Technical Action Research as a Validation Method in Information Systems Design Science

Roel Wieringa[1] and Ayşe Moralı[2]

[1] Department of Electrical Engineering, Mathematics, and Computer Science, University of Twente, The Netherlands
r.j.wieringa@utwente.nl,
http://www.ewi.utwente.nl~roelw
[2] PwC, Gent, Belgium
ayse.morali@pwc.be

Abstract. Current proposals for combining action research and design science start with a concrete problem in an organization, then apply an artifact to improve the problem, and finally reflect on lessons learned. The aim of these combinations is to reduce the tension between relevance and rigor. This paper proposes another way of using action research in design science, which starts with an artifact, and then tests it under conditions of practice by solving concrete problems with them. The aim of this way of using action research in design science is to bridge the gap between the idealizations made when designing the artifact and the concrete conditions of practice that occur in real-world problems.

The paper analyzes the role of idealization in design science and compares it with the requirements of rigor and relevance. It then proposes a way of bridging the gap between idealization and practice by means of action research, called technical action research (TAR) in this paper. The core of TAR is that the researcher plays three roles, which must be kept logically separate, namely of artifact developer, artifact investigator, and client helper. Finally, TAR is compared to other approaches of using action research in design science, and with canonical action research.

1 Introduction

Design science is the study of artifacts in context [1]. Interest for it in the information systems community arose out of a desire to make research results more relevant, by adding a problem-solving cycle to a theory-building cycle. *Action research,* taken very generally, is the intervention in a social situation in order to both improve this situation and to learn from it [2]. It arose shortly after World War II out of dissatisfaction with social science as mere diagnosis [3]. In addition to diagnosis, action research includes an intervention by scientists to improve the situation, and to learn from what happened in this "social experiment".

Both research approaches, design science and action research, are motivated by the desire to increase the relevance of research by incorporating a social problem-solving activity in research without sacrificing rigor [4,5]. This has motivated several authors to propose mergers of action research and design science

K. Peffers, M. Rothenberger, and B. Kuechler (Eds.): DESRIST 2012, LNCS 7286, pp. 220–238, 2012.

research. Lee [6] proposes an elegant extension of the framework of March & Smith [7] in which actions are treated as artifacts to be built, evaluated, theorized about, and justified. Baskerville et al. [8] integrate both action research and design science in Checkland's soft systems approach. Sein et al. [9] propose an integrated cycle that combines building, intervention and evaluation from design science research with reflection and learning from action research, emphasizing the interleaving of building and IT artifact with intervening in an organization. All of these approaches start from an organizational problem to be solved by action research, and then design an artifact to solve this concrete problem. They then reflect on this experience to draw generalizable lessons learned from this.

In this paper, we will start at the opposite end, namely artifact design, and then look for organizational problems that could be solved by this artifact. The goal of the researcher is to develop this artifact for use in a class of situations imagined by the researcher. Typically, then, the artifact is first tested, not on a real-world problem, but on toy problems under idealized circumstances in a laboratory. Next, it is scaled up to conditions of practice by solving more realistic problems with it, until it can be tested by using it in one or more concrete client organizations to solve concrete problems.

We propose to use action research in the last stages of this process of scaling up to practice. The only way to leave the idealized circumstances of the laboratory is to enter the real world. This is very similar to the way new medicines are transferred to practice, after first testing them under idealized conditions in a lab, then testing them with healthy volunteers and, eventually, with patients. To mark this artifact-driven action research off from the problem-driven action research mentioned above, we call it *technical action research* (TAR).

The primary motivation for this way of using action research is to bridge the gap between the idealizations of initial design and the concrete conditions of practice. We discuss this gap in the next section, and describe our way of using action research in more detail afterwards.

2 Relevance, Rigor and Idealization

TAR is intended to increase the relevance of artifacts just as other forms of action research aim to increase the relevance of knowledge. But the relevance gap that is bridged by TAR is the one between idealized conditions and practical conditions. To explain this, we return for a moment to the problem of rigor and relevance as originally introduced by Schön [10]. Schön is interested in problem-solving in the professions, such as in architecture, health care or law and he contrasts this with problem-solving in the technical sciences. To draw this contrast, he introduces what he calls "technical rationality", defined by him as a problem-solving approach in which possible alternative solutions are compared with respect to goals, before one solution is selected to be implemented. He identifies four assumptions about the problem made by technical rationality [10, pages 40–42]:

- The problem is framed,
- it is an example of a problem class,

– it is stable, and
– it has unambiguous goals.

Problems in the technical engineering sciences, Schön says, satisfy these assumptions, but in some professional practices, such as in the practice of architecture or law, problems do not satisfy these idealizing assumptions [10, page 42]. Technical rationality, rigorous as it is, is then not relevant for these problems and a more "artistic" way of coping with the problem is required, using Schön's words.

Schön himself frames this as the dilemma of rigor "versus" relevance, but stated like this, this is a false dilemma. First, as also acknowledged by Schön, there are many practical problems in the professions that *do* satisfy the assumptions of technical rationality, and for these problems, technical rationality can produce relevant solutions in a rigorous way. So in these cases, rigor is not opposed to relevance.

