Appendices

Appendix A
Exercises

Explanations of the different types of exercises can be found in the Preface. In summary, the objective is to provide four types of exercises:

Understanding	These exercises aim at highlighting the most important issues from each chapter. Exercises are available in Chaps. 1–11.
Training	The objective of these exercises is to encourage practicing experimentation. This includes setting up hypotheses and performing the statistical analysis.
Reviewing	Chapters 12 and 13 include examples of experiments. The intention of this part is to provide help in reviewing and reading published experiments.
Assignments	These exercises are formulated to promote an understanding of how experiments can be used in software engineering to evaluate methods and techniques.

The understanding type questions can be found at the end of each chapter, while the three other types of exercises can be found in this appendix.

A.1 Training

The exercises are preferably solved either using a statistical program package or tables from books in statistics. The tables in Appendix B may be used, but the tables provided are only for the 5% significance level, so if other significance levels are used then other sources must be used. It should be remembered that Appendix B has primarily been provided to explain the examples in Chap. 10.

C. Wohlin et al., *Experimentation in Software Engineering*,
DOI 10.1007/978-3-642-29044-2, © Springer-Verlag Berlin Heidelberg 2012

A.1.1 Normally Distributed Data

The probably most complicated example of the statistical methods in Chap. 10 is the goodness of fit test for the normal distribution, see Sect. 10.2.12. Thus, it is appropriate to ensure a good understanding of that test.

1. Carry out the goodness of fit test, on the same data, see Table 10.20, using 12 segments instead.

A.1.2 Experience

In Chap. 12, the outcome of the Personal Software Process course is compared with the background of the students taking the course. The analysis conducted in Chap. 12 is only partial. The full data set is provided in Tables A.2 and A.3. In Table A.1, the survey material handed out at the first lecture is presented. The outcome of the survey is presented in Table A.2. The outcome of the PSP course is presented in Table A.3, where the following seven measures have been used to measure the outcome of the course:

Size	The number of new and changed lines of code for the ten programs.
Time	The total development time for the ten programs.
Prod.	The productivity measured as number of lines of code per development hour.
Faults	The number of faults logged for the ten programs. This includes all faults found, for example, including compilation faults.
Faults/KLOC	The number of faults for each 1,000 lines of code.
Pred. Size	The absolute relative error in predicting program size. The figures show the error in absolute percentages, for example, both over- and underestimates with 20% are shown as 20% without any sign indicating the direction of the estimation error.
Pred. Time	The absolute relative error in predicting the development time.

Based on the presentation in Chap. 12 and the data in Tables A.2 and A.3 answer the following questions.

1. How can the survey be improved? Think about what constitutes good measures of background, experience and ability.
2. Define hypotheses, additional to those in Chap. 12, based on the available data. Motivate why these hypotheses are interesting.
3. What type of sampling has been used?
4. Analyze the hypotheses you have stated. What are the results?
5. Discuss the external validity of your findings. Can the results be generalized outside the PSP? Can the results be generalized to industrial software engineers?

Table A.1 Student characterization

Area	Description	Answer
Study program (denoted Line)	Answer: Computer Science and Engineering or Electrical Engineering	
General knowledge in computer science and software engineering (denoted SE)	1. Little, but curious about the new course 2. Not my speciality (focus on other subjects) 3. Rather good, but not my main focus (one of a couple of areas) 4. Main focus of my studies	
General knowledge in programming (denoted Prog.)	1. Only 1–2 courses 2. 3 or more courses, no industrial experience 3. A few courses and some industrial experience 4. More than three courses and more than 1 year industrial experience	
Knowledge about the PSP (denoted PSP)	1. What is it? 2. I have heard about it 3. A general understanding of what it is 4. I have read some material	
Knowledge in C (denoted C)	1. No prior knowledge 2. Read a book or followed a course 3. Some industrial experience (less than 6 months) 4. Industrial experience	
Knowledge in C++ (denoted C++)	1. No prior knowledge 2. Read a book or followed a course 3. Some industrial experience (less than 6 months) 4. Industrial experience	
Number of courses (denoted Courses)	A list of courses was provided and the students were asked to put down a yes or no whether they had taken the course or not. Moreover, they were asked to complement the list of courses if they had read something else they thought was a particularly relevant course	

A.1.3 Programming

In an experiment, 20 programmers have developed the same program, where 10 of them have used programming language A and 10 have used language B. Language A is newer and the company is planning to change to language A if it is better than language B. During the development, the size of the program, the development time, the total number of removed defects and the number of defects removed in test have been measured.

Table A.2 Information from background survey

Subject	Line	SE	Prog.	PSP	C	C++	Courses
1	1	2	1	2	1	1	2
2	1	3	2	1	2	1	4
3	2	3	2	2	2	2	7
4	1	3	2	3	2	1	3
5	1	3	2	3	2	1	5
6	2	4	3	2	1	1	7
7	2	3	2	2	1	2	7
8	1	3	2	2	1	1	4
9	2	4	3	2	1	1	9
10	2	4	2	1	1	1	7
11	1	2	2	1	2	1	3
12	2	4	3	2	1	1	9
13	2	4	3	2	3	3	8
14	2	3	2	2	1	1	6
15	1	3	2	2	1	1	5
16	2	4	2	1	1	1	10
17	1	3	3	1	1	1	5
18	2	4	3	2	1	3	6
19	2	4	3	3	3	3	8
20	1	1	1	1	1	1	2
21	2	3	3	2	2	2	10
22	2	3	2	3	1	1	5
23	1	3	2	2	1	1	4
24	1	2	1	1	1	1	3
25	2	4	3	1	2	2	7
26	1	3	2	2	1	1	5
27	2	4	3	2	3	2	7
28	1	3	2	3	1	1	2
29	2	4	2	3	1	1	7
30	2	3	3	1	2	3	6
31	1	3	2	2	2	2	5
32	2	3	3	1	2	2	10
33	2	4	3	1	1	1	5
34	1	2	2	1	2	2	3
35	1	2	1	1	1	1	2
36	1	2	1	2	1	1	2
37	1	2	2	2	2	2	2
38	2	4	2	2	2	1	6
39	1	2	1	2	1	1	2
40	2	4	3	1	4	4	7
41	2	3	3	2	2	2	8
41	2	4	3	2	2	2	9

(*continued*)

Table A.2 (continued)

Subject	Line	SE	Prog.	PSP	C	C++	Courses
43	1	3	2	1	1	1	3
44	1	4	3	2	3	2	7
45	2	4	2	2	2	1	6
46	2	2	4	2	4	4	7
47	2	4	3	2	3	2	7
48	1	2	2	2	1	1	2
49	1	3	3	1	1	1	3
50	2	3	2	3	1	1	8
51	2	4	2	4	2	2	8
52	2	4	3	3	3	2	8
53	2	4	3	3	2	2	10
54	1	2	1	2	1	1	2
55	1	2	2	2	1	1	4
56	2	3	2	1	1	1	8
57	1	2	3	1	1	1	4
58	2	4	3	3	1	1	6
59	1	2	2	2	2	1	4

The programmers have been randomly assigned a programming language and the objective of the experiment is to evaluate if the language has any effect on the four measured variables. The collected data can be found in Table A.4. The data is fictitious.

1. Which design has been used in the experiment?
2. Define the hypotheses for the evaluation.
3. Use box plots to investigate the differences between the languages in terms of central tendency and dispersion with respect to all four factors. Is there any outlier and if so should it be removed?
4. Assume that parametric tests can be used. Evaluate the effect of the programming language on the four measured variables. Which conclusions can be drawn from the results?
5. Evaluate the effect of the programming language on the four measured variables using a non-parametric test. Which conclusions can be drawn from the results? Compare the results to those achieved when using parametric tests.
6. Discuss the validity of the results and if it is appropriate to use a parametric test.
7. Assume that the participating programmers have chosen the programming language themselves. What consequences does this have on the validity of the results? Do the conclusions still hold?

