance scoring rubric

Below is the rubric we used to score programming performance. Three researchers independently scored the quality of the software applications developed by the participants, from which average scores were calculated for each of the five dimensions. Each individual dimension's score was then included as an indicator, with respect to the programming

performance variable, as part of the measurement model in our structural equation modeling analysis.

	0–25	25–50	50–75	75–100
Completeness	The assignment was incomplete or completed without regard to instructions	The assignment was only partially completed per instructions	The assignment was moderately completed per instructions	The assignment was fully completed per instructions
Correctness	The program did not perform per instructions	The program performed only partially per instructions	The program performed moderately per instructions	The program performed fully per instructions
Requirements met	Adheres to less than 70% of standard	Adheres to between 70 and 80% of standard	Adheres to between 80 and 90% of standard	Adheres to between 90 and 100% of standard
Ease of use	Required user to reread before understood	Required user to reread to confirm understood	Reread was not required to confirm understood	Immediately understood
Code clarity	Code is unclear or too specific to stated purpose to be revised	Code is enough to revise	Code is clear and modular enough to ease revision	Code is clear and general enough to simplify revision

Appendix 4: Coefficients for measurement instrument validation

Loadings, weights, cross-loadings, cross-weights, and indicator effect sizes are summarized in Table 6. Loadings, shown in bold, are from a structure matrix and thus unrotated; cross-loadings, shown in italics, are from a pattern matrix and thus oblique-rotated (Ehremberg and Goodhart 1976; Thompson 2004). This combination of structure and pattern matrices' loadings allows for easy identification of possible validity problems, while at the same time obviating the need for a potentially distorting normalization procedure (Ferguson 1981; Kock 2015b; Ogasawara 1999).

R-squared, adjusted *R*-squared, composite reliability, Cronbach's alpha, average variance extracted, and *Q*-squared coefficients are listed in Table 7. Composite reliability and Cronbach's alpha coefficients are reliability measures (Fornell and Larcker 1981; Nunnaly 1978; Nunnally and Bernstein 1994). Average variances extracted are sometimes used for convergent validity assessment, in addition to loadings (Fornell and Larcker 1981). *Q*-squared coefficients are used, together with *R*-squared coefficients, for predictive validity assessment (Geisser 1974; Kock 2015b; Stone 1974).

Latent variable correlations and square roots of average variances extracted are listed in Table 8. These coefficients are used for discriminant validity assessment; that is, to assess whether measures associated with each latent variable are not confused by respondents with measures associated with other latent variables (Fornell and Larcker 1981; Kock 2014; Schumacker and Lomax 2004).

Table 9 shows variance inflation factors (Hair et al. 2009) from a full collinearity test. In a full collinearity test, variance inflation factors are calculated for all of the variables in the model (Kock and Lynn 2012). This allows for the assessment of whole-model collinearity in the presence of

Table 6 Combined loadings and cross-loadings, weights, and effect sizes

	Loadings and cross-loadings			Weights			ES
	Stress	Perf	*P* value	Stress	Perf	*P* value	
Stress1	**0.945**	*−0.002*	< 0.001	**0.240**	*0*	0.002	0.226
Stress2	**0.883**	*0.113*	< 0.001	**0.222**	*0*	0.003	0.196
Stress3	**0.970**	*0.009*	< 0.001	**0.241**	*0*	0.002	0.234
Stress4	**0.957**	*−0.024*	< 0.001	**0.249**	*0*	0.001	0.239
Stress5	**0.639**	*−0.024*	< 0.001	**0.164**	*0*	0.023	0.105
Perf1	*0.031*	**0.958**	< 0.001	*0*	**0.216**	0.004	0.207
Perf2	*0.041*	**0.936**	< 0.001	*0*	**0.208**	0.006	0.194
Perf3	*0.003*	**0.967**	< 0.001	*0*	**0.221**	0.003	0.214
Perf4	*0.016*	**0.935**	< 0.001	*0*	**0.206**	0.006	0.193
Perf5	*−0.039*	**0.924**	< 0.001	*0*	**0.208**	0.005	0.192

Loadings and weights shown in bold, cross-loadings and cross-weights in italics; Stress, perceived stress; Perf, programming performance; ES, effect size

Table 7 Latent variable coefficients

	Stress	Perf
R-squared coefficients	0.057	0.366
Adjusted *R*-squared coefficients	0.050	0.337
Composite reliability coefficients	0.948	0.976
Cronbach's alpha coefficients	0.927	0.969
Average variances extracted	0.788	0.891
Q-squared coefficients	0.059	0.364

Stress, perceived stress; Perf, programming performance; ES, effect size

variables measured through single indicators. Full collinearity variance inflation factors can also be used in common method bias tests (Kock 2015c).

