



# Empirical Software Engineering Research

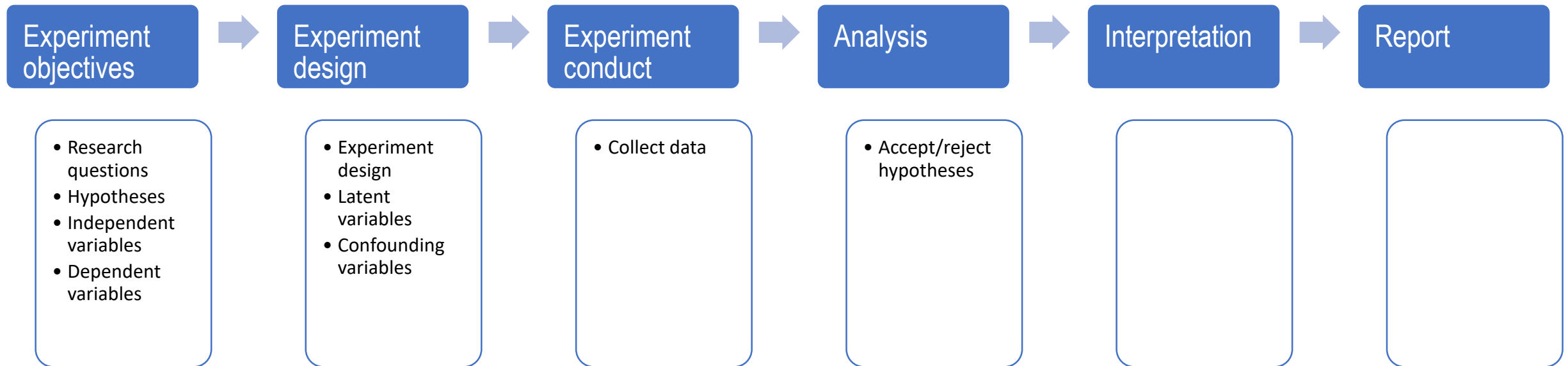
## *Experiment Design*

Norman Peitek, Annabelle Bergum, Lina Lampel, Sven Apel

# Learning Goals

- Gain an understanding of different experiment designs
- Familiarize yourself with choosing the right experiment design
- Be able to distinguish between-subject and within-subject designs

# Overview



Quality of the experiment can only degrade, not improve (a great analysis does not save bad design)

- There are many ways an experiment can go wrong
  - What if all skilled/motivated programmers end up in one group (potentially for a reason)?
  - Sampling: Where do we get participants from? Are they representative? Are students representative?
  - Skill: Do all participants have the same/comparable skills? How do we test that?
  - Task: Is the task well-chosen? Is it representative?
- In short, there are many ways that influence the quality of an experiment
- We need to design experiments in a way to reduces bias

- Many, many different designs
- In this course, we are going to focus on the major, simple designs
  - These are typical for SE research
  - Complex designs are more prevalent in other sciences
- How to select the right one?
  - Depends on sample size and effect size
  - If both are large, the confounding factors have less of an impact
  - But typically, samples tend to be small and effect sizes unknown → experiment design is critical

Will discuss  
tomorrow

# Experiment Designs: Basic Terms

- Within-subject and between-subject design
  - = One group versus two (or more) groups
- With vs. without repeated measures
  - Multiple (repeated) tasks with different stimuli
- One-factorial or multi-factorial design
  - = One vs. several independent variables
- Univariate versus multi-variate design
  - = One vs. several dependent variables

# Experiment Designs: Why?

- Instruction to act
- Makes communication easier
- Decision for statistical analysis

# One-Factorial Designs





# Between-Subject Design

- Participants are divided into two (or more) groups
- As many groups as there are levels of the independent variable
- Results are compared between groups

Novice Programmers	Expert Programmers
Program Comprehension	Program Comprehension

# Between-Subject Design: Potential Issues

- Requires large(r) number of participants
- Variance between participants (inter-individual differences)
- Requires a decent balancing between groups
  - For example, balancing, randomization, ...
  - If participants are aware of the different groups, they can react in a biasing way, for example:
    - Compensatory rivalry
    - Resentful demoralization



# Within-Subject Design

- All participants are in one group
- Each participant experiences two (or more) levels of an independent variable
- Within-subject design controls for potential inter-individual differences
- Example: program comprehension versus syntax errors

(All) Programmers	
Program Comprehension	Syntax Errors

# Within-Subject Design: Potential issues

- Learning effects
  - Especially problematic with creative tasks
  - Requires different, but similar tasks, which can be quite challenging

Heineken



Walk on a  
straight line



Krombacher



Walk on a  
straight line



→ Krombacher is  
stronger than  
Heineken?