Second, Schön assumes that in the technical sciences, knowledge is developed in isolation from practice [10, pages 23–26]. All that a technical problem-solver has to do, according to Schön, is to select whatever he or she can use from this knowledge to solve the technical problem at hand. However, in the technical sciences too, knowledge developed in isolation from practice is not relevant. Cartwright [11] argues at length that the laws of physics are literally false, because they make idealizing assumptions, such as the existence of point masses and frictionless surfaces, which are false in the real world. Rather than being about the real world, these laws can better be viewed as being about abstract, "nomological machines" [12]. McMullin [13] calls this "Galilean idealization". The purposes of idealization are to enhance insight into an idealized phenomenon in isolation, and to enhance our capacity to reason and compute about this isolated idealized phenomenon. In the laboratory, great effort is exerted to approximate the idealizations assumed by these nomological machines.

This poses a relevance problem for the technical sciences, analogous to the relevance problem indicated by Schön for the social sciences. Technical professionals must design and investigate machines in the real world, where there is no budget or inclination to enforce the idealizations of nomological machines [14,15]. How then can they use scientific knowledge to develop artifacts for the real world outside a laboratory? By designing and investigating artifacts first under idealized conditions, and then scaling up in small steps to conditions of practice [16,15]. Prototypes of artifacts are first tested in the idealized conditions of the laboratory, and then these conditions are gradually relaxed until a realistic version of the artifact is tested in a realistic environment. This allows the technical scientist to develop knowledge about the behavior of artifacts in practice. The technical approach to scaling up increases relevance without sacrificing rigor.

In medical research too, medicine are first tested in safe, artificial conditions, after which the context is scaled up to more realistic conditions of practice, by first testing it on healthy volunteers, and then ill volunteers.[1] And this too is done according to rigorous standards of scientific methodology.

[1] `http://www.fda.gov/Drugs/DevelopmentApprovalProcess/SmallBusinessAssistance/ucm053131.htm`

The problem of application-oriented research in the technical and medical sciences is, then, not one of rigor against relevance, but one of idealization versus practice. And this problem is dealt with in these sciences by starting artifact development in idealized circumstances and slowly scaling up to conditions of practice.

If this approach can be used successfully in the technical and medical sciences, then it is worth trying to do also use it in information systems design science. In this paper we will approach problem solving with Schön's technical rationality, in the form of an *engineering cycle* in which artifacts are compared and evaluated against stable goals. We will discuss the engineering cycle in more detail later, but for now it suffices to say that it is a rational decision process in which the designer generates and evaluates alternative designs by comparing them against design goals. In the first iterations through the engineering cycle, the designer makes idealizing assumptions to make it easier to find a design at all. After a proof of concept has been given, the designer improves the design by iterating through the cycle, gradually relaxing the idealizing assumptions, until all remaining assumptions can easily be satisfied in practice. Our use of TAR will be in the last stages of this iterative process, namely when the researcher tests an artifact by using it to solve a client's problem.

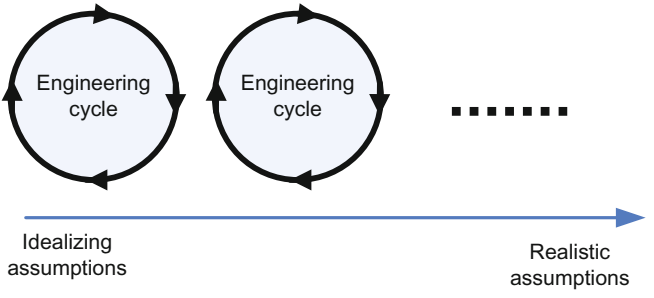

Fig. 1. Scaling up by iterating through the engineering cycle, making more realistic assumptions in later iterations

This requires a clear distinction between problem-solving for the client, artifact development by the researcher, and knowledge acquisition by the researcher. For this, we use the refinement of the framework for design science of Hevner et al. [1], discussed next.

3 A Framework for Design Science

Figure 2 shows an adaptation of the top-level framework of Hevner et al. [1], showing design science and its interfaces with the environment and the scientific knowledge base. This framework has been presented and motivated in more detail

earlier, and we here briefly summarize the aspect necessary for the purpose of this paper [17].

In the new framework, the environment is the source of design goals and of a budget to achieve them. This budget can consist of money and instruments, but will at least consist of the time of the researcher, which has to be paid for in one way or the other. In return, the design researcher delivers artifacts that can be used to solve problems in the environment, i.e. to achieve goals in the environment.

The important change with respect to the framework of Hevner et al. is that the design science activity has been split into two, solving improvement problems and answering knowledge questions.

