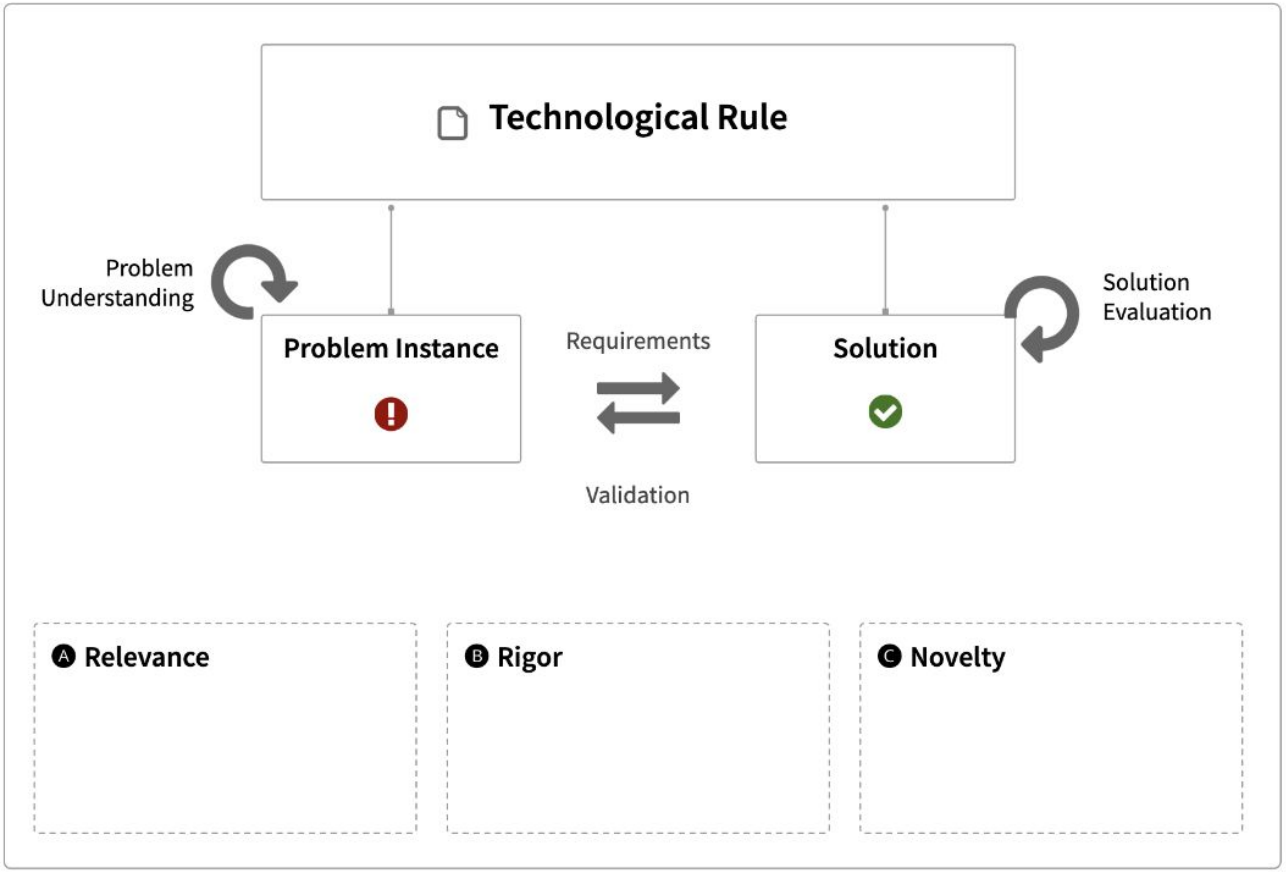
Design Science Revisited (Project #1!)

