2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement

Using a Visual Abstract as a Lens for Communicating and Promoting Design Science Research in Software Engineering

Margaret-Anne Storey
University of Victoria
BC, Canada
mstorey@uvic.ca

Emelie Engström
Lund University
Sweden
emelie.engstrom@cs.lth.se

Martin Höst
Lund University
Sweden
martin.host@cs.lth.se

Per Runeson
Lund University
Sweden
per.runeson@cs.lth.se

Elizabeth Bjarnason
Lund University
Sweden
elizabeth@cs.lth.se

***Abstract*—Empirical software engineering research aims to generate prescriptive knowledge that can help software engineers improve their work and overcome their challenges, but deriving these insights from real-world problems can be challenging. In this paper, we promote design science as an effective way to produce and communicate prescriptive knowledge. We propose using a visual abstract template to communicate design science contributions and highlight the main problem/solution constructs of this area of research, as well as to present the validity aspects of design knowledge. Our conceptualization of design science is derived from existing literature and we illustrate its use by applying the visual abstract to an example use case. This is work in progress and further evaluation by practitioners and researchers will be forthcoming.**

1. Introduction

It's often a challenge to articulate the contributions that come out of software engineering research and highlight their value to practitioners. This is partially due to the interdisciplinary nature of software engineering research which can build on and contribute to many fields, including mathematics, engineering, design, and the social sciences. These disciplines also rely on very different research paradigms and often hold contradictory views on what is considered valid and reliable research. Furthermore, as software engineering is a professional field, practitioners and other researchers often question the relevance of the problems investigated and the value of the proposed solutions.

Similar challenges are experienced in other research disciplines, such as information systems and management science. *Design science research* has been widely adopted in these and other engineering disciplines as a way to frame research by highlighting both the problems addressed and the interventions proposed [1], [2], [3]. Design science supports researchers in conveying how they build on and contribute to an existing knowledge base.

Despite the applied and interdisciplinary nature of software engineering, design science has never gained traction despite attempts to popularize it within the community [4], [5]. This is unfortunate as many research contributions are not accepted due to unclear benefits or disagreements about what constitutes a valuable "software engineering research contribution". Excellent research contributions also go unnoticed by practitioners that could benefit from those insights.

Many software engineering research papers do not explicitly describe the class of problems addressed, nor do they sufficiently describe the relevance to practitioners. They tend to focus on problem solving instances but do not sufficiently reason about how the knowledge can be generalized [6]. We feel that design science should be promoted as it is a powerful way to frame prescriptive software engineering research—it prompts researchers to uncover the relevance of the problems being addressed, as well as clearly expose how the solutions help.

To promote design science and communicate the benefits it brings to empirical software engineering, we present a *visual abstract* template that describes key design science concepts. This abstract is a work in progress that was formed following a year-long discussion among the authors on how design science can be used to frame our own (and other) software engineering research. We feel the visual abstract has two benefits. First, it can be used to capture the scope, process, takeaways, and value of a study that produces prescriptive knowledge on how to solve a given problem. That is, the abstract forms a lens for describing design science research. Second, based on our research and industry colleagues' preliminary reactions to the abstract, we believe the visual abstract may lead to more efficient and accurate dissemination of research findings with industry as well as between researchers.

In Section 2, we present our view of design science, aggregating and building on but rejecting some concepts from other conceptualizations of this paradigm. This sets the stage for Section 3, where we present a visual template for design science research and provide guidelines for applying the template to software engineering. In Section 4, we demonstrate the visual abstract by applying it to a previously published paper. In Section 5, we discuss how we arrived at this particular visual abstract for design science and some limitations we faced using the template and its associated guidelines. We conclude the paper in Section 6.