Table A.3 Outcome from the PSP course

Subject	Size	Time	Prod.	Faults	Faults/KLOC	Pred. size	Pred. time
1	839	3,657	13.8	53	63.2	39.7	20.2
2	1,249	3,799	19.7	56	44.8	44.1	21.2
3	968	1,680	34.6	71	73.3	29.1	25.1
4	996	4,357	13.7	35	35.1	24.3	18.0
5	794	2,011	23.7	32	40.3	26.0	13.2
6	849	2,505	20.3	26	30.6	61.1	48.2
7	1,455	4,017	21.7	118	81.1	36.5	34.7
8	1,177	2,673	26.4	61	51.8	34.6	32.5
9	747	1,552	28.9	41	54.9	51.0	18.2
10	1,107	2,479	26.8	59	53.3	22.6	14.0
11	729	3,449	12.7	27	37.0	26.9	52.0
12	999	3,105	19.3	63	63.1	26.0	19.8
13	881	2,224	23.8	44	49.9	47.9	39.9
14	730	2,395	18.3	94	128.8	63.0	20.3
15	1,145	3,632	18.9	70	61.1	33.3	34.8
16	1,803	3,193	33.9	98	54.4	52.9	21.8
17	800	2,702	17.8	60	75.0	34.3	26.7
18	1,042	2,089	29.9	64	61.4	49.3	41.5
19	918	3,648	15.1	43	46.8	49.7	71.5
20	1,115	6,807	9.8	26	23.3	34.1	22.4
21	890	4,096	13.0	108	121.3	19.3	34.8
22	1,038	3,609	17.3	98	94.4	21.4	52.0
23	1,251	6,925	10.8	498	398.1	21.8	34.1
24	623	4,216	8.9	53	85.1	40.5	36.3
25	1,319	1,864	42.5	92	69.7	43.7	45.0
26	800	4,088	11.7	74	92.5	42.6	36.2
27	1,267	2,553	29.8	88	69.5	53.0	30.1
28	945	1,648	34.4	42	44.4	33.3	17.9
29	724	4,144	10.5	49	67.7	32.8	17.8
30	1,131	2,869	23.7	102	90.2	29.2	15.5
31	1,021	2,235	27.4	49	48.0	18.0	25.0
32	840	3,215	15.7	69	82.1	85.6	54.0
33	985	5,643	10.5	133	135.0	27.3	31.0
34	590	2,678	13.2	33	55.9	83.0	20.0
35	727	4,321	10.1	48	66.0	17.0	22.7
36	955	3,836	14.9	76	79.6	33.3	36.8
37	803	4,470	10.8	56	69.7	18.2	27.7
38	684	1,592	25.8	28	40.9	35.0	34.1
39	913	4,188	13.1	45	49.3	25.3	27.5
40	1,200	1,827	39.4	61	50.8	31.6	20.9
41	894	2,777	19.3	64	71.6	21.3	22.4
42	1,545	3,281	28.3	136	88.0	35.0	16.1

(*continued*)

Table A.3 (continued)

Subject	Size	Time	Prod.	Faults	Faults/KLOC	Pred. size	Pred. time
43	995	2,806	21.3	71	71.4	15.6	38.3
44	807	2,464	19.7	65	80.5	43.3	26.4
45	1,078	2,462	26.3	55	51.0	49.1	51.6
46	944	3,154	18.0	71	75.2	59.0	39.2
47	868	1,564	33.3	50	57.6	50.4	45.2
48	701	3,188	13.2	31	44.2	21.2	49.7
49	1,107	4,823	13.8	86	77.7	19.3	28.4
50	1,535	2,938	31.3	71	46.3	29.6	20.7
51	858	7,163	7.2	97	113.1	58.4	32.9
52	832	2,033	24.6	84	101.0	48.4	25.6
53	975	3,160	18.5	115	117.9	29.5	31.5
54	715	3,337	12.9	40	55.9	41.7	26.6
55	947	4,583	12.4	99	104.5	41.0	22.3
56	926	2,924	19.0	77	83.2	32.5	34.7
57	711	3,053	14.0	78	109.7	22.8	14.3
58	1,283	7,063	10.9	186	145.0	46.5	26.6
59	1,261	3,092	24.5	54	42.8	27.4	45.3

A.1.4 Design

This exercise is based on data obtained from an experiment carried out by Briand, Bunse and Daly. The experiment is further described by Briand et al. [28].

An experiment is designed in order to evaluate the impact of quality object-oriented design principles when intending to modify a given design. The quality design principles evaluated are the principles provided by Coad and Yourdon [35]. In the experiment two systems are used with one design for each system. One of the designs is a 'good' design made using the design principles and the other is a 'bad' design not using the principles. The two designs are documented in the same way in terms of layout and content and are of the same size, i.e. they are developed to be as similar as possible except for following or not following the design principles. The objective of the experiment is to evaluate if the quality design principles ease impact analysis when identifying changes in the design.

The task for each participant is to undertake two separate impact analyses, one for each system design. Marking all places in the design that have to be changed but not actually change them makes the impact analyses. The first impact analysis is for a changed customer requirement and the second is for an enhancement in the systems functionality. Four measures are collected during the task:

Mod_Time: Time spent on identifying places for modification.
Mod_Comp: Represents the completeness of the impact analysis and is defined as:

Table A.4 Data for programming exercise

Programming language	Program size (LOC)	Development time (min)	Total number of defects	Number of test defects
A	1,408	3,949	89	23
A	1,529	2,061	69	16
A	946	3,869	170	41
A	1,141	5,562	271	55
A	696	5,028	103	39
A	775	2,296	75	29
A	1,205	2,980	79	11
A	1,159	2,991	194	28
A	862	2,701	67	27
A	1,206	2,592	77	15
B	1,316	3,986	68	20
B	1,787	4,477	54	10
B	1,105	3,789	130	23
B	1,583	4,371	48	13
B	1,381	3,325	133	29
B	944	5,234	80	25
B	1,492	4,901	64	21
B	1,217	3,897	89	29
B	936	3,825	57	20
B	1,441	4,015	79	18

$$\text{Mod\_Comp} = \frac{\text{Number of correct places found}}{\text{Total number of places to be found}}$$

Mod_Corr: Represents the correctness of the impact analysis and is defined as:

$$\text{Mod\_Corr} = \frac{\text{Number of correct places found}}{\text{Total number of places indicated as found}}$$

Mod_Rate: The number of correct places found per time unit, that is:

$$\text{Mod\_Rate} = \frac{\text{Number of correct places found}}{\text{Time for identification}}$$

The experiment is conducted at two occasions, in order to let each participant work with both the good design and the bad design. The subjects were randomly assigned to one of two groups, A or B. Group A worked with the good design at the first occasion and the bad design in the second. Group B studied the bad design first and then the good design. The collected data can be found in Table A.5.

