Discussion: How Do API Documentation and Static Typing Affect API Usability?

Stefan Endrikat, Stefan Hanenberg, Romain Robbes, and Andreas Stefik

But First: Context

A Controlled Experiment To Assess the Benefits of Procedure Argument Type Checking

L. Prechelt and W. F. Tichy. April 1988.

ANSI Vs. K&R C

```
K&R C:
int foo(s, f, b)
  char* s;
  float f;
  struct Baz * b;
{
  return 5;
}

foo(3, 4, 5); // OK
```

```
ANSI C:
int foo(char* s,
        float f,
        struct Baz * b)
{
  return 5;
}

foo(3, 4, 5); // ERROR: 3 is not a char *…
```

Example from https://jameshfisher.com/2016/11/27/c-k-and-r/

Hypotheses

- Hypothesis 1.Type checking increases Interface Use Productivity.
- Hypothesis 2.Type checking reduces the number of Interface Defects in delivered programs.
- Hypothesis 3.Type checking reduces Interface Defect Lifetimes.

Participants

- The 34 subjects had the following education. Two were postdoctorates in computer science (CS); 19 were PhD students in CS and had completed an MS degree in CS; another subject was also a CS PhD student but held an MS in physics; 12 subjects were CS graduate students with a BS in CS.

- The subjects had between 4 and 19 years of programming experience ($\mu = 10$) and all but 11 of them had written at least 3,000 lines in C (all but one at least 300 lines). Only eight of the subjects had some programming experience with X-Windows or Motif; only three of them had written more than 300 lines in X-Windows or Motif.

Tasks

- Defined at a low level of granularity

```
/* Register callback–function ‘button_pushed’ for the
‘invert’ button with the number 1 as ‘client_data’ */
```

It can be implemented thus:

```
  XtAddCallbackF(invert,
                 XmCactivateCallback,
                 button_pushed,
                 (XtPointer)1);
```

Task Evaluation

- Hand-assessed (one person, for consistency)

