Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise

E. Arisholm et al. 2007

Ivan Ítalo Ituassú

Department of Computer Science - Federal University of Minas Gerais (UFMG)

Software Quality and Measurement - 2015

Overview

- **Authors:**
  Erik Arisholm, Hans Gallis, Tore Dyba and Dag I.K. Sjøberg

- **Objective:**

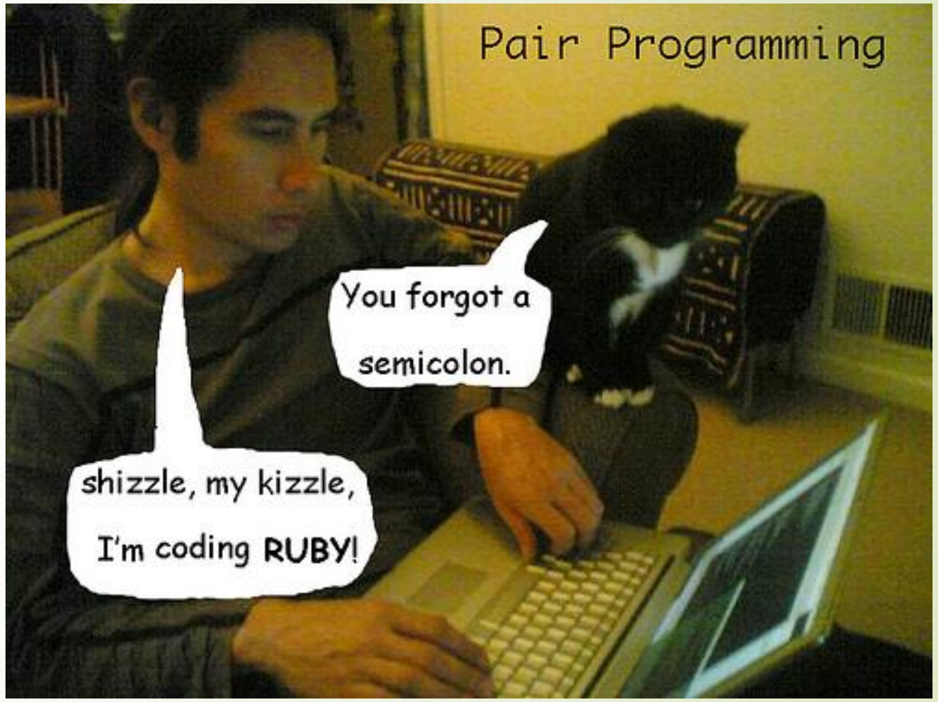
Perform an Empirical experiment on Pair Programming to evaluate hypotheses about the benefits of it

Agenda

- Introduction/Motivation
- Concepts
- Empirical Study
- Threats to Validity
- Analysis
- Conclusion

Introduction

- Pair Programming (PP) is not new, but focus only came due to **Extreme Programming**.
- PP is expected to **increase software quality** without impacting time to deliver.
- Some PP studies concluded that **correctness increased and time duration decreased** (Time to Market). However, one experiment didn't show any positive effects of PP.



Motivation

The problems:

- Most of the existing studies **can't be compared directly**;
- They haven't accounted for the moderating effect of the **complexity** of the programming tasks;
- System complexity and programmer expertise would have a significant impact on **when** and **how** PP is beneficial compared with individual programming.

Motivation

Thus, the paper’s Quasi-Experiment:

- *What is the effect regarding **duration**, **effort**, and **correctness** of pair programming for **various levels** of system **complexity** and programmer **expertise** when performing change tasks?*

Concepts

- **Pair Programming**

Two programmers work on the same task using one computer and keyboard.

Two distinct roles:

1) **a driver**, who types at the keyboard and focuses on the details of the coding
2) **a navigator**, who actively observes the work of the driver, looking for tactical and strategic defects, thinking of alternatives, writing down "things-to-do," and looking up references.

In addition to coding, PP also involves other phases of the software development process, such as design and testing.

Concepts

- **Quasi-Experiment**

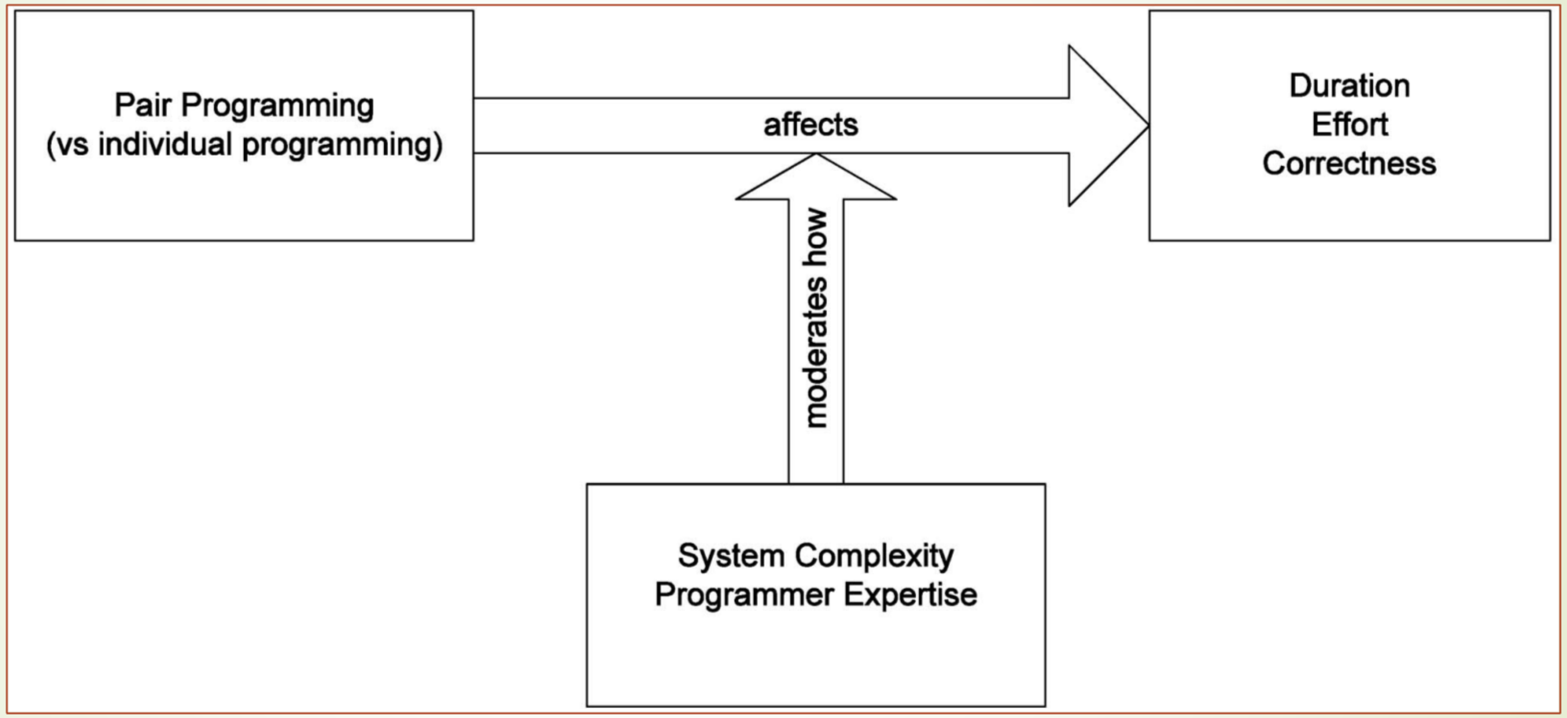
Aims to determine whether a program or intervention has the intended effect on a study's participants.