Table A.5 Data for design exercise

		Good object-oriented design				Bad object-oriented design			
Participant	Group	Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate	Mod_Time	Mod_Comp	Mod_Corr	Mod_Rate
P01	B	–	0.545	0.75	–	–	0.238	0.714	–
P02	B	–	0.818	1	–	–	0.095	1	–
P03	A	20	0.409	1	0.45	25	0.19	1	0.16
P04	B	22	0.818	1	0.818	25	0.238	1	0.2
P05	B	30	0.909	1	0.667	35	0.476	0.909	0.286
P07	A	–	0	–	–	38	0.476	1	0.263
P09	A	–	0.455	1	–	–	0.476	1	–
P10	B	–	0.409	0.9	–	–	0.381	1	–
P11	A	45	0.545	0.923	0.267	50	0.714	1	0.3
P12	B	–	0.773	1	–	–	0.714	1	–
P13	A	40	0.773	1	0.425	40	0.762	1	0.4
P14	B	30	0.909	1	0.667	30	0.333	0.875	0.233
P15	B	–	0.864	1	–	40	0.238	1	0.125
P16	B	30	0.773	1	0.567	–	–	–	–
P17	B	–	0.955	1	–	–	0.286	0.75	–
P18	B	–	0	–	–	–	0.19	1	–
P19	A	29	0.818	1	0.621	27	0.667	1	0.519
P20	A	9	0.591	1	1.444	15	0.19	0.8	0.267
P21	B	20	0.591	1	0.65	35	0.19	1	0.114
P22	B	30	0.682	1	0.5	20	0.714	1	0.75
P23	B	–	0.818	1	–	–	0.476	1	–
P24	A	30	0.773	1	0.567	40	0.762	1	0.4
P25	A	–	0.955	1	–	–	0.667	0.875	–
P26	B	25	0	0	0	25	0.095	0.5	0.08
P27	A	27	0.773	0.944	0.63	36	0.389	0.7	0.194
P28	A	25	0.773	1	0.68	30	0.667	1	0.467
P29	B	44	0.773	1	0.386	23	0.762	1	0.696
P31	A	–	0.409	1	–	–	0.286	0.75	–
P32	A	30	0.909	1	0.667	–	0.5	1	–
P33	A	65	0.818	1	0.277	–	0.619	1	–
P34	A	50	0.636	0.933	0.28	30	0.4	0.889	0.267
P35	A	10	0.591	1	1.3	10	0.667	1	1.4
P36	A	13	1	1	1.692	–	0.619	1	–

1. Which design has been used in the experiment?
2. Define the hypotheses for the evaluation.
3. How should the missing values in Table A.5 be treated?
4. Assume that parametric tests can be used. Evaluate the effect of the quality design principles on the four measured variables. Which conclusions can be drawn from the results?
5. Evaluate the effect of the quality design principles on the four measured variables using non-parametric tests. Which conclusions can be drawn from the results? Compare the results to those achieved when using parametric tests.
6. Discuss the validity of the results and if it is appropriate to use parametric tests.
7. The participants in the experiment are students taking a software engineering course that have volunteered to be subjects. From which population is the sample taken from? Discuss how this type of sampling will affect the external validity of the experiment? How can the sampling be made differently?

A.1.5 Inspections

This exercise refers to the example experiment in Chap. 13.

1. Rewrite the abstract in Chap. 13 to be a structured abstract, as defined in Chap. 11.
2. Conduct the scoping and planning steps for an *exact* replication of the experiment. Especially, define how many subjects should be enrolled to achieve a given level of confidence in the analysis.
3. Conduct the scoping step for a *differentiated* replication of the experiment. Define three different goal templates for three alternative replications. Discuss pros and cons of each alternative with respect to costs, risks and gains (see also Fig. 2.1).

A.2 Reviewing

Below is a list of questions, which are important to consider when reading or reviewing an article presenting an experiment. Use the list and review the examples presented in Chaps. 12 and 13, and also some experiment presented in the literature.

The list below should be seen as a checklist in addition to normal questions when reading an article. An example of a normal question may be; is the abstract a good description of the content of the paper? Some specific aspects to consider when reading an experiment article are:

- Is the experiment understandable and interesting in general?
- Does the experiment have any practical value?
- Are other experiments addressing the problem summarized and referenced?

- What is the population in the experiment?
- Is the sample used representative of the population?
- Are the dependent and independent variables clearly defined?
- Are the hypotheses clearly formulated?
- Is the type of design clearly stated?
- Is the design correct?
- Is the instrumentation described properly?
- Is the validity of the experiment treated carefully and convincing?
- Are different types of validity threats addressed properly?
- Has the data been validated?
- Is the statistical power sufficient, are there enough subjects in the experiment?
- Are the appropriate statistical tests applied? Are Parametric or non-parametric tests used and are they used correctly?
- Is the significance level used appropriate?
- Is the data interpreted correctly?
- Are the conclusions correct?
- Are the results not overstated?
- Is it possible to replicate the study?
- Is data provided?
- Is it possible to use the results for performing a meta-analysis?
- Is further work and experimentation in the area outlined?

A.3 Assignments

These assignments are based on the following general scenario. A company would like to improve their way of working by changing the software process. You are consulted as an expert in evaluating new techniques and methods in relation to the existing process. The company would like to know whether or not to change their software process.

You are expected to search for appropriate literature, review the existing literature on the subject, apply the experiment process and write a report containing a recommendation for the company. The recommendation should discuss both the results of the experiment and other relevant issues for taking the decision whether or not to change the process. Other relevant issues include costs and benefits for making the change. If you are unable to find the correct costs, you are expected to make estimates. The latter may be in terms of relative costs.

The assignments are intentionally fairly open-ended to allow for interpretation and discussion. Each assignment is described in terms of prerequisites needed to perform the assignment and then the actual task is briefly described. It should be noted that the assignments below are examples of possible experiments that can be conducted. The important issue to hold in mind is that the main objective is that the assignments should provide practice in using experiments as part of an evaluation procedure.

Finally, it should be noted that some organizations provide what is called lab packages that can be used to replicate experiments. Lab packages are important as they allow us to build upon work by others and hence hopefully come to more generally valid results by replication. Some lab packages can be found by a search on the Internet. It may also be beneficial to contact the original experimenter to get support and maybe also a non-published lab package.

A.3.1 Unit Test and Code Reviews

The company wants to evaluate if it is cost-effective to introduce code reviews. Unit testing is done today, although on non-reviewed code. Is this the best way to do it?

Prerequisites

- Suitable programs with defects that can be found during either reviews or testing.
- A review method, which may be ad hoc, but preferable it should be something more realistic, for example, a checklist-based approach. In this case, a checklist is needed.
- A testing method, which also may be ad hoc, but preferably it is based on, for example, usage or equivalence partitioning.

Task

- Evaluate if it is cost-effective to introduce code reviews.

A.3.2 Inspection Methods

Several different ways of conducting reviews are available. The company intends to introduce the best inspection method out of two possible choices. Which of the two methods is the best to introduce for the company?

Prerequisites

- Suitable software artifacts to review should be available.
- Two review methods with appropriate support in terms of, for example, checklists or description of different reading perspectives, see also Appendix A.1.5.

Task

- Assume that the company intends to introduce reviews of the chosen software artifacts, which method should they introduce? Determine which of the inspection methods that is best in finding defects. Is the best method also cost effective?

A.3.3 Requirements Notation

It is important to write requirements specification so that all readers interpret them easily and in the same way. The company has several different notations to choose from. Which is the best way of representing requirements?

Prerequisites

- A requirements specification written in several different notations, for example, natural language and different graphical representations.

Task

- Evaluate if it is beneficial to change the company's notation for requirements specifications. Assume that the company uses natural language today.

Appendix B
Statistical Tables

This appendix contains statistical tables for a significance level of 5%. More elaborated tables can be found in most books on statistics, for example [119], and tables are also available on the Internet. The main objective here is to provide some information, so that the tests that are explained in Chap. 10 become understandable and so that the examples provided can be followed. This is important even if statistical packages are used for the calculations, since it is important to understand the underlying calculations before just applying the different statistical tests. It is also worth noting the tables are a shortcut, for example, the values for the t-test, F-test and Chi-2 can be calculated from the respective distributions.