- Solving improvement problems corresponds, roughly, to building artifacts in the framework of Hevner et al. Here, we define an *improvement problem* as a difference between the actual state of the world and the world as desired by some stakeholder, where this stakeholder has made available a budget to reduce this gap. If a stakeholder has a desire but is not willing to make a budget available to achieve it, in the form of time, money or other resources, then we do not consider this desire to be a stakeholder goal. The improvement problems we are concerned with consist of designing and evaluating artifacts.
- Answering knowledge questions corresponds, roughly, to developing theories in the framework of Hevner et al. We define a *knowledge question* is a lack of knowledge about some aspects of the real world. This includes the development of theories but it can include more, such as the development of rules of thumb or of design guidelines [18].

The distinction between improvement problems and knowledge questions is important, because the attempt to solve them require the problem solver to do different things. Attempting to solve an improvement problem requires the problem solver to identify the relevant stakeholders, their goals, and criteria for the improvement, and to design a treatment that aims to change the real world in the direction of stakeholder goals. Attempting to answer a knowledge question, by contrast, requires us to identify the questions and unit of study, and define measurements that will provide the quantitative or qualitative data by which we can answer these questions.

Not only should we do different things to solve improvement problems or answer knowledge questions, the results of these efforts are also evaluated

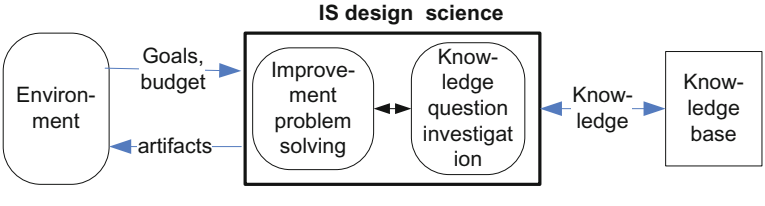

Fig. 2. Framework for design science

differently. Evaluating the result of solving an improvement problem involves the application of criteria to see if the improvement has been achieved. These criteria indicate effectiveness (has a change been achieved) and utility (has the change led to an improvement). Evaluating the result of answering a knowledge question, by contrast, involves assessing the truth of these answers and our degree of certainty about this, as well as assessing the scope of these answers.

These differences appear in the formulation of improvement problems and knowledge questions. Improvement problems can always be formulated in how-to-do form, such as

– "How to assess confidentiality risks in outsourcing IT management?"

Going one step further, they can also always be reformulated as design assignments, such as

– "Improve the assessment of confidentiality risks in outsourcing IT management."

Knowledge questions, by contrast, leave the state of the world as it is. They always can be rephrased into descriptive, explanatory, or predictive questions about the world, where descriptive questions may concern the current state of the world or its history:

– "What are the confidentiality risks in this outsourcing arrangement?" (descriptive question about current state of the world)
– "What events have led to this outsourcing arrangement?" (descriptive question about history of the world)
– "Why have these decisions been made?" (explanatory question)
– "What would be the effect of applying this new confidentiality assessment technique?" (predictive question)

Having distinguished improvement problems from knowledge questions within design science, we now turn to the engineering cycle, which is a rational way to solve improvement problems.

4 The Engineering Cycle

In the engineering cycle, an improvement problem is investigated, alternative treatment designs generated and validated, a design is selected and implemented, and experience with the implementation is evaluated (figure 3). The structure of this cycle has been extensively motivated elsewhere [19].[2] Here we give a brief summary.

[2] There, engineering cycle has been called the *regulative cycle*, following Van Strien [20].

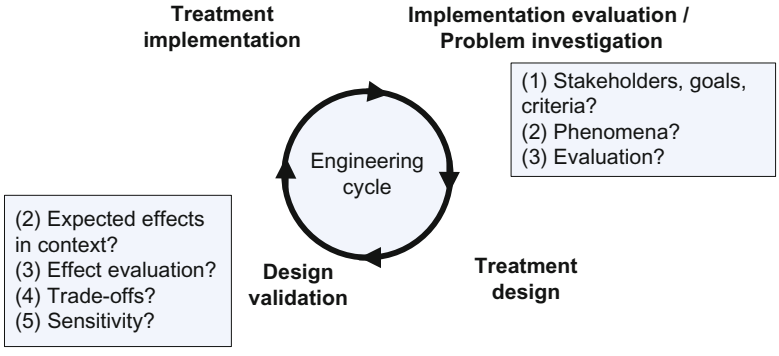

Fig. 3. The engineering cycle

4.1 Problem Investigation

In the *problem investigation* task,

(1) stakeholders and their goals are identified, and these are operationalized into criteria.
(2) Phenomena relevant for the improvement problem must be investigated, and
(3) it must be assessed how well these phenomena agree with the goals of the stakeholders.

This provides the researcher with a map of the needs for improvement.

4.2 Treatment Design

The design scientist then designs one or more *treatments*, which we here assume consist of an *artifact* interacting with a *problem context* (figure 4). This is a useful way of conceptualizing artifacts. Any IT artifact, whether it is an information system or a method or technique for developing, implementing, maintaining or using information systems, is used by inserting it in a problem context, with which it then starts interacting. Artifacts may consist of software or hardware, or they may be conceptual entities such as methods and techniques or business processes.

