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*—Much empirical software engineering research aims at producing prescriptive knowledge that helps software engineers improve their work or solve their problems. But deriving general knowledge from real world problem solving instances can be challenging. In this paper, we promote design science as a paradigm to support producing and communicating prescriptive knowledge. We propose a visual abstract template to communicate design science contributions and to highlight the main problem and solution constructs of their research, as well as present validity aspects of design knowledge. Our conceptualization of design science is derived from existing literature and were together with the visual abstract derived iteratively as we applied them to different examples of design science research. We present and discuss one example application of the visual abstract. This is work in progress and further evaluation by practitioners and researchers is encouraged.**

1. Introduction

Articulating research contributions in software engineering and highlighting their value is often a challenging endeavour. In part, this is due to the interdisciplinary nature of our research which may build on and contribute to many fields including mathematics, engineering, design, and the social sciences. These disciplines rely on very different research paradigms and contradictory world views on what may be valid and reliable research across these disciplines. As software engineering is furthermore a professional field of practice, the relevancy of the problems researched and the value of prescriptions to solve these problems are often called into question by practitioners and other researchers.

Similar challenges in articulating and agreeing on research contributions have been experienced in many research disciplines and are not unique to software engineering. Examples of such disciplines include information systems and management science. *Design Science Research* has been widely adopted in these and engineering disciplines as a way to frame research by highlighting both the problem addressed and the intervention proposed [1], [2], [3]. Design science supports a researcher in conveying how they build on and contribute to an existing knowledge base.

Despite the applied and interdisciplinary natures of our discipline, design science is under utilized in software engineering despite attempts to popularize it within our community [4], [5]. This is unfortunate as many research contributions are not accepted due to unclear research contributions or due to disagreements about what constitutes a valuable "software engineering research contribution". Furthermore, many research contributions go unnoticed by practitioners that could benefit from those insights.

We propose that design science is a powerful way to frame prescriptive software engineering research and should be promoted. We observe that many software engineering research papers do not explicitly describe the class of problems addressed, nor do they sufficiently describe the problem's relevance to practitioners. They tend to focus on problem solving instances but do not sufficiently reason about how the knowledge can be generalized [7]. Design science prompts researchers to uncover the relevance of the class of problems addressed, as well as clearly surface the knowledge of how solutions may help.

To promote adoption of design science and to communicate its benefits for empirical software engineering, we present a *visual abstract* template that captures key concepts in design science. This visual abstract is a work in progress and 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 argue that the visual abstract has two benefits. Firstly, it can be used to capture the takeaway, the scope and process, and the value from a study which produces design knowledge in terms of a prescription to solve a given problem. That is, the abstract forms a lens for describing design science research. Secondly, based on preliminary reactions from research and industry colleagues to the abstract, we suggest that the visual abstract may lead to more efficient and more accurate dissemination of research findings with industry as well as between researchers.

Our paper is organized as follows. In the next section (Section 2), we present our view of design science, building and aggregating but rejecting some concepts from other conceptualizations of design science. This view of design science helps set the stage for the next section in the paper (Section 3) where we present a visual template for design science research. We also provide guidelines for applying the template to design science research. In Section 4, we

demonstrate and discuss the visual abstract in action by applying it to a previously published paper. Finally, in Section 5, we discuss how we arrived at this particular visual abstract template for design science and share our experiences using and iterating on the template and its associated guidelines. We also list limitations of when design science and in turn how our abstract may be applied. Finally, in Section 6, we conclude with future work.

2. A brief conceptualization of design science

Software engineering research should ultimately produce knowledge that helps software engineering professionals design solutions to their problems. This implies that the knowledge produced is prescriptive, i.e. it provides advice on how to act in various situations. Such knowledge is by necessity context dependent and relative to a defined setting [4]. It shares characteristics with what is referred to as design knowledge or the product of design science research, in other disciplines such as management research [2]. Although the conceptualization varies between fields and design science researchers, these researchers share the view of knowledge as being holistic and heuristic, and justified by in-context evaluations. We build on this basis and select concepts and terminology from several sources on design science, as defined below.

The term holistic is used by van Aken [2] and refers to the "magic" aspect of design knowledge where we may partly understand why a certain solution works in a specific context, but never fully. 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 having an effect on an instance of a problem-solution pair [6]. Similarly, we can never prove the effect of a heuristic prescription conclusively, but have to rely on in-context evaluations. By evaluating multiple instances of problem-solution pairs matching a given prescription, our understanding about that prescription increases.

Such holistic and heuristic prescriptions are, in design science research, expressed in terms of *technological rules* [2]. A technological rule captures general knowledge about the *mapping* between a *problem* and a proposed *solution*. Furthermore 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's and Hevner's work [9].