The following statistical tables are included:

- t-test (see Sects. 10.3.4, 10.3.7, and Table B.1)
- Chi-2 (see Sect. 10.3.12 and Table B.2)
- Mann-Whitney (see Sect. 10.3.5 and Table B.3)
- Wilcoxon (see Sect. 10.3.8 and Table B.4)
- F-test (see Sects. 10.3.6, 10.3.10, Table B.5)

C. Wohlin et al., *Experimentation in Software Engineering*,
DOI 10.1007/978-3-642-29044-2, © Springer-Verlag Berlin Heidelberg 2012

Table B.1 Critical values two-tailed t-test (5%), see Sects. 10.3.4 and 10.3.7

Degrees of freedom	t-value
1	12.706
2	4.303
3	3.182
4	2.776
5	2.571
6	2.447
7	2.365
8	2.306
9	2.262
10	2.228
11	2.201
12	2.179
13	2.160
14	2.145
15	2.131
16	2.120
17	2.110
18	2.101
19	2.093
20	2.086
21	2.080
22	2.074
23	2.069
24	2.064
25	2.060
26	2.056
27	2.052
28	2.048
29	2.045
30	2.042
40	2.021
60	2.000
120	1.980
∞	1.960

Table B.2 Critical values one-tailed Chi2-test (5%), see Sect. 10.3.12

Degrees of freedom	χ^2
1	3.84
2	5.99
3	7.81
4	9.49
5	11.07
6	12.59
7	14.07
8	15.51
9	16.92
10	18.31
11	19.68
12	21.03
13	22.36
14	23.68
15	25.00
16	26.30
17	27.59
18	28.87
19	30.14
20	31.41
21	32.67
22	33.92
23	35.17
24	36.42
25	37.65
26	38.88
27	40.11
28	41.34
29	42.56
30	43.77
40	55.76
60	79.08
80	101.88
100	124.34

Table B.3 Critical values two-tailed Mann-Whitney (5%), see Sect. 10.3.5

N_A \ N_B	5	6	7	8	9	10	11	12
3	0	1	1	2	2	3	3	4
4	1	2	3	4	4	5	6	7
5	2	3	5	6	7	8	9	11
6		5	6	8	10	11	13	14
7			8	10	12	14	16	18
8				13	15	17	19	22
9					17	20	23	26
10						23	26	29
11							30	33
12								37

Table B.4 Critical values two-tailed matched-pair Wilcoxon test (5%), see Sect. 10.3.8

n	T
6	0
7	2
8	3
9	5
10	8
11	10
12	13
13	17
14	21
15	25
16	29
17	34
18	40
19	46
20	52
22	66
25	89

Please note that in Table B.3, N_A is for the smaller sample and N_B for the larger sample.

Please note that Table B.5 provides the upper 0.025% point of the F distribution with f_1 and f_2 being the degrees of freedom. This is equivalent to $F_{0.0025,f_1,f_2}$.

Table B.5 Critical values two-tailed F-test (5%), see Sect. 10.3.6. For ANOVA, this is equivalent to a significance level of 2.5%, see Sect. 10.3.10

f_1 / f_2	1	2	3	4	5	6	7	8	9	10	12	15	20	30	40	60	120	∞
1	648	800	864	900	922	937	948	957	963	969	977	985	993	1,001	1,006	1,010	1,014	1,018
2	38.5	39.0	39.2	39.2	39.3	39.3	39.4	39.4	39.4	39.4	39.4	39.4	39.4	39.5	39.5	39.5	39.5	39.5
3	17.4	16.0	15.4	15.1	14.9	14.7	14.6	14.5	14.5	14.4	14.3	14.2	14.2	14.1	14.0	14.0	14.0	13.9
4	12.2	10.6	9.98	9.60	9.36	9.20	9.07	8.98	8.90	8.84	8.75	8.66	8.56	8.46	8.41	8.36	8.31	8.26
5	10.0	8.43	7.76	7.39	7.15	6.98	6.85	6.76	6.68	6.62	6.52	6.43	6.33	6.23	6.18	6.12	6.07	6.02
6	8.81	7.26	6.60	6.23	5.99	5.82	5.70	5.60	5.52	5.46	5.37	5.27	5.17	5.07	5.01	4.96	4.90	4.85
7	8.07	6.54	5.89	5.52	5.29	5.12	4.99	4.90	4.82	4.76	4.67	4.57	4.47	4.36	4.31	4.25	4.20	4.14
8	7.57	6.06	5.42	5.05	4.82	4.65	4.53	4.43	4.36	4.30	4.20	4.10	4.00	3.89	3.84	3.78	3.73	3.67
9	7.21	5.71	5.08	4.72	4.48	4.32	4.20	4.10	4.03	3.96	3.87	3.77	3.67	3.56	3.51	3.45	3.39	3.33
10	6.94	5.46	4.83	4.47	4.24	4.07	3.95	3.85	3.78	3.72	3.62	3.52	3.42	3.31	3.26	3.20	3.14	3.08
12	6.55	5.10	4.47	4.12	3.89	3.73	3.61	3.51	3.44	3.37	3.28	3.18	3.07	2.96	2.91	2.85	2.79	2.72
15	6.20	4.76	4.15	3.80	3.58	3.41	3.29	3.20	3.12	3.06	2.96	2.86	2.76	2.64	2.59	2.52	2.46	2.40
20	5.87	4.46	3.86	3.51	3.29	3.13	3.01	2.91	2.84	2.77	2.68	2.57	2.46	2.35	2.29	2.22	2.16	2.09
30	5.57	4.18	3.59	3.25	3.03	2.87	2.75	2.65	2.57	2.51	2.41	2.31	2.20	2.07	2.01	1.94	1.87	1.79
40	5.42	4.05	3.46	3.13	2.90	2.74	2.62	2.53	2.45	2.39	2.29	2.18	2.07	1.94	1.88	1.80	1.72	1.64
60	5.29	3.93	3.34	3.01	2.79	2.63	2.51	2.41	2.33	2.27	2.17	2.06	1.94	1.82	1.74	1.67	1.58	1.48
120	5.15	3.80	3.23	2.89	2.67	2.52	2.39	2.30	2.22	2.16	2.05	1.94	1.82	1.69	1.61	1.53	1.43	1.31
∞	5.02	3.69	3.12	2.79	2.57	2.41	2.29	2.19	2.11	2.05	1.94	1.83	1.71	1.57	1.48	1.39	1.27	1.00

References

1. Anastas, J.W., MacDonald, M.L.: Research Design for the Social Work and the Human Services, 2nd edn. Columbia University Press, New York (2000)
2. Andersson, C., Runeson, P.: A spiral process model for case studies on software quality monitoring – method and metrics. Softw. Process: Improv. Pract. **12**(2), 125–140 (2007). doi: 10.1002/spip.311
3. Andrews, A.A., Pradhan, A.S.: Ethical issues in empirical software engineering: the limits of policy. Empir. Softw. Eng. **6**(2), 105–110 (2001)
4. American Psychological Association: Ethical principles of psychologists and code of conduct. Am. Psychol. **47**, 1597–1611 (1992)
5. Avison, D., Baskerville, R., Myers, M.: Controlling action research projects. Inf. Technol. People **14**(1), 28–45 (2001). doi: 10.1108/09593840110384762. URL http://www.emeraldinsight.com/10.1108/09593840110384762
6. Babbie, E.R.: Survey Research Methods. Wadsworth, Belmont (1990)
7. Basili, V.R.: Quantitative evaluation of software engineering methodology. In: Proceedings of the First Pan Pacific Computer Conference, vol. 1, pp. 379–398. Australian Computer Society, Melbourne (1985)
8. Basili, V.R.: Software development: a paradigm for the future. In: Proceedings of the 13th Annual International Computer Software and Applications Conference, COMPSAC'89, Orlando, pp. 471–485. IEEE Computer Society Press, Washington (1989)
9. Basili, V.R.: The experimental paradigm in software engineering. In: H.D. Rombach, V.R. Basili, R.W. Selby (eds.) Experimental Software Engineering Issues: Critical Assessment and Future Directives. Lecture Notes in Computer Science, vol. 706. Springer, Berlin Heidelberg (1993)
10. Basili, V.R.: Evolving and packaging reading technologies. J. Syst. Softw. **38**(1), 3–12 (1997)
11. Basili, V.R., Weiss, D.M.: A methodology for collecting valid software engineering data. IEEE Trans. Softw. Eng. **10**(6), 728–737 (1984)
12. Basili, V.R., Selby, R.W.: Comparing the effectiveness of software testing strategies. IEEE Trans. Softw. Eng. **13**(12), 1278–1298 (1987)
13. Basili, V.R., Rombach, H.D.: The TAME project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. **14**(6), 758–773 (1988)
14. Basili, V.R., Green, S.: Software process evaluation at the SEL. IEEE Softw. **11**(4), pp. 58–66 (1994)
15. Basili, V.R., Selby, R.W., Hutchens, D.H.: Experimentation in software engineering. IEEE Trans. Softw. Eng. **12**(7), 733–743 (1986)
16. Basili, V.R., Caldiera, G., Rombach, H.D.: Experience factory. In: J.J. Marciniak (ed.) Encyclopedia of Software Engineering, pp. 469–476. Wiley, New York (1994)