May best be defined as lacking key components of a true experiment.

While a true experiment includes (1) ***pre-post test design***, (2) a ***treatment group and a control group***, and (3) ***random assignment*** of study participants, quasi-experimental studies lack one or more of these design elements.

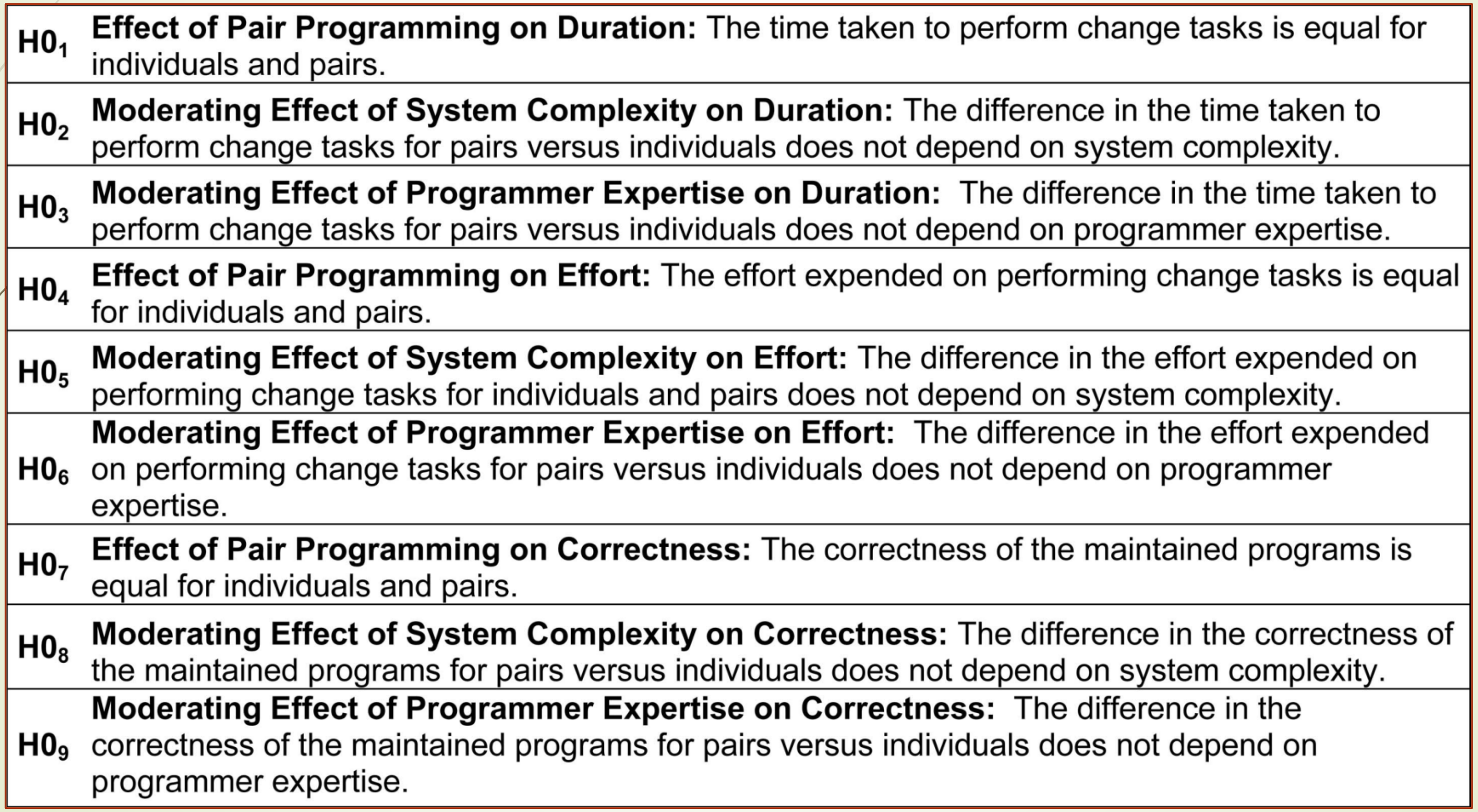
Design of the Experiment

Conceptual Model and Hypotheses



- Fig. 1. Conceptual research model of the hypothesized effects of pair programming.

Conceptual Model and Hypotheses



$H0_1$	**Effect of Pair Programming on Duration:** The time taken to perform change tasks is equal for individuals and pairs.
$H0_2$	**Moderating Effect of System Complexity on Duration:** The difference in the time taken to perform change tasks for pairs versus individuals does not depend on system complexity.
$H0_3$	**Moderating Effect of Programmer Expertise on Duration:** The difference in the time taken to perform change tasks for pairs versus individuals does not depend on programmer expertise.
$H0_4$	**Effect of Pair Programming on Effort:** The effort expended on performing change tasks is equal for individuals and pairs.
$H0_5$	**Moderating Effect of System Complexity on Effort:** The difference in the effort expended on performing change tasks for individuals and pairs does not depend on system complexity.
$H0_6$	**Moderating Effect of Programmer Expertise on Effort:** The difference in the effort expended on performing change tasks for pairs versus individuals does not depend on programmer expertise.
$H0_7$	**Effect of Pair Programming on Correctness:** The correctness of the maintained programs is equal for individuals and pairs.
$H0_8$	**Moderating Effect of System Complexity on Correctness:** The difference in the correctness of the maintained programs for pairs versus individuals does not depend on system complexity.
$H0_9$	**Moderating Effect of Programmer Expertise on Correctness:** The difference in the correctness of the maintained programs for pairs versus individuals does not depend on programmer expertise.