978-1-5090-4039-1/17 $31.00 © 2017 IEEE
DOI 10.1109/ESEM.2017.28

2. A Conceptualization of Design Science

Most software engineering research aims to produce knowledge that helps software engineering professionals solve their problems—prescriptive knowledge that provides advice on how to act in various situations. Such knowledge is, by necessity, context dependent and relative to a defined setting [4], and shares characteristics with what other disciplines refer to as design knowledge [2]. Although the meaning of design knowledge varies across fields and among design science researchers, they all share the view that knowledge is holistic, heuristic, and justified by in-context evaluation. We build on this understanding and select concepts and terminology from several sources on design science.

The term holistic is used by van Aken [2] and refers to the "magic" aspect of design knowledge where we never fully understand why a certain solution works in a specific context. Various context factors impact the effect of a solution, and some of them may be better understood than others. However, there will always be hidden context factors that affect a problem-solution pair [7]. Similarly, we can never prove the effect of a heuristic prescription conclusively without in-context evaluations. By evaluating multiple problem-solution pairs matching a given prescription, our understanding about that prescription increases.

Such holistic and heuristic prescriptions in design science research are expressed in terms of *technological rules* [2], which are rules that capture general knowledge about the *mappings* between *problems* and proposed *solutions*. Depending on the abstraction level of the technological rule, the solution may refer to the use of a specific artifact or the application of a more general approach. In both cases, the design knowledge within the technological rule aims to help professionals design customized solutions to their own software engineering problems—it frames the research product in terms of desired effects and interventions rather than in terms of a solution to a problem. The term *technological rule* originates from Bunge [8] and corresponds to "methods" in Gregor and Hevner's work [9].

While adhering to the design science paradigm puts the focus on how to *produce* and *assess* design knowledge (i.e., technological rules), our visual abstract template is designed to help researchers effectively *communicate* as well as justify design knowledge: e.g., which instantiations of the rule have been studied and how were they evaluated, how was problem understanding achieved, and what are the foundations for the proposed solution?

In line with van Aken [2], our visual abstract template emphasizes technological rules as the main takeaway of design science within software engineering research. A technological rule can be expressed in the form: *to achieve <Effect > in <Situation > apply <Intervention>*. Here, a class of software engineering problems is generalized to a stakeholder's desired effect of applying a potential intervention in a specified situation. Making this problem generalization explicit helps the researcher identify and communicate

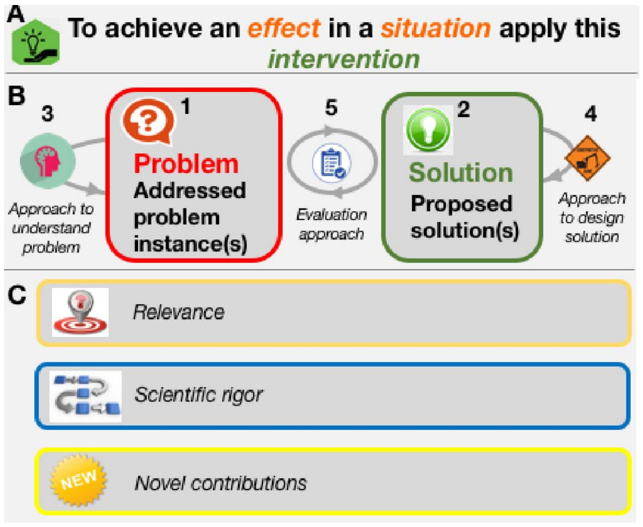
A
To achieve an effect in a situation apply this intervention
B
3
Approach to understand problem
1
Problem
Addressed problem instance(s)
5
Evaluation approach
2
Solution
Proposed solution(s)
4
Approach to design solution
C
Relevance
Scientific rigor
NEW
Novel contributions

Figure 1. A visual abstract template for design science research. The researcher should fill this in using the guidelines provided. Part A denotes the technological rule. Part B shows the problems, proposed solutions, and the empirical and research steps followed. Part C shows three design knowledge assessment criteria.

the different value-creating aspects of a research study or program.