DOI 10.1007/978-3-642-29044-2

17. Basili, V.R., Caldiera, G., Rombach, H.D.: Goal Question Metrics paradigm. In: J.J. Marciniak (ed.) Encyclopedia of Software Engineering, pp. 528–532. Wiley (1994)
18. Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M.V.: The empirical investigation of perspective-based reading. Empir. Soft. Eng. **1**(2), 133–164 (1996)
19. Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M.V.: Lab package for the empirical investigation of perspective-based reading. Technical report, Univeristy of Maryland (1998). URL http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html
20. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Trans. Softw. Eng. **25**(4), 456–473 (1999)
21. Baskerville, R.L., Wood-Harper, A.T.: A critical perspective on action research as a method for information systems research. J. Inf. Technol. **11**(3), 235–246 (1996). doi: 10.1080/026839696345289
22. Benbasat, I., Goldstein, D.K., Mead, M.: The case research strategy in studies of information systems. MIS Q. **11**(3), 369 (1987). doi: 10.2307/248684
23. Bergman, B., Klefsjö, B.: Quality from Customer Needs to Customer Satisfaction. Studentlitteratur, Lund (2010)
24. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. **80**(4), 571–583 (2007). doi: 10.1016/j.jss.2006.07.009
25. Brereton, P., Kitchenham, B.A., Budgen, D.: Using a protocol template for case study planning. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering. University of Bari, Italy (2008)
26. Briand, L.C., Differding, C.M., Rombach, H.D.: Practical guidelines for measurement-based process improvement. Softw. Process: Improv. Pract. **2**(4), 253–280 (1996)
27. Briand, L.C., El Emam, K., Morasca, S.: On the application of measurement theory in software engineering. Empir. Softw. Eng. **1**(1), 61–88 (1996)
28. Briand, L.C., Bunse, C., Daly, J.W.: A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs. IEEE Trans. Softw. Eng. **27**(6), 513–530 (2001)
29. British Psychological Society: Ethical principles for conducting research with human participants. Psychologist **6**(1), 33–35 (1993)
30. Budgen, D., Kitchenham, B.A., Charters, S., Turner, M., Brereton, P., Linkman, S.: Presenting software engineering results using structured abstracts: a randomised experiment. Empir. Softw. Eng. **13**, 435–468 (2008). doi: 10.1007/s10664-008-9075-7
31. Budgen, D., Burn, A.J., Kitchenham, B.A.: Reporting computing projects through structured abstracts: a quasi-experiment. Empir. Softw. Eng. **16**(2), 244–277 (2011). doi: 10.1007/s10664-010-9139-3
32. Campbell, D.T., Stanley, J.C.: Experimental and Quasi-experimental Designs for Research. Houghton Mifflin Company, Boston (1963)
33. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI(R): Guidelines for process integration and product improvement. Technical report, SEI (2003)
34. Ciolkowski, M., Differding, C.M., Laitenberger, O., Münch, J.: Empirical investigation of perspective-based reading: A replicated experiment. Technical report, 97-13, ISERN (1997)
35. Coad, P., Yourdon, E.: Object-Oriented Design, 1st edn. Prentice-Hall, Englewood (1991)
36. Cohen, J.: Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. Psychol. Bull. **70**, 213–220 (1968)
37. Cook, T.D., Campbell, D.T.: Quasi-experimentation – Design and Analysis Issues for Field Settings. Houghton Mifflin Company, Boston (1979)
38. Corbin, J., Strauss, A.: Basics of Qualitative Research, 3rd edn. SAGE, Los Angeles (2008)
39. Cruzes, D.S., Dybå, T.: Research synthesis in software engineering: a tertiary study. Inf. Softw. Technol. **53**(5), 440–455 (2011). doi: 10.1016/j.infsof.2011.01.004

40. Dalkey, N., Helmer, O.: An experimental application of the delphi method to the use of experts. Manag. Sci. **9**(3), 458–467 (1963)
41. DeMarco, T.: Controlling Software Projects. Yourdon Press, New York (1982)
42. Demming, W.E.: Out of the Crisis. MIT Centre for Advanced Engineering Study, MIT Press, Cambridge, MA (1986)
43. Dieste, O., Grimán, A., Juristo, N.: Developing search strategies for detecting relevant experiments. Empir. Softw. Eng. **14**, 513–539 (2009). URL http://dx.doi.org/10.1007/s10664-008-9091-7
44. Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C., Lindeberg, O.: Cooperative method development. Empir. Softw. Eng. **13**(3), 231–260 (2007). doi: 10.1007/s10664-007-9057-1
45. Doolan, E.P.: Experiences with Fagan's inspection method. Softw. Pract. Exp. **22**(2), 173–182 (1992)
46. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9-10), 833–859 (2008). doi: DOI:10.1016/j.infsof.2008.01.006
47. Dybå, T., Dingsøyr, T.: Strength of evidence in systematic reviews in software engineering. In: Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '08, Kaiserslautern, pp. 178–187. ACM, New York (2008). doi: http://doi.acm.org/10.1145/1414004.1414034
48. Dybå, T., Kitchenham, B.A., Jørgensen, M.: Evidence-based software engineering for practitioners. IEEE Softw. **22**, 58–65 (2005). doi: http://doi.ieeecomputersociety.org/10.1109/MS.2005.6
49. Dybå, T., Kampenes, V.B., Sjøberg, D.I.K.: A systematic review of statistical power in software engineering experiments. Inf. Softw. Technol. **48**(8), 745–755 (2006). doi: 10.1016/j.infsof.2005.08.009
50. Easterbrook, S., Singer, J., Storey, M.-A., Damian, D.: Selecting empirical methods for software engineering research. In: F. Shull, J. Singer, D.I. Sjøberg (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
51. Eick, S.G., Loader, C.R., Long, M.D., Votta, L.G., Vander Wiel, S.A.: Estimating software fault content before coding. In: Proceedings of the 14th International Conference on Software Engineering, Melbourne, pp. 59–65. ACM Press, New York (1992)
52. Eisenhardt, K.M.: Building theories from case study research. Acad. Manag. Rev. **14**(4), 532 (1989). doi: 10.2307/258557
53. Endres, A., Rombach, H.D.: A Handbook of Software and Systems Engineering – Empirical Observations, Laws and Theories. Pearson Addison-Wesley, Harlow/New York (2003)
54. Fagan, M.E.: Design and code inspections to reduce errors in program development. IBM Syst. J. **15**(3), 182–211 (1976)
55. Fenton, N.: Software measurement: A necessary scientific basis. IEEE Trans. Softw. Eng. **3**(20), 199–206 (1994)
56. Fenton, N., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thomson Computer Press, London (1996)
57. Fenton, N., Pfleeger, S.L., Glass, R.: Science and substance: A challenge to software engineers. IEEE Softw. **11**, 86–95 (1994)
58. Fink, A.: The Survey Handbook, 2nd edn. SAGE, Thousand Oaks/London (2003)
59. Flyvbjerg, B.: Five misunderstandings about case-study research. In: Qualitative Research Practice, concise paperback edn., pp. 390–404. SAGE, London (2007)
60. Frigge, M., Hoaglin, D.C., Iglewicz, B.: Some implementations of the boxplot. Am. Stat. **43**(1), 50–54 (1989)
61. Fusaro, P., Lanubile, F., Visaggio, G.: A replicated experiment to assess requirements inspection techniques. Empir. Softw. Eng. **2**(1), 39–57 (1997)
62. Glass, R.L.: The software research crisis. IEEE Softw. **11**, 42–47 (1994)
63. Glass, R.L., Vessey, I., Ramesh, V.: Research in software engineering: An analysis of the literature. Inf. Softw. Technol. **44**(8), 491–506 (2002). doi: 10.1016/S0950-5849(02)00049-6