PP

$H0_1$, $H0_4$, $H0_7$

SC

$H0_2$, $H0_5$, $H0_8$

PE

$H0_3$, $H0_6$, $H0_9$

Design of the Experiment

- A **two-phase** experiment with a total sample of 295 Java consultants (98 pairs and 99 individuals) from 29 consultancy companies.
- The **first phase** of the experiment was conducted on individual developers in 2001.
- the **second phase** was conducted on developer pairs in the second half of 2004 and the first half of 2005.
- First, all subjects performed a **pretest task**, the results of which were used to adjust for skill differences between these two groups. Subsequently, the subjects performed change tasks on two alternative Java systems based on a **centralized or delegated control style**, respectively.

Dependent Variables

- **Duration:** elapsed time taken to perform a set of change tasks, considering only duration for subjects whose work was correct.
- **Effort:** total number of programmer hours taken to develop a correct program. Thus, effort equals duration for the individuals and twice the duration for the pair programmers.
- **Correctness:** whether or not the final, maintained program, possessed the required functionality.

Population and Sampling Procedure

- A **power analysis** was performed to calculate sample size (N).
- The target population of the experiment was **professional Java consultants**. (99 individuals and 98 pairs).
- A **project manager** in each company **selected** the subjects from the company's pool of consultants **and rated** them according to their Java programming experience.
- The pairs were formed based on the individuals' programmer **category and their PP experience**. Only 10 subjects claimed to have PP experience, which constituted five of the 98 pairs.

Population and Sampling Procedure

- The pairs were constituted from two programmers within the same company.
- In the second phase of the experiment (2004/2005), it was difficult to recruit junior developers, probably because the companies employed few new graduates after the decline in the IT market in 2001/2002.

Tasks

- The experiment included the programming of six change tasks: a training task, a pretest task (t1), and four (incremental) main experimental tasks (t2, t3, t4, and t5).
- **Individual Training Task:** All the subjects were asked to change a small program so that it could read numbers from the keyboard and print them out in reverse order.
- **Individual Pretest Task (Task 1: ATM):** The change consisted of adding transaction log functionality and printing an account statement for a bank teller machine and was not related to the main experiment tasks.

Tasks

- **Main Experiment Tasks (Tasks 2-5: Coffee Machine):**

Based on two Java systems: centralized and delegated control.

Centralized: a few large ”control classes” coordinate a set of simple classes.
Delegated: responsibilities are distributed among a number of classes.

The tasks consisted of four incremental changes to the coffee machine:

- **t2.** Implement a coin return button.
- **t3.** Introduce bouillon as a new drink choice.
- **t4.** Check whether all ingredients are available for the selected drink.
- **t5.** Make one’s own drink by selecting from the available ingredients.

Tasks

- **Special Last Task (the Time Sink Task):**

The final change task in an experiment needs special attention as a result of potential "ceiling effects."

Not included in the analysis!

The presence of the last task helped to put time pressure on the subjects during the experiment.

Design of the Experiment

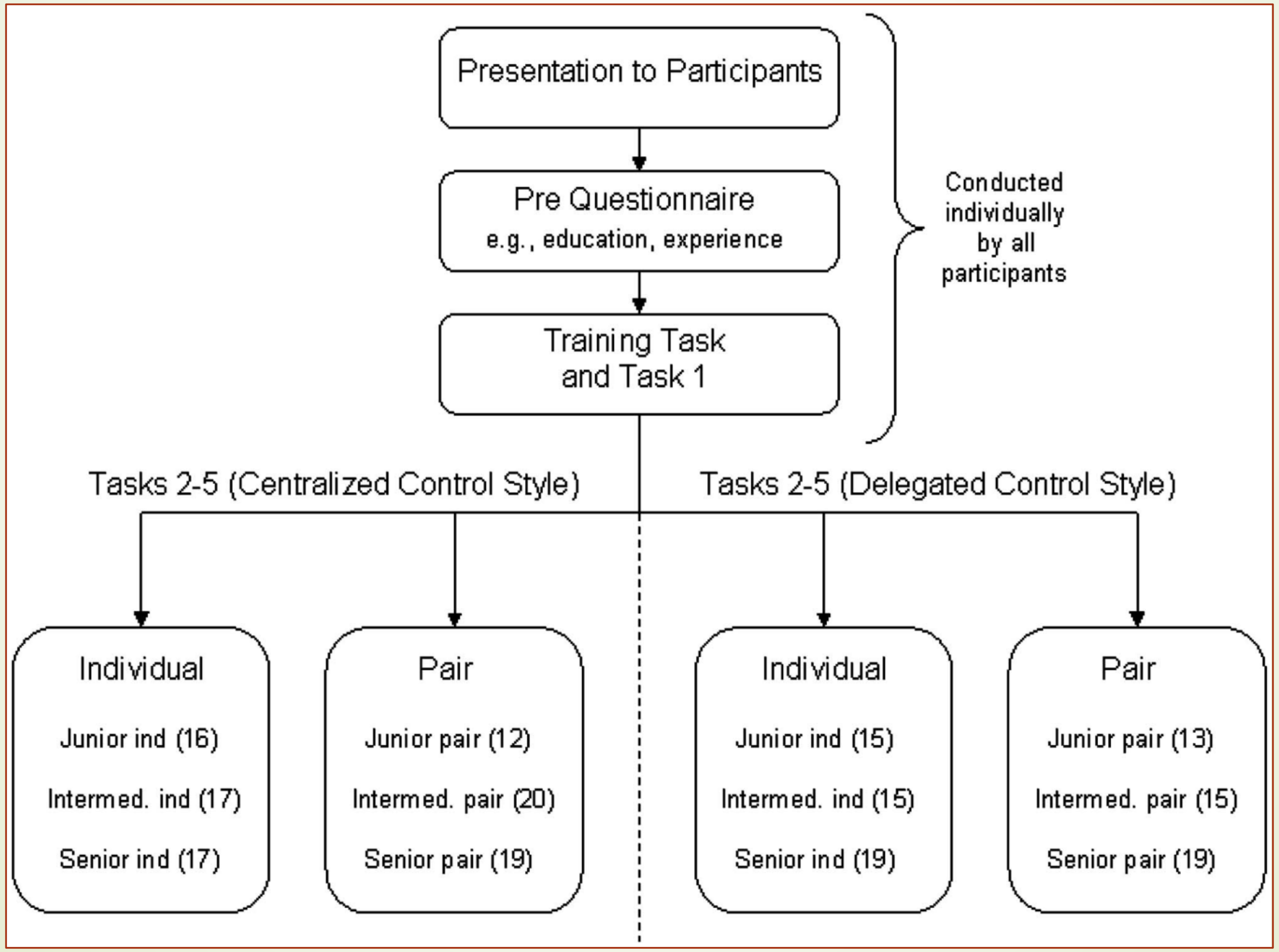


Fig. 2.

Experimental design.

Execution

Execution

- The experiment was conducted incrementally in 27 separate sessions on separate days (for the 29 companies).
- The experiment was conducted in the subjects' own offices, where each developer would normally work, or in offices at Simula Research Laboratory.
- Work at Simula was similar to working at a client's site. The subjects had access to the Internet, printers, libraries, coffee, and so on, as in any other project they might be working on.
- Each subject also used a Java development tool of their own choice, e.g., JBuilder, Eclipse, IntelliJ, NetBeans, or Emacs and Javac.