A design science research study creates new knowledge by investigating one or more instances of a solution in context [4] through action research (i.e., within one context) [10], through multiple case studies (i.e., across contexts) [2], or through design cycles [9]. Each instantiation (i.e., a case study or an iteration of action research) constitutes a unit of analysis in relation to that rule and the new knowledge is communicated and assessed from that perspective. The technological rule may be either a new proposal or an existing one that is refined or validated by the current study.

3. A Visual Abstract for Design Science

Our proposed **visual abstract** template (see Fig. 1) relies on the concepts described in Section 2. It captures three main aspects of design science contributions: A) the theory proposed or refined in terms of a technological rule; B) the empirical contribution of the study in terms of one or more instances of a problem-solution pair and the corresponding design and evaluation cycles; and C) the assessment of the value of the produced knowledge in terms of relevance, rigor, and novelty.

The template provides guidelines for how to create a visual abstract of a particular instance of design science research in software engineering. As described in Section 2, each iteration or case constitutes an instantiation of a problem-solution pair and is the unit of analysis in design science research. Our visual abstract template highlights the need for a careful description of these units of study. In the visual abstract template, the researcher is encouraged to reflect on how a study adds new knowledge to the general technological rule and, to be aware of the relationship between the general rule and its instantiation, articulating both.

The italicized text in the template should be replaced with text pertinent to the research being communicated through the template. Later, we show the visual abstract template in action with a specific research example.

3.1. The Technological Rule

The technological rule is the crucial part of the visual abstract because it captures the main theoretical contribution from the study in a single logical phrase (see Fig. 1, part A). It has three key constructs: the desired **effect** in a given **situation** along with a proposed **intervention** to achieve that effect.

Taking sufficient care to formulate a technological rule forces the researcher to reflect on the knowledge contribution, which in turn helps to articulate the problem they address in their research. By crisply describing a potential generalization of the problem-solution pair under investigation, they take a step back from the current case to look at the research from an envisioned recipient's point of view, which helps to extract information relevant to them.

3.2. Research Scope and Process

The main body of the abstract focuses on the scope and process of a study or research program and is composed of five constructs (refer to the numbered elements in Fig. 1, part B).

(1) The **problem instance** box captures the instances of the problem under study and how the scope of the problem is considered in the research. The description in this box should be an instantiation of the general problem as captured by the technological rule. The researcher should check that the technological rule and the problem instance are consistent with each other.

(2) The proposed **solution(s)** should be described in terms of how the general intervention was applied to address the problem. However, it should only be briefly described in the solution box as the reader can refer to the paper for more details. Which details to include here will depend on the intended audience: a researcher, practitioner, or manager will have different information needs.

(3) The **problem understanding approach** supports the reader in assessing the relevance of the rule in relation to their own context. It may be that problem understanding is based on existing literature or real-world knowledge, or it may be that new descriptive knowledge arises through empirical means to understand more about the nature of the problem instances addressed. Note that insights may also come once the solutions are evaluated (item 5).

(4) The **design approach** provides an opportunity for the researcher to describe which empirical steps they followed during the design of their solution, before they evaluate their solution using the problem instance.

(5) The **evaluation approach** describes how a researcher applies their solution(s) to the problem instances to evaluate the solution but also to bring further insights on the problem. The knowledge that is gathered from this evaluation (and other evaluations of other problem solving instances) is essential to validate the technological rule. We encourage the researcher to include the main elements of the evaluation steps that were followed or note if the evaluation needs to occur at a later date.

Although we present the problem to the left of the proposed solution, the researcher may not initiate their research study with a particular problem in mind—they may instead start their study with a tool or technique and then search for a problem to solve with their hypothesized solution.

3.3. Design Knowledge Assessment

Discerning researchers and practitioners will have many questions concerning the design knowledge captured by the technological rule. They will want to know if the class of problems addressed has real-world relevance, whether the problem understanding or solution design and evaluation approaches can be trusted, or whether the design knowledge reported is sufficiently novel and/or mature to warrant attention.