The interaction between an artifact and the problem context is the treatment that is expect to improve the context, just as a medicine (artifact) is inserted in a context (human body) which starts a treatment that is expected to improve the context. In our case we assume that the problem context is a social system, such as an organization, containing information systems. The problem to be solved by inserting an IT artifact in this problem context, is that some stakeholder goals need to be achieved.

Stakeholders are legal or biological persons who are affected by the artifact, and are part of the problem context. *Practitioners* are people designing particular

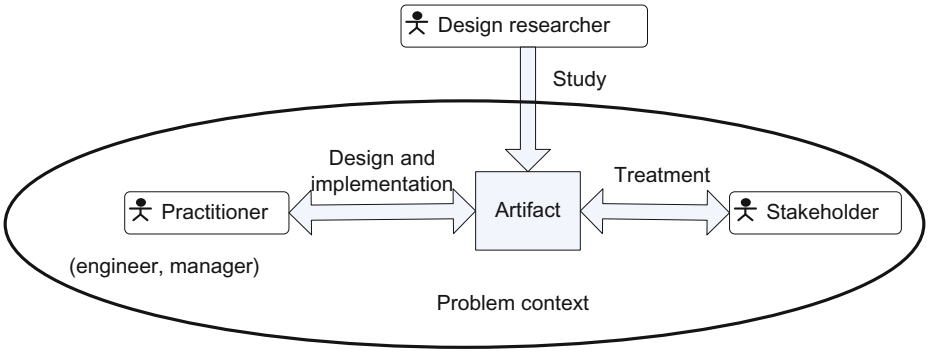

Fig. 4. Treatments and artifacts

treatments for particular problems, or managing the attempt to solve a particular problem. They are part of the problem context too. Their interaction with the artifact consists of designing and implementing it in that particular situation.

It is important here to distinguish *particular problems* that exist at a particular place and time from *problem classes.* The problem that company A has, today, of assessing confidentiality risks of company A when outsourcing the management of ERP systems to company B, is a particular problem. The problem concerns individual, named companies, and it exists at a certain time and place. But this particular problem is an instance of a problem class, namely the generic problem how confidentiality risks in outsourcing are to be assessed. In fact, a particular problem can be viewed as an instance of many related but different problem classes. We can consider the class of problems of confidentiality in outsourcing in general, or the problem class of outsourcing IT management, or the problem class of of outsourcing ERP management.

This distinction is important in design science because we are aiming at *general* knowledge about classes of problems, and so we have to be clear about the class we want to generalize to. The design scientist can study particular problems, containing particular practitioners, artifacts and stakeholders, in order to learn something about a class of problems of similar structure.

4.3 Design Validation

In the engineering cycle, when a treatment is designed, it is validated before it is implemented. In validation, stakeholders and their goals are assumed to have been identified in the problem investigation, so we skip that question. The two core knowledge questions of validation can be stated as follows:

(2) Expected effects: What will be the effects of the artifact in a problem context?
(3) Expected value: How well will these effects satisfy the criteria?

This allows for the possibility that there is more than one effect, and for the possibility that these may satisfy the criteria only partially, and that some effects may even violate some criteria.

Before implementation, two other questions must be asked, namely

(4) Trade-offs: How does this treatment perform compared to other possible treatments?
(5) Sensitivity: Would the treatment still be effective and useful if the problem changes?

The trade-off question (4) includes comparison between reduced versions of the treatment, as well as with existing treatments. The sensitivity question includes assessment of what happens when the problem grows larger (e.g. more stakeholders, more data, etc.) or when the problem is compounded with other problems.

4.4 Treatment Implementation and Evaluation

Implementation in this paper is transfer to the environment (figure 2). When a treatment is actually used in the real-world, then it can be *evaluated* by asking the same questions as before, about

(1) stakeholders,
(2) effects,
(3) value and
(4) sensitivity to problem context.

Note that in problem investigation, we ask for phenomena, which may be effects of existing technology in a context, whereas in implementation evaluation, we specifically ask for the effects of the treatments and artifacts under evaluation. The questions in both cases are the same, but the focus in different: In one case it is on problematic phenomena, whereas in the other on the effects of implemented technology.

This finishes our review of the engineering cycle. We next present TAR as one way to perform the validation task in the engineering cycle.

5 Artifact Validation by Technical Action Research

5.1 Designing an Artifact and Helping a Client with It

As explained earlier, what we call *technical action research* in this paper is the attempt to scale up a treatment to conditions of practice by actually using it in a particular problem. Figure 5 shows that TAR consists of two engineering cycles. In one engineering cycle, the researcher aims at improving a *class* of problems; in the other, the researcher improves a *particular* problem. We explain this by means of an example.

The problem in the left-hand engineering cycle is to improve an artifact, for example to improve a technique for assessing confidentiality risks in outsourcing

Fig. 5. Engineering cycles for artifact development and client helping

[21]. Confidentiality risks exist because outsourcing places sensitive information assets at the premises of a third party. In addition to the risk posed by placing information assets outside the premises of the outsourcing client, another risk is introduced because some employees of the outsourcing provider have legitimate access to confidential data of the outsourcing client, independent of where this information is placed. This creates further risks because these employees are outside the reach of internal control of the outsourcing client. Current service level agreements (SLAs) are often not sufficient to satisfy auditors of the outsourcing client that the client is in control of its information assets. However, the outsourcing provider will not allow these auditors on their premises because of the provider's confidentiality requirements; an outsourcing provider may provide outsourcing services to different clients that are each other's competitors.

The technique introduced by Morali and Wieringa [21] consists of some notations to represent the structure of the outsourcing architecture, and techniques to reason about vulnerabilities of this architecture, and about the value of information flowing through this network.

Consider the situation where these techniques have been designed, and tested on some artificial examples by the researcher. These tests have been successful and the researcher thinks these techniques can be used to solve a real-world problem. How to validate this claim? One way to validate it is that the researcher uses the technique to actually do a real-world confidentiality risk assessment for a particular, real-world client. This is the right-hand engineering cycle in figure 5.

In this cycle, an intake is done (stakeholders in the client company, client goals, relevant phenomena and their evaluation) and a treatment plan is agreed on. Part of this agreement is a justification that this plan is expected to indeed deliver the results expected by the client, and this justification consists of an application of what the researcher knows about this treatment. In the very first application of the treatment in a client company, the researcher's knowledge will be abstract and uncertain, and so the risk that the expected results will not be achieved is high; but this is a risk to be taken if a real-world test of the techniques is ever to take place. The researcher then uses the technique to help the client, and evaluates the results with the client.

The goal of the client cycle in figure 5 is to answer one or more validation questions in the researcher's cycle. This is a research goal, and to make this explicit we add an empirical cycle, as described in the next section.

5.2 Inserting the Research Cycle in TAR

The four validation questions in the researcher's cycle (expected effects, expected value, trade-offs, sensitivity) are knowledge questions. Specifically, they are prediction questions: They ask for what *would* happen if the artifact *would* be transferred to the real world. In figure 6 we have inserted a research cycle to make this logic explicit.

The research cycle shown in figure 6 has the same rational problem solving structure as the engineering cycle, but this time, the goal is not to design and implement a useful artifact, but the goal is to design and implement a way of answering knowledge questions.

Research Problem Investigation. In research problem investigation, we determine what the unit of study is, what concepts we use to state the research questions about the unit of study, and what we already know about these questions. In design science, the unit of study is an artifact in context (figure 4) and the research questions are elaborations of one or more knowledge questions that appear in the engineering cycle. In this paper, we are concerned with validation research, and so the research questions will be one or more of the four design validation questions, or elaborations of these questions.

The set of all units of study make up a population. It is often difficult in design science to state in advance exactly what the population is. The precise delimitation of the population may require continued research. For example, we may be interested in the effectiveness and utility of our techniques to assess confidentiality risks, but part of our ignorance at the start of this research is that we do not know precisely for which class of companies and outsourcing relationships the technique will be effective and useful. So at the start of our research, the population is not known precisely. In the TAR approach proposed in this paper, we start from clear problem instances, i.e. we apply the technique in a client company of which we think it clearly falls inside the imperfectly known population.