Execution

- The authors wanted the subjects to perform the tasks with satisfactory quality in as short a time as possible because most software engineering jobs impose relatively severe time constraints on the tasks to be performed.

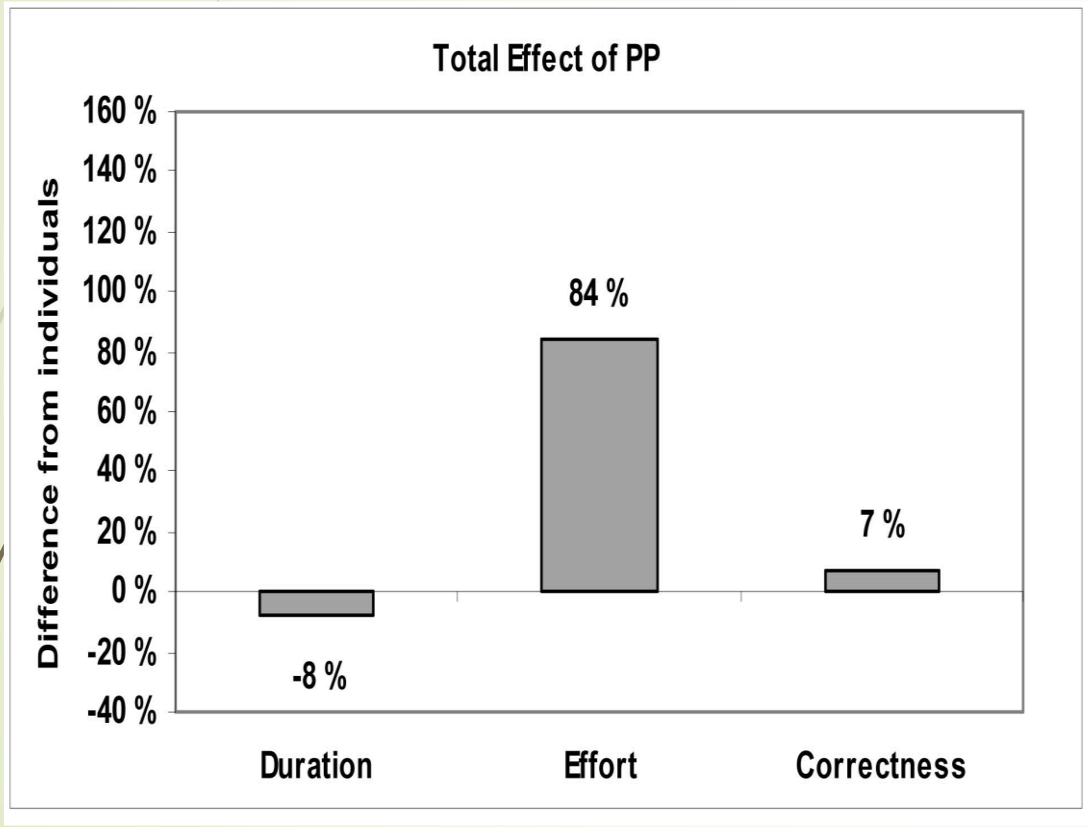
1) Instead of offering an hourly rate, they offered a ”fixed” honorarium based on an estimate that the work would take 5 hours to complete.
2) They introduced a special last, time-sink task.
3) The subjects were allowed to leave when they finished. Those who did not finish had to leave after 8 hours.

Execution

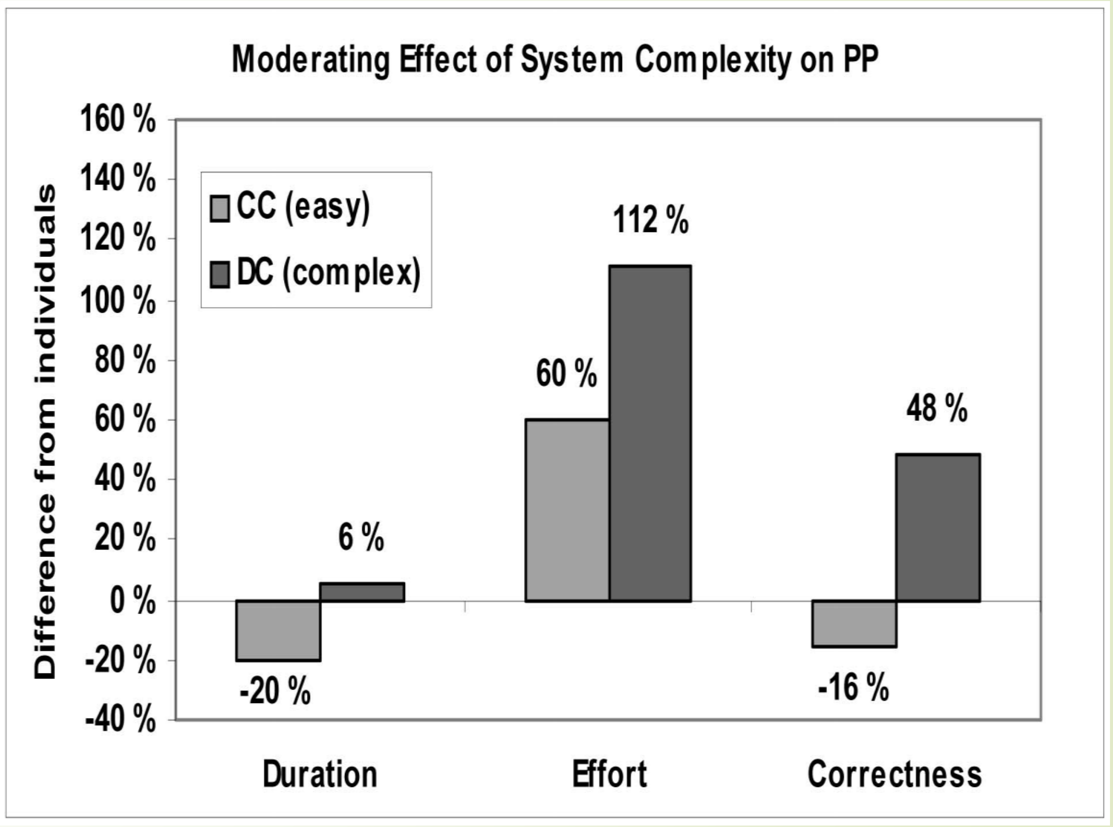
- **To perform the correctness analysis**, one consultant first developed a tool that automatically unpacked and built the source code corresponding to each task solution. Each one was tested using an automated test script.
- **To perform the final grading** of the task solutions, a grading tool was developed that enabled the consultants to view the source code, the difference in source code, the expected and actual test case output, and the difference between the two.
- One consultant was responsible for phase 1, the other for phase 2, but, essentially, the **correctness scores of both phases were based on a consensus** between the two consultants.