Our visual abstract template provides a way to assess the design knowledge produced. This helps the industry practitioner or peer researcher effectively assess the value of a technological rule in relation to their own situation. Drawing from design science literature, we select three criteria for research evaluation: *relevance, rigor*, and *novelty*. However, although these criteria are commonly accepted in most research communities, the interpretation varies depending on the research paradigm and has its own denotation in the context of design science research.

We refer the reader to Part C of the visual abstract template shown in Fig. 1. Deciding what text to place in these three boxes can be challenging and may lead the researcher to consider uncomfortable questions about their research, while at the same time providing them a way to showcase the relevance, rigor, and novelty of their research. We provide more guidance on these three boxes next.

3.3.1. Relevance of the technological rule. For this part of the abstract, the researcher should discuss relevance in terms of the class of problems and solutions captured by the technological rule, and how those problem-solution pairs are relevant to a broader set of professional software engineers. The relevance box helps a researcher convince their *target stakeholders* that the problem being addressed is relevant for them, and that the proposed interventions are actionable in their context. Arguing relevance at the general level can be harder to do than arguing relevance for the stakeholders involved in a specific study.

3.3.2. Scientific rigor. When it comes to filling out this part, rigor can be considered at the level of the technological rule as well as at the level of the problem solving instance.

This may not seem natural to many software engineering researchers as it is more usual to focus on rigor for specific evaluation steps. For this more general perspective, rigor of the technological rule may be discussed in terms of maturity or saturation of the rule [2], [9] (i.e., a summary of both current and previous empirical contributions related to the general rule). The researcher filling out the template may consider and share what other problem instances have been considered and whether other solution alternatives were considered. They may also consider if side effects of applying the prescription were considered, and if not, highlight future work opportunities.

3.3.3. Novel contributions. Research novelty may be assessed with respect to the novelty of the problem insights provided, the solution suggested, or how the mapping of an existing solution to a known problem can bring value. We think this latter point is particularly important because many researchers assert that when both the solution and problem are well known, applying the solution to the problem is merely "routine design". We disagree with this and stress that if the solution is shown to address the given problem instance(s), this knowledge can be used to build, reinforce or to refine an existing technological rule. Furthermore, novel contributions can arise in terms of a refinement or further validation of an existing technological rule (as we will discuss later in Section 4). In summary, novelty should be assessed in terms of the entire technological rule rather than how novel the problems or solutions are in isolation.

Next we show these guidelines in action by using the template to communicate the research in a previously published paper. We discuss the decisions made and trade-offs faced while filling out this template.

4. The Design Science Visual Abstract in Action

During the iterative design of our visual abstract template, we applied different variants of the template to 11 existing software engineering papers (we were involved in four of the studies and had no involvement in the others). We share the application of the template for one of these studies in this short paper (see Fig. 2): we chose an example where we could share insights as an author of the work (Runeson), as well as share feedback from others (Runeson's co-authors) that have experience in research and industry.

Our chosen paper addresses a **problem** identified by several observational studies: manually assigning bugs to teams is labor intensive and error prone [11]. The **solution** applies established machine learning (ML) techniques (ensemble learners, stack generalization) to assign bugs to the right team. The solution was **evaluated** with large datasets consisting of real data from two industry domains.

The insights gained from applying the solution to these two **problem instances** are generalized to the **technological rule** presented in Fig.2.

After applying the template, the resulting abstract shows that our example paper addresses the **relevant** challenge of bug assignment in industrial projects such as Eclipse [13]. It builds on existing research that proposes that machine learning is a useful approach for bug assignment [13]. The visual abstract reveals that the researchers' problem understanding activities and evaluation were conducted in large-scale industrial contexts, contributing to the **scientific rigor** of the work, which in turn adds credibility to the technological rule showcased by applying the template. The abstract also reveals that the application of automated bug assignment techniques on a particular scale and at a particular level of precision is the key **novel contribution** of the work.