EMSE-UVic Fall 2020

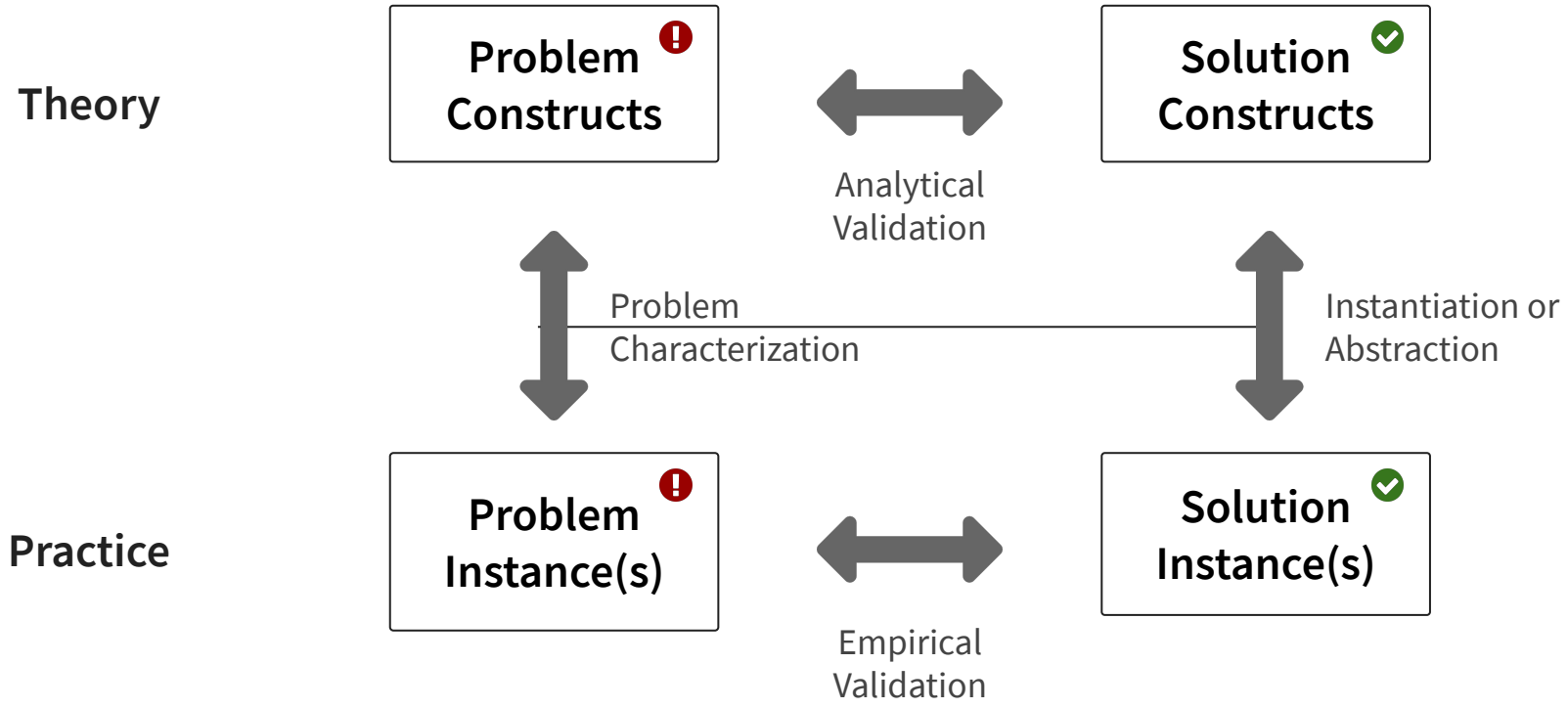
Margaret-Anne Storey

Week 5, Oct 9/2020



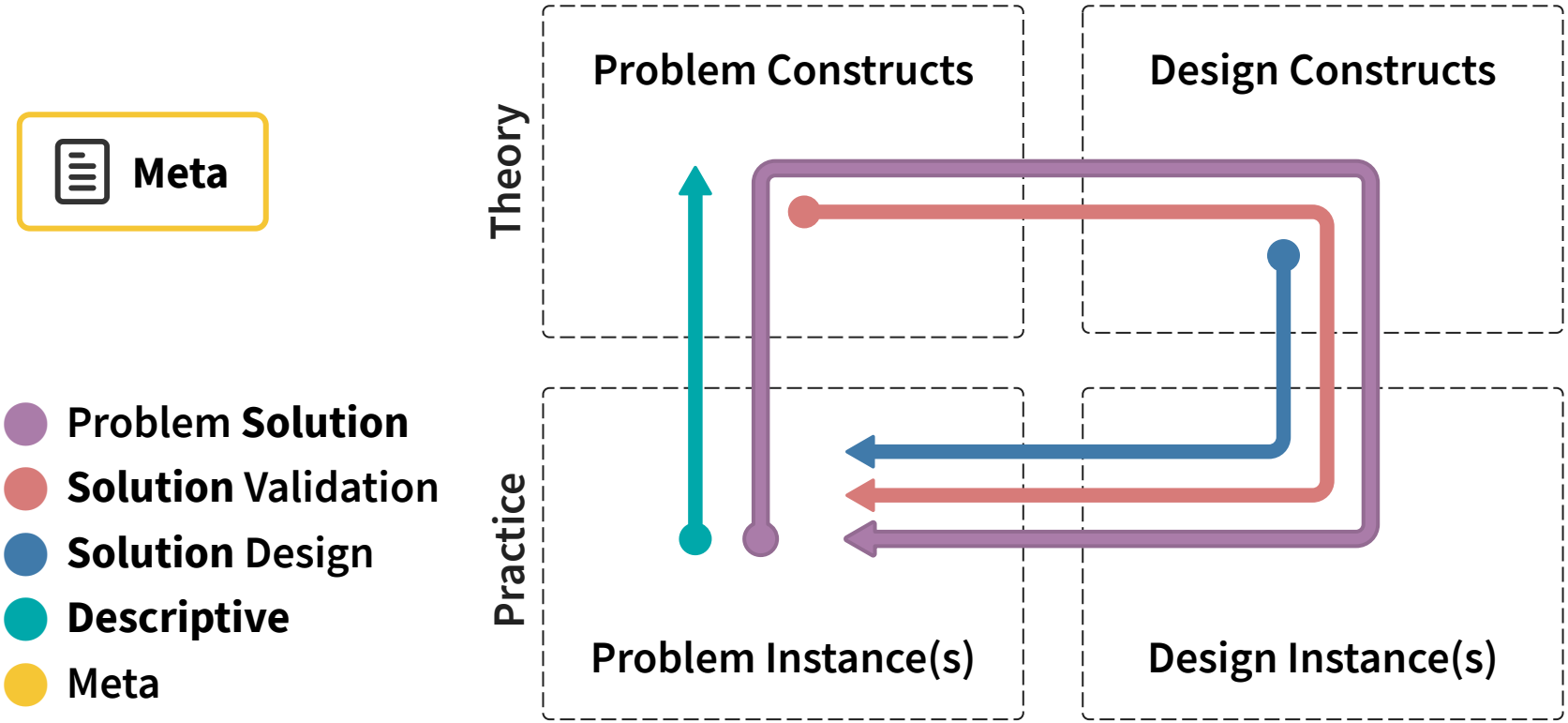
Design Science Visual Abstract Template

Storey et al., 2017



Design Science - One View

Problem Solution

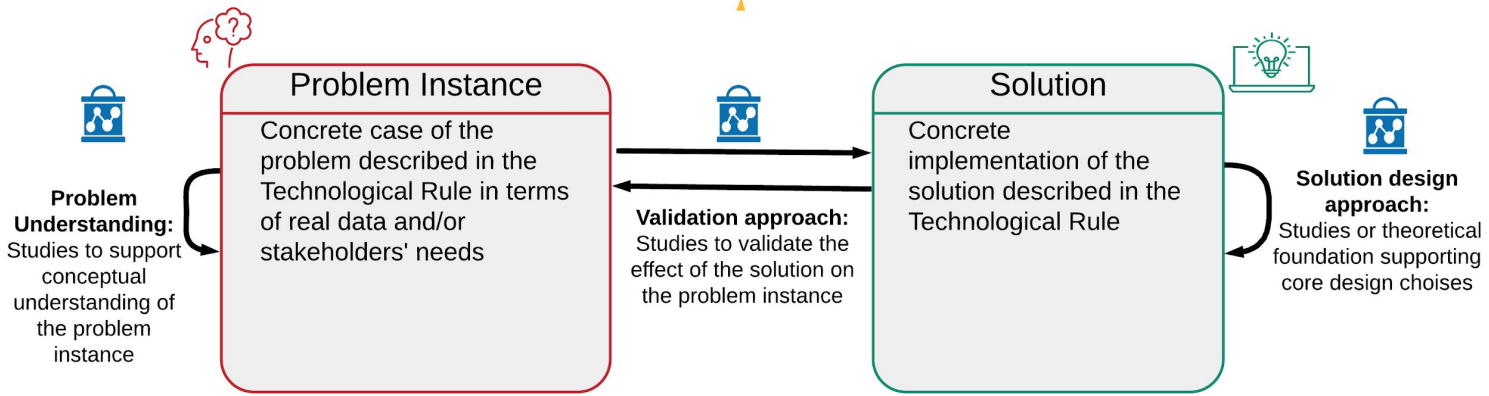


4 | Different kinds of contributions

Engström et al., 2019

Generic Visual Abstract

Technological rule: To achieve effect/change in situation/context apply solution/intervention



Relevance: Characteristics of the context that are likely to impact applicability and potential value of the proposed solution

Rigor: Characteristics of the three knowledge creating activities (problem understanding, solution design and in context evaluation) that adds to the strength of the empirical support of the Technological Rule

Novelty: Positioning of the Technological Rule in terms of previous knowledge

Problem-Solution

Paper ID: Nistor2015

Technological rule: To fix performance bug problems caused by unnecessary loop execution use CARAMEL, which adds a break within loops

Problem Understanding: Literature references to papers on performance bugs and a classification of CondBreak fixes

Problem Instance

Performance bugs are programming errors that slow down program execution. The study aims to study how likely developers are to fix a performance bug.

Validation approach: CARAMEL was evaluated on 11 JAVA and 4 C/C++ applications through two implementations respectively CARAMEL-J and CARAMEL-C

Solution

CARAMEL, a novel static technique that detects and fixes performance bugs that have non-intrusive fixes likely to be adopted by developers. Typically, these bugs are fixed by adding one line of code inside the loop, i.e., if (cond) break, called a CondBreak fix.

Solution design approach: A set of RI (result instructions) have been classified into 6 types. They have been used to implement the CARAMEL algorithm.

Relevance: Programming errors due to performance bugs. Relevant for programmers for which performance is an issue

Rigor: The validation pointed out that CARAMEL, used on 11 JAVA applications, and 4 C/C++ applications identified 61 performance bugs in the JAVA applications and 89 in C/C++ ones. Of these, 10+24 were not found before.

Novelty: (i) detect performance bugs whose fixes clearly offer more benefits than drawbacks to developers; (ii) identify a family of performance bugs; (iii) CARAMEL as tecnique for detecting performance bugs that have CondBreak fixes;

Paper Title: CARAMEL: Detecting and Fixing Performance Problems That Have Non-Intrusive Fixes

Solution Validation

Paper ID: Loncaric2018

Technological rule: To synthesize data structures that track subsets and aggregations of multiple related collections alternate steps of query synthesis and incrementalization

Problem Understanding: Observing that others tried to solve the problem

Problem Instance

Four example cases are described, implicitly pointing to a problem

Validation approach: Cozy's effect on development time, correctness, and efficiency was evaluated for the four real cases

Solution

A tool Cozy, was implemented that can synthesize implementations for complex multi-collection data structures from high-level specifications. It iteratively improves the data structure specification using two cooperating components: a query synthesizer that selects a better representation and implementation for each query method and an incrementalization step that ensures the new representation is kept up-to-date when an update method is called.

Solution design approach: Not reported - focus on the final solution

Relevance: Data structure problems, especially in domains like user interfaces or web services where software must manage some internal state and also handle asynchronous events.

Rigor: Proof of concept demonstrated in four real cases

Novelty: It is a new technique for data structure synthesis that overcomes many of the limitations of previous work

Paper Title: Generalized Data Structure Synthesis

Solution Design

Paper ID: Avgerino2014

Technological rule: To increase efficiency and effectiveness when using symbolic execution alternate between static and dynamic symbolic execution

Problem Understanding: The problem is well known and reported in several studies.

Problem Instance

How to increase bug detection in created benchmarks (using binaries from Debian Wheezy and Sid)

Validation approach: Use of benchmarks to compare with previous work, Applied MergePoint with veritesting to 33,248 binaries

Solution

Implement MergePoint with veritesting. Veritesting alternates between SSE and DSE. The alternation mitigates the difficulty of solving formulas, while alleviating the high overhead associated with a path-based DSE approach.