Results

Results

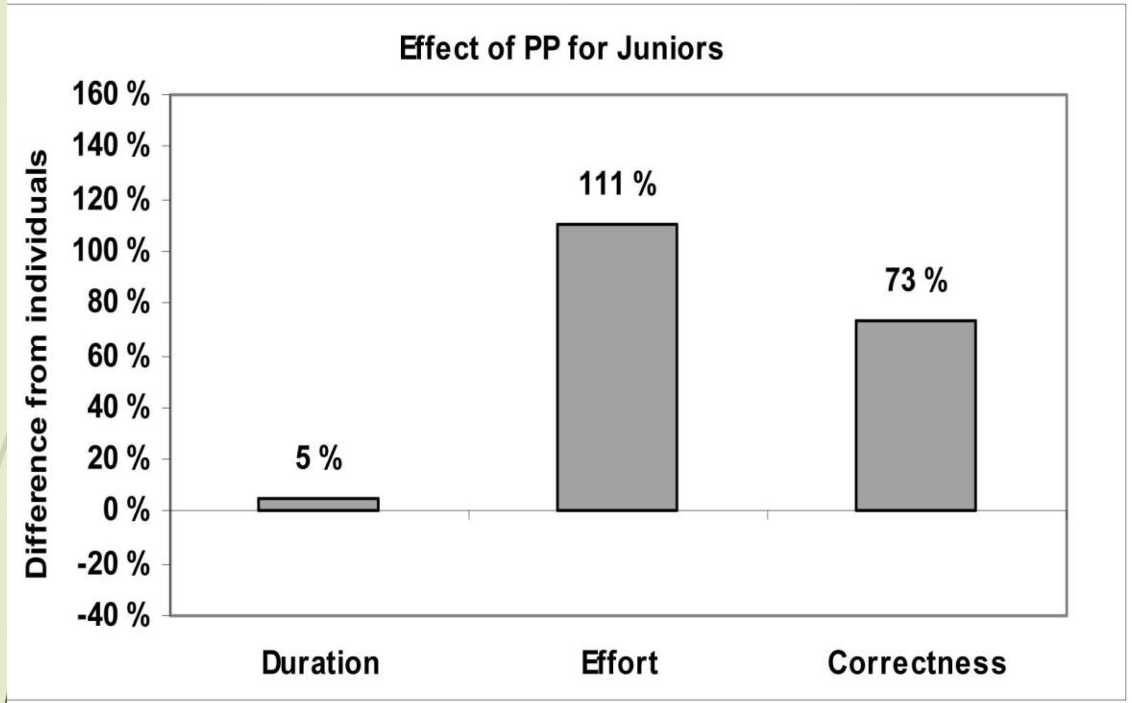


Total Effect of PP
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
-8 %
84 %
7 %
Duration
Effort
Correctness

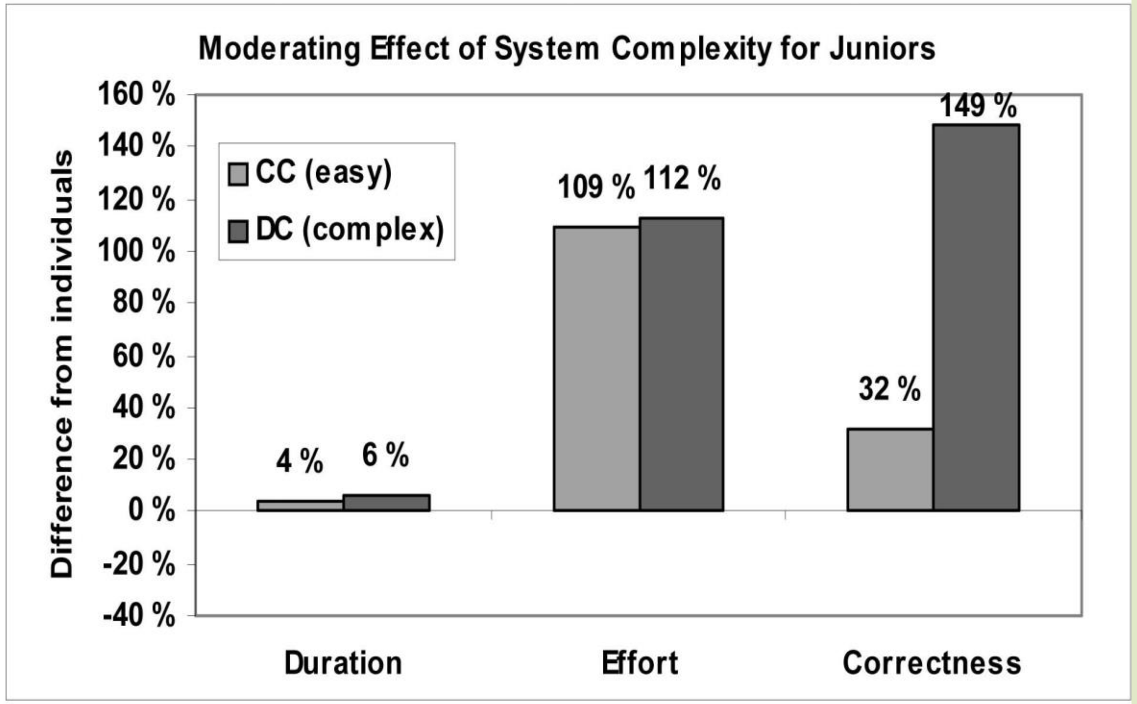


Moderating Effect of System Complexity on PP
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
CC (easy)
DC (complex)
-20 %
6 %
60 %
112 %
-16 %
48 %
Duration
Effort
Correctness