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

In line with van Aken [2], we emphasize, in our visual abstract template, the technological rule as the main take away of software engineering design science 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 an envisioned stakeholder's desired effect of a potential intervention in a specified situation. Making this problem generalization explicit helps the researcher identify and communicate 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 solutions in context [4], through action research (i.e. within one context) [10], or through multiple case studies (across contexts) [2] or through design cycles [9]. This new knowledge is coupled with a technological rule and should be communicated and assessed from that perspective. The technological rule may either be a new proposal or an existing one which is refined or validated by the current study.

Due to the holistic nature of the technological rule, each such instantiation (i.e., in a case study or an iteration of action research) constitutes a unit of analysis in relation to that rule. This means that to add new empirical knowledge to a general technological rule, the rule must be instantiated (i.e. an instance of the technological rule must be applied to solve a real problem or improve a real situation) and reflected on as whole. In the visual abstract template the researcher is encouraged to reflect on how the current 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. In our visual abstract, the scope of study constitutes the main body and captures this instantiation in terms of a description of the problem instance(s) and the solution instance(s). Furthermore, refinements or validation of the technological rule may be derived from any one of the three processes of *problem understanding*, *solution design* or *solution evaluation*, applied in each instantiation and is therefore also part of this main body. Note that the scope of study is not the same as the scope to which generalizability is claimed which should rather be captured in the articulation of the technological rule.

Finally our visual abstract template also comprises the assessment of the produced design knowledge. This is to support the recipient of the research, i.e., the industry practitioner or peer researcher, to effectively assess the value of a technological rule in relation to his or her own situation. Drawing from design science literature, we select three criteria for research evaluation: *relevance, rigor*, and *novelty*. However, although they are commonly accepted research assessment criteria in most research communities, the interpretation of these criteria tends to vary. We discuss these criteria further in the next section where we also propose a visual abstract template for communicating research in software engineering that is congruent with the conceptualization of design science we just described.

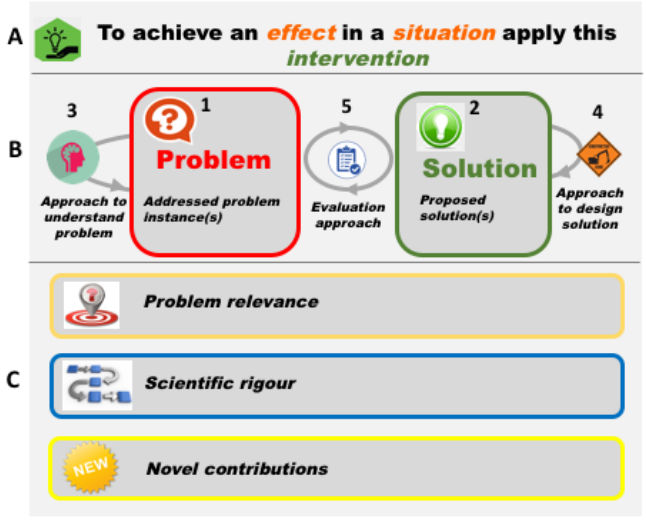

Figure 1. A visual abstract template for design science research. The researcher should fill this in using the guidelines provided. Part A denotes the technological rule. Part B shows the problem instance(s), solution(s) proposed and the empirical and research steps followed. Part C shows three design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

3. A Visual Abstract for Design Science

Our proposed **visual abstract** (see Fig. 1) relies on the concepts described above. It captures three main aspects of design science contributions: A) the general take away in terms of a technological rule; B) the research scope and process, in terms of one or more instances of problem-solution pair(s) and corresponding design and evaluation cycle(s); and C) assessment of the value of the produced knowledge, in terms of relevance, rigor and novelty. We provide guidelines for creating a visual abstract of a particular instance of design science research in software engineering based on this template. The italics text in the template should be replaced with text pertinent to the research being communicated through the template. Later, we show the visual abstract in action with a specific research example.

3.1. Take Away: The Proposed Technological Rule

We see the technological rule as the crucial part of the visual abstract because it captures the main contribution or take away from the study in a single logical phrase (see Fig. 1, part A). A technological rule has three key constructs: (1) the **effect** that is desired in (2) a given **situation** along with (3) a proposed **intervention** to achieve the desired effect in a given situation.

Taking care to formulate a technological rule forces the researcher to reflect on the knowledge contribution, which in turn helps to articulate the problem that they address in their research accordingly. 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 scope and process of the current study or research program and is composed of five constructs (once again see Fig. 1, Part B): 1) the problem instance(s) addressed; 2) the corresponding solution(s) proposed; 3) the empirical and research work conducted to understand the problem; 4) the empirical or design work to arrive at the proposed solutions from alternative options; and 5) the empirical work conducted or proposed to evaluate the proposed solution(s) when applied to one or more problem instances. Although we present these five constructs in a particular order, this order does not imply how the research steps were conducted. It may even be that the researcher doesn't start with a problem in mind, but may start with a tool or technique and then search for a problem instance to apply a hypothesized solution.