ANSI C Makes the Second Task Faster

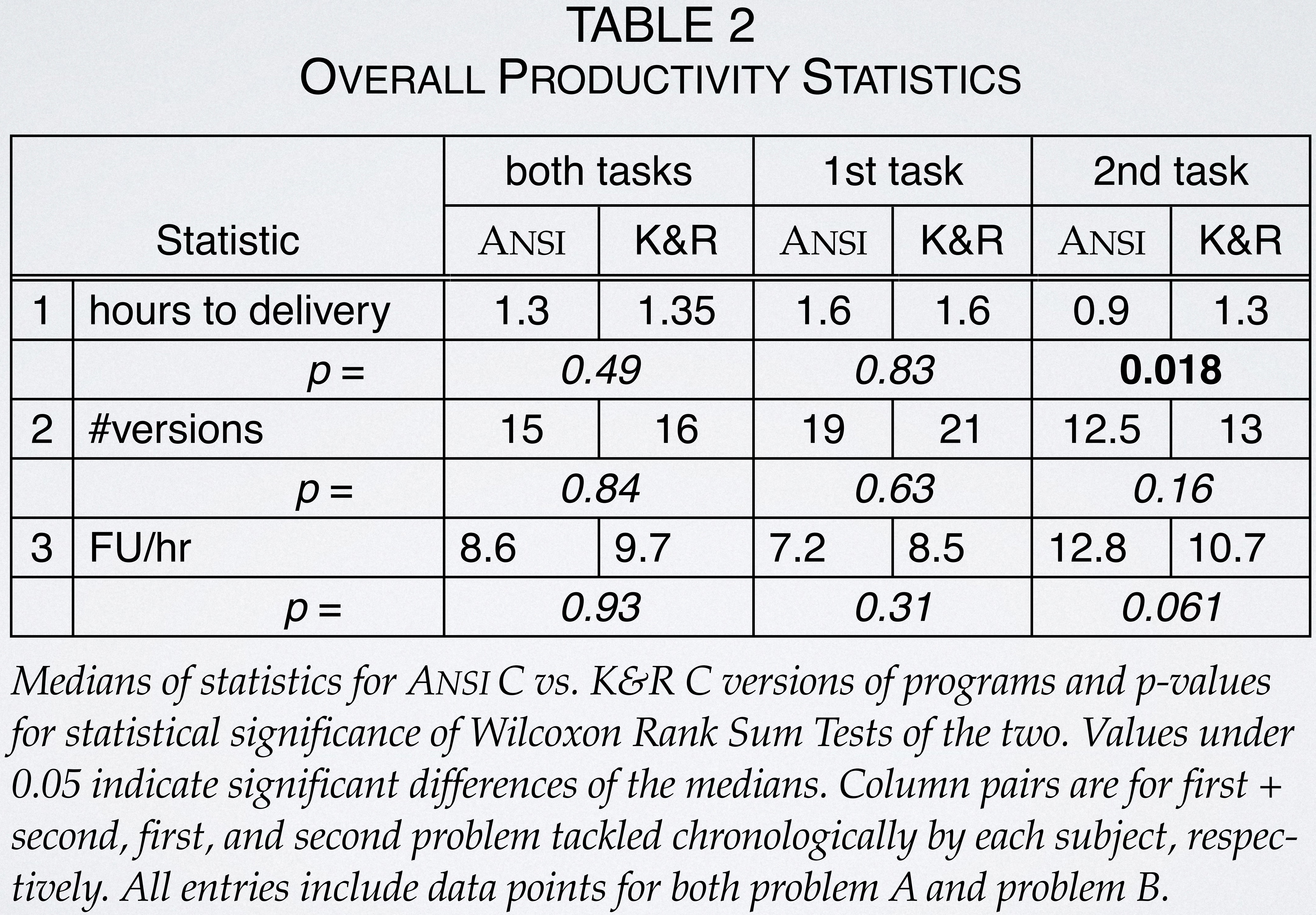


TABLE 2
OVERALL PRODUCTIVITY STATISTICS

Statistic		both tasks		1st task		2nd task	
		ANSI	K&R	ANSI	K&R	ANSI	K&R
1	hours to delivery	1.3	1.35	1.6	1.6	0.9	1.3
	p =	*0.49*		*0.83*		**0.018**	
2	#versions	15	16	19	21	12.5	13
	p =	*0.84*		*0.63*		*0.16*	
3	FU/hr	8.6	9.7	7.2	8.5	12.8	10.7
	p =	*0.93*		*0.31*		*0.061*	

Medians of statistics for ANSI C vs. K&R C versions of programs and p-values for statistical significance of Wilcoxon Rank Sum Tests of the two. Values under 0.05 indicate significant differences of the medians. Column pairs are for first + second, first, and second problem tackled chronologically by each subject, respectively. All entries include data points for both problem A and problem B.