# Within-Subject Design: Potential issues

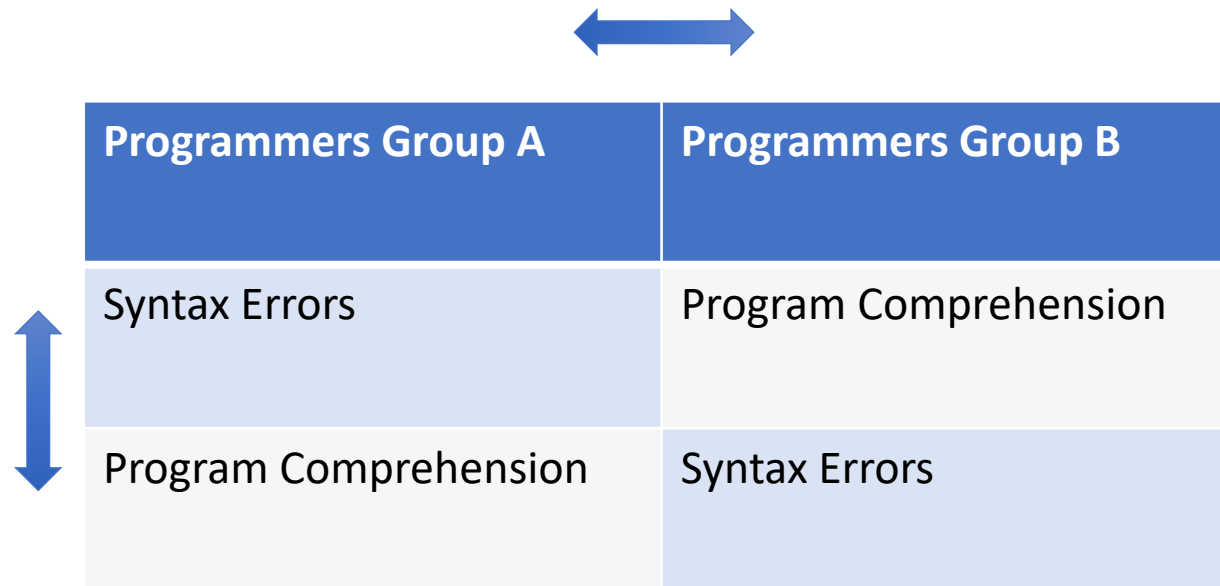
- Ordering effects
  - For example, Python → Java might be different than Java → Python
- Intra-individual differences
  - Fatigue
  - Motivation
- Mortality
  - If multiple sessions, some participants might drop out over time
  - Increases with larger gaps and number of gaps

# Between-Subject & Within-Subject Design

- In medicine, ~80% of the studies are between-subject design
- In SE, the ratio is likely the other way around
  - One reason: Programming is a complex task with substantial individual differences

# Crossover Design

- Combination of within-subject and between-subject design



- Increased complexity that combines some issues, but also helps with other issues
- Quickly grows in complexity with more variables