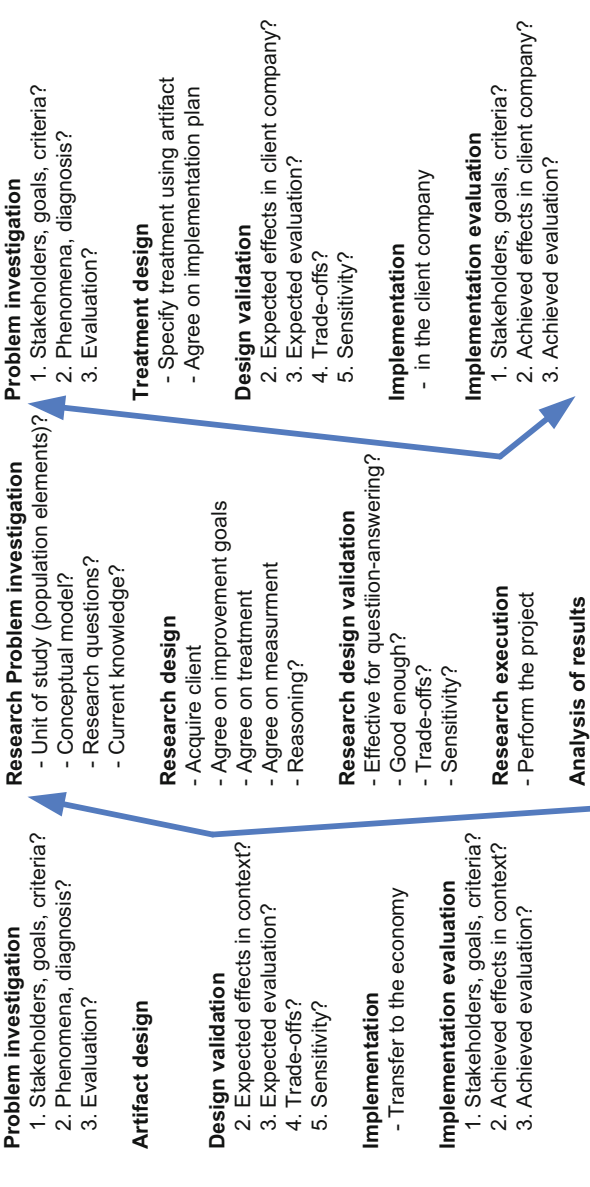

Fig. 6. The structure of technical action research

Research Design. Research design in TAR consists of acquiring access to a client company, agreeing on an improvement goal for the client cycle, agreeing on what the researcher will do for the company and on how the researcher will collect data. In TAR, data may be collected by means of a researcher's diary, interviews with stakeholders in the client company, logs of tools used in the client cycle, etc. The researcher should also determine how she will reason about these qualitative or quantitative data. How will the results of interviews be coded? How will the researcher reason from the data about a single case to claims about the entire population? Such claims are full of uncertainty and are fallible. The entire TAR exercise is based on the assumption that what the researcher learns in this particular case, will provide lessons learned that will be usable in the next case [22,23,24]. And so during research design, the researcher should think about how she wants to draw lessons learned that will be applicable to other cases too, even if they are full of uncertainty.

Research Design Validation. Research design validation is best construed as a risk assessment of not being able to answer the research questions if the researcher executes this research design. For example, will the measurement plan yield enough data to be able to answer the research questions? Is the client company representative in the relevant way of other companies with similar problems? The answers to questions like these may motivate the researcher to adapt her research design to reduce the risk of not being able to answer the research questions.

Research Execution. Research execution consists of the execution of the client cycle, part of which is the operationalization of the treatment plan already agreed on in the research design. Here, resources, people, time and places have to be agreed on to perform the tasks of the treatment. The client cycle includes an evaluation with the client whether and to which extent the client's goals have been achieved.

Result Evaluation. After completing the client cycle, the researcher returns to the research cycle and analyzes the results. Observations are extracted from the raw data, possible explanations are searched for, research questions answered, and generalizations to other cases from the same problem class hypothesized. Limitations of these outcomes are stated explicitly, and the increment of knowledge achieved identified.

A major threat to validity of the answers to the research questions is that the researcher, who developed the artifact, is able to use it in a way that no one else can. This would imply a lack of generalizability to any case where other people than the researcher are using the artifact. This threat can be mitigated by teaching others to use the artifact so that they can use it to perform the client cycle. A second major threat is that stakeholders may answer interview questions in a socially desirable way or, in a variant of this threat, that the researcher herself interprets answers in a desirable way. This introduces a risk that positive

evaluations are of limited value, but it does not diminish the value of improvement suggestions and of observations of failures of the artifact. The threat can be mitigated by having others than the researcher do the data collection and coding.

Finally, the researcher identifies consequences for the top-level engineering cycle in which the treatment was designed for a problem class. For example, some elements of the technique may have turned out to be unusable or useless, and in any case the researcher may have acquired ideas for changes that would improve the artifact even further.

6 Discussion

6.1 Validations of TAR

TAR is an approach to validate new artifacts under conditions of practice. Validation research aims to answer effectiveness and utility questions about an artifact in context, and to investigate the robustness of the answers under changes of artifact (trade-off analysis) and changes in context (sensitivity analysis). What about the validation of the TAR approach itself? This too has been done by action research, where now the artifact to be validated is TAR, and it has to be validated in the context of scaling up some technique to conditions of practice.

TAR has been used in several research projects, three of which have been published. Morali and Wieringa [21] describe a project in which the researcher herself used a newly developed confidentiality risk assessment technique for outsourced IT management to do a risk assessment in two different client companies. Zambon et al. [25] describe a similar project, this time about techniques to assess availability risks in outsourcing, tested by the researcher at one client company. Engelsman and Wieringa [26] describe a project in which a technique to relate business objectives to enterprise architecture was used by enterprise architects in a large government organization to redesign their enterprise architecture in a traceable way.

Any claims that we make about the effectiveness and utility of TAR as a validation research method (design validation questions (2) and (3) of the engineering cycle) are subject to the same limitations as have been noted above for TAR: Maybe we are the only ones able to use TAR; maybe we have interpreted the data about its utility too favorably. The first threat has been mitigated by teaching TAR to Master's and PhD students, who then use it in their own research. Experience so far indicates that others can use TAR too.

We have found the use of TAR useful because it provides a structured checklist of things to do when validating artifacts in practice, and it does indeed allow us to find out how an artifact performs in practice, which, as stated in the introduction, is our goal. For researchers not interested in learning about how a technique, not yet transferred to the real world, would perform in practice, TAR would not be useful.

6.2 Rationality, Rigor and Relevance

TAR can teach us whether an artifact is applicable in practice, and by being used in one client company, it shows that the artifact is at least relevant to that company. However, relevance comes in degrees. The two companies where we used our confidentiality risk assessment technique found the confidentiality risk assessment results relevant for their goals, and they have used them. But they did not see enough of a business case to set aside resources to acquire tools and expertise to regularly use the technique. On the other hand, the government organization that used our enterprise architecture (re)design technique, invested resources to regularly use the technique.