Runeson applied the template to this work and then asked his co-authors, both from industry and academia, to assess the resulting visual abstract: they noted that the cost/benefit aspect of the research was not sufficiently clear and they questioned the level of detail used to present the solution in the abstract. From an industry perspective, the co-authors noted that the key novelty of the research came from showing that the precision of automated assignment is on par with the manual process. However, from an academic perspective, the co-authors mentioned that the reviews they received from their manuscript submission indicated that the classifiers and their performance were considered more significant. We see the industry goal is captured more clearly by the overall technological rule, while more detailed contributions that researchers may care about are presented as a solution. We discuss the points raised by this example and other insights next.

5. Discussion

Our proposed visual abstract for design science culminates from a year-long study of how design science has been used in software engineering, while also trying to understand why it has not been adopted. During this time, we (the authors) met regularly to discuss different papers on design science and to arrive at a shared conceptualization (as discussed in Section 2).

Our view is similar to the view of design science proposed by Wierenga *et al.* [4], but we place more emphasis on the technological rule construct that is also stressed by van Aken *et al.* [2]. What we have noticed in much of the prescriptive software engineering research is that the proposed **intervention** is often well described in terms of a method, an algorithm, a set of guidelines, or a tool, while the **problem** addressed (at the instance level) or the general technological rule is left as implicit or assumed. Even when many details are given for the problem instance, the scope of the study may not be well defined and there may be a mismatch between the description of the problem and what is actually evaluated or targeted in the solution design. Because of this, we believe that articulating a technological rule may improve how such research findings are communicated.

Another point of confusion around design science concerns *action research* [10], where the main goal is to make a change for a particular real-world problem instance. With design science, the main goal is to arrive at knowledge

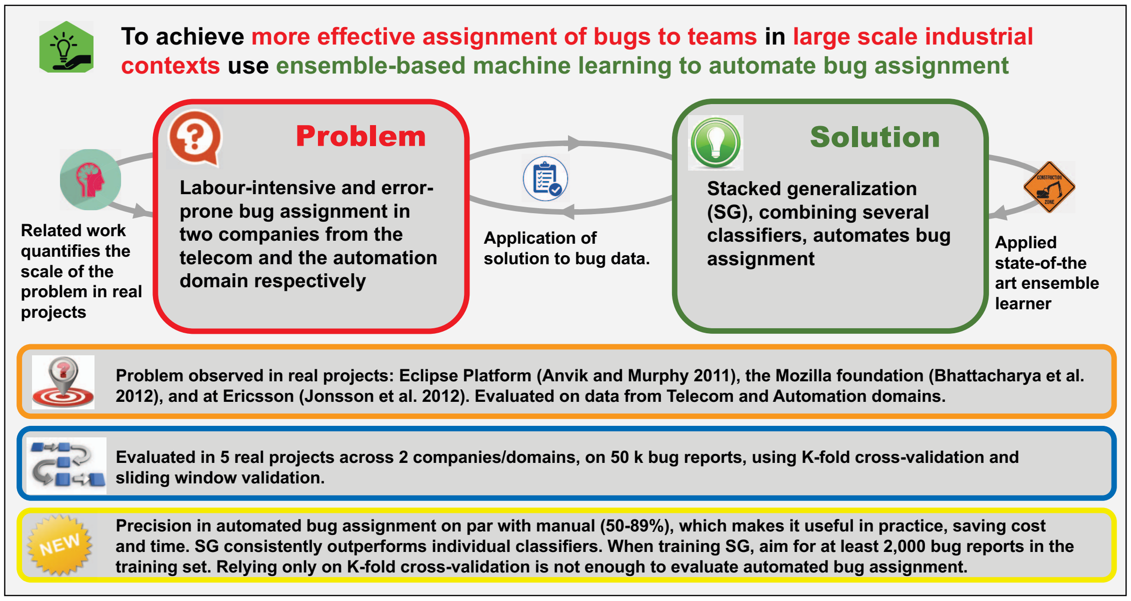
To achieve more effective assignment of bugs to teams in large scale industrial contexts use ensemble-based machine learning to automate bug assignment
Problem
Labour-intensive and error-prone bug assignment in two companies from the telecom and the automation domain respectively
Related work quantifies the scale of the problem in real projects
Application of solution to bug data.
Solution
Stacked generalization (SG), combining several classifiers, automates bug assignment
Applied state-of-the art ensemble learner
Problem observed in real projects: Eclipse Platform (Anvik and Murphy 2011), the Mozilla foundation (Bhattacharya et al. 2012), and at Ericsson (Jonsson et al. 2012). Evaluated on data from Telecom and Automation domains.
Evaluated in 5 real projects across 2 companies/domains, on 50 k bug reports, using K-fold cross-validation and sliding window validation.
NEW
Precision in automated bug assignment on par with manual (50-89%), which makes it useful in practice, saving cost and time. SG consistently outperforms individual classifiers. When training SG, aim for at least 2,000 bug reports in the training set. Relying only on K-fold cross-validation is not enough to evaluate automated bug assignment.