# Crossover Design: Issues

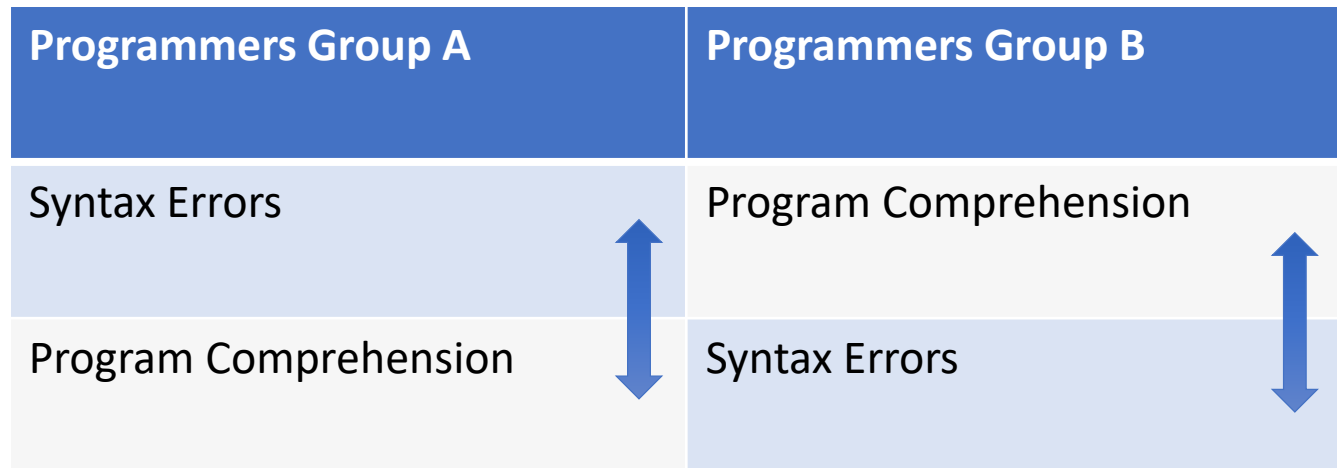
## Benefits

- Check for learning effects
- Check for ordering effects

## Issues

- Intra-individual differences
- Inter-individual differences
- Mortality

Programmers Group A	Programmers Group B
Syntax Errors	Program Comprehension
Program Comprehension	Syntax Errors





In addition to the already mentioned ones, there are many potential threats to validity that can affect an experiment!

- History: experiment circumstances could affect more so than the experiment
  - For example, Task A on Monday morning, Task B on Friday afternoon
- Maturation: participants react differently through the course of an experiment
  - For example, participant getting tired throughout the experiment
  - Or, participant learning throughout the experiment
- Ambiguity of causality: Does A cause B, B cause A, or C cause A and B?
  - For example, does high program complexity cause a high error rate, or vice versa? Or does a complex problem cause both, high program complexity and a high error rate?

# Threats to Validity: Multiple Groups

- Population bias: participants for each group may bias the results
- Never ask participants in which group they want to be!
  - For example, do you want to complete the experiment in your known programming language or the super cool new programming language I made up?
  - Participants selecting the novel choice will likely have different characteristics that influence the results more than the actual difference in programming language

- Mono-operation bias: If the experiment includes a single independent variable/treatment, it may under-represent the construct
  - For example, study program comprehension on a single snippet (rather than multiple snippets)
- Mono-method bias: If there is only one type of measurement it might introduce some bias
- Hypothesis guessing: participants may try to “figure out” the experiment and change their behavior

# Repeated Measures Design



# Repeated Measures Design

In combination with all the experiment design choices, researchers can conduct a repeated measures design.

- By taking multiple measures for the same variable, we can reduce bias

Novice Programmers	Expert Programmers
Program Comprehension 1	Program Comprehension 1
Program Comprehension 2	Program Comprehension 2
Program Comprehension 3	Program Comprehension 3

- Repeated measures design are useful with small numbers of participants
- Depending on the task length, it can be quite helpful or impossible
  - For example, when the task is 30 minutes, it is unrealistic to ask participants for multiple measures as fatigue and mortality would outweigh the benefits
  - But, if the task is one minute, the preparation takes way longer → might as well keep the participants busy for 20-30 minutes
- But, be careful about ordering and learning effects even within the same kind of task!

# Repeated Measures Design: Aggregation



- Let's imagine a study that tests a new tool to fix bugs
- The researcher conducts a study that lets programmers use the new tool
  - One proposed study design lets one programmer do one bug fix task
  - One proposed study design lets five programmers do one bug fix task
  - An alternative study design lets five programmers do five bug fix tasks
- Which one is preferable?

# Repeated Measures Design: Aggregation

- The keyword is aggregation of data
  - Data can be aggregated along participants
  - Data can be aggregated along tasks
  - Or both
- For 2012-2020, of 144 papers...
  - 22 investigated individual tasks/participants
  - 71 investigated aggregated results
  - 51 used a combined approach
- We investigated 12 studies that only used an individual analysis
  - For half of them, the aggregated analysis led to a (partially) changed conclusion

Participant	Task 1	Task 2	Task 3	Total
Expert 1	5	2	8	5
Expert 2	4	4	4	4
Expert 3	2	3	4	3
Expert 4	5	4	6	5
Experts	4	3.25	5.5	4.25
Novice 1	3	6	9	6
Novice 2	3	3	3	3
Novice 3	6	4	5	5
Novice 4	8	6	10	8
Novices	5	4.75	6.75	5.5



# Repeated Measures Design: Aggregation

- Too much aggregation can be problematic, too
- “Do background colors improve program comprehension in the #ifdef hell?”
  - <https://link.springer.com/article/10.1007/s10664-012-9208-x>

```
1 static int __rep_queue_filedone(dbenv, rep, rfp)
2   DB_ENV *dbenv;
3   REP *rep;
4   __rep_fileinfo_args *rfp; {
5   #ifndef HAVE_QUEUE
6     COMPQUIET(rep, NULL);
7     COMPQUIET(rfp, NULL);
8     return (__db_no_queue_am(dbenv));
9   #else
10    db_pgno_t first, last;
11    u_int32_t flags;
12    int empty, ret, t_ret;
13    #ifdef DIAGNOSTIC
14      DB_MSGBUF mb;
15    #endif
16    // over 100 lines of additional code
17  #endif
18 }
```



```
1 static int __rep_queue_filedone(dbenv, rep, rfp)
2   DB_ENV *dbenv;
3   REP *rep;
4   __rep_fileinfo_args *rfp; {
5   #ifndef HAVE_QUEUE
6     COMPQUIET(rep, NULL);
7     return (__db_no_queue_am(dbenv));
8   #else
9     db_pgno_t first, last;
10    u_int32_t flags;
11    int empty, ret, t_ret;
12    #ifdef DIAGNOSTIC
13      DB_MSGBUF mb;
14    #endif
15    // over 100 lines of additional code
16  #endif
17 }
```

Kind of Annotation	Maintenance Tasks		
	Task	Time [s]	Correct / Incorrect
Background Colors	M1	414	21 / 1
	M2	342	21 / 1
	M3	468	19 / 3
	M4	1404	15 / 7
Average/sum		657.13	76 / 12
#ifdefs	M1	432	19 / 2
	M2	354	19 / 2
	M3	396	12 / 9
	M4	882	12 / 9
Average/sum		515.99	62 / 22
t test/ $\chi^2$		-2.49	3.52
p value		0.012	0.061

# Increasing Validity



- In a perfect world, every empirical study would be perfect: right theory for our observations, study is planned and executed perfectly, data analysis is done correctly, interpretation is accurate and comprehensive
- But we do not live in a perfect world, which is okay. But we must
  - evaluate the quality of the research as an integral part
  - provide information to allow replication

# Increasing Validity: Minimum

To enable other researchers to assess the validity of your experiment, you must:

- Report all necessary information regarding methods and materials, including participants, tasks, etc.
- Thoroughly report the observations and separate your interpretation
- Discuss threats to validity

Often boring, technical and repetitive, but important! It makes your report scientific.

# Increasing Validity: Group Assignment

A good group assignment is:

- Uniform: vary between groups in relevant factors, within groups only in irrelevant factors
- Equal: every participant in a group is treated equally
- Balanced: all groups should be of similar size
- Large: As a rule of thumb, plan with at least 10-15 participants per group
- Randomized: assignment is free of bias (talked about options yesterday)

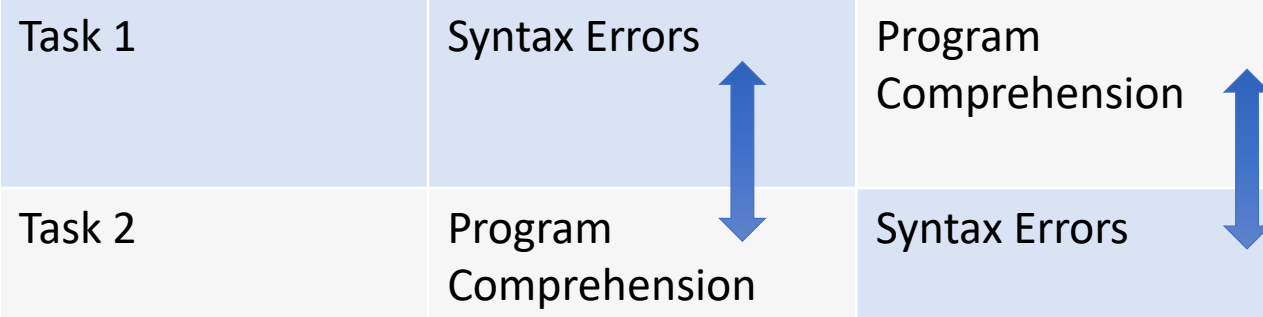
# Multi-Factorial Designs



# Latin-Square Design

- Special case of a crossover design
- Different tasks in each session (= task is second independent variable)