64. Gómez, O.S., Juristo, N., Vegas, S.: Replication types in experimental disciplines. In: Proceedings of the 4th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano-Bozen (2010)
65. Gorschek, T., Wohlin, C.: Requirements abstraction model. Requir. Eng. **11**, 79–101 (2006). doi: 10.1007/s00766-005-0020-7
66. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A model for technology transfer in practice. IEEE Softw. **23**(6), 88–95 (2006)
67. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: Industry evaluation of the requirements abstraction model. Requir. Eng. **12**, 163–190 (2007). doi: 10.1007/s00766-007-0047-z
68. Grady, R.B., Caswell, D.L.: Software Metrics: Establishing a Company-Wide Program. Prentice-Hall, Englewood (1994)
69. Grant, E.E., Sackman, H.: An exploratory investigation of programmer performance under on-line and off-line conditions. IEEE Trans. Human Factor Electron. **HFE-8**(1), 33–48 (1967)
70. Gregor, S.: The nature of theory in information systems. MIS Q. **30**(3), 491–506 (2006)
71. Hall, T., Flynn, V.: Ethical issues in software engineering research: a survey of current practice. Empir. Softw. Eng. **6**, 305–317 (2001)
72. Hannay, J.E., Sjøberg, D.I.K., Dybå, T.: A systematic review of theory use in software engineering experiments. IEEE Trans. Softw. Eng. **33**(2), 87–107 (2007). doi: 10.1109/TSE.2007.12
73. Hannay, J.E., Dybå, T., Arisholm, E., Sjøberg, D.I.K.: The effectiveness of pair programming: a meta-analysis. Inf. Softw. Technol. **51**(7), 1110–1122 (2009). doi: 10.1016/j.infsof.2009.02.001
74. Hayes, W.: Research synthesis in software engineering: a case for meta-analysis. In: Proceedings of the 6th International Software Metrics Symposium, Boca Raton, pp. 143–151 (1999)
75. Hetzel, B.: Making Software Measurement Work: Building an Effective Measurement Program. Wiley, New York (1993)
76. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
77. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. Empir. Softw. Eng. **5**(3), 201–214 (2000)
78. Höst, M., Wohlin, C., Thelin, T.: Experimental context classification: Incentives and experience of subjects. In: Proceedings of the 27th International Conference on Software Engineering, St. Louis, pp. 470–478 (2005)
79. Höst, M., Runeson, P.: Checklists for software engineering case study research. In: Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement, Madrid, pp. 479–481 (2007)
80. Hove, S.E., Anda, B.: Experiences from conducting semi-structured interviews in empirical software engineering research. In: Proceedings of the 11th IEEE International Software Metrics Symposium, pp. 1–10. IEEE Computer Society Press, Los Alamitos (2005)
81. Humphrey, W.S.: Managing the Software Process. Addison-Wesley, Reading (1989)
82. Humphrey, W.S.: A Discipline for Software Engineering. Addison Wesley, Reading (1995)
83. Humphrey, W.S.: Introduction to the Personal Software Process. Addison Wesley, Reading (1997)
84. IEEE: IEEE standard glossary of software engineering terminology. Technical Report, IEEE Std 610.12-1990, IEEE (1990)
85. Iversen, J.H., Mathiassen, L., Nielsen, P.A.: Managing risk in software process improvement: an action research approach. MIS Q. **28**(3), 395–433 (2004)
86. Jedlitschka, A., Pfahl, D.: Reporting guidelines for controlled experiments in software engineering. In: Proceedings of the 4th International Symposium on Empirical Software Engineering, Noosa Heads, pp. 95–104 (2005)
87. Johnson, P.M., Tjahjono, D.: Does every inspection really need a meeting? Empir. Softw. Eng. **3**(1), 9–35 (1998)

88. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation. Springer, Kluwer Academic Publishers, Boston (2001)
89. Juristo, N., Vegas, S.: The role of non-exact replications in software engineering experiments. Empir. Softw. Eng. **16**, 295–324 (2011). doi: 10.1007/s10664-010-9141-9
90. Kachigan, S.K.: Statistical Analysis: An Interdisciplinary Introduction to Univariate and Multivariate Methods. Radius Press, New York (1986)
91. Kachigan, S.K.: Multivariate Statistical Analysis: A Conceptual Introduction, 2nd edn. Radius Press, New York (1991)
92. Kampenes, V.B., Dyba, T., Hannay, J.E., Sjø berg, D.I.K.: A systematic review of effect size in software engineering experiments. Inf. Softw. Technol. **49**(11–12), 1073–1086 (2007). doi: 10.1016/j.infsof.2007.02.015
93. Karahasanović, A., Anda, B., Arisholm, E., Hove, S.E., Jørgensen, M., Sjøberg, D., Welland, R.: Collecting feedback during software engineering experiments. Empir. Softw. Eng. **10**(2), 113–147 (2005). doi: 10.1007/s10664-004-6189-4. URL http://www.springerlink.com/index/10.1007/s10664-004-6189-4
94. Karlström, D., Runeson, P., Wohlin, C.: Aggregating viewpoints for strategic software process improvement. IEE Proc. Softw. **149**(5), 143–152 (2002). doi: 10.1049/ip-sen:20020696
95. Kitchenham, B.A.: The role of replications in empirical software engineering – a word of warning. Empir. Softw. Eng. **13**, 219–221 (2008). URL 10.1007/s10664-008-9061-0
96. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (version 2.3). Technical Report, EBSE Technical Report EBSE-2007-01, Keele University and Durham University (2007)
97. Kitchenham, B.A., Pickard, L.M., Pfleeger, S.L.: Case studies for method and tool evaluation. IEEE Softw. **12**(4), 52–62 (1995)
98. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8), 721–734 (2002). doi: 10.1109/TSE.2002.1027796. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1027796
99. Kitchenham, B., Fry, J., Linkman, S.G.: The case against cross-over designs in software engineering. In: Proceedings of the 11th International Workshop on Software Technology and Engineering Practice, Amsterdam, pp. 65–67. IEEE Computer Society, Los Alamitos (2003)
100. Kitchenham, B.A., Dybå, T., Jørgensen, M.: Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software Engineering, Edinburgh, pp. 273–281 (2004)
101. Kitchenham, B.A., Al-Khilidar, H., Babar, M.A., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., Zhu, L.: Evaluating guidelines for reporting empirical software engineering studies. Empir. Softw. Eng. **13**(1), 97–121 (2007). doi: 10.1007/s10664-007-9053-5. URL http://www.springerlink.com/index/10.1007/s10664-007-9053-5
102. Kitchenham, B.A., Jeffery, D.R., Connaughton, C.: Misleading metrics and unsound analyses. IEEE Softw. **24**, 73–78 (2007). doi: 10.1109/MS.2007.49
103. Kitchenham, B.A., Brereton, P., Budgen, D., Turner, M., Bailey, J., Linkman, S.G.: Systematic literature reviews in software engineering – a systematic literature review. Inf. Softw. Technol. **51**(1), 7–15 (2009). doi: 10.1016/j.infsof.2008.09.009. URL http://www.dx.doi.org/10.1016/j.infsof.2008.09.009
104. Kitchenham, B.A., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., Linkman, S.: Systematic literature reviews in software engineering – a tertiary study. Inf. Softw. Technol. **52**(8), 792–805 (2010). doi: 10.1016/j.infsof.2010.03.006
105. Kitchenham, B.A., Sjøberg, D.I.K., Brereton, P., Budgen, D., Dybå, T., Höst, M., Pfahl, D., Runeson, P.: Can we evaluate the quality of software engineering experiments? In: Proceedings of the 4th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, Bolzano/Bozen (2010)
106. Kitchenham, B.A., Budgen, D., Brereton, P.: Using mapping studies as the basis for further research – a participant-observer case study. Inf. Softw. Technol. **53**(6), 638–651 (2011). doi: 10.1016/j.infsof.2010.12.011