Let us now return to Schön's idealizing assumptions of technical rationality, listed in section 2. To what extent does TAR depend on these? TAR assumes that the problem for which an artifact is designed, is not unique. It also assumes that there is a conceptual model of these problems, i.e. that the problems have been framed, and that all problems in the relevant class can be framed the same way. (It does allow that each of these problems can be framed in different ways too.) Both engineering cycles in TAR assume that agreement on goals can be achieved. And the basic assumption of validation research is that the world is stable enough to allow prediction of what would happen if the artifact were used. We conclude that TAR makes the assumptions of technical rationality identified by Schön [10].

To the extent that these assumptions are violated in a problematic situation, TAR helps us to find out whether or not this particular artifact can still be used in a particular client organization. But if all of these assumptions are violated, TAR is not applicable and other approaches should be searched, such as the combination of action research and design research in soft systems methodology proposed by Baskerville et al. [8].

6.3 Comparison with Related Work

Most approaches to action research follow the action research cycle proposed by Susman & Evered [2], which consists of diagnosing – action planning – action taking – evaluating – specifying learning. Figure 7 overlays this cycle with the multilevel structure of TAR.

This immediately makes clear that Susman & Evered's cycle is problem-driven: The cycle is triggered by a concrete need experienced by a particular client, and works bottom-up by iterating through the client cycle, to lessons learned that go beyond that particular client. Indeed, the cycle revolves around a client-system infrastructure in which the researcher and the client have built a mutual relationship of helping and learning. This contrasts with TAR, in which the researcher has identified a *class* of problems, and aims to develop an artifact to mitigate those problems. Validation of one artifact will require more than one TAR project, each for different clients. A client-researcher relationship needs to be built up for every separate client cycle. Action Design Research (ADR) introduced by Sein et al. [9] also is problem-driven and aims to build design principles based on iterative client cycles for the same client.

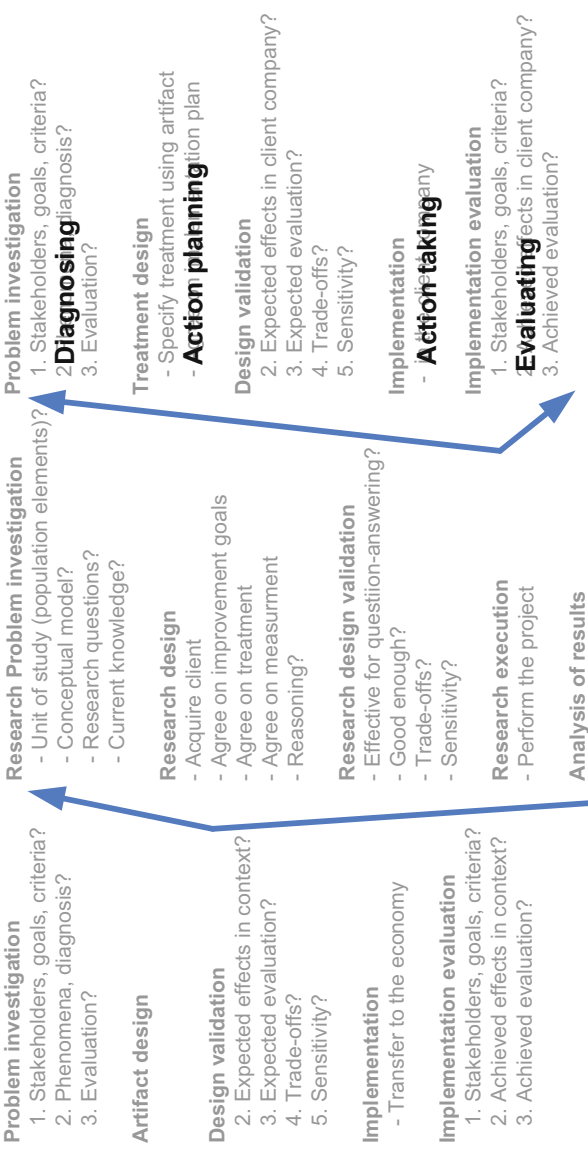

Fig. 7. The action research cycle of Susman & Evered [2] overlaid on TAR

TAR satisfies the principles of canonical action research (CAR) defined by Davison et al. [27], except that CAR assumes a problem-driven approach, whereas TAR follows an artifact-driven approach. CAR has five principles.

- For each client cycle, TAR requires a client-researcher agreement (principle 1 of CAR).
- TAR follows a cyclical model (principle 2) in two senses: The client engineering cycle can be performed iteratively for one client, and also the researcher's cycle can be performed iteratively. In each iteration of the researcher's cycle, the researcher performs a validation at a different client, and the researcher may redesign the artifact based on lessons learned so far.
- In this paper we have not discussed the use of theory (principle 3). However, specifying a hypothesis about the effect or utility of an artifact requires theory, and gaining experience in one or more client cycles can justify adaptations to theory. We will elaborate on generalization from action research in future work.
- TAR is built on the principle of change through action (principle 4) in two ways: the client's context is improved in a client cycle, and the artifact is improved in the researcher's cycle.
- The principle of learning through reflection (principle 5) is realized by inserting the research cycle between the two engineering cycles. Reflection on the client cycle consists of analyzing the results of the client cycle with a view to answering validation research questions, and drawing conclusions from this about what we have learned about the use of the artifact in context, and about possible improvements of the artifact.