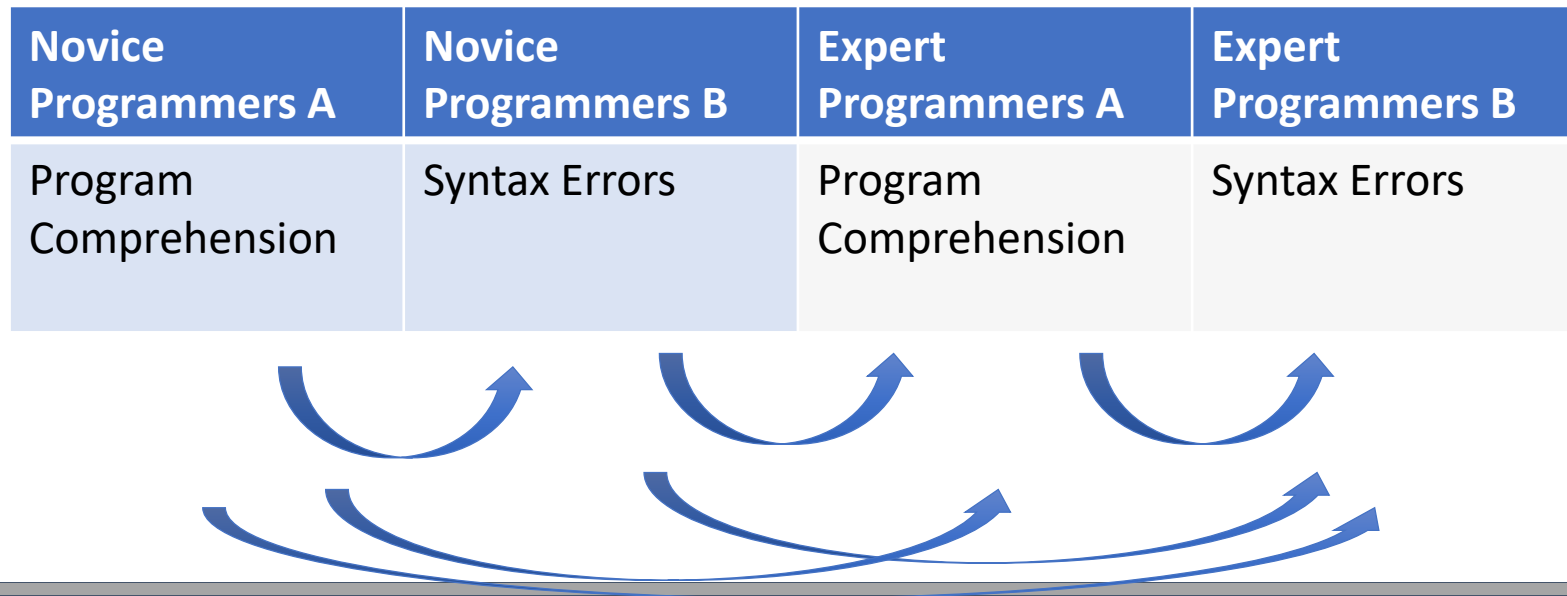
	Programmers Group A	Programmers Group B
Task 1	Syntax Errors	Program Comprehension
Task 2	Program Comprehension	Syntax Errors



# Two-Factorial, Between-Subject Design

Independent variables:

- Programmer experience
- Cognitive process





# Two-Factorial, Within-Subject Design

Independent variables:

- Programmer experience
- Cognitive process

Novice Programmers A	Novice Programmers B	Expert Programmers A	Expert Programmers B
Program Comprehension	Syntax Errors	Program Comprehension	Syntax Errors
Syntax Errors	Program Comprehension	Syntax Errors	Program Comprehension

# Multi-Factorial designs

- Only if shown experiment designs are insufficient
- For example, a 4-factorial design with 2x2x3x2 factors
- Typically requires *many* samples

		C <sub>1</sub>		C <sub>2</sub>		C <sub>3</sub>	
		B <sub>1</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>2</sub>
A <sub>1</sub>	D <sub>1</sub>						
	D <sub>2</sub>						
A <sub>2</sub>	D <sub>1</sub>						
	D <sub>2</sub>						



By default, go for as simple as possible

A well-balanced simple design is much preferable over a convoluted complex experiment design



Carefully consider benefits and drawbacks



Consider resource constraints (realistically)

If your planned experiment design requires 50 participants for each of the 10 groups, decide whether that is realistic. If in doubt, simplify.

## **What Drives the Reading Order of Programmers? An Eye Tracking Study**

Norman Peitek  
Leibniz Institute for Neurobiology  
Magdeburg, Germany

Janet Siegmund  
Chemnitz University of Technology  
Chemnitz, Germany

Sven Apel  
Saarland University  
Saarbrücken, Germany

- What kind of experiment design was used?

# Further Reading

<https://www.se.cs.uni-saarland.de/publications/docs/SPA+20.pdf>