Solution design approach: It builds on and improves previous work (MergePoint) wrt the path explosion problem

Relevance: Lack of efficiency when using symbolic execution

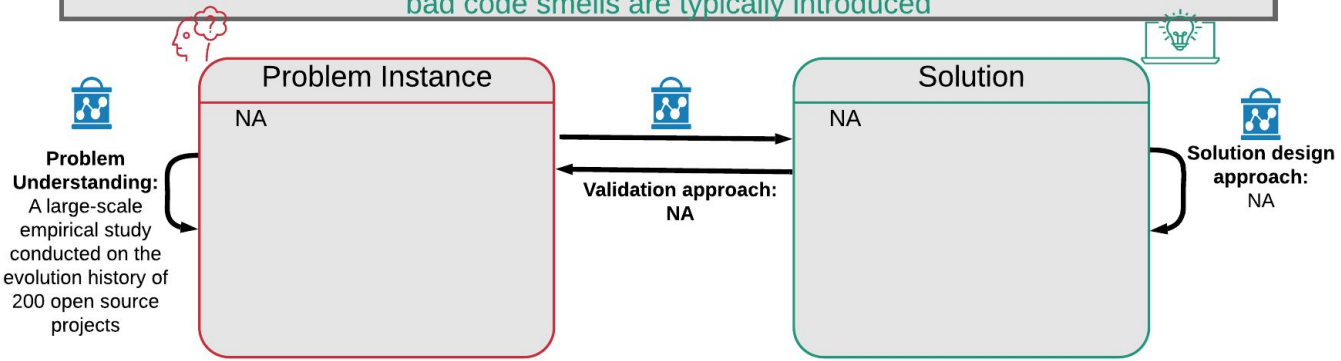
Rigor: Large-scale experiment on 33,248 programs from Debian Linux. MergePoint generated billions of SMT queries, hundreds of millions of test cases, millions of crashes, and found 11,687 distinct bugs

Novelty: Tested an order of magnitude more applications than have been tested by prior symbolic execution research. We analyzed each application for less than 15 minutes per experiment. We improve open source software by finding over 10,000 bugs and generating millions of test cases.

Paper Title: Enhancing Symbolic Execution with Veritesting

Paper ID: Tufano2015

Technological rule: To better plan activities for improving design and source code quality when developing software utilise the knowledge gained in this study about when and why bad code smells are typically introduced



Relevance: The study context is the change history of 200 projects belonging to three software ecosystems, namely Android, Apache, and Eclipse

Rigor: Validity is ensured because a large set of 200 projects concerning the analysis of code smells and oftheir evolution has been investigated. Projects were extracted from three ecosystems: android, apache and eclips.

Novelty: First comprehensive empirical investigation into when and why code smells are introduced in software projects.

Paper Title: When and Why Your Code Starts to Smell Bad

Meta

Paper ID: Floyd 2017

Technological rule: To understand how human brain processes software engineering tasks when making subjective human judgments use medical imaging techniques (fMRI - functional magnetic resonance imaging)

Problem Understanding: Controlled experiment monitored with fMRI technique to gain understanding.

Problem Instance

Code comprehension, code review and review using functional magnetic resonance imaging

Validation approach: N/A

Solution

Use medical imaging techniques to understand code comprehension and code review and the relationship to natural language and expertise.

Solution design approach: N/A

Relevance: Code review and comprehension vs prose review activities

Rigor: In a controlled experiment involving 29 participants, authors examine code comprehension, code review and prose review using functional magnetic resonance imaging.

Novelty: Measure brain activity for carrying out different tasks

Paper Title: Decoding the representation of code in the brain: An fMRI study of code review and expertise

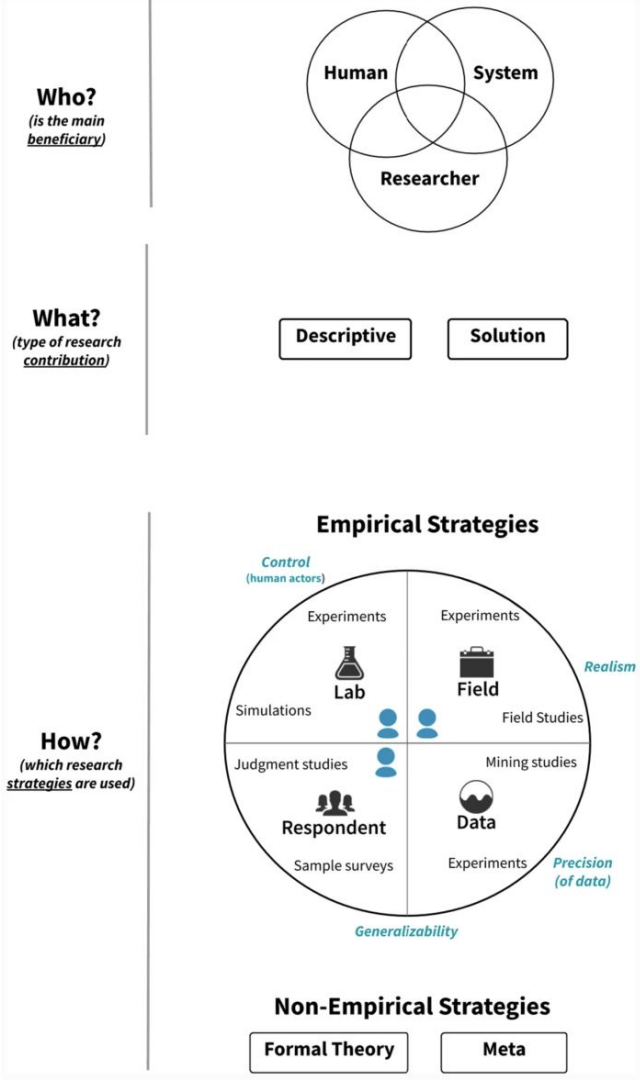
Who What How Framework (Preparation for Project #1!)

