Writing a Report About Design Research

Roel Wieringa

University of Twente, the Netherlands

16th February 2007

1. Introduction

This short note provides guidelines on how to report about design research. It is intended as a guide for students who need to write a paper about a design research assignment, e.g. in the ARE course. It can also be used to report about Master's projects or even about Ph.D. projects.

The most difficult part of writing the report—and indeed of doing the research—is figuring out what problems you need to solve. Is it a practical problem or a knowledge problem? How many problems are there? How is your answer evaluated? Section 2 treats the problem of the problem, and it is the largest section. Once you understand the problem to be solved, the rest is relatively easy. Sections 3 and 4 describe how you report about your solution to a practical problem and to a knowledge problem, respectively. The subjects treated here are treated more elaborately in the course Problem Analysis and Solution Requirements (PASR).

2. Knowledge Problems and Practical Problems

The basic distinction to make is that between knowledge problems and practical problems.

A knowledge problem is a difference between what stakeholders know about the world, and what they would like to know.

Stakeholders in a knowledge problem want to know something: For example, which products are currently on the market, and what are their performance characteristics; what are the reported bugs in this program; what is the cause of this bug; what is the architecture of this system; what are the interface requirements of that system; etc.

There are two guidelines to recognize a knowledge problem:

1. ***Knowledge problems can always be reformulated in journalistic form,*** i.e. as if you were a journalist reporting about facts; or as if you were a detective investigating the crime scene. If you think of yourself as a detective, it is clear that you don't want to disturb anything that might cause you to give a wrong answer to the knowledge problem. You try not to interfere with the evidence, and to leaver the crime scene in a state as if you had not been there. This is strictly spoken not possible, for everyone, even a detective, leaves traces. But it is an excellent ideal to strive for when answering a knowledge problem.

Knowledge problems can be restated in one of the following forms:

- What was the case? What happened?
- Where was it the case? Where did it happen?
- When was it the case? When did it happen?
- Who was involved?
- Why was it the case? Why did it happen? What were the reasons for actions, or the causes of events?

These are journalistic questions. Example:

- How do software architects in company X ensure their software is maintainable?
    - The architectural decisions have been made (and still are being made) in company X. You will probably investigate past decisions; but depending on your research design you may also act as participant observer in a current project, in order to observe in real life how these decisions are currently being made.
    - The decisions have been made in company X (*where*) in several completed projects (*when*).
    - The architects are involved, but possibly also analysts, builders, maintenance personnel, customers, managers, etc.
    - Why are they making these decisions? Reasons may be experience, habit, literature, patterns, prejudice, standards, company rules, etc.

2. ***The answer to a knowledge problem is judged by only one criterion: Is it true or false?***

Example:

- Have these architectural decisions really been made?
- Is it true that they were made in this company (not in a subcontracting company) and in these projects (and not in other projects)?
- Are these really all involved actors (or did we miss anyone?) Are they really involved (or are they only saying so)?
- Did we find the real reasons for actions (or did we misunderstand) and the real causes of events (or are the real causes hidden for us)?

Usually there are qualifications to the answer, e.g.

- I found this to be true in cases A, B and C; I cannot make a claim about other cases; I do believe it is true in other cases too, because … (some argument)… but I have not collected evidence that this is, in fact, the case.
- I found this to be true in X% of the cases investigated.
- I found this to be true in these cases, but there are some threats to the validity of this claim, namely …. (discussion of threats to validity)…

A practical problem is a difference between the way stakeholders experience the world and the way they would like to experience it.

Stakeholders in a practical problem want to change the world. They may want to update their software, change their maintenance procedures, improve their architecture decisions, improve customer relationships, increase their market share, cut costs, etc.

There are two guidelines to recognize practical problems.

1. **A practical problem can always be reformulated in engineering form**:
    - How to do X
    - Do X

In the first case, we have a *design problem:* the solution to a design problem is a description of how to do X, also called a specification of X. For example, a doctor may describe a treatment plan, a bridge engineer may specify how a bridge is to be built, and you may specify a plan for a holiday you want to take soon.
In the second case, we have an *implementation problem.* The solution to this problem is an implementation of X. For example, a treatment plan is executed, a bridge is built according to specification, you go on holiday as planned.