107. Laitenberger, O., Atkinson, C., Schlich, M., El Emam, K.: An experimental comparison of reading techniques for defect detection in UML design documents. J. Syst. Softw. **53**(2), 183–204 (2000)
108. Larsson, R.: Case survey methodology: quantitative analysis of patterns across case studies. Acad. Manag. J. **36**(6), 1515–1546 (1993)
109. Lee, A.S.: A scientific methodology for MIS case studies. MIS Q. **13**(1), 33 (1989). doi: 10.2307/248698. URL http://www.jstor.org/stable/248698?origin=crossref
110. Lehman, M.M.: Program, life-cycles and the laws of software evolution. Proc. IEEE **68**(9), 1060–1076 (1980)
111. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying software engineers: data collection techniques for software field studies. Empir. Softw. Eng. **10**, 311–341 (2005)
112. Linger, R.: Cleanroom process model. IEEE Softw. pp. 50–58 (1994)
113. Linkman, S., Rombach, H.D.: Experimentation as a vehicle for software technology transfer – a family of software reading techniques. Inf. Softw. Technol. **39**(11), 777–780 (1997)
114. Lucas, W.A.: The case survey method: aggregating case experience. Technical Report, R-1515-RC, The RAND Corporation, Santa Monica (1974)
115. Lucas, H.C., Kaplan, R.B.: A structured programming experiment. Comput. J. **19**(2), 136–138 (1976)
116. Lyu, M.R. (ed.): Handbook of Software Reliability Engineering. McGraw-Hill, New York (1996)
117. Maldonado, J.C., Carver, J., Shull, F., Fabbri, S., Dória, E., Martimiano, L., Mendonça, M., Basili, V.: Perspective-based reading: a replicated experiment focused on individual reviewer effectiveness. Empir. Softw. Eng. **11**, 119–142 (2006). doi: 10.1007/s10664-006-5967-6
118. Manly, B.F.J.: Multivariate Statistical Methods: A Primer, 2nd edn. Chapman and Hall, London (1994)
119. Marascuilo, L.A., Serlin, R.C.: Statistical Methods for the Social and Behavioral Sciences. W. H. Freeman and Company, New York (1988)
120. Miller, J.: Estimating the number of remaining defects after inspection. Softw. Test. Verif. Reliab. **9**(4), 167–189 (1999)
121. Miller, J.: Applying meta-analytical procedures to software engineering experiments. J. Syst. Softw. **54**(1), 29–39 (2000)
122. Miller, J.: Statistical significance testing: a panacea for software technology experiments? J. Syst. Softw. **73**, 183–192 (2004). doi: http://dx.doi.org/10.1016/j.jss.2003.12.019
123. Miller, J.: Replicating software engineering experiments: a poisoned chalice or the holy grail. Inf. Softw. Technol. **47**(4), 233–244 (2005)
124. Miller, J., Wood, M., Roper, M.: Further experiences with scenarios and checklists. Empir. Softw. Eng. **3**(1), 37–64 (1998)
125. Montgomery, D.C.: Design and Analysis of Experiments, 5th edn. Wiley, New York (2000)
126. Myers, G.J.: A controlled experiment in program testing and code walkthroughs/inspections. Commun. ACM **21**, 760–768 (1978). doi: http://doi.acm.org/10.1145/359588.359602
127. Noblit, G.W., Hare, R.D.: Meta-Ethnography: Synthesizing Qualitative Studies. Sage Publications, Newbury Park (1988)
128. Ohlsson, M.C., Wohlin, C.: A project effort estimation study. Inf. Softw. Technol. **40**(14), 831–839 (1998)
129. Owen, S., Brereton, P., Budgen, D.: Protocol analysis: a neglected practice. Commun. ACM **49**(2), 117–122 (2006). doi: 10.1145/1113034.1113039
130. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model for software. Technical Report, CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburgh (1993)
131. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, Electronic Workshops in Computing (eWIC). BCS, University of Bari, Italy (2008)
132. Petersen, K., Wohlin, C.: Context in industrial software engineering research. In: Proceedings of the 3rd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, pp. 401–404 (2009)