Figure 2. Visual abstract for the paper on automated bug assignment [11].

that can be applied in broader settings. Such knowledge is captured through the technological rule, generalizing the prescription beyond the specific problem or solution instances considered.

We recognized early on that simply arguing for broader acceptance of design science may not be sufficient and that we needed a way to demonstrate how design science can bring benefits to researchers and practitioners. Our visual abstract tailored for presenting design science findings that may otherwise be buried can help popularize design science in software engineering.

5.1. Why a Visual Abstract

We were inspired to use the visual abstract approach which has been popularized by Andrew M. Ibrahim, a surgeon whose main goal is to improve surgical care[1]. The visual abstract approach aims to highlight the key points and findings contained in a scientific abstract in a more visual manner. Ibrahim notes that the visual abstract, when used by the Annals of Surgery and many other medical journals, is not meant to replace the reading of an article but to draw attention to the study and the findings with the goal of improving patient care.

Structured abstract formats[2] address the same challenge of buried findings but are textual and better suited for experiments or studies that are best presented in a linear format. Moreover, structured abstract formats are for inclusion in papers, whereas we suggest that the visual abstract is more suited to posters or presentations, or as an addition to blog posts (such as http://neverworkintheory.org) and evidence briefings [14]. Early feedback from several of our industry partners was positive in terms of how the visual abstract may help disseminate research in today's highly visual world.

As mentioned, we refined the visual abstract by applying it to different research papers, discussing the template with other researchers and practitioners and refining it throughout. As we were designing the abstract, we initially attempted to use diagrams in key design science papers (notably by Hevner et al. and Wierenga et al.), but found the diagrams these authors provided were more useful in describing the research process followed rather than communicating and highlighting the research contributions. The factors we finally included in the template were selected due to their ability to communicate the most important aspects of design science research. Of course, as we evaluate it further (in ongoing work), the abstract may change but it stabilized in our latest attempts to apply it.

5.2. Limitations

It is important to stress that our visual abstract is a lens through which we can highlight research that is congruent with a design science view. For example, we have found that it isn't that useful to present research that is of a mainly descriptive nature (although for some papers it helped to highlight that fact), nor is it that useful for presenting research where a technological rule has yet to be developed (again highlighting the immaturity of the work). We tried to use it to describe the work contained in this paper but quickly realized that our work is too preliminary to suggest a technological rule that visual abstracts should be used for more rapidly communicating design science research

1. https://www.surgeryredesign.com/
2. See https://www.nlm.nih.gov/bsd/policy/structured_abstracts.html

in software engineering. We also asked a colleague to use the visual abstract to describe a research project spanning 10 years—he failed after a couple of hours and felt the abstract was better suited to more contained studies, such as in a single paper. He did say that with more time he could perhaps use it to present his career-long research and that it could be a useful exercise. Finally, the visual abstract relies on the skills of the researcher to decide what to include and what to leave out, just as writing relies on such skills. Consideration for who the recipient is (e.g., industry versus academia) is paramount for arriving at a compelling abstract.