ANSI C Helps Remove Defects Faster

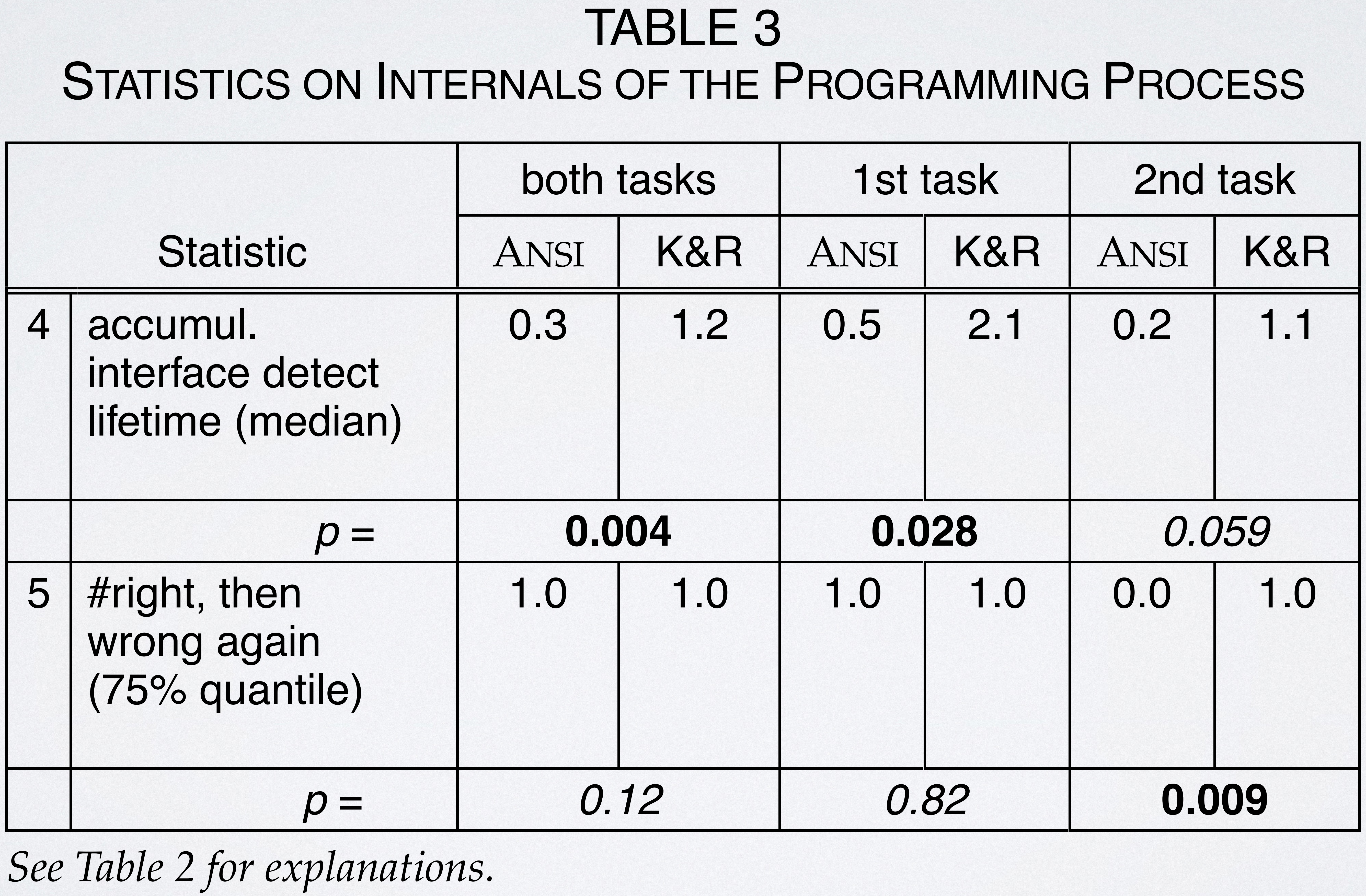


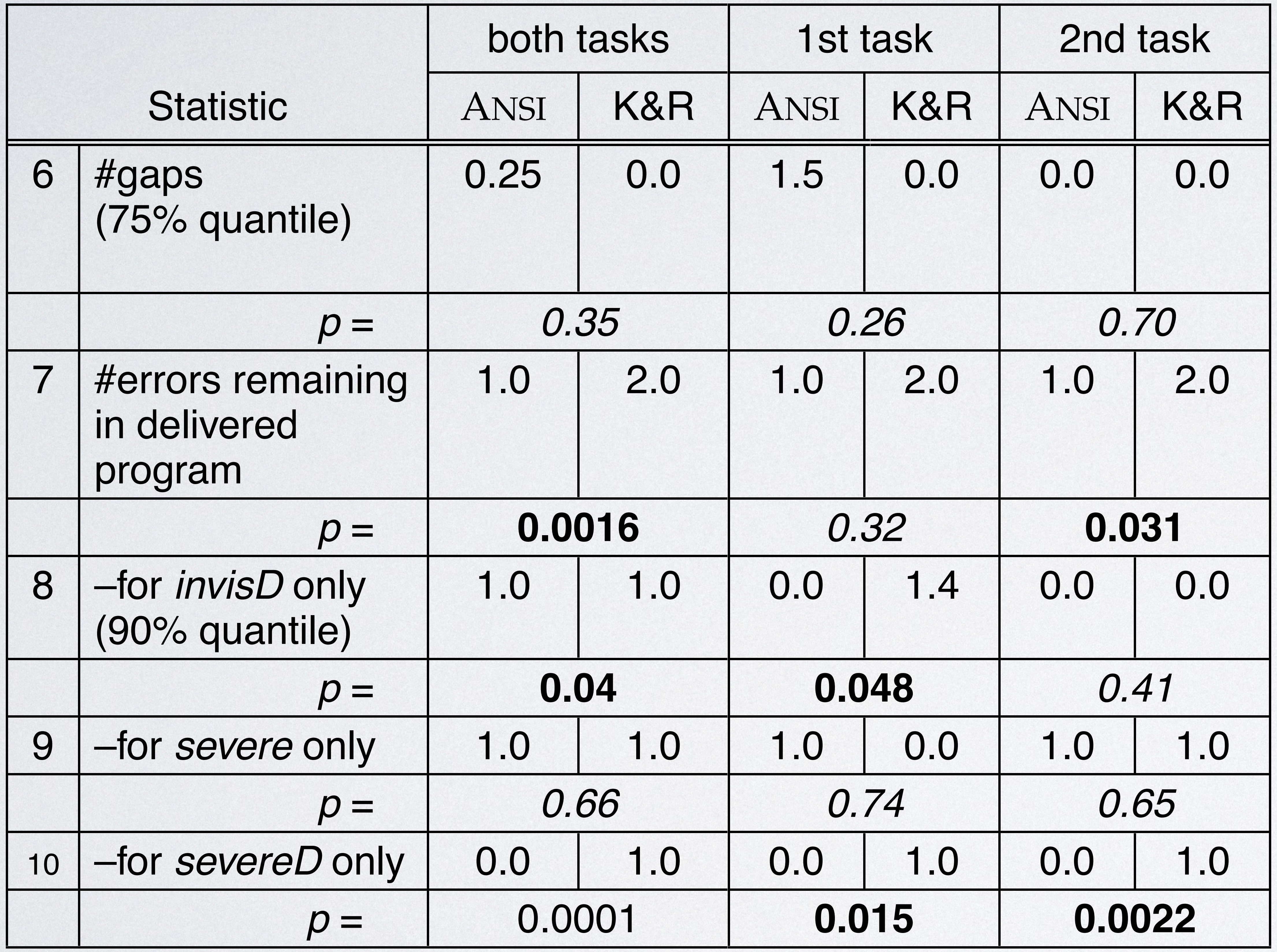
TABLE 3
STATISTICS ON INTERNALS OF THE PROGRAMMING PROCESS

		both tasks		1st task		2nd task	
	Statistic	ANSI	K&R	ANSI	K&R	ANSI	K&R
4	accumul. interface detect lifetime (median)	0.3	1.2	0.5	2.1	0.2	1.1
	p =	**0.004**		**0.028**		*0.059*	
5	#right, then wrong again (75% quantile)	1.0	1.0	1.0	1.0	0.0	1.0
	p =	*0.12*		*0.82*		**0.009**	

See Table 2 for explanations.

ANSI C Results in Better Programs

TABLE 4
Statistics on the Delivered Program



	Statistic	both tasks		1st task		2nd task	
		Ansi	K&R	Ansi	K&R	Ansi	K&R
6	#gaps (75% quantile)	0.25	0.0	1.5	0.0	0.0	0.0
	p =	*0.35*		*0.26*		*0.70*	
7	#errors remaining in delivered program	1.0	2.0	1.0	2.0	1.0	2.0
	p =	**0.0016**		*0.32*		**0.031**	
8	–for *invisD* only (90% quantile)	1.0	1.0	0.0	1.4	0.0	0.0
	p =	**0.04**		**0.048**		*0.41*	
9	–for *severe* only	1.0	1.0	1.0	0.0	1.0	1.0
	p =	*0.66*		*0.74*		*0.65*	
10	–for *severeD* only	0.0	1.0	0.0	1.0	0.0	1.0
	p =	0.0001		**0.015**		**0.0022**	

See Table 2 for explanations. Lines 6 and 8 do not list medians but other quantiles instead, as indicated.