A major problem untouched by this paper is the role of theory in action research. What role does theory have in designing improvements of artifacts, designing an action research project, and in drawing lessons learned from a client cycle? Can we generalize from a single case using, for example, reasoning by analogy? These questions are topics of our current research.

Acknowledgments. The authors wish to thank the anonymous referees for their stimulating comments on an earlier version of this paper.

References

1. Hevner, A., March, S., Park, J., Ram, S.: Design science in information system research. MIS Quarterly 28(1), 75–105 (2004)
2. Susman, G., Evered, R.: An assessment of the scientific merits of action research. Administrative Science Quarterly 23(4), 582–603 (1978)
3. Lewin, K.: Action research and minority problems. Journal of Social Issues 2, 34–46 (1946)
4. Baskerville, R.: What design science is not. European Journal of Information Systems 17, 441–443 (2008)

5. Järvinen, P.: Action research is similar to design science. Quality and Quantity 41(1), 37–54 (2007)
6. Lee, A.: Action is an artifact: What action research and design science offer to each other. In: Kock, N. (ed.) Information Systems Action Research: An Applied View of Emerging Concepts and Methods, pp. 43–60. Springer (2007)
7. March, A., Smith, G.: Design and natural science research on information technology. Decision Support Systems 15(4), 251–266 (1995)
8. Baskerville, R., Pries-Heje, J., Venable, J.: Soft design science methodology. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST 2009, pp. 9:1–9:11. ACM Press (2009)
9. Sein, M., Henfridsson, O., Purao, S., Rossi, M., Lindgren, R.: Action design research. MIS Quarterly 35(2), 37–56 (2011)
10. Schön, D.: The Reflective Practitioner: How Professionals Think in Action. Arena (1983)
11. Cartwright, N.: How the Laws of Physics Lie. Oxford University Press (1983)
12. Cartwright, N.: The Dappled World. A Study of the Boundaries of Science. Cambridge University Press (1999)
13. McMullin, E.: Galilean idealization. Studies in the History and Philosophy of Science 16(3), 247–273 (1985)
14. Boon, M.: How science is applied in technology. International Studies in the Philosophy of Science 20(1), 27–47 (2006)
15. Laymon, R.: Applying idealized scientific theories to engineering. Synthese 81, 353–371 (1989)
16. Küppers, G.: On the relation between technology and science—goals of knowledge and dynamics of theories. The example of combustion technology, thermodynamics and fluid dynamics. In: Krohn, W., Layton, E., Weingart, P. (eds.) The Dynamics of Science and Technology. Sociology of the Sciences, II, pp. 113–133. Reidel (1978)
17. Wieringa, R.: Relevance and Problem Choice in Design Science. In: Winter, R., Zhao, J.L., Aier, S. (eds.) DESRIST 2010. LNCS, vol. 6105, pp. 61–76. Springer, Heidelberg (2010)
18. Vincenti, W.: What Engineers Know and How They Know It. Analytical Studies from Aeronautical History. Johns Hopkins (1990)
19. Wieringa, R.J.: Design science as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, pp. 1–12. ACM, New York (2009)
20. Van Strien, P.: Towards a methodology of psychological practice: The regulative cycle. Theory & Psychology 7(5), 683–700 (1997)
21. Morali, A., Wieringa, R.J.: Risk-based confidentiality requirements specification for outsourced it systems. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE 2010), Sydney, Australia, Los Alamitos, California, pp. 199–208. IEEE Computer Society (September 2010)
22. Lee, A., Baskerville, R.: Generalizing generalizability in information systems research. Information Systems Research 14(3), 221–243 (2003)
23. Seddon, P., Scheepers, R.: Other-settings generalizability in IS research. In: International Conference on Information Systems (ICIS), pp. 1141–1158 (2006)

24. Seddon, P., Scheepers, R.: Towards the improved treatment of generalization from knowledge claims in IS research: drawing general conclusions from samples. European Journal of Information Systems, 1–16 (2011), doi:10.1057/ejis.2011.9
25. Zambon, E., Etalle, S., Wieringa, R.J., Hartel, P.H.: Model-based qualitative risk assessment for availability of IT infrastructures. Software and Systems Modeling 10(4), 553–580 (2011)
26. Engelsman, W., Wieringa, R.: Goal-Oriented Requirements Engineering and Enterprise Architecture: Two Case Studies and Some Lessons Learned. In: Regnell, B., Damian, D. (eds.) REFSQ 2011. LNCS, vol. 7195, pp. 306–320. Springer, Heidelberg (2012)
27. Davison, R., Martinsons, M., Kock, N.: Principles of canonical action research. Information Systems Journal 14, 65–86 (2004)