If you are solving a practical problem, you are like a doctor helping a patient. You figure out what to do (medicine? Which one? Referral to specialist?) and then you do it. The goal is to change the state of the patient. Unlike the detective, you do want to change the world, but in a controlled way.

Example:
Which architecture decisions improve maintainability of software? The design form of this question is

- How to make architectural decisions that improve maintainability?

If we have a specification X of these decisions, then the corresponding implementation question is:

- Implement the set of architectural guidelines X in our organization.

2. **The answer to a practical problem is not true or false but it is useful or useless (and all shades in between).**
    - The answer to a practical problem is judged by criteria that have been identified by problem analysis. In general, these criteria are different for different problems. These criteria operationalize the usefulness of a solution: The better a solution satisfies these criteria, the more useful the solution is.
    - In general, if C is the set of criteria for solutions to a particular practical problem, then for any proposed solution we can ask ``How well does this solution satisfy C?'' For example, how well does our specification of X satisfy the criteria? Or how well does our implementation of X satisfy the criteria?
    - Usually, there is not one criterion but many. And usually, it is not possible to satisfy all criteria equally well.
    - Even if there is only one criterion, there are usually many possible solutions to a practical problem, that satisfy this criterion better or worse. How many solutions there are, depends on our creativity.
    - If there is more than one criterion, which is the usual case, then different solutions score differently on each criterion. For example, one architecture decision may increase maintainability and may also increase initial implementation cost.

Importance of the distinction.

The distinction between knowledge problems and practical problems is important because the problem-solver must do different things to solve each of them, and must evaluate his or her answer according to criteria that are different for knowledge problems and for practical problems.

- To solve a knowledge problem, the problem-solver gathers facts, reads the literature, performs research, asks experts, etc. while trying not to disturb the object of knowledge.
- To solve a practical problem, the problem-solver specifies a change to be applied to the world, and then applies the change. This changes the state of the world.

In addition to the difference in problem-solving approach, there is a difference in the way solutions are evaluated.

- The answer to a knowledge problem is evaluated by only one criterion: Is it true?
- The answer to a practical problem is evaluated to a problem-dependent criterion: Does it bring us closer to the (problem-specific) goal?

The following table summarizes some important distinctions between knowledge problems and practical problems.

Knowledge problems	**Practical problems**
Find truth	Do something useful
Avoid interference with the world	Interfere with the world
Goal is knowledge	Goal is change in the world
Any change in the world is a side-effect (to be minimized)	Any knowledge gained is a side-effect (to be cherished)
Ethical rules not applied to the answer (unpleasant, revolting and offensive truths are still truths)	Ethical rules are applied to the answer (If you change the world you are accountable for what you do)

Difficulty of the distinction.

The distinction between knowledge- and practical problems is difficult for two reasons.

1. ***People confuse the two problems.***
   This is a recursive reason: Because others are confused, you will get confused too. Many practical problems are stated in the form of knowledge problems and vice versa. Examples:
   - Do a quick scan of the software engineering processes of company X.
     Strictly spoken this is indeed an implementation problem: Do the quick scan. However, it is more enlightening to treat the *real* problem as a knowledge problem, despite its initial word “do”. The problem owner wants to know what the status of SE processes in X is, and tells the problem-solver to use a particular knowledge acquisition tool for it, namely a quick scan.
   - What is an IT architecture that would support our e-business?
     This is a practical problem of the how-to-do kind. There is no IT architecture somewhere in the world that has to be investigated. The problem-solver is not like a detective entering a crime scene to see what is there; he or she is like a doctor figuring out what to do to heal this patient. Treating this like a knowledge problem leads to nonsense approaches like formulating a hypothesis and checking whether it is true, and forgetting to identify criteria or evaluating the usefulness of the solution.
2. **Problems are composed of subproblems that are of a different type.**
   This makes the distinction really hard. Each problem has subproblems. To solve the knowledge problem ``what is the state of our SE processes?'' the problem-solver has to *do* something, e.g. to perform a quick scan, or to do participatory research, do a case study, etc. In general, to answer a knowledge problem, we may have to design a research plan, and then execute the research plan. Designing and executing a research are practical problems.