Results

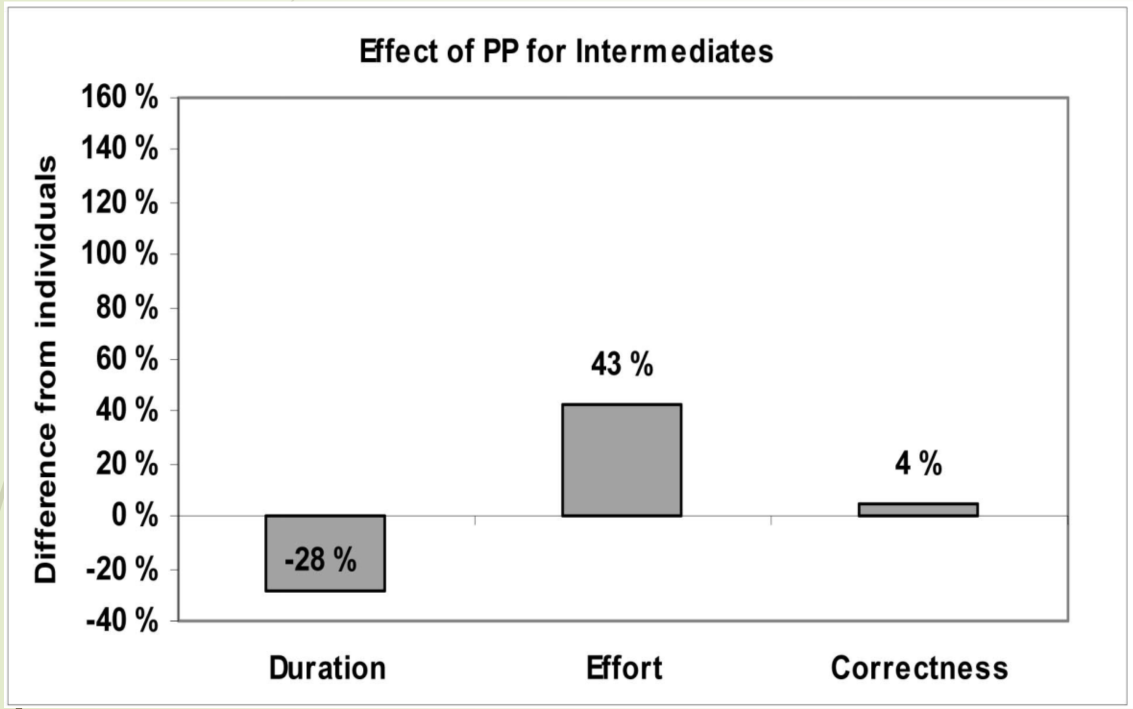


Effect of PP for Juniors
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
5 %
111 %
73 %
Duration
Effort
Correctness

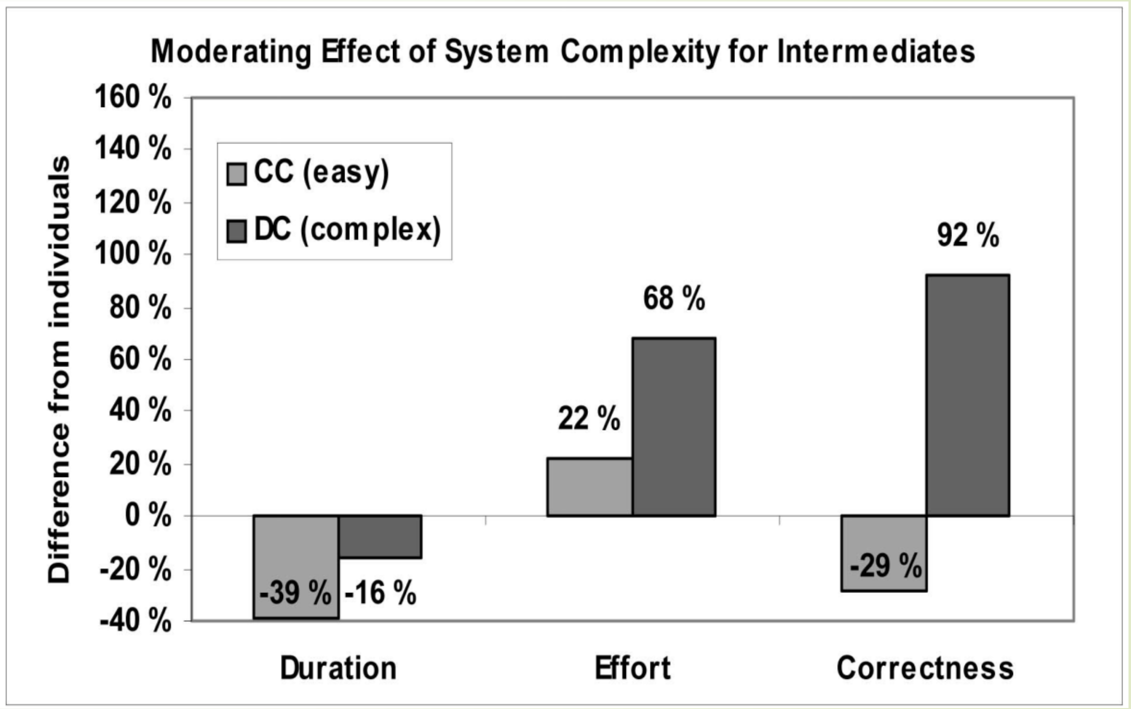


Moderating Effect of System Complexity for Juniors
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
CC (easy)
DC (complex)
4 %
6 %
109 %
112 %
32 %
149 %
Duration
Effort
Correctness

Results

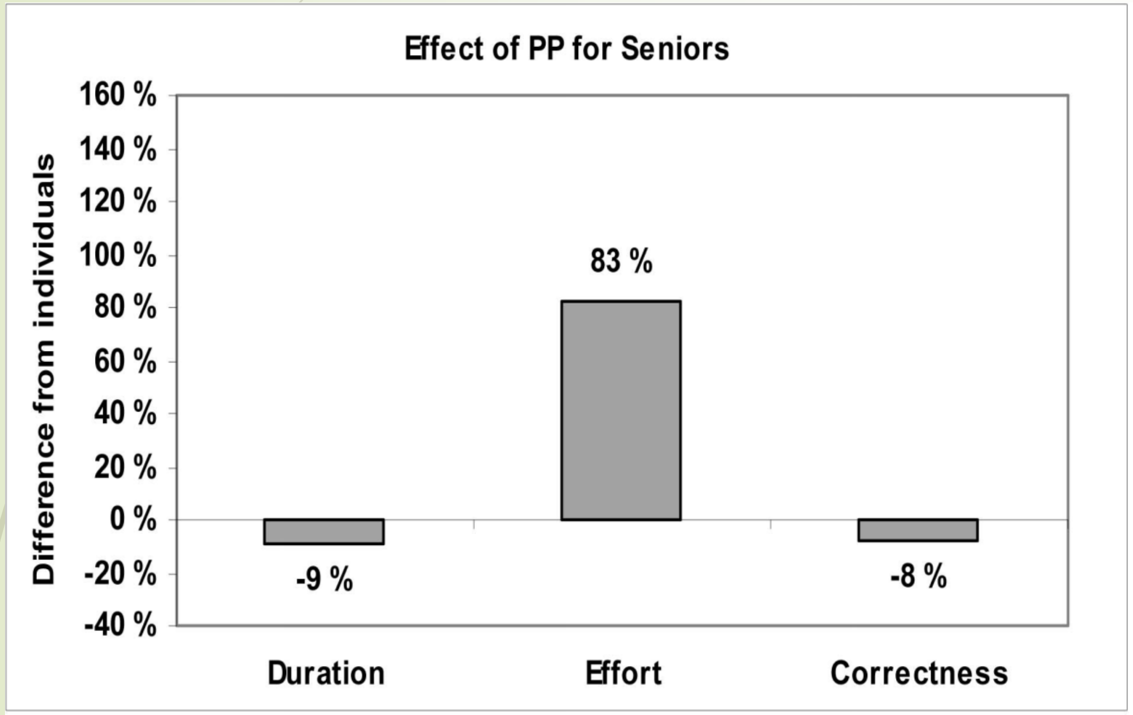


Effect of PP for Intermediates
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
-28 %
43 %
4 %
Duration
Effort
Correctness