The measurement model assessment results summarized in the tables above suggest that the measurement instrument presents acceptable convergent validity, discriminant validity, and reliability. These also suggest that the measurement instrument presents acceptable predictive validity. Finally, these results above suggest that the measurement instrument is free from model-wide collinearity and that common method variance does not have a significant biasing effect in the analysis.

Table 8 Latent variable correlations and square roots of average variances extracted

	Stress	Perf	GPA	Age	Sex (M/F)	Exp
Stress	**0.888**	−0.524	−0.204	0.221	−0.154	−0.142
Perf	−0.524	**0.944**	0.144	−0.246	0.158	0.212
GPA	−0.204	0.144	**1**	−0.001	0.026	0.071
Age	0.221	−0.246	−0.001	**1**	−0.124	0.135
Sex (M/F)	−0.154	0.158	0.026	−0.124	**1**	0.210
Exp	−0.142	0.212	0.071	0.135	0.210	**1**

Square roots of average variances extracted shown on diagonal within bold; Stress, perceived stress; Perf, programming performance; Exp, programming experience

Table 9 Full collinearity variance inflation factors

Stress	Perf	GPA	Age	Sex (M/F)	Exp	Exp*Stress
1.452	1.576	1.051	1.186	1.092	1.448	1.423

Stress, perceived stress; Perf, programming performance; Exp, programming experience; Exp*Stress, interaction variable used to implement the moderating effect: Exp → (Stress → Perf)

References

Akerstedt T, Gillberg M (1990) Subjective and objective sleepiness in the active individual. Int J Neurosci 52(1–2):29–37

Bailey J, Mitchell RB (2006) Industry perceptions of the competencies needed by computer programmers: technical, business, and soft skills. J Comput Inf Syst 47(2):28–33

Barnes RF, Raskind M, Gumbrecht G, Halter JB (1982) The effects of age on the plasma catecholamine response to mental stress in man. J Clin Endocrinol Metab 54(1):64–69

Bartlett F (1932) Remembering: a study in experimental and social psychology. Cambridge University Press, Cambridge

Bartlett F (1958) Thinking: an experimental and social study. Basic Books, New York

Beckers JJ, Rikers RM, Schmidt HG (2006) The influence of computer anxiety on experienced computer users while performing complex computer tasks. Comput Hum Behav 22(3):456–466

Bera AK, Jarque CM (1981) Efficient tests for normality, homoscedasticity and serial independence of regression residuals: Monte Carlo evidence. Econ Lett 7(4):313–318

Bergin S, Reilly R (2005) Programming: factors that influence success. ACM SIGCSE Bull 37(1):411–415

Billings AG, Moos RH (1982) Work stress and the stress-buffering roles of work and family resources. J Organ Behav 3(3):215–232

Brosnan MJ (1998) The impact of computer anxiety and self-efficacy upon performance. J Comput Assist Learn 14(3):223–234

Burgess GA (2005) Introduction to programming: blooming in America. J Comput Sci Coll 21(1):19–28

Byrne P, Lyons G (2001) The effect of student attributes on success in programming. ACM SIGCSE Bull 33(3):49–52

Caplan LJ, Schooler C (1990) The effects of analogical training models and age on problem-solving in a new domain. Exp Aging Res 16(3):151–154

Catherine BC, Wheeler DD (1994) The Myers-Briggs personality type and its relationship to computer programming. J Res Comput Educ 26(3):358–370

Chan DKC, Yang SX, Hamamura T, Sultan S, Xing S, Chatzisarantis NL, Hagger MS (2015) In-lecture learning motivation predicts students' motivation, intention, and behaviour for after-lecture learning: examining the trans-contextual model across universities from UK, China, and Pakistan. Motiv Emot 39(6):908–925

Cohen J (1988) Statistical power analysis for the behavioral sciences. Lawrence Erlbaum, Hillsdale

Cohen S, Kamarck T, Mermelstein R (1983) A global measure of perceived stress. J Health Soc Behav 24(4):385–396

Cohen I, Brinkman WP, Neerincx MA (2015) Modelling environmental and cognitive factors to predict performance in a stressful training scenario on a naval ship simulator. Cogn Technol Work 17(4):503–519

Cossete P, Audet M (1992) Mapping of an idiosyncratic schema. J Manage Stud 29(3):325–348

Czaja SJ (1995) Aging and work performance. Rev Public Pers Adm 15(2):46–61

Dermentzi E, Papagiannidis S, Toro CO, Yannopoulou N (2016) Academic engagement: differences between intention to adopt Social Networking Sites and other online technologies. Comput Hum Behav 61(1):321–332

Dibiase D, Kidwai K (2010) Wasted on the young? Comparing the performance and attitudes of younger and older US adults in an online class on geographic information. J Geogr High Educ 34(3):299–326

