Beyond Herding Cats: Aligning Quantitative Technology Evaluation in Large-Scale Research Projects

Michael Kläs[1], Thomas Bauer[1], Ubaldo Tiberi[2]

[1] Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
{michael.klaes,thomas.bauer}@iese.fraunhofer.de

[2] Volvo Group Trucks Technology, Advanced Technology & Research, Göteborg, Sweden
ubaldo.tiberi@volvo.com

Abstract. A large-scale research project involving many research and industry organizations working on a common goal should be an ideal basis for profound technology evaluations. The possibility for industrial case studies in multiple settings ought to enable reliable quantitative assessment of the performance of new technologies in various real-world settings. However, due to diverse challenges, such as internal agendas, implicit constraints, and unaligned objectives, leveraging this potential goes beyond the usual challenge of cat-herding in such projects. Based on our experience from coordinating technology evaluations in several research projects, this paper sketches the typical issues and outlines an approach for dealing with them. Although new in its composition, this approach brings together principles and techniques perceived to have been useful in earlier projects (e.g., cross-organizational alignment, abstract measures, and internal baselining). Moreover, as we are currently applying the approach in a large research project, this paper presents first insights into its applicability and usefulness.

Keywords. Empirical study, Multiple case studies, Baselines, GQM+Strategies

1 MOTIVATION

Besides bringing together specialists working on one specific topic, a key advantage of large-scale research projects is the availability of a large number of "customer" companies that provide an opportunity to perform empirical evaluations under real conditions.

In software engineering, *case studies* are a commonly used empirical method for investigating phenomena in such real-world settings. They contribute to a better understanding of software engineering technologies and their practical performance [9]. This is especially important when new software engineering technology is to be introduced in a company, because observations collected in vitro, for instance via controlled experiments, do not necessary scale up to realistic applications in a company setting, and introducing a new technology not sufficiently evaluated beforehand can have serious consequences [11].

In recent years, a second term, action research, has been used increasingly in the context of software engineering studies investigating the effect of new technologies and process improvement activities. Runeson et al. use this term to separate (action research) studies investigating a process improvement activity from case studies investigating the current state before or after an improvement action [16]. However, based on their literature survey on action research, Santos et al. conclude that action research is rarely used in software engineering [17], which matches our observation that most studies conducted in the context of process improvement activities are published as (industrial) cases studies. In this paper, we therefore do not differentiate between action and case study research when talking about technology evaluation and subsume both methods under the more common term *case study* (CS).

We agree with Kitchenham et al. [9] that CSs are not only needed in software engineering to investigate "how" and "why" questions but also to check whether a new technology is better in a given real-world context and, if so, how strong this improvement effect is. However, some inherent characteristics of CSs [16] limit their validity in such applications. First of all, since most CSs represent only one 'case', statistical tests as a typical means for assuring *conclusion validity* can rarely be applied. Moreover, the fact that a CS is conducted in a real-world setting limits the degree to which potential confounding factors can be controlled (*internal validity*). Finally, since a CS usually considers one specific project in one company, it is difficult to determine the degree to which study findings can be generalized to other situations (*external validity*).

One solution for addressing these weaknesses, at least partially, is to *conduct multiple CSs*, which would provide more cases and a more diverse context. However, such multiple CS research – an example is presented by Mohagheghi and Conradi [13] – is still rare. Reasons that typically inhibit such studies are the limited number of available cases and the difficulty to get data measured consistently for available cases that can be aggregated into more general statements.

An alternative might be found in performing a *meta-analysis* for existing case studies; however, a "meta-analysis is seldom possible […] because there are often insufficient primary studies" [8]. For instance, only four papers (5%) in the area of model-based testing (MBT) present industrial applications [15]. In consequence, Davis et al. (like many others) exclude CS results from their meta-analysis [4].

On the other hand, *large-scale research projects* seem to provide a good opportunity to perform a higher number of CSs addressing one topic. However, in reality, various reasons (which we will discuss in more detail in the following section) limit the practicability of getting combinable CS-based evaluation results in such projects. Therefore, the majority of projects provide survey-based summarizing results in the best case (e.g., [3][14]).

The aim of this paper is to support persons coordinating and setting up empirical technology evaluations in large research projects by presenting an approach that addresses the most common challenges we and other researchers have typically been faced with in the past. The intention of this paper is not to present a closed framework for CS-based research in the context of large-scale research projects. Rather, it contributes by highlighting major challenges that need attention, presents an approach

based on a set of mutually supporting recommendations, illustrates its application in an ongoing large-scale research project, and reports on our experiences.

The remainder of this paper is structured as follows: First, existing work and guidelines in the area are discussed (Section 2), followed by challenges that make technology evaluation in large-scale projects difficult and which motivate our work (Section 3). The approach and its application are presented in Section 4 and the paper closes with a summary and an outlook on future work in Section 5.

2 Related Work

Available guidelines for conducting CSs in software engineering are scientifically elaborated and well detailed but focus mainly on qualitative research [12] and researchers planning to conduct a single CS [16]. Thus, they address the challenges occurring in large-scale research projects only insufficiently.

Only few papers report on multiple CSs [13]. Moreover, they focus on reporting their specific results and not on providing guidelines for conducting such studies.

Besides the general CS guidelines for technology evaluation provided by Kitchenham et al. [9], we are aware of only one contribution that explicitly deals with the challenges of industrially based multiple CSs [1]. In contrast to our objective of technology evaluation, their guidelines focus on and were applied to explorative studies. In explorative studies, however, the challenges addressed in this paper such as baselining are of less importance and are thus not elaborately discussed or evaluated.

In education and social science, designs for multiple CSs (more formally called *multiple site, structured case study designs*) were developed, for instance, by Greene and David as an extension to the more traditional research strategies [6]. As in our case, their major intention behind conducting multiple CSs in a structured way was to increase the generalizability of their results. However, in accordance with the social science understanding of CS research, their focus is on collecting and combining qualitative data (such as information extracted from interviews). In consequence, they do not deal with issues occurring when defining measures and determining baselines, which are essential for quantitative technology evaluation.

A research direction that is related to this work since it is also driven by the goal of supporting quantitative technology evaluation across multiple independent cases is presented by Williams et al. [20]. They propose the concept of technology-dependent frameworks for industrial case study research. They instantiated their concept idea by developing a framework for evaluating extreme programming, which comprises a core set of metrics and context factors [19]. Although defining a fixed set of metrics seems appealing, in large-scale research projects we have observed two major issues with such an approach: (1) Since the metrics are predefined, they cannot be simply substituted by other metrics that would be easier to collect or more appropriate in a specific case; (2) the proposed set of metrics comprises metrics demanding absolute numbers regarding defects and productivity, which most companies are not willing to reveal due to confidentiality issues (see also challenges discussed in the next section).

3 Observed Challenges

This section summarizes typical challenges that can be observed during quantitative technology evaluation and sketches how they are addressed in our approach.

In order to enrich and back up our own opinions, we conducted a focus group meeting [10] where we identified challenges observed by the participants in the context of quantitative technology evaluation. Eleven researchers with practical experience in empirical research participated; most of them also had experience in organizing its application in large-scale research projects. In total, 17 challenges were mentioned; 16 of them could be clustered into four major groups: *organizational issues* (5), *collecting the right data* (4), *providing combinable data* (3), and *defining a baseline* (4). Since organizational issues deal with scheduling, distribution of efforts, and empowerment of roles, they have to be addressed in the organization of the project. The remaining three groups are easier to target with methodological solutions and are therefore presented in more detail:

1. *Collecting (the right) data* to quantify the obtained improvements is often a major challenge for several reasons. We will focus on the most critical ones: (1) Data collection has to be well *motivated*, especially in a large research project where potential CS providers cannot be forced to collect data but have to be convinced that they will personally benefit from such data collection. In our approach, we address this issue by providing the means for aligning measurement activities not only with the objectives of the research project but also with company-specific businesses goals. (2) In order to be useful, the measures have to be clearly defined and address the goals of the research project. In our approach, we define what data are relevant by developing consolidated goal and strategy graphs for the project objectives with abstract measures. Additionally, we develop examples of suitable CS-specific measurement plans and conduct workshops to teach goal-oriented measurement.
2. *Providing (combinable) data:* In order to get more general statements on the effects of new technologies, CS-specific measurement results have to be provided and combined. In large-scale projects, this is usually difficult due to data confidentiality and the different environments in which these data were collected. To assure that such data can be *combined* (at least at a certain level of abstraction), we define in our approach a common set of general but abstract measures that quantify consolidated project goals but can be implemented with different strategies and measurement plans. The issue that certain types of data (e.g., defect numbers, productivity data, etc.) can be collected but not *provided* to persons outside the organization due to confidentiality reasons is addressed in the definