Same Results: Drag-and-Drop Refactoring: Intuitive and Efficient Program Transformation

Summary: The authors describe an approach and tool (DNDRefactoring) for drag-and-drop refactoring, and compare their tool to the standard refactoring workflow in Eclipse. The authors found differences in response times between refactoring with Eclipse and DNDRefactoring for all but one refactoring task [**?**].

- Independent variable: Refactoring approach (2 levels, operationalized with Eclipse and DNDRefactoring)
- Tasks: 7 refactoring tasks in 4 categories:
  - Extract Method (1 task)
  - Move Method (4 task)
  - Three-step refactoring (1 task)
  - Extract class (1 task)
- Dependent variable: Task completion time [metric scale]
- Null hypothesis:
  - There is no difference in response times between Eclipse and DNDRefactoring
- Results:
  - For the extract method refactoring, there is no difference. For all other refactorings, there is a significant difference

In our reanalysis, we evaluated for each task whether there was a significant effect. Specifically, we conducted a t-test for the three-step task, and a Wilcoxon test for all other tasks, since the according data deviate from a normal distribution. Furthermore, we adjusted the significance level with an FDR correction. In Table 1, we summarize the data and results of the significance tests. With our task-wise analysis, we come to the same conclusion as the authors, namely that, except for the extract method refactoring, DNDRefactoring reduces the response times of the participants to complete the refactoring tasks. We found one issue with the analysis, namely that the authors did not correct for multiple testing, but after applying FDR correction, the results did not change. In summary, we found that aggregating the data did not change the results.

Table 1: Task completion times for DNDRefactoring. Gray cells contain values computed by us.

Participant	Refactoring:	Extract Method	Move Method	Move Method	Move Method	Move Method	Three-step Refactoring	Extract Class	Total
#1	Eclipse	42.3	48.3	22.9	18.5	16.6	29.9	42.1	219.8
	DNDR	18.6	12.7	13.4	13.8	14.3	20.9	miss	93.7
#2	Eclipse	106.4	71	10.7	7.3	4.4	113.4	32.5	345.7
	DNDR	13.8	6.7	2.8	13.2	4.9	22.8	17.6	81.8
#3	Eclipse	18.6	65.4	10.7	8.1	4	23.9	151.9	282.6[1]
	DNDR	33.5	3.7	2.9	2.3	1.4	16.8	8.9	69.5
#4	Eclipse	53.3	11.7	18.4	13.1	4	87.1	40.5	228.1
	DNDR	55.9	5.8	3.5	2.1	2.3	39.5	23.5	132.6
#5	Eclipse	23.7	93	8.9	29.8	8.2	95.9	41.5	301
	DNDR	63.1	5	6.4	1.8	2	13.9	11.4	103.6
#6	Eclipse	10	100.5	3	10.2	2.7	100.4	24.2	251
	DNDR	31.3	26.8	1.5	1	1.1	15	15.3	92
#7	Eclipse	22.6	46.3	2.8	10.7	5.2	69	25.1	181.7
	DNDR	22.8	3.4	1.6	1.6	0.9	23	7.6	60.9
#8	Eclipse	18.8	136.7	4.1	6.7	2.7	77.5	23.7	270.2
	DNDR	17.6	1	2.1	1.7	4.1	6.	21.8	55.1
#9	Eclipse	7	50.7	3.1	4.7	2.3	43	24	134.8
	DNDR	12.6	1.5	1.5	1.6	1.	13.7	12.9	45.4
Average	Eclipse	33.6	69.3	9.4	12.1	5.6	71	45.06[1]	246.1
	DNDR	29.9	7.4	4	4.3	3.6	19.2	14.9	81.6
Speed up[3]		1.1	9.4	2.4	2.8	1.5	3.7	4.5	3.0
p value (aggr.)		0.715		**0.002**			**0.002**	**0.004**	**0.004**
Speed up (aggr.)		1.1		5			3.7	4.5	3.0
W/t value		18	45	45	40	41	4.81 (df=8)	36	8.17 (df=8)
p value		0.652[2]	**0.004**	**0.004**	**0.039**	**0.027**	**0.001**[2]	**0.008**[2]	**0.000004**[2]

[1]These values are reported as 228.1 (total) and 66.75 (average) in the original study, which is incorrect according to the raw data of the study (Table IV in the original paper).
[2]The p values differ from the original study, but we cannot be sure why. Part of the explanation is that we used a t test instead of a Wilcoxon test for the three-step refactoring and the total response time.
[3]Eclipse average divided by DNDRefactoring average, as in the original study.