Dollinger SMC (1995) Mental rotation performance: age, sex, and visual field differences. Dev Neuropsychol 11(2):215–222

Dönmez D, Grote G, Brusoni S (2016) Routine interdependencies as a source of stability and flexibility. A study of agile software development teams. Inf Organ 26(3):63–83

Duschl KC, Gramß D, Obermeier M, Vogel-Heuser B (2015) Towards a taxonomy of errors in PLC programming. Cogn Technol Work 17(3):417–430

Dyck JL, Smither JAA (1994) Age differences in computer anxiety: the role of computer experience, gender and education. J Educ Comput Res 10(3):239–248

Ehremberg ASC, Goodhart GJ (1976) Factor analysis: limitations and alternatives. Marketing Science Institute, Cambridge

Elias SM, Smith WL, Barney CE (2012) Age as a moderator of attitude towards technology in the workplace: work motivation and overall job satisfaction. Behav Inf Technol 31(5):453–467

Ferguson GA (1981) Statistical analysis in psychology and education. McGraw-Hill, New York

Fornell C, Larcker DF (1981) Evaluating structural equation models with unobservable variables and measurement error. J Mark Res 18(1):39–50

Gardner H (1985) The mind's new science. Basic Books, New York

Garstka TA, Schmitt MT, Branscombe NR, Hummert ML (2004) How young and older adults differ in their responses to perceived age discrimination. Psychol Aging 19(2):326–335

Geisser S (1974) A predictive approach to the random effects model. Biometrika 61(1):101–107

Gel YR, Gastwirth JL (2008) A robust modification of the Jarque-Bera test of normality. Econ Lett 99(1):30–32

Gilroy FD, Desai HB (1986) Computer anxiety: sex, race and age. Int J Man Mach Stud 25(6):711–719

Gioia DA, Manz CC (1985) Linking cognition and behavior: a script processing interpretation of vicarious learning. Acad Manag Rev 10(3):527–539

Gnambs T (2015) What makes a computer wiz? Linking personality traits and programming aptitude. J Res Pers 58(3):31–34

González A, Ramírez MP, Viadel V (2012) Attitudes of the elderly toward information and communications technologies. Educ Gerontol 38(9):585–594

Haenlein M, Kaplan AM (2004) A beginner's guide to partial least squares analysis. Underst Stat 3(4):283–297

Hagan D, Markham S (2000) Does it help to have some programming experience before beginning a computing degree program? ACM SIGCSE Bull 32(3):25–28

Hair JF, Black WC, Babin BJ, Anderson RE (2009) Multivariate data analysis. Prentice Hall, Upper Saddle River

Hannah L (2014) The rise of the modern firm. Bus Hist 56(5):845–846

Hasan B (2003) The influence of specific computer experiences on computer self-efficacy beliefs. Comput Hum Behav 19(4):443–450

Hetherington EM, Blechman EA (2014) Stress, coping, and resiliency in children and families. Psychology Press, New York

Huang LK (2015) Exploring factors affecting top management support of IT implementation: a stakeholder perspective in hospital. J Inf Technol Manag 26(1):31–45

Jaradat MIRM, Faqih KM (2014) Investigating the moderating effects of gender and self-Efficacy in the context of mobile payment adoption: a developing country perspective. Int J Bus Manag 9(11):147

Jarque CM, Bera AK (1980) Efficient tests for normality, homoscedasticity and serial independence of regression residuals. Econ Lett 6(3):255–259

John RR (2014) The computer boys take over: computers, programmers, and the politics of technical expertise. Bus Hist 56(5):846–847

Johnson KM (2015) Non-technical skills for IT professionals in the landscape of Social Media. Am J Bus Manag 4(3):102–122

Khan IA, Brinkman WP, Hierons RM (2011) Do moods affect programmers' debug performance? Cogn Technol Work 13(4):245–258