The **problem instance** box captures which instances of the problem and how the scope of the studied problem are considered in the reported 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. Similarily, the **solution(s)** proposed should be described in terms of how the general intervention was applied to address the problem in the specific case. 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.

The **problem understanding approach** should support 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 instance(s) addressed. Note that insights on the problem(s) may also come from the solution evaluation.

The **evaluation approach** should describe 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 the solution evaluation (and other evaluations of other problem solving instances) is essential in validating the technological rule. Here, we encourage the researcher to include the main elements of the evaluation steps that have been followed, or to mention if the solution evaluation remains future work.

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

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, or 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. 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 may be quite challenging and may lead the researcher to consider uncomfortable questions about their research, while at the same time providing them with a way to showcase what they have done in their research to demonstrate relevance, rigor and novelty. We provide more guidance on these three boxes next.

3.3.1. Problem relevance. 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. This may help the recipient to generalize from the case context presented to their own potential needs and assess whether the contribution is relevant to them. Arguing relevance at the general level can be harder to do than arguing relevance for the stakeholders involved in a specific study. 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.

3.3.2. Scientific rigor. When it comes to filling out this part, rigor could 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 in this box.

3.3.3. Novel contributions. Novelty of the presented design science research 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 solution is new and has been shown to bring value. We think this latter point is particularly important, because many researchers assert that when both the solution and problem are well known, that applying the solution to the problem is merely "routine design" We disagree with this view, 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 sum, 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 and show how the template can be used to communicate the research in a previously published paper. We discuss the decisions made and tradeoffs faced while filling out this template.

4. Design Science Visual Abstract in Action

During our iterative process of designing the visual abstract, we have applied different variants of our abstract to examples of software engineering research (four studies we were involved in and seven studies where we had no involvement) but can only share one example in this short paper. We chose an example where we could share insights as an author of the work (Per Runeson) and we could also share feedback from other co-authors that have experience in research and industry.

Our demonstration example (see Fig. 2) addresses a problem, identified by several observational studies, that manual assignment of bugs to teams for repair is labor-intensive and error-prone [11]. The **solution** is to use established Machine Learning (ML) techniques (ensemble learners, Stack Generalization) to dispatch issues to the right team. The solution is **evaluated** with real, large data sets from two industry domains.

Due to the sensitivity to specific data sets in ML [12], the quantitative results in the evaluation can only be assessed in the local context, while the more generic knowledge related to the solution is considered more general. The insights gained from applying the solution to these two **problem instances**, have thus been generalized to the presented **technological rule** in Fig.2.

The work presented in our example paper addresses the **relevant** challenge of bug assignment in industrial projects, such as Eclipse, as shown by Anvik *et al.* [**?**] and others. It builds on existing research that proposes that machine learning is a useful approach for bug assignment [**?**]. However, this previous work did not demonstrate that this approach scales to large scale projects. The visual abstract communicates that the contribution of the work is in the application of known techniques to a known problem, while it has not been demonstrated to work on this industrial scale before and so is **novel**. The problem understanding and evaluation have been conducted in a large scale industrial context, which adds to the **rigor** of the solution. This validation adds credibility to the technological rule we present in this abstract.

Runeson defined the visual abstract for this work, and asked the co-authors, both from industry and academia, to assess the presentation. One industry perspective was that the cost/benefit aspect was not sufficiently clear in this abstract. The level of detail at which the solution should be presented in the abstract was also raised. From the industry perspective, the novelty lies in that the precision of automated assignment is on par with the manual process. But from an academic perspective, the specifics of the classifiers and their performance were considered more