6. Conclusions and Future Work

Our contributions from this paper are twofold. First, we provide a brief conceptualization of design science research and an argument of the benefits of using design science as a research paradigm in software engineering. Second, we present a visual abstract template and guidelines for how to apply it to research that is of a prescriptive design nature. Software engineering research often involves interdisciplinary perspectives—the design science view we provide helps to tie together diverse empirical steps and relate them according to an overall research goal. The template also helps to surface the contributions in a visual and less linear manner, and also helps the researcher to reason about and assess their contributions. We also believe that the design science visual abstract helps emphasize the main takeaway—the technological rule—that is sometimes left implicit or even missing from many research papers.

We don't claim the visual abstract is unique to software engineering, in fact we believe it would be useful for other domains. That said, we derived it only on consideration of 11 software engineering research papers and feedback from software practitioners and researchers. Our future work involves applying it to many more papers and asking other researchers to apply it to their work, while seeking feedback from authors and research stakeholders on its benefits and how it may be improved.

In conclusion, we hope that our work will lead to the increased adoption of design science in software engineering and that the visual abstract template will assist in more rapid communication of research contributions in our community.

Acknowledgments. We thank Greg Wilson for suggesting the visual abstract approach and Cassandra Petrachenko for her editing support. We also thank Bjorn Regnell, Sigrid Eldh, Leif Jonsson, Kristian Sandahl and Markus Borg for providing feedback on the visual abstract design.

References

[1] A. Hevner and S. Chatterjee, *Design Research in Information Systems: Theory and Practice*, 2010th ed. New York ; London: Springer, May 2010.

[2] J. E. v. Aken, "Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules," vol. 41, no. 2, pp. 219–246, 2004.

[3] S. Gregor, "The nature of theory in information systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, 2006.

[4] R. J. Wieringa, "What Is Design Science?" in *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, 2014, pp. 3–11.

[5] C. Wohlin, "Empirical software engineering research with industry: Top 10 challenges," in *2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI)*, May 2013, pp. 43–46.

[6] J. Siegmund, N. Siegmund, and S. Apel, "Views on internal and external validity in empirical software engineering," in *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, vol. 1. IEEE, 2015, pp. 9–19.

[7] T. Dybå, D. Sjøberg, and D. S. Cruzes, "What works for whom, where, when, and why? On the role of context in empirical software engineering," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Sep. 2012, pp. 19–28.

[8] M. Bunge, *Philosophy of Science: Volume 2, From Explanation to Justification*, 1st ed. New Brunswick, N.J: Routledge, Feb. 1998.

[9] S. Gregor and A. R. Hevner, "Positioning and Presenting Design Science Research for Maximum Impact," vol. 37, no. 2, pp. 337–356, Jun. 2013.

[10] R. Wieringa and A. Moral, "Technical Action Research as a Validation Method in Information Systems Design Science," in *Design Science Research in Information Systems. Advances in Theory and Practice*. Springer, Berlin, Heidelberg, May 2012, pp. 220–238.

[11] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh, and P. Runeson, "Automated Bug Assignment: Ensemble-based Machine Learning in Large Scale Industrial Contexts," vol. 21, no. 4, pp. 1579–1585, 2016.

[12] M. Borg and P. Runeson, "IR in Software Traceability: From a Bird's Eye View," in *In Proceedings Empirical Software Engineering and Measurements*, 2013, pp. 243–246.

[13] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 361–370.

[14] B. Cartaxo, G. Pinto, E. Vieira, and S. Soares, "Evidence Briefings: Towards a Medium to Transfer Knowledge from Systematic Reviews to Practitioners," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '16. Ciudad Real, Spain: ACM, 2016, pp. 57:1–57:10.