133. Pfleeger, S.L.: Experimental design and analysis in software engineering part 1–5. ACM Sigsoft, Softw. Eng. Notes, **19**(4), 16–20; **20**(1), 22–26; **20**(2), 14–16; **20**(3), 13–15; **20**, (1994)
134. Pfleeger, S.L., Atlee, J.M.: Software Engineering: Theory and Practice, 4th edn. Pearson Prentice-Hall, Upper Saddle River (2009)
135. Pickard, L.M., Kitchenham, B.A., Jones, P.W.: Combining empirical results in software engineering. Inf. Softw. Technol. **40**(14), 811–821 (1998). doi: 10.1016/S0950-5849(98)00101-3
136. Porter, A.A., Votta, L.G.: An experiment to assess different defect detection methods for software requirements inspections. In: Proceedings of the 16th International Conference on Software Engineering, Sorrento, pp. 103–112 (1994)
137. Porter, A.A., Votta, L.G.: Comparing detection methods for software requirements inspection: a replicated experiment. IEEE Trans. Softw. Eng. **21**(6), 563–575 (1995)
138. Porter, A.A., Votta, L.G.: Comparing detection methods for software requirements inspection: a replicated experimentation: a replication using professional subjects. Empir. Softw. Eng. **3**(4), 355–380 (1998)
139. Porter, A.A., Siy, H.P., Toman, C.A., Votta, L.G.: An experiment to assess the cost-benefits of code inspections in large scale software development. IEEE Trans. Softw. Eng. **23**(6), 329–346 (1997)
140. Potts, C.: Software engineering research revisited. IEEE Softw. pp. 19–28 (1993)
141. Rainer, A.W.: The longitudinal, chronological case study research strategy: a definition, and an example from IBM Hursley Park. Inf. Softw. Technol. **53**(7), 730–746 (2011)
142. Robinson, H., Segal, J., Sharp, H.: Ethnographically-informed empirical studies of software practice. Inf. Softw. Technol. **49**(6), 540–551 (2007). doi: 10.1016/j.infsof.2007.02.007
143. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioners-Researchers, 1st edn. Blackwell, Oxford/Cambridge (1993)
144. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioners-Researchers, 2nd edn. Blackwell, Oxford/Madden (2002)
145. Runeson, P., Skoglund, M.: Reference-based search strategies in systematic reviews. In: Proceedings of the 13th International Conference on Empirical Assessment and Evaluation in Software Engineering. Electronic Workshops in Computing (eWIC). BCS, Durham University, UK (2009)
146. Runeson, P., Höst, M., Rainer, A.W., Regnell, B.: Case Study Research in Software Engineering. Guidelines and Examples. Wiley, Hoboken (2012)
147. Sandahl, K., Blomkvist, O., Karlsson, J., Krysander, C., Lindvall, M., Ohlsson, N.: An extended replication of an experiment for assessing methods for software requirements. Empir. Softw. Eng. **3**(4), 381–406 (1998)
148. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. **25**(4), 557–572 (1999)
149. Selby, R.W., Basili, V.R., Baker, F.T.: Cleanroom software development: An empirical evaluation. IEEE Trans. Softw. Eng. **13**(9), 1027–1037 (1987)
150. Shepperd, M.: Foundations of Software Measurement. Prentice-Hall, London/New York (1995)
151. Shneiderman, B., Mayer, R., McKay, D., Heller, P.: Experimental investigations of the utility of detailed flowcharts in programming. Commun. ACM **20**, 373–381 (1977). doi: 10.1145/359605.359610
152. Shull, F.: Developing techniques for using software documents: a series of empirical studies. Ph.D. thesis, Computer Science Department, University of Maryland, USA (1998)
153. Shull, F., Basili, V.R., Carver, J., Maldonado, J.C., Travassos, G.H., Mendonça, M.G., Fabbri, S.: Replicating software engineering experiments: addressing the tacit knowledge problem. In: Proceedings of the 1st International Symposium on Empirical Software Engineering, Nara, pp. 7–16 (2002)
154. Shull, F., Mendoncça, M.G., Basili, V.R., Carver, J., Maldonado, J.C., Fabbri, S., Travassos, G.H., Ferreira, M.C.: Knowledge-sharing issues in experimental software engineering. Empir. Softw. Eng. **9**, 111–137 (2004). doi: 10.1023/B:EMSE.0000013516.80487.33

155. Shull, F., Carver, J., Vegas, S., Juristo, N.: The role of replications in empirical software engineering. Empir. Softw. Eng. **13**, 211–218 (2008). doi: 10.1007/s10664-008-9060-1
156. Sieber, J.E.: Protecting research subjects, employees and researchers: implications for software engineering. Empir. Softw. Eng. **6**(4), 329–341 (2001)
157. Siegel, S., Castellan, J.: Nonparametric Statistics for the Behavioral Sciences, 2nd edn. McGraw-Hill International Editions, New York (1988)
158. Singer, J., Vinson, N.G.: Why and how research ethics matters to you. Yes, you! Empir. Softw. Eng. **6**, 287–290 (2001). doi: 10.1023/A:1011998412776
159. Singer, J., Vinson, N.G.: Ethical issues in empirical studies of software engineering. IEEE Trans. Softw. Eng. **28**(12), 1171–1180 (2002). doi: 10.1109/TSE.2002.1158289. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1158289
160. Simon S.: Fermat's Last Theorem. Fourth Estate, London (1997)
161. Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.-K., Rekdal, A.C.: A survey of controlled experiments in software engineering. IEEE Trans. Softw. Eng. **31**(9), 733–753 (2005). doi: 10.1109/TSE.2005.97. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1514443
162. Sjøberg, D.I.K., Dybå, T., Anda, B., Hannay, J.E.: Building theories in software engineering. In: Shull, F., Singer, J., Sjøberg D. (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
163. Sommerville, I.: Software Engineering, 9th edn. Addison-Wesley, Wokingham, England/Reading (2010)
164. Sørumgård, S.: Verification of process conformance in empirical studies of software development. Ph.D. thesis, The Norwegian University of Science and Technology, Department of Computer and Information Science, Norway (1997)
165. Stake, R.E.: The Art of Case Study Research. SAGE Publications, Thousand Oaks (1995)
166. Staples, M., Niazi, M.: Experiences using systematic review guidelines. J. Syst. Softw. **80**(9), 1425–1437 (2007). doi: 10.1016/j.jss.2006.09.046
167. Thelin, T., Runeson, P.: Capture-recapture estimations for perspective-based reading – a simulated experiment. In: Proceedings of the 1st International Conference on Product Focused Software Process Improvement (PROFES), Oulu, pp. 182–200 (1999)
168. Thelin, T., Runeson, P., Wohlin, C.: An experimental comparison of usage-based and checklist-based reading. IEEE Trans. Softw. Eng. **29**(8), 687–704 (2003). doi: 10.1109/TSE.2003.1223644
169. Tichy, W.F.: Should computer scientists experiment more? IEEE Comput. **31**(5), 32–39 (1998)
170. Tichy, W.F., Lukowicz, P., Prechelt, L., Heinz, E.A.: Experimental evaluation in computer science: a quantitative study. J. Syst. Softw. **28**(1), 9–18 (1995)
171. Trochim, W.M.K.: The Research Methods Knowledge Base, 2nd edn. Cornell Custom Publishing, Cornell University, Ithaca (1999)
172. van Solingen, R., Berghout, E.: The Goal/Question/Metric Method: A Practical Guide for Quality Improvement and Software Development. McGraw-Hill International, London/Chicago (1999)
173. Verner, J.M., Sampson, J., Tosic, V., Abu Bakar, N.A., Kitchenham, B.A.: Guidelines for industrially-based multiple case studies in software engineering. In: Third International Conference on Research Challenges in Information Science, Fez, pp. 313–324 (2009)
174. Vinson, N.G., Singer, J.: A practical guide to ethical research involving humans. In: Shull, F., Singer, J., Sjøberg, D. (eds.) Guide to Advanced Empirical Software Engineering. Springer, London (2008)
175. Votta, L.G.: Does every inspection need a meeting? In: Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering, ACM Software Engineering Notes, vol. 18, pp. 107–114. ACM Press, New York (1993)
176. Wallace, C., Cook, C., Summet, J., Burnett, M.: Human centric computing languages and environments. In: Proceedings of Symposia on Human Centric Computing Languages and Environments, Arlington, pp. 63–65 (2002)

177. Wohlin, C., Gustavsson, A., Höst, M., Mattsson, C.: A framework for technology introduction in software organizations. In: Proceedings of the Conference on Software Process Improvement, Brighton, pp. 167–176 (1996)
178. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer, Boston (2000)
179. Wohlin, C., Aurum, A., Angelis, L., Phillips, L., Dittrich, Y., Gorschek, T., Grahn, H., Henningsson, K., Kågström, S., Low, G., Rovegård, P., Tomaszewski, P., van Toorn, C., Winter, J.: Success factors powering industry-academia collaboration in software research. IEEE Softw. (PrePrints) (2011). doi: 10.1109/MS.2011.92
180. Yin, R.K.: Case Study Research Design and Methods, 4th edn. Sage Publications, Beverly Hills (2009)
181. Zelkowitz, M.V., Wallace, D.R.: Experimental models for validating technology. IEEE Comput. **31**(5), 23–31 (1998)
182. Zendler, A.: A preliminary software engineering theory as investigated by published experiments. Empir. Softw. Eng. **6**, 161–180 (2001). doi: http://dx.doi.org/10.1023/A:1011489321999

Index

C. Wohlin et al., *Experimentation in Software Engineering*,
DOI 10.1007/978-3-642-29044-2, © Springer-Verlag Berlin Heidelberg 2012