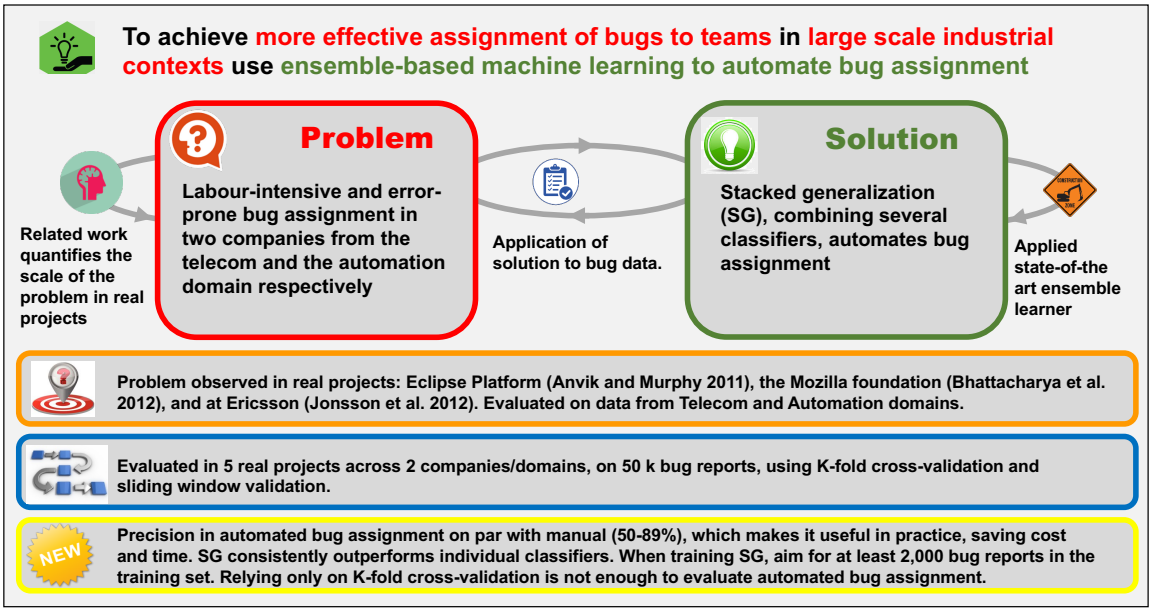

Figure 2. Visual abstract for the paper on automated bug assignment [11].

significant contributions. We see the industry goal captured more clearly with 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 that time, we (all the authors) met regularly to discuss different papers on design science and to arrive at a shared conceptualization of design science (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 software engineering research that is of a prescriptive nature, 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 instance level) or the general technological rule is left as implicit or assumed. Even when many details are given on the problem instance(s), 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. Articulating a technological rule, we believe, may improve how such research findings are communicated.

Another point of confusion around design science concerns *Action Research* [10]. With Action Research the main goal is to make a change for a particular real world problem instance, whereas with Design Science the main goal is to arrive at knowledge that can be applied in broader settings. Such knowledge is captured through the technological rule, generalizing the prescription beyond 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 approach to help popularize design science in software engineering is to propose a visual abstract, tailored for presenting design science findings that may otherwise be buried.

Structured abstract formats[1] address the same challenge of buried findings but they are textual and are more suited for experiments or studies that are best presented in a linear format. Blogs (such as http://neverworkintheory.org) and evidence briefings [13] have also been proposed, but they also present the results through text in a linear fashion. Feedback from our industry partners suggest that visual presentations may help disseminate research in today's highly visual world. Thus we turned to a visual abstract template[2] to highlight the key points and findings contained in a scientific abstract. The visual abstract, when used by the Annals of Surgery and many other medical journals, is not meant to replace reading of an article but to draw attention to the study and the findings with the goal of improving patient care.

The visual abstract we propose was designed by iteratively trying it out on different research papers, discussing and refining the template with other researchers and prac-

1. See https://www.nlm.nih.gov/bsd/policy/structured_abstracts.html
2. Popularized by Andrew M. Ibrahim, a surgeon whose main goal is to improve surgical carehttps://www.surgeryredesign.com/

titioners. We started with diagrams used in design science methodology papers (notably by Hevner et al., and Wierenga et al.,) but we found the diagrams they provided were more useful in describing the research process followed rather than communicating and highlighting the research contributions. The factors we included in the template in the end were evaluated by 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.

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). For example, 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 in software engineering. We also asked a colleague to use the visual abstract to describe a 10 year research project – he failed after a couple of hours and felt the abstract was more 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 (industry versus academia, for example) is paramount at arriving at a compelling abstract.

6. Conclusions and Future Work

Our contributions from this paper are twofold. Firstly, we provide a brief conceptualization of design science research and provide an argument of the benefits of using design science as a research paradigm in software engineering. Secondly, 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, that is 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. 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 editing support. We also thank Bjorn Regnell, Sigrid Eldh, Leif Jonsson, Kristian Sandahl and Markus Borg providing feedback on various versions of the visual abstract.

References

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

[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.

[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, pp. 3–11. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-662-43839-8_1

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

[6] T. Dyb{\textbackslash}a a, D. I. K. Sjberg, 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*, pp. 19–28.

[7] 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.

[8] M. Bunge, *Philosophy of Science: Volume 2, From Explanation to Justification*, 1st ed. Routledge.

[9] S. Gregor and A. R. Hevner, "Positioning and presenting design science research for maximum impact," vol. 37, no. 2, pp. 337–356. [Online]. Available: http://dl.acm.org/citation.cfm?id=2535658.2535660

[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, pp. 220–238. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-29863-9_17

[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.

[12] M. Borg and P. Runeson, "IR in software traceability: From a bird's eye view," in *In Proceedings Empirical Software Engineering and Measurements*, pp. 243–246.

[13] 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. ACM, pp. 57:1–57:10. [Online]. Available: http://doi.acm.org/10.1145/2961111.2962603