EMSE-UVic Fall 2020

Margaret-Anne Storey

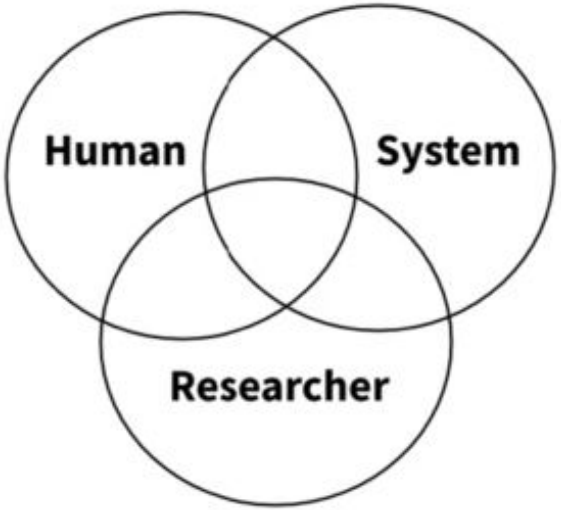
Week 5, Oct 9/2020

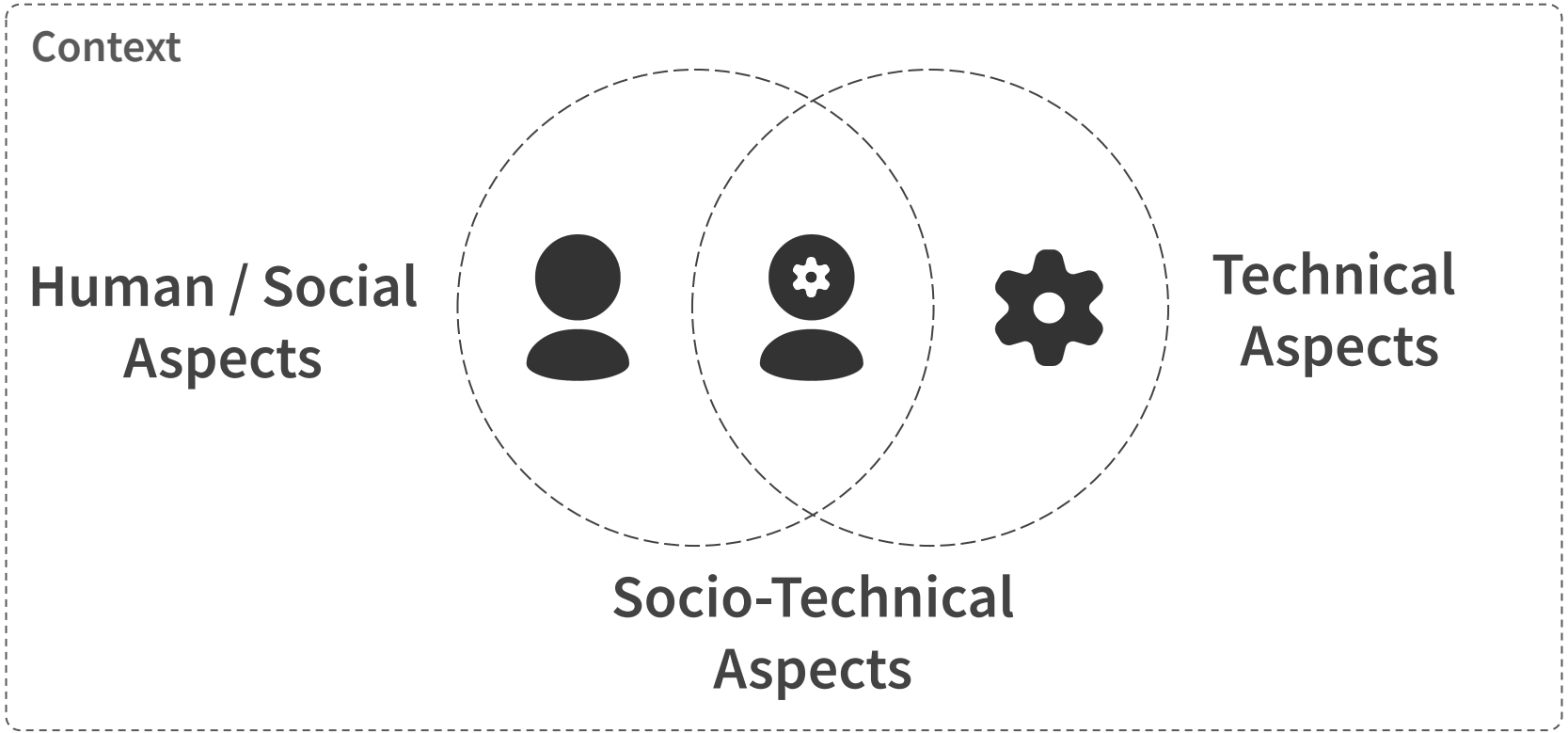
Who
What
How
Framework



Who: is the reported *beneficiary* of the research

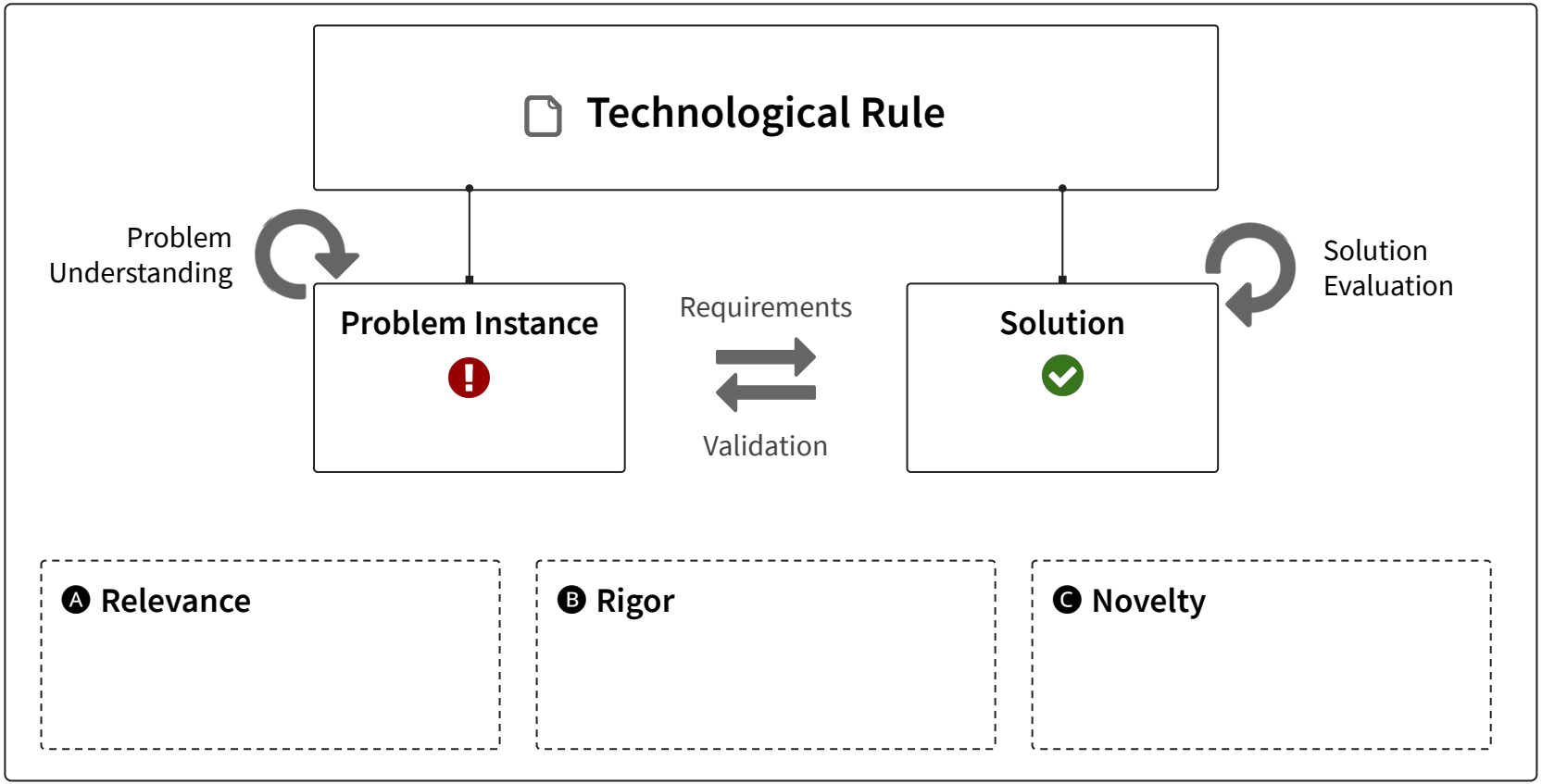
Who?
(is the main beneficiary)