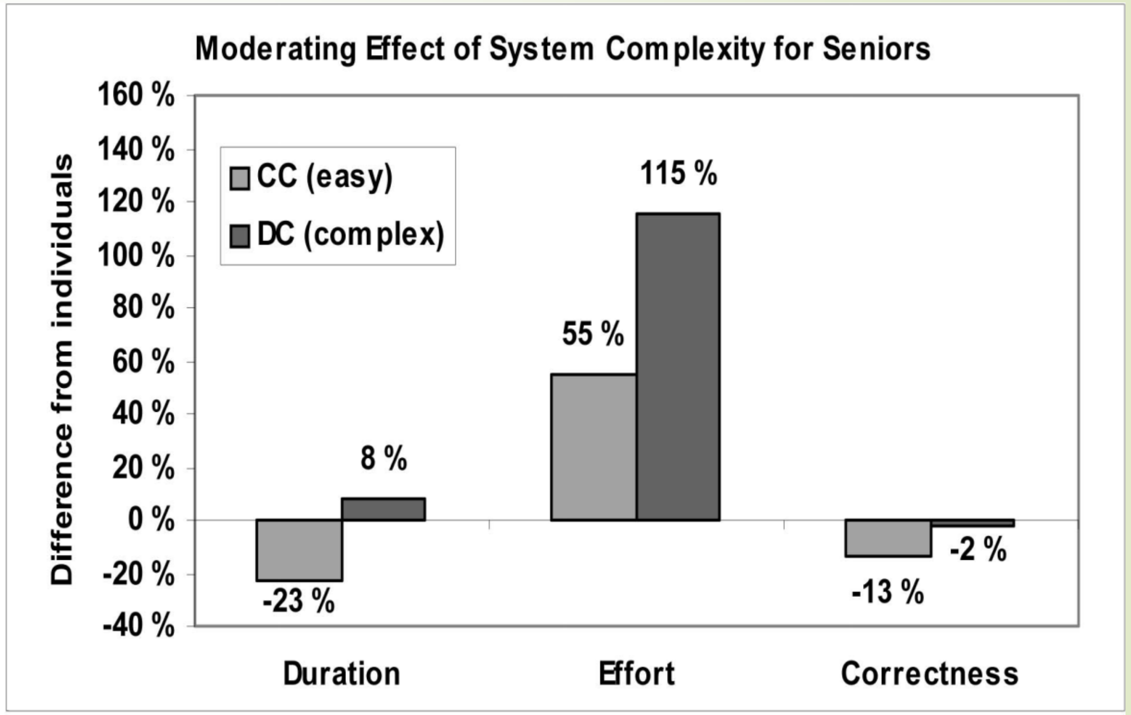


Moderating Effect of System Complexity for Intermediates
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
CC (easy)
DC (complex)
-39 %
-16 %
22 %
68 %
-29 %
92 %
Duration
Effort
Correctness

Results

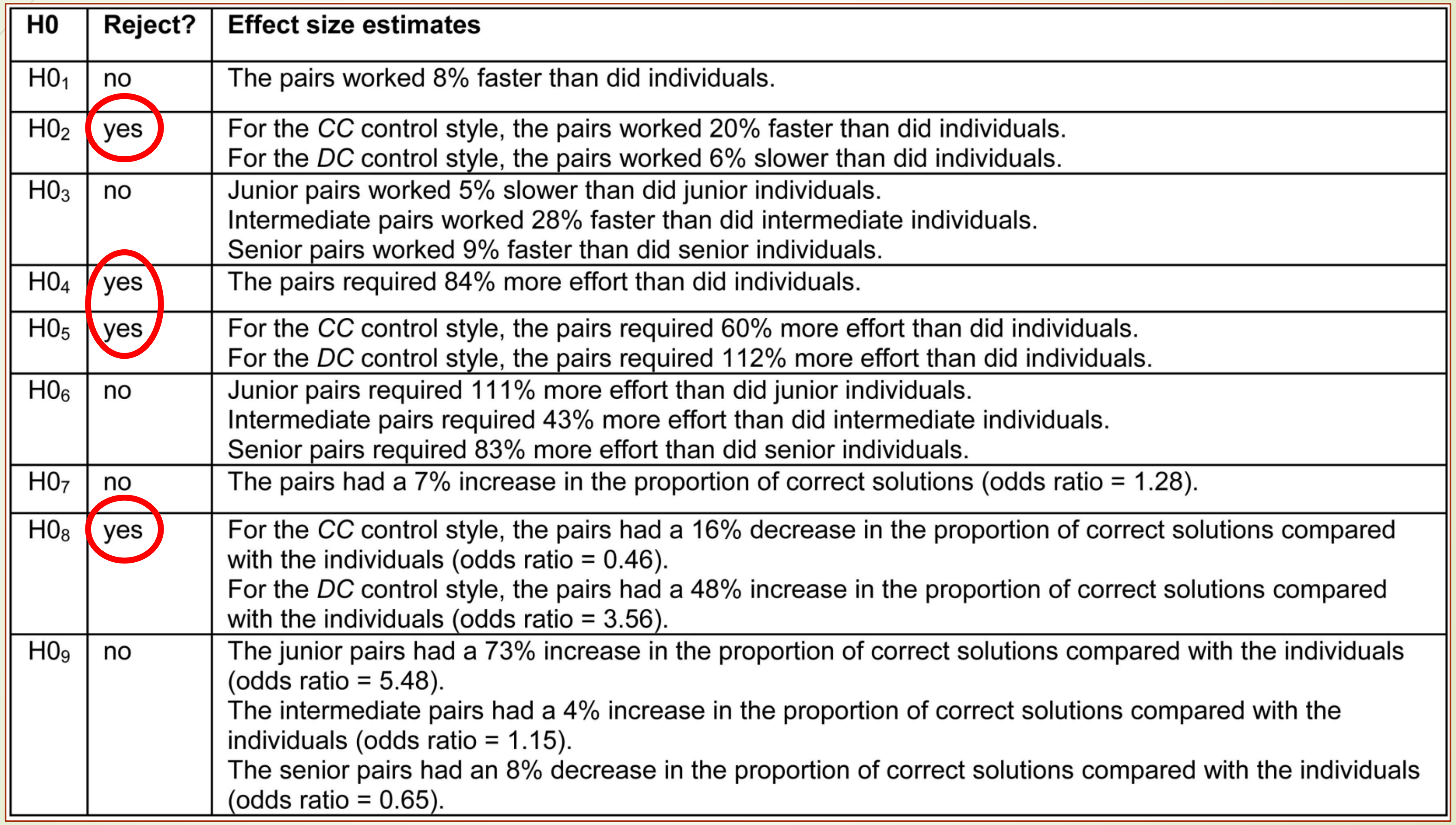


Effect of PP for Seniors
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
83 %
-9 %
-8 %
Duration
Effort
Correctness