   Conversely, every practical problem contains knowledge problems. To design a research plan, the problem-solver must know which research techniques there are; must learn more about the subject to be studied and about the availability of data about the subject. Can he or she use questionnaires? Is case study material available? etc.

More generally, to answer a design problem—a practical problem of the how-to-do kind—the problem-solver needs to

- investigate the problem (itself a knowledge problem: who are the stakeholders, what are their goals, which criteria are relevant for the solution?),
- propose possible solutions (this contains several subproblems itself) and
- validate those solutions (again a knowledge problem: does our solution satisfy the criteria?).

Proposing possible solutions contains itself knowledge problems (which solutions already exist?) but also practical problems (invent a novel solution; or adapt an existing solution to this problem).

Any practical problem contains knowledge problems and practical problems as subproblems, and vice versa. At first sight, this should not confuse us. If a red box R contains a blue box, we do not conclude that R is blue: No, it *contains* a box that is blue, but it is itself still red. And if my house contains a wooden floor, this does not imply that my house is made of wood. In the physical world, containment does not confuse us. But in the world of problem-solving, we are easily confused by subproblem hierarchies.

3. Reporting About Practical Problems

Practical problems to be solved by our students usually concern software technology: software, methods, tools, notations, techniques, procedures, and models to be used in software development or maintenance (Pfleeger).

A report about solving a practical problem should have the following structure. Typically, a student project covers only the first three of these topics, or even less, depending on the time available. However, if validation and/or implementation are not performed, the student may still be able to offer advice about how to perform validation and implementation.

1. **Problem investigation:** What is the problem?
   Examples of questions to be asked here are the following. Note that they are all knowledge questions.
   - Who are the stakeholders?
   - What are their goals?
   - What are the problematic phenomena, that make the stakeholders want to solve the problem?
   - What are the causes of these phenomena?
   - What are the criteria to be applied to the solution? These criteria are to be derived from the stakeholders' goals as well as our diagnosis of the problem. Often, these criteria are called *quality attributes*, e.g. how expensive, easy to implement, quickly available, robust, reliable, understandable, maintainable, complex, simple, fast, efficient, interoperable, portable, usable, secure, safe etc. must the solution be?
   - What are the priorities of the relevant criteria? How much does a stakeholder win (or lose) if a criterion is satisfied (or not satisfied)?
   - How urgent is the problem? Is there a deadline, and how hard is it? Or is there a periodic deadline (solution to be delivered before the end of this month, or before the end of next month, etc.). Or is it now or never? It may come as a surprise to some, but some deadlines are *not* negotiable.

2. **Solution design:** Which solution alternatives are available?
   In many cases the student does not have to invent something new but make an inventory of solutions available on the market or in the company, or described in engineering literature. This is a knowledge problem. Alternatively, the student can assemble a solution from known elements. That is a knowledge question (which elements are available) and a creative problem (how to assemble them?).
3. **Solution validation:** Which alternative best solves the problem?
   This subproblem itself contains two knowledge problems:
   - *What are the properties of the solution(s)?*
     To investigate the properties of a solution not yet implemented, we can do a mathematical analysis (e.g. estimation of computational complexity), make a model (throw-away prototype) and do experiments with it, collect experiences of others with the same solution, ask experts' opinion about the specified solution, etc.
   - *How do the solutions score on the criteria?*
     Once we know (with a certain degree of uncertainty) the properties of the solutions that we specified, then we can rank the solutions on the criteria. How expensive, maintainable, complex, simple, fast, interoperable, portable, usable, secure, safe etc. is our solution?