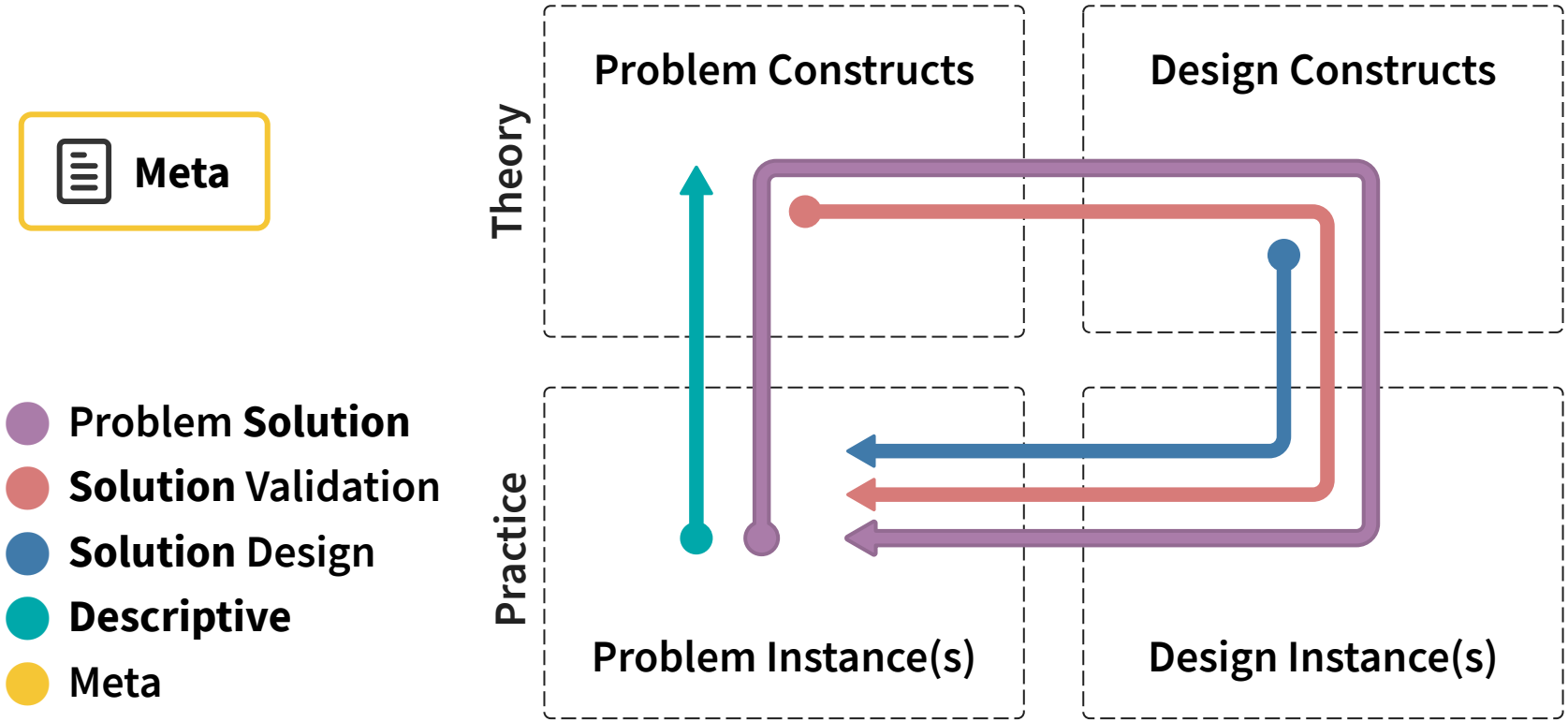
Software Engineering **Design** Space

What: type of research *contribution*



16 | Design Science Visual Abstract Template

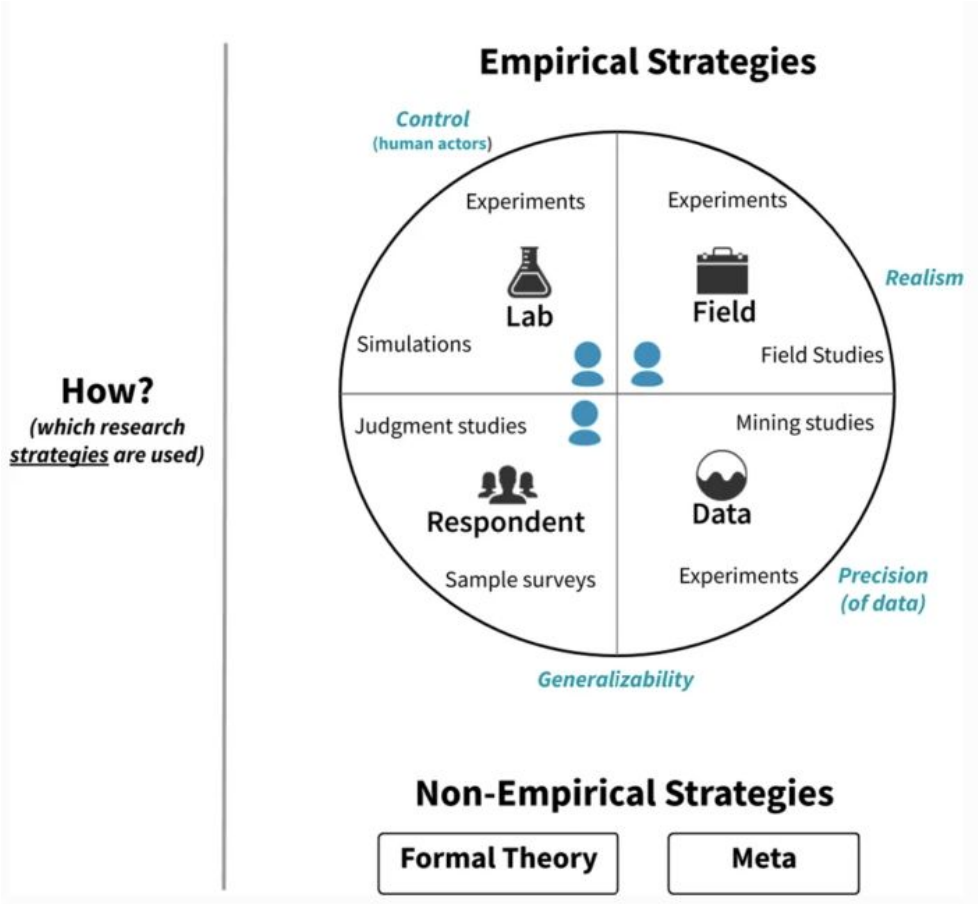
Storey et al., 2017



Different kinds of contributions

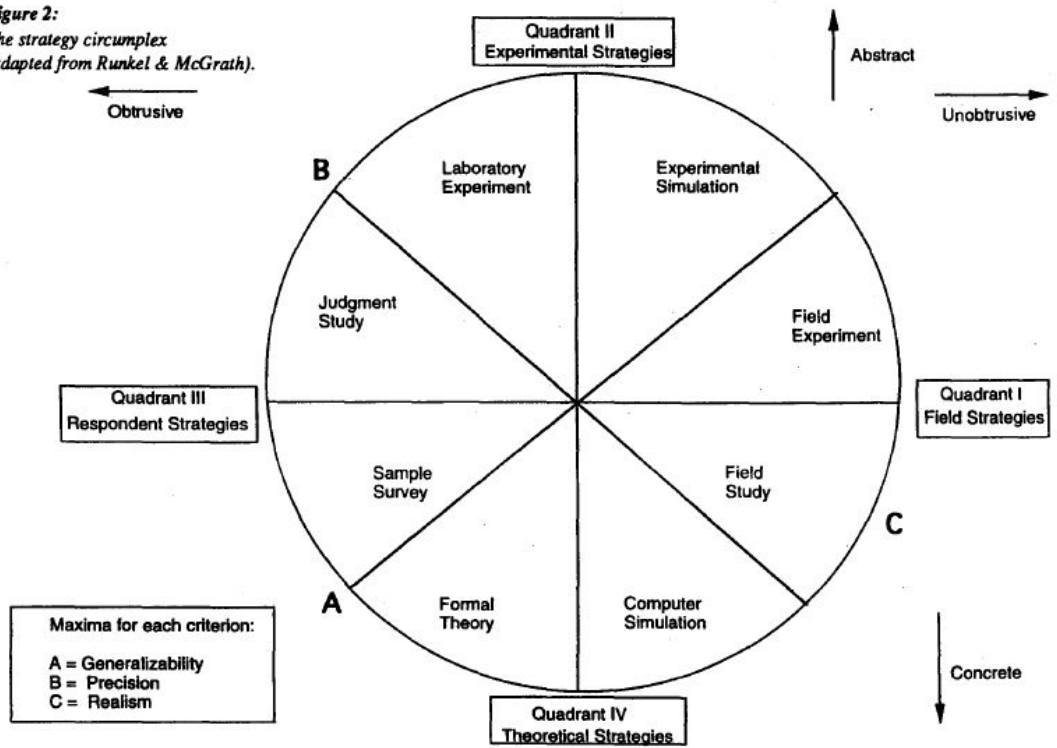
Engström et al., 2019

What: type of research *contribution*



Circumflex in the paper!
(to avoid confusion listen up!)