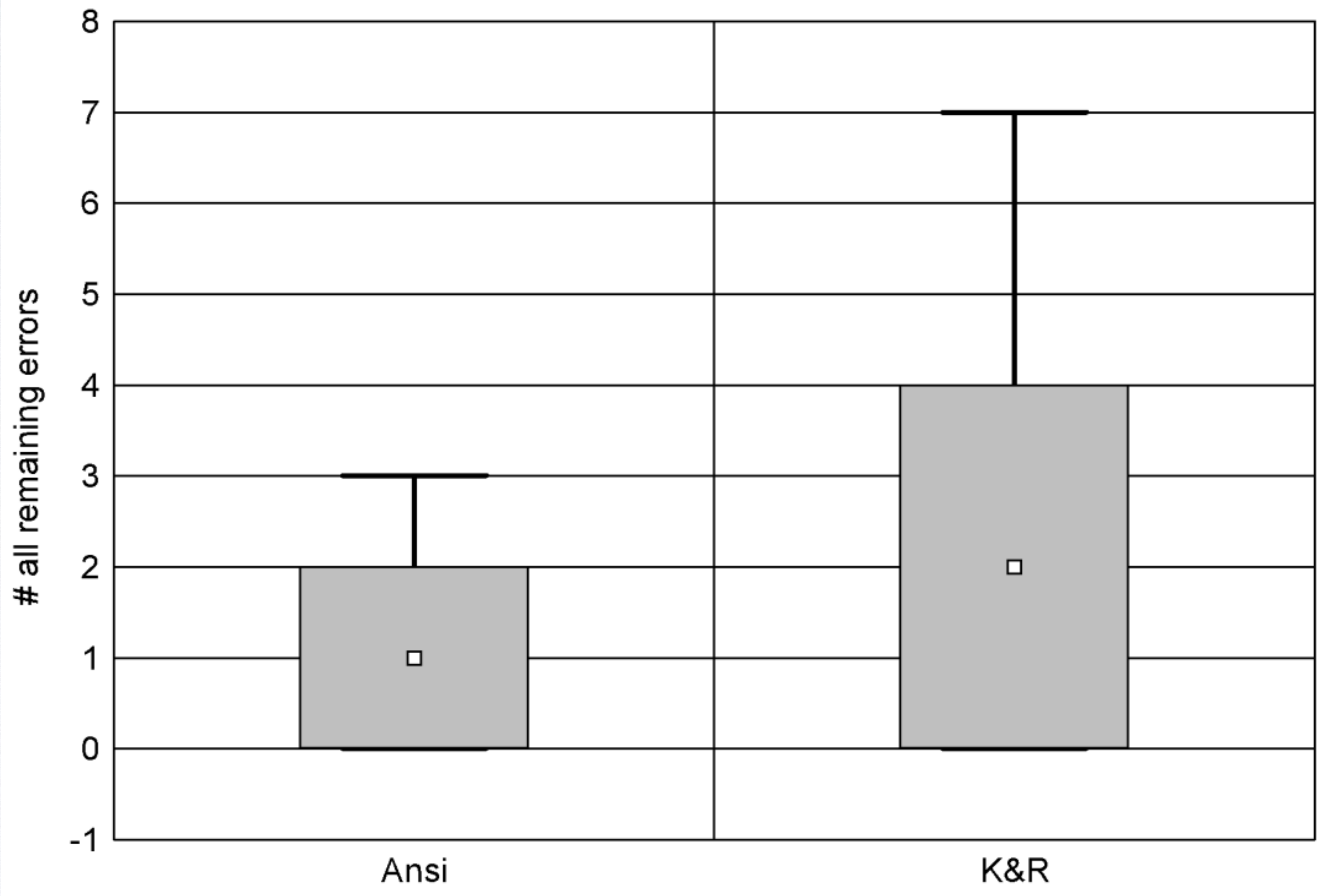


Fig. 7. Boxplots of total number of remaining defects in delivered programs over both tasks.