Kline RB (1998) Principles and practice of structural equation modeling. The Guilford Press, New York
Kock N (2014) Advanced mediating effects tests, multi-group analyses, and measurement model assessments in PLS-based SEM. Int J e-Collab 10(3):1–13
Kock N (2015a) A note on how to conduct a factor-based PLS-SEM analysis. Int J e-Collab 11(3):1–9
Kock N (2015b) WarpPLS 5.0 user manual. ScriptWarp Systems, Laredo
Kock N (2015c) Common method bias in PLS-SEM: a full collinearity assessment approach. Int J e-Collab 11(4):1–10
Kock N (2016) Non-normality propagation among latent variables and indicators in PLS-SEM simulations. J Mod Appl Stat Methods 15(1):299–315
Kock N, Chatelain-Jardón R (2016) Surprise-enhanced and technology-mediated learning: a two-country study. Cogn Technol Work 18(1):105–119
Kock N, Lynn GS (2012) Lateral collinearity and misleading results in variance-based SEM: an illustration and recommendations. J Assoc Inf Syst 13(7):546–580
Kock N, Mayfield M (2015) PLS-based SEM algorithms: the good neighbor assumption, collinearity, and nonlinearity. Inf Manag Bus Rev 7(2):113–130
Kock N, Sexton S (2017) Variation sharing: a novel numeric solution to the path bias underestimation problem of PLS-based SEM. Int J Strateg Decis Sci 8(4):46–68
Kraft P (2012) Programmers and managers: the routinization of computer programming in the United States. Springer, New York
Lohmöller J-B (1989) Latent variable path modeling with partial least squares. Physica, Heidelberg
Lord RG, Maher KJ (1990) Alternative information-processing models and their implications for theory, research, and practice. Acad Manag Rev 15(1):9–28
Magsamen-Conrad K, Upadhyaya S, Joa CY, Dowd J (2015) Bridging the divide: using UTAUT to predict multigenerational tablet adoption practices. Comput Hum Behav 50(3):186–196
Maier C, Laumer S, Weinert C, Weitzel T (2015) The effects of technostress and switching stress on discontinued use of social networking services: a study of Facebook use. Inf Syst J 25(3):275–308
Martin MA (2007) Bootstrap hypothesis testing for some common statistical problems: a critical evaluation of size and power properties. Comput Stat Data Anal 51(12):6321–6342
Morrell W, Park DC, Mayhorn CB, Kelley CLR (2000) Effects of age and instructions on teaching older adults to use Eldercomm, an electronic bulletin board system. Educ Gerontol 26(3):221–235
Neumark D (2003) Age discrimination legislation in the United States. Contemp Econ Policy 21(3):297–317
Neumark D (2009) The Age Discrimination in Employment Act and the challenge of population aging. Res Aging 31(1):41–68
Nunnally JC, Bernstein IH (1994) Psychometric theory. McGraw-Hill, New York
Nunnaly J (1978) Psychometric theory. McGraw Hill, New York
Ogasawara H (1999) Standard errors for the direct oblimin solution with Kaiser's normalization. Jpn J Psychol 70(4):333–338
Oh SY, Bailenson J, Weisz E, Zaki J (2016) Virtually old: embodied perspective taking and the reduction of ageism under threat. Comput Hum Behav 60(3):398–410
Paxton P, Curran PJ, Bollen KA, Kirby J, Chen F (2001) Monte Carlo experiments: design and implementation. Struct Equ Model 8(2):287–312
Perry EL, Simpson PA, NicDomhnaill OM, Siegel DM (2003) Is there a technology age gap? Associations among age, skills, and employment outcomes. Int J Sel Assess 11(2):141–149
Potosky D (2002) A field study of computer efficacy beliefs as an outcome of training: the role of computer playfulness, computer knowledge, and performance during training. Comput Hum Behav 18(3):241–255
Ramalingam V, Wiedenbeck S (1998) Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. J Educ Comput Res 19(4):367–381
Robert C, Casella G (2013) Monte Carlo statistical methods. Springer, New York
Rosenthal R, Rosnow RL (2007) Essentials of behavioral research: methods and data analysis. McGraw Hill, Boston
Rubio MA, Romero-Zaliz R, Mañoso C, Angel P (2015) Closing the gender gap in an introductory programming course. Comput Educ 82(2):409–420
Rumelhart DE (1978) Schemata: the building blocks of cognition. Center for Human Information Processing, University of California, San Diego, San Diego
Schumacker RE, Lomax RG (2004) A beginner's guide to structural equation modeling. Lawrence Erlbaum, Mahwah
Sorensen LJ, Stanton NA (2015) Exploring compatible and incompatible transactions in teams. Cogn Technol Work 17(3):367–380
Soror AA, Hammer BI, Steelman ZR, Davis FD, Limayem MM (2015) Good habits gone bad: explaining negative consequences associated with the use of mobile phones from a dual-systems perspective. Inf Syst J 25(4):403–427
Stone M (1974) Cross-validatory choice and assessment of statistical predictions. J Roy Stat Soc B 36(1):111–147
Thompson B (2004) Exploratory and confirmatory factor analysis: understanding concepts and applications. American Psychological Association, Washington
Tse D, Langston RF, Kakeyama M, Bethus I, Spooner PA, Wood ER, Witter MP, Morris RG (2007) Schemas and memory consolidation. Science 316(5821):76–82
Vauclair CM, Lima ML, Abrams D, Swift HJ, Bratt C (2016) What do older people think that others think of them, and does it matter? The role of meta-perceptions and social norms in the prediction of perceived age discrimination. Psychol Aging 31(7):699
Whitbourne SK (2012) The aging body: physiological changes and psychological consequences. Springer, New York