Moderating Effect of System Complexity for Seniors
Difference from individuals
160 %
140 %
120 %
100 %
80 %
60 %
40 %
20 %
0 %
-20 %
-40 %
CC (easy)
DC (complex)
115 %
55 %
8 %
-23 %
-13 %
-2 %
Duration
Effort
Correctness

Results



H0	Reject?	Effect size estimates
$H0_1$	no	The pairs worked 8% faster than did individuals.
$H0_2$	yes	For the *CC* control style, the pairs worked 20% faster than did individuals. For the *DC* control style, the pairs worked 6% slower than did individuals.
$H0_3$	no	Junior pairs worked 5% slower than did junior individuals. Intermediate pairs worked 28% faster than did intermediate individuals. Senior pairs worked 9% faster than did senior individuals.
$H0_4$	yes	The pairs required 84% more effort than did individuals.
$H0_5$	yes	For the *CC* control style, the pairs required 60% more effort than did individuals. For the *DC* control style, the pairs required 112% more effort than did individuals.
$H0_6$	no	Junior pairs required 111% more effort than did junior individuals. Intermediate pairs required 43% more effort than did intermediate individuals. Senior pairs required 83% more effort than did senior individuals.
$H0_7$	no	The pairs had a 7% increase in the proportion of correct solutions (odds ratio = 1.28).
$H0_8$	yes	For the *CC* control style, the pairs had a 16% decrease in the proportion of correct solutions compared with the individuals (odds ratio = 0.46). For the *DC* control style, the pairs had a 48% increase in the proportion of correct solutions compared with the individuals (odds ratio = 3.56).
$H0_9$	no	The junior pairs had a 73% increase in the proportion of correct solutions compared with the individuals (odds ratio = 5.48). The intermediate pairs had a 4% increase in the proportion of correct solutions compared with the individuals (odds ratio = 1.15). The senior pairs had an 8% decrease in the proportion of correct solutions compared with the individuals (odds ratio = 0.65).

Threats to Validity

- **Trade-off**

1) To ensure realism (reduce threats to external validity);
2) To ensure control (to reduce threats to internal validity);

Threats to Validity

Internal

- The main threat to internal validity in this experiment was the **lack of random assignment** to the two treatment groups: individual programming and pair programming.
- Another related issue is that, for the analyses of duration and effort, they removed subjects with incorrect solutions, thereby introducing a potential bias, particularly since **they removed a larger proportion of individuals than pairs**.

Threats to Validity

Construct Validity: whether the sampling particulars of a study can be defended as measures of general constructs.

- Future experiments should consider using **heterogeneous pairs** and programmers that have already reached their **maximum level of combined efficiency**.
- the **reliability** of the categorization is **questionable**;
- correctness can be measured in many ways, It is possible that the **results would have been different** had we used more **fine-grained** measures of correctness

Threats to Validity

External

- The scope of this study is limited to situations in which the **developers have no prior knowledge of the system** to be changed.
- It is possible that the **results do not apply** to situations in which the **developers are also the original designers**.
- A related issue is whether the **short-term effects** observed in this experiment are representative of **long-term development**, in particular, due to **pair jelling**.

Analysis and Discussion

- *“Anecdotal and empirical evidence reported in the literature suggest **several organizational and personal benefits of PP** over individual programming, such as reduced time to market, reduced development costs, improved quality of the software, reduced costs of training new personnel, and enhanced trust, motivation, information and knowledge transfer among developers”.*

Analysis - Duration

- In previous experiments, the differences in the time taken varied from no difference, an insignificant 29% decrease, a 42.5% decrease, a 46.6% decrease, and a 52% decrease in favor of PP.

- This paper suggest an **overall insignificant decrease** in duration of 8%.
- When considering the moderating effects of system complexity: sig. 20% decrease for the CC design and insig. 6% increase for the DC design.
- When also accounting for different levels of programmer expertise: 39% decrease (intermediates on the CC design) to a slight increase of 8% (for seniors on the DC design).

Analysis - Effort

- In this context, effort is simply the same as the duration for the individuals and the duration multiplied by two for the pairs.

- The results in this paper suggest an **overall significant 84% increase** in effort.
- When considering the moderating effects of both system complexity and programmer expertise, the difference in effort ranged from an insignificant 22% increase (for intermediates on the CC design) to a significant 115% increase (for seniors on the DC design), **both in favor of individual programming**.

Analysis - Correctness

- In previous studies, the results vary from apparently no increase in correctness when using PP, a significant 15% increase in correctness, an insignificant 29% increase, and a significant 33% increase.

- This paper suggest an **overall 7% insignificant increase** in correctness (72% and 76% correct solutions for individuals and pairs).
- When considering the moderating effects of system complexity and expertise, our results suggest a significant, overall 48% increase in correctness for the DC design, but this effect was mostly due to the observations for **junior pair programmers,** who had a 149% increase in correctness for the DC design.

Analysis – Different Results

- There are several differences between the studies that complicate the above attempt to directly compare the results.

- only considered time/effort data for subjects with **correct programs**;
- **pair jelling**;
- the previous experiments all considered initial development tasks, whereas this considered **maintenance tasks** on systems of which the programmers had **no prior knowledge**;
- The **difference in the subjects**' ability, education, experience, training, etc., in general, and in PP in particular, may be a major cause of the different results;

Conclusion

- The existing body of empirical evidence indicates that **PP affects duration, effort, and correctness**, and we are reasonably sure that these effects are not simply due to chance.
- The existing results constitute necessary and useful steps toward being able to predict ***when*** PP might be beneficial.
- However, we are still far from being able to explain ***why*** we observe the given effects.
- Results from social psychology also suggest that, when, including **more complex and involving tasks**, social laboring is possible.

Paper Review

- The paper describes very well its experiment's steps and phases;
- The authors didn't ignore the experiment's deficiencies, explaining throughly the study's limitations;
- Correlation with other studies is very well explored;
- From a statistical view, design, methodologies and execution are solid;

- First phase was in 2001, but second phase 3-4 years later...
- Many results would be better explained through charts instead of tables;

Thank you

for your attention!

Questions?

Impressions?