Now, Back to the Present

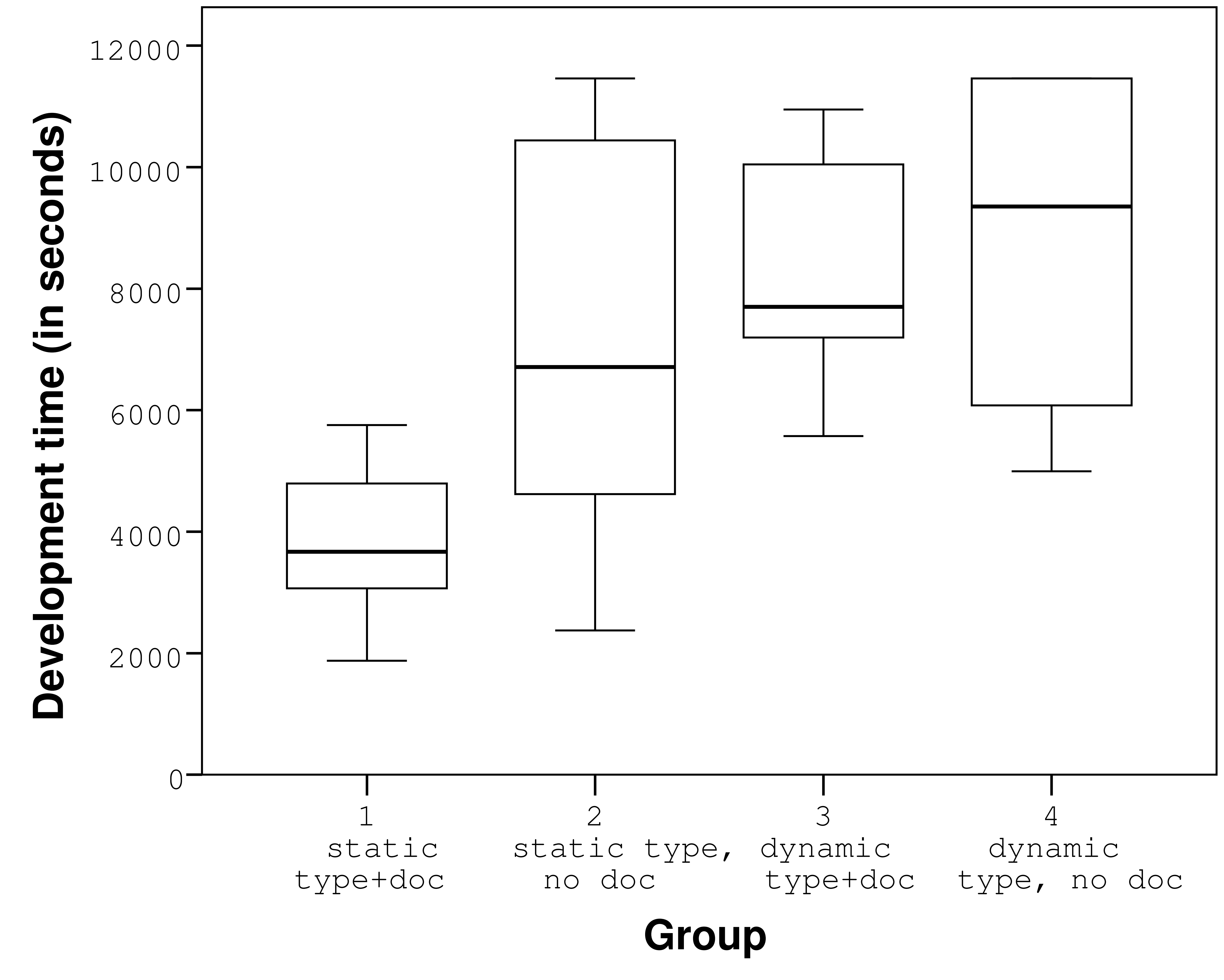
Experiment Notes

- Five hours max duration!
- 20-30 participants
- Documentation: plain text (?)
- Language: Dart (?)

Design

- 2x2 design
  - IVs: type system, documentation
  - DV: development time

Results



Development time (in seconds)
12000
10000
8000
6000
4000
2000
0
1
static
type+doc
2
static type,
no doc
3
dynamic
type+doc
4
dynamic
type, no doc
Group

Problem: Too Many Variables

- Type of documentation (in this study: text only)
- Choice of language (Dart)
- Choice of type system (Optional types in Dart)
- IDE features: autocomplete, and now Copilot
- Choice of tasks

Discussion questions

Q1: compare threats to validity

Q2: How might we mitigate these threats? What future studies should we do?

Group Discussion

- Mitigating these threats may require a fresh study! Pick one threat, and:
  - Develop a hypothesis (great hypotheses, if validated, can lead to tools)
  - Design a study
  - Propose a tool that could help if your hypothesis is true

- Type of documentation (in this study: text only)
- Choice of language (Dart)
- Choice of type system (Optional types in Dart)
- IDE features: autocomplete, and now Copilot
- Choice of tasks