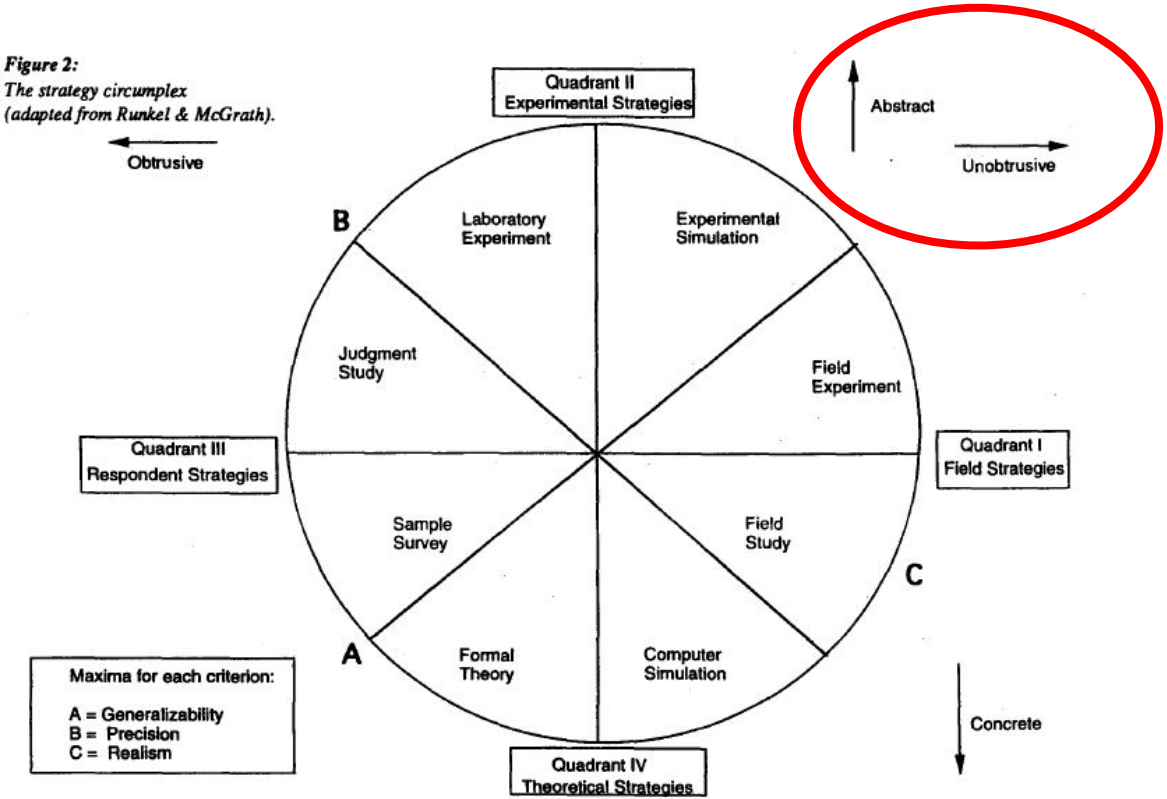
Figure 2:
The strategy circumplex
(adapted from Runkel & McGrath).

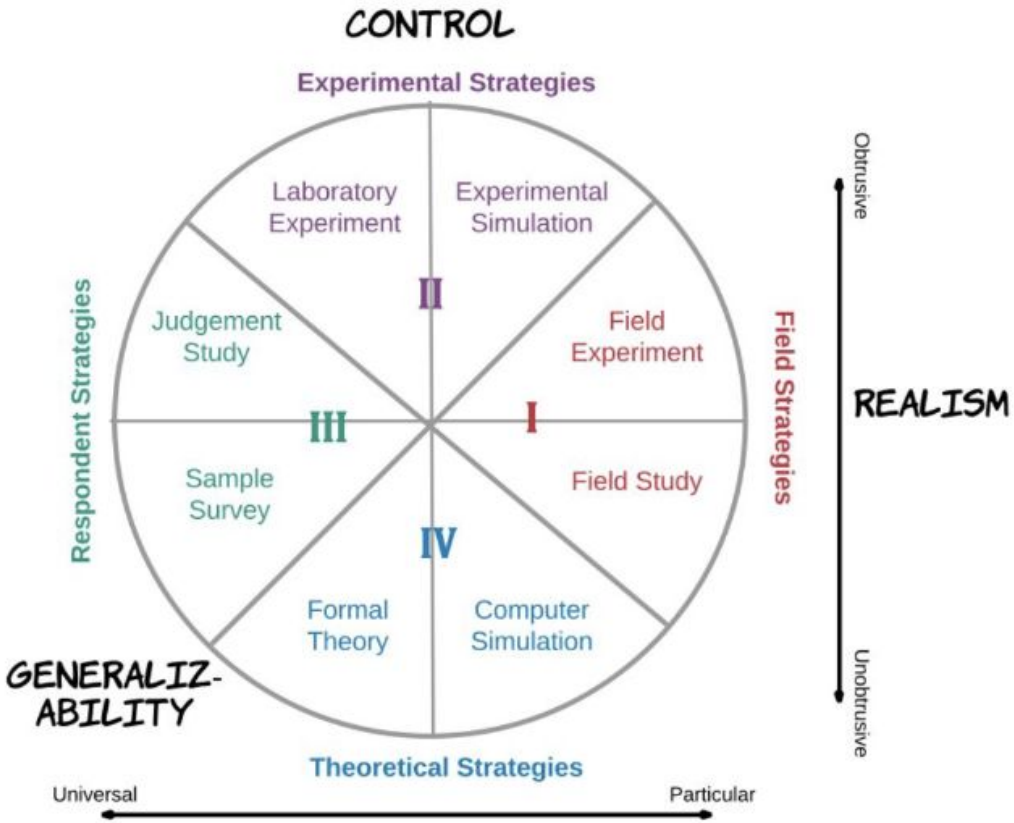


Circumflex in the paper!
(to avoid confusion listen up!)

These axes were switched!

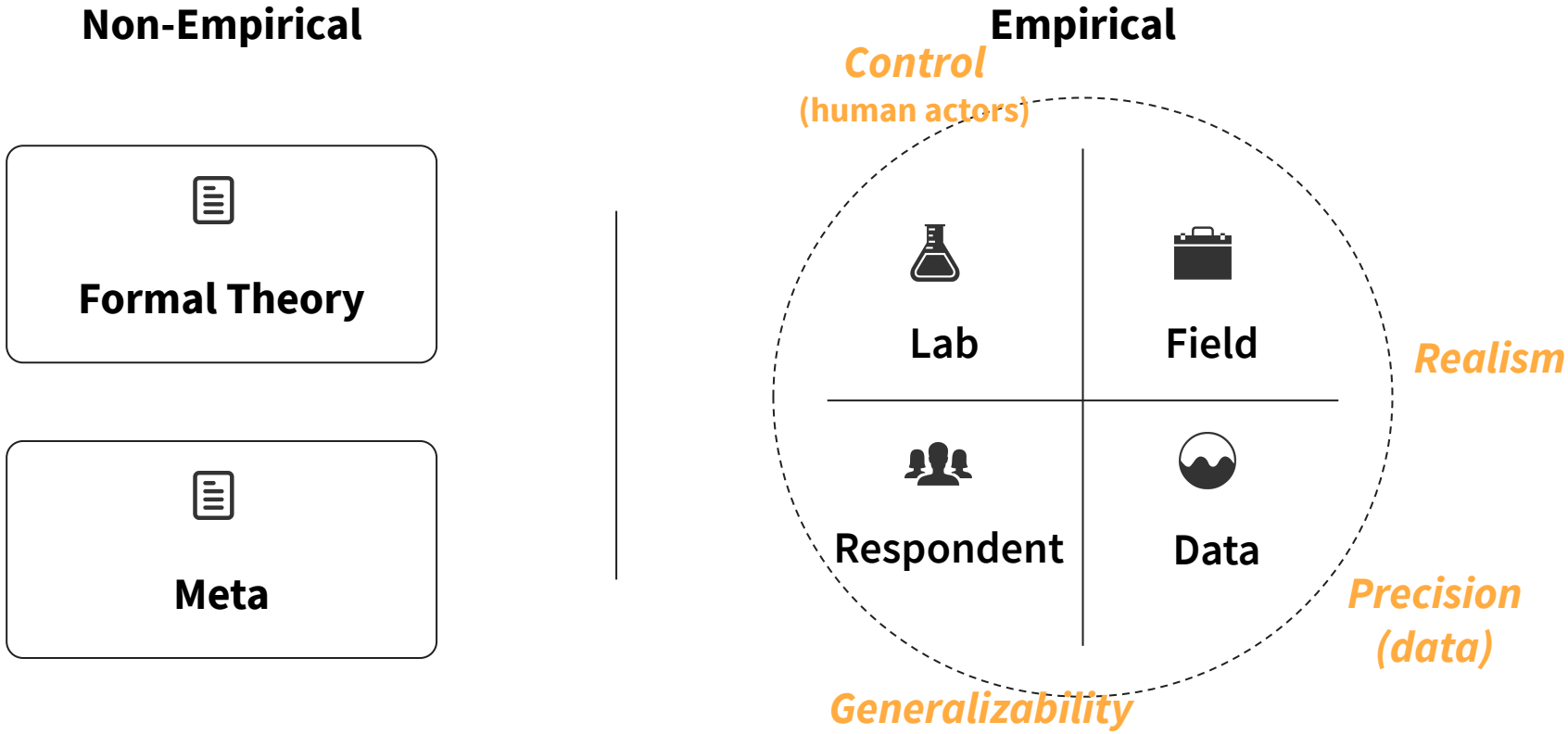
Figure 2:
The strategy circumplex
(adapted from Runkel & McGrath).