     Additional questions that can be answered under this heading are
   - *How sensitive is the solution to changes in the problem?*
     Answering this question is called *sensitivity analysis.* For example, what happens if the number of users or amount of data or number of software system grows? How scalable is our solution?
   - *Which solution is to be preferred, according to which criteria?*
     This is often called *trade-off analysis.* If you have found three acceptable solutions, the usually none of them scores well on all criteria. What are the trade-offs involve in choosing one over the other?
   - *Are we sure we can apply our conclusions to real solution behavior?*
     For example, if we validated a new method by discussing it with stakeholders and experts, and gaining their approval, how certain can we be that the method will work in realistic situations? Or If we investigated solution properties by modeling (e.g. a throw-away prototype), how certain are we that we can infer from this what the properties of a real solution will be? This is sometimes called *similarity analysis:* How similar is the model to the real thing?
   - *Whose goals are achieved by this solution?*
   - *Which stakeholders win most and who lose most with a solution?*
   - *What is the improvement, overall, with respect to the current situation?*
   - *What is the support for this solution in the organization?*
4. **Solution implementation**
   This starts a new round of practical problems. We have a specification of a solution and must implement it. Now we have to solve additional practical problems:
   - How many resources are needed?
   - Which significant milestones can be defined?
   - Which strategy do we apply, big bang or iterative? If iterative, do we work in a risk-driven or in a benefit-driven way?
   - What are the risks, anyway?
   - etc.

   All of these are lower-level practical- and knowledge problems within the implementation subproblem of your original practical problem.
5. **Implementation evaluation:** How well did it solve the problem?
   This is a classical knowledge problem investigated by management scientists: How does some implemented solution perform? Books about business

research methods (e.g. Cooper & Schindler) provide plenty of examples of how to do this kind of research.

4. Reporting About Knowledge Problems

Some knowledge problems may be easy to answer by taking relatively simple actions:

- ask an expert
- read the literature
- search the Web.

A common feature of these problem-solving strategies is that you trust the source of your answer. There are some justifications for this, but we need to be careful.

- Experts are trusted because they have a reputation to lose. However, some experts may be more expert at maintaining their reputation than at maintaining their knowledge level.
- Scientific books and papers are always peer-reviewed, so that others have already judged their quality before you read the document. However, peers may all come from one in-crowd, and may be prejudiced. In some cases, the peer group may be a mutual admiration society: ``If I praise you in public, then you praise me in public.''
- The web is an unlimited swamp of texts without serious quality control. Any one can put documents on the web. Current technology allows you to let documents grow to any size and duplicate them without restriction without significant effort. Companies distribute information that puts them in a favorite light. Conspiracy theorists, religious fanatics, Elvis believers, politicians and terrorists all have found the web as a place to crank out documents that serve their goals. Publishers manipulate links so that newly published books end up on top in search results (not as sponsored links). The Wikipedia has some public quality control mechanism, but nevertheless, politicians have found out how they can favorably rewrite their biographies in Wikipedia, companies describe their products in Wikipedia entries in seemingly neutral language (check out the entry for RUP) and attempts by some people to actually correct their own biography in the Wikipedia are edited away by other people who think they know better. In addition, documents on the Web may disappear, be relocated, or change when you are not looking, so that if ever you refer to a document on the web in your report, you must include the date and time of access. Your reader may not be able to find the document anymore, or may find a document that has changed since you last accessed it. The web is a great source of information—for example, the UT provides access to most relevant publications of IEEE, ACM and Springer, which save many of us an enormous amount of time. But don't believe everything you find there. And don't use documents found on the web as if they were scientific, peer-reviewed references, unless they *are* scientific, peer-reviewed documents (such an IEEE journal that is available on-line).

The basic rule for solving knowledge problems is this:
Check every possible way in which your answer could be false.
Check all the ways in which the expert, book, paper, or web document that you consulted could be wrong. Are the observations on which they base their knowledge repeatable? Are they relevant for your knowledge question? The book by Wohlin et al. listed in the references is an excellent source for a list of all ways in which your best answer to a knowledge question can still be false. Check under the heading “threats to validity”.

To rub the point home: Conspiracy theorists, religious fanatics, Elvis believers, politicians, terrorist and those who have been abducted by aliens, are not in the habit of figuring out every possible way in which they could be wrong; they do not prefer to put their answers to the test; they do not work through peer-reviewing (but they resemble mutual admiration groups); on the contrary, many of them tend to become quite upset when you do not believe what they believe. The basic attitude to be learnt when answering knowledge problems with full intellectual integrity is to learn to live with uncertainty:

- "The scientist has a lot of experience with ignorance and doubt and uncertainty, and this experience is of very great importance, I think.... in order to progress we must recognize our ignorance and leave room for some doubt.''[1]

This also happens to be the central competence of great designers;

- "I have, perhaps, one real talent: that is that I don't mind at all living in the area of total uncertainty.'' Designer Ted Happold. [2]

If there is no expert or document that you can trust to answer your knowledge question, then you must perform research to answer it. There is a large variety of research methods, such as laboratory experiments, simulations, field experiments, field studies, case studies and action research methods. This is not an exhaustive list. The set of research methods is only bound by our own creativity. More information about research methods and their structure is given in the PASR course. In addition, there are many books written on research design; the book by Cooper & Schindler listed in the references is a good example.

A report about research should have the following structure.

1. **Research problem investigation**
   - What research problem do we have? What do we want to know?
   - Why do we want to know this? Is there a practical problem that provides the context for this particular knowledge problem? Are investigating a practical problem, or validating a proposed solution, or evaluating an implemented solution?
2. **Research design**
   - How are we going to answer the research problem? Do we choose one of the existing research methods, or do we invent another method for this particular problem?
   - What, in more detail, will we do to answer the research problem? The answer to this question is often called a research protocol.
3. **Research design validation.**
   - Why should this research protocol answer the research questions we have?
   - In what ways could we fail to answer the question if we follow this protocol?
4. **Research**
   Performing the research as planned may lead to a new set of subproblems to be solved, e.g. to design a questionnaire, to get access to a case study, to gain sufficient trust to perform action research, to get experimental subjects, to set up a simulation, etc.
5. **Evaluation**
   - What is the answer to our research questions? Do we need a conceptual or statistical analysis of the answers?

[1] R.P. Feynman. *What Do You Care What Other People Think?* Bantam, 1988. P. 245.

[2] Quoted in N. Cross. *Engineering Design Methods: Strategies for Product Design*. 2nd edition. Wiley, 1994. P. 17.

- In which ways can our answers be wrong? E.g. are our observations and analysis reliable? What are the threats to validity of our answers? Applicability of our answer to our own question is called *internal validity*.
- What is the explanation for our results? Can we explain the results by pointing at underlying causes? Are there other explanations?
- Can we generalize our answers? Are they applicable to other cases, similar but different to ours? Applicability to other cases is called *external validity.*

5. More Words of Advice

1. Write for the reader. Who is your reader? What is he/she interested in? Why should he/she read your report? What information does he/she need to understand it?
2. Can you use shorter words? Can you omit words? Can you use shorter sentences? Can you omit sentences? Can you omit paragraphs, sections, or chapters? For every word in your text: Why is it there? Modern word processing allows you to grow a text to any size. This was also possible in the days of typewriters but back then, it was a lot more work to increase the size of your text. You could easily add words to the end of your text, but adding them in the middle was difficult. So: Can you reduce the size of your text? In general, the shorter the text, the easier it is for the reader—but the more difficult it is to write. Conversely, the longer the text, the easier to write, and the more boring it is to read. Only those who are forced to do so, will read it; and some of those will read rather diagonally, or only read the headings. Who do you want to please: Yourself or your reader?
3. Are your diagrams readable? Modern word processing allows you to make a diagram (say, in Visio) and then include it in your text. Your word processor makes it very easy for you to scale the diagram down to a size that fits your line length. For example, Word does it for you while inserting, and many authors are not even aware that their diagrams are scaled. But the result is easily recognizable:
   - A diagram with letters in point size 4 or smaller, unreadable for anyone over 24 years of age.
   - A diagram that looks like spaghetti, or rather like a bundle of hair pulled out of a hair brush
   - Or a diagram with point size 4, and 95% white space. The white space shows that there is plenty of room to make the letters readable. The author simply did not bother to do so.

   If a diagram is not readable, it should be omitted. A speaker who cannot be heard; a television station that broadcasts black images; a manager who does not manage, all because they found it easier this way, should be swept off the stage, put out of business, and fired. The guideline is clear: Do not scale a diagram. Draw it in exactly the size at which it must appear in your document.

6. References

D.R. Cooper, P.S. Schindler.
Business research methods, 8th edition. Irwin/McGraw-Hill 2003.

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Weslén.
Experimentation in Software Engineering: An Introduction. Kluwer 2002.