Empirical Research in Software Engineering

Stefan Kapferer

June 14, 2019

University of Applied Sciences of Eastern Switzerland (HSR FHO)

Table of contents

Problem & Motivation

Why do we need Software Engineering?

Software Engineering Definition

Software engineering aims to provide systematic methods and tools to overcome the challenges we face in software development.

- Software development is a complex process.
- It is based on human-based activities and creativity.
- “Software is developed and not produced.” [3]

> *“(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).” - IEEE [1]*

Why Empirical Research in Software Engineering?

Why Empirical Research in Software Engineering?

These methods and tools applied to software development evolve and change.

- Continuous improvement
- Evaluation of applicability and effectiveness
- The similarities to social and behavioral sciences do not allow to be as formal as in mathematics or physics.
  - Software engineering is a discipline **highly based on human activities and creativity**.

The Empirical Method

Empirical research strategies can help measuring and analyzing the impact of new methods and tools in software engineering:

- Is a variation of the **scientific method** [3, 5].
- Uses statistical and qualitative methods.
- Knowledge is gained by **observation or experiments applied to practice**.

Applying Empirical Research

Empirical research strategies aim for understanding a phenomenon in the “real world” context.

- They can be used to evaluate results of research and thesis projects in the industry.
- For example: understand and analyze the capabilities of a new tool.

Application Example

Context Mapper [6]: An Example Thesis Project

A Domain-specific Language for Context Mapping & Service Decomposition[1]

CONTEXT MAPPER

- Modeling Domain-driven Design (DDD) Context Maps
- Practitioners using this DDD concept have to draw these context maps by hand (no other tools available yet).

```
ContextMap {
  /* Add Bounded Contexts to Context Map */
  contains CustomerManagement, CustomerSelfService, DebtCollection

  /* Define Bounded Context Relationships: */

  CustomerSelfService [D,C]<-[U,S] CustomerManagement

  PolicyManagement [SK]<->[SK] DebtCollection
}
```

[1] https://contextmapper.github.io/

Context Mapper [6]: An Example Thesis Project

Our hypothesis:

> *"Software architects and DDD adopters can benefit from a tool which supports the creation of DDD-based models in a formal and expressive way. Thereby, the models can be transformed and evolved iteratively, which increases the process and productivity."*

Context Mapper [6]: An Example Thesis Project

Example research questions to be answered by using empirical studies:

- **RQ1:** Does the tool fulfill the practitioners requirements regarding usability and expressiveness to model their context maps with the tool?
- **RQ2:** Does the tool reduce the effort required by the software architects and improves the productivity over the project lifecycle?
- **RQ3:** Which factors are relevant whether a company would use the tool or not?

Empirical Strategies

Overview of Empirical Strategies

Fundamental strategies of empirical software engineering [2, 8]:

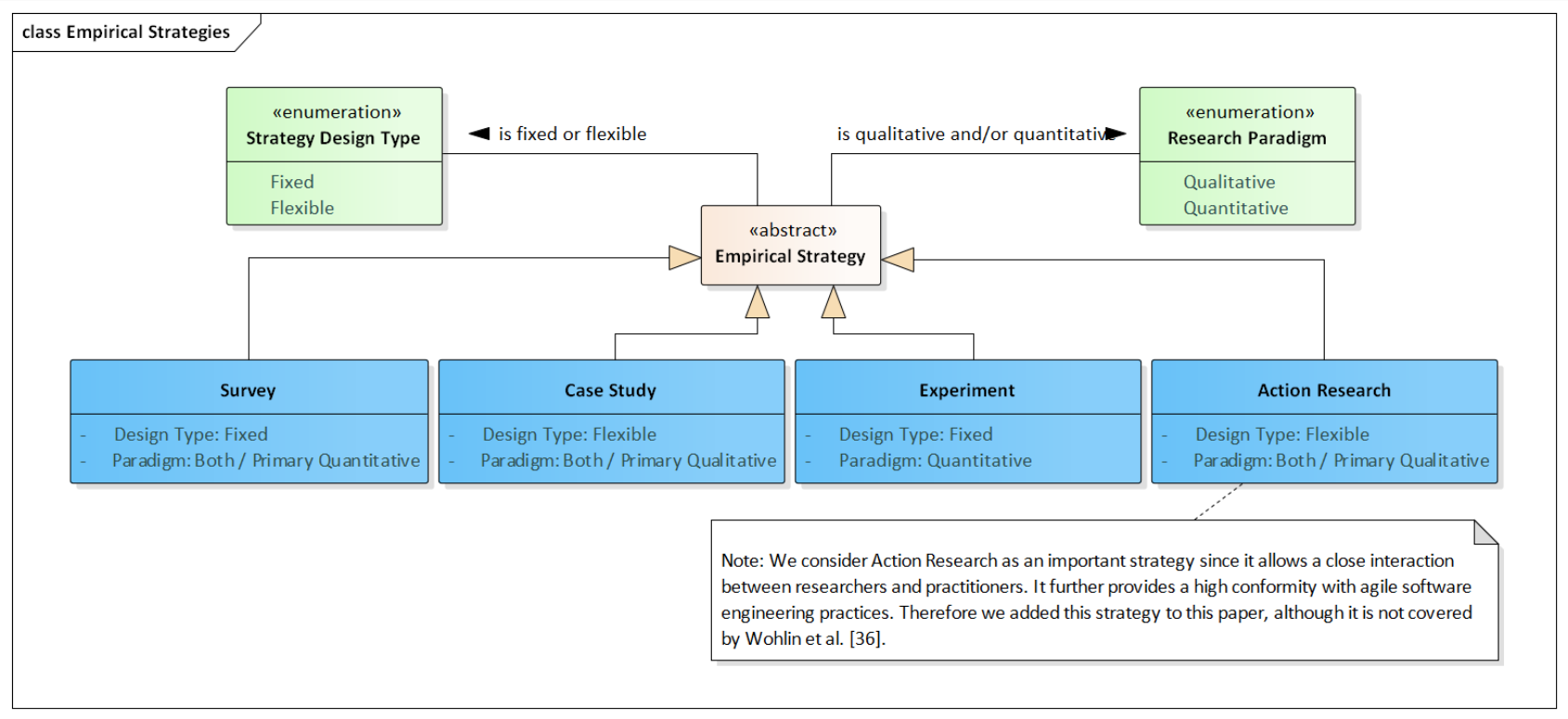


Figure 1: Empirical Strategies in Software Engineering

Survey

With **surveys** we can:

- Collect data from or about people.
- Derive a generalized opinion of a broad population of individuals.

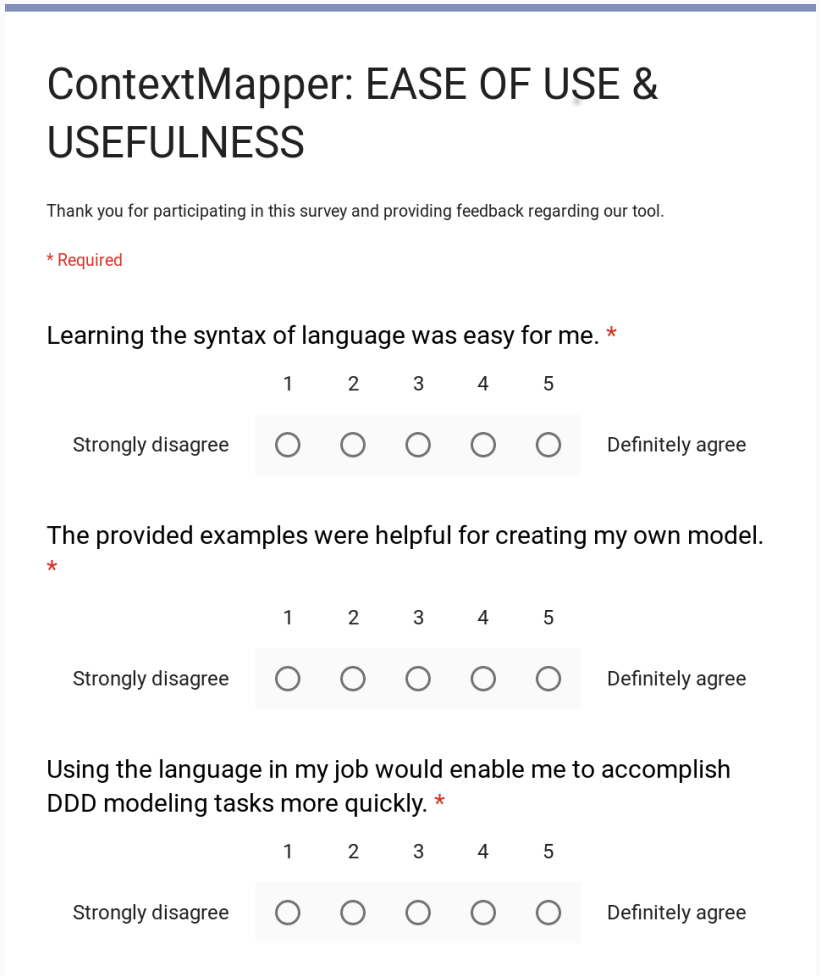
Surveys collect data with **questionnaires** or **interviews**.