Circumflex

Joseph. E. McGrath.
Methodology matters: Doing research in the behavioral and social sciences. 1972



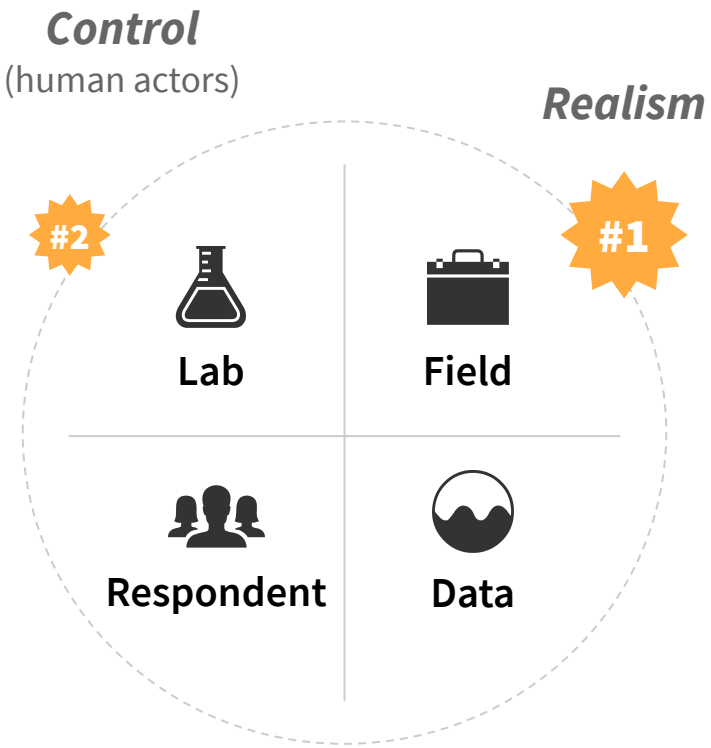
Socio-Technical Research Framework

Extending: Runkel & McGrath:
Research on Human Behavior: A Systematic Guide to Method, 1972

Triangulation: “The basic idea underpinning the concept of triangulation is that the phenomena under study can be understood best when approached with a variety or a combination of research methods. Triangulation is most commonly used in data collection and analysis techniques, but it also applies to sources of data. It can also be a rationale for multiple investigators in team research.”

http://methods.sagepub.com/reference/sage-encyc-qualitative-research-methods/n468.xml

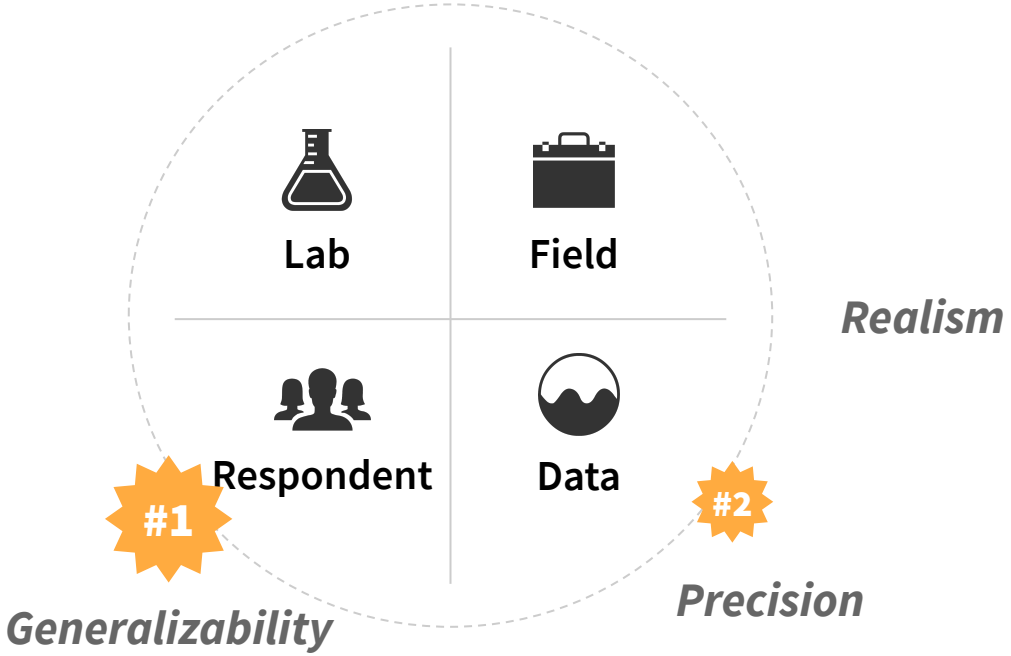
CloudBuild



The Methods We Chose

Lebeuf, Voyloshnikova, Herzig & Storey:
"Debugging, and Optimizing Distributed Software Builds: A Design Study", ICMSE 2018

GitHub



The Methods We Chose

Gousios, Storey & Bacchelli,
"Work Practices and Challenges in Pull-Based Development: The Contributor's Perspective", ICSE 2016

Sequential explanatory strategy: e.g., quantitative analysis of trace data followed by qualitative analysis of interview data (latter helps explain the former)

Sequential exploratory strategy: e.g., analysis of qualitative data from surveys followed by analysis of quantitative trace data (for testing emerging theory, explain early exploratory findings)

Concurrent triangulation strategy: different methods used concurrently, improve validity

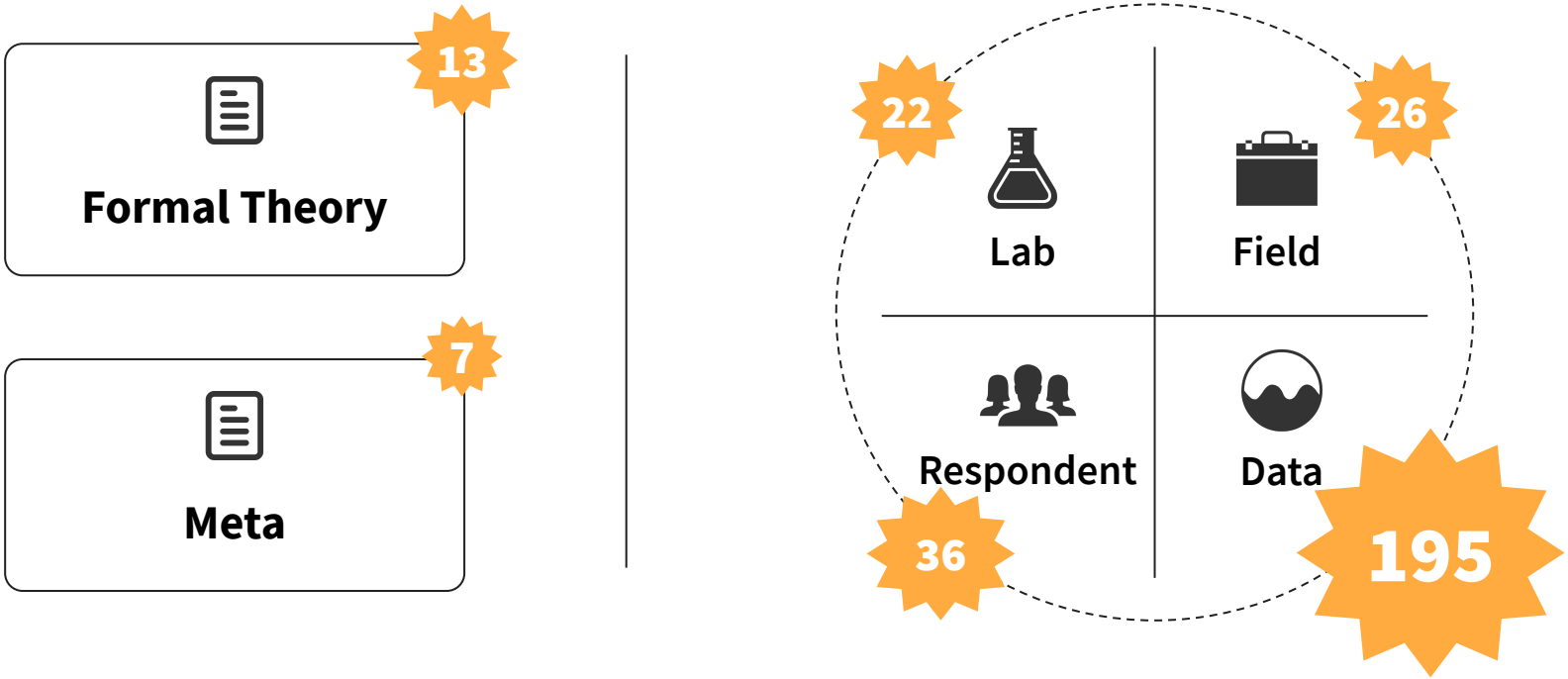
 | Mixed method designs (see Creswell)