- Descriptive: examine characteristics of population
- Explanatory: identify reasons for certain behavior
- Explorative: open survey where RQs are not clear yet (pre-study)

Survey

For example:

- As a pre-study: What do architects think about the idea of formalizing DDD context maps? (RQ3)
  - Investigate expectations of target user group.
- Evaluate user impressions regarding usability and expressiveness of DSL. (RQ1)



ContextMapper: EASE OF USE & USEFULNESS

Thank you for participating in this survey and providing feedback regarding our tool.

* Required

Learning the syntax of language was easy for me. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Definitely agree

The provided examples were helpful for creating my own model. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Definitely agree

Using the language in my job would enable me to accomplish DDD modeling tasks more quickly. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Definitely agree

Case Study

Case studies **“investigate contemporary phenomena in their context”** [4, 7, 9]:

- Observe how a new tool or method behaves in practice.
  - Apply it in an industry project.
- Typically used in cases where it is not possible to isolate the phenomenon from its context.
- Uses the following methods to collect data:
  - Interviews
  - Observations
  - Archival Data

Case Study

For example use the Context Mapper tool in a company which already uses DDD context maps to:

- Apply observation methods using A/B testing and usability tests to **detect unfulfilled requirements**. (RQ1)
- Conduct interviews to **study user satisfaction**. (RQ1)
- Recording user actions to **compare “effort-completion time”**. (RQ2)

Action Research

Action research has **similarities to case studies**, but in this case the researcher **actively participates in the project to improve the tool or process**.

> *"In action research, the emphasis is more on what the practitioners do than what they say they do." [2]*

- Experiment together with the practitioners.
- Use the user feedback to directly improve the tool or process.
- Improve and test with users again.

Relationships with Agile Practices

Relationships with Agile Practices

What is the relationship to (agile) software engineering practices? Are combinations possible?

1. Software engineering approaches are the **major subject** to empirical studies in our field.

2. **Agility** of empirical strategies
   - How ”agile“ are they?
   - Is it possible to apply them and proceed in an ”agile“ manner?

How “agile” are these Empirical Strategies?

Can we use the presented strategies and proceed in an “agile” way?

Important agile practices[2]:

- Ability to respond to change.
- Regular retrospectives.
- Short feedback loops with customers (practitioners).
- Self empowerment of development teams.

[2] https://www.agilealliance.org/

How “agile” are these Empirical Strategies?

Agility of empirical strategies:

- Action research is already very agile by design.
- Case studies a bit less, since the researcher not participates in the project.
- Experiments and surveys do not conform due to their fixed design.

Possible solution to introduce “agility” for all strategies:

- Conduct smaller studies and iterate.
- Use results of one study as input for next one.

For example: (Context Mapper)

- Conduct multiple small case studies and proceed in an iterative way.

Conclusion

Summary & Conclusion

- The **main concepts of empirical software engineering** are:
  - Survey
  - Case Study
  - Experiment
  - Action Research

- The approaches are **applicable to research and thesis projects** to **evaluate software engineering methods and tools**.

- It is possible to **apply the strategies in an agile way**.

Questions?

Discussion

Appendix

Experiment

Experiments provide **more control** than a case study and is typically conducted in a **laboratory environment**.

- In experiments researchers measure the impact of changing one or more variables.
- The rest of the environment must be keept at a fixed level.
  - Therefore only applicable in a laboratory and not in a “real world” environment.

For example:

- Let software architects or engineers create DDD context maps in a laboratory environment.
- Compare *time* and *effort* needed to fulfill specific tasks with the tool and without the tool.
  - Thereby measure if productivity increases as predicted.

References i


IEEE Standard Glossary of Software Engineering Terminology.
IEEE Std 610.12-1990, pages 1–84, Dec 1990.


D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen.
Action research.
Commun. ACM, 42(1):94–97, Jan. 1999.


V. R. Basili.
The experimental paradigm in software engineering.
In H. D. Rombach, V. R. Basili, and R. W. Selby, editors, *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, pages 1–12, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

References ii

I. Benbasat, D. K. Goldstein, and M. Mead.
The case research strategy in studies of information systems.
MIS Q., 11(3):369–386, Sept. 1987.

R. L. Glass.
The software-research crisis.
IEEE Softw., 11(6):42–47, Nov. 1994.

S. Kapferer.
A domain-specific language for service decomposition.
Term Project, University of Applied Sciences of Eastern Switzerland (HSR FHO), 2018.
Publication: `https://eprints.hsr.ch/722/`.

References iii


C. Robson.
Real World Research: A Resource for Social Scientists and Practitioner-Researchers.
Regional Surveys of the World Series. Wiley, 2002.


C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln.
Experimentation in Software Engineering.
Springer Publishing Company, Incorporated, 2012.


R. Yin.
Case Study Research: Design and Methods.
Applied Social Research Methods. SAGE Publications, 2009.