253

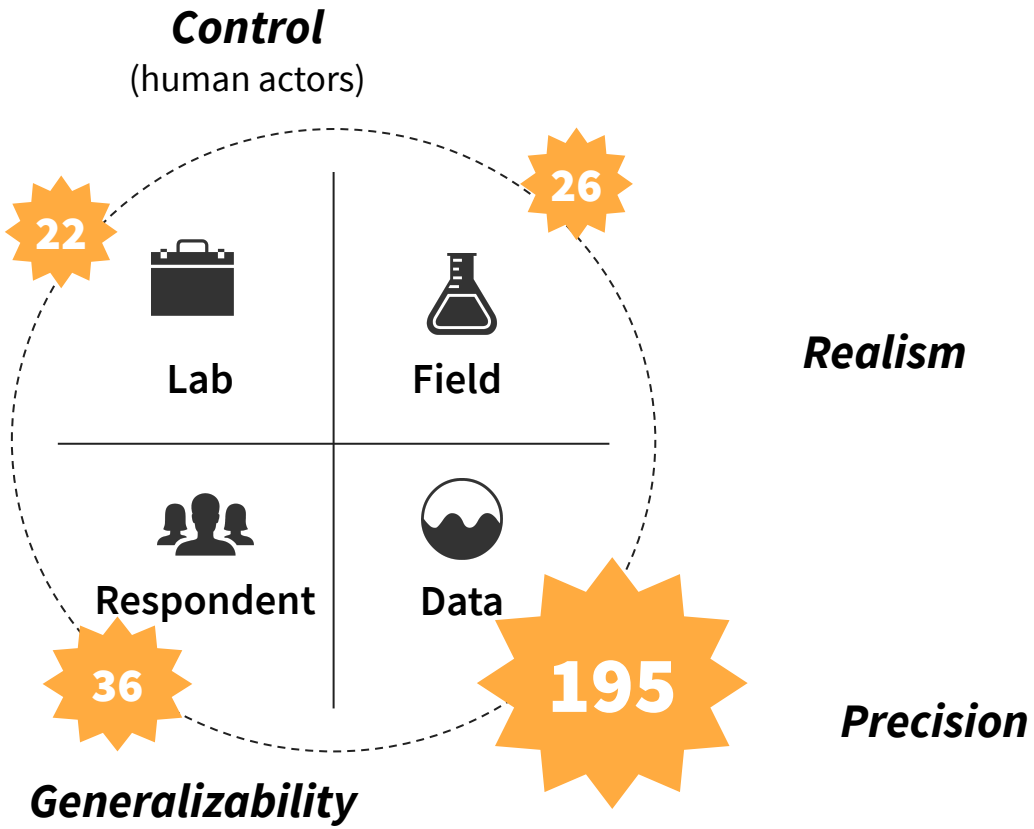
Technical Track Papers
2015 to 2017

Categorizing ICSE Paper Research Methods

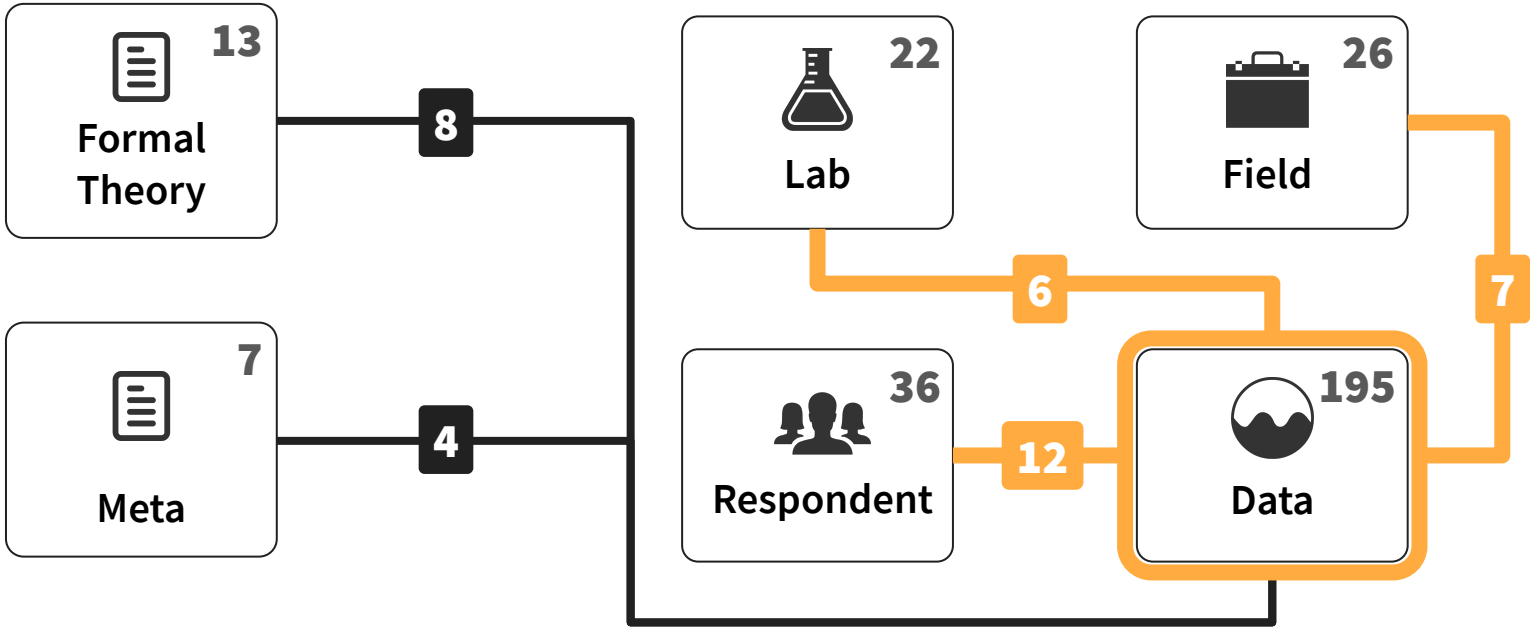
Williams et al., 2019



Categorizing ICSE Paper Research Methods

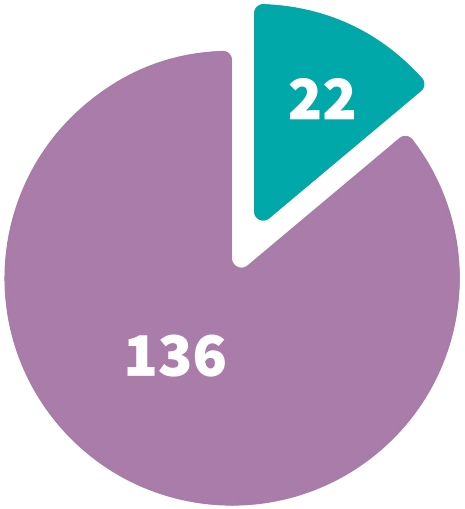


Categorizing Research Methods



37 Data Papers Used Triangulation

Data Only Papers



Contribution Type

Data Only Papers

Of these...

110

70%

of 158

Authors Mention

Developers

(But they don't study them!)

According to the Authors

What has changed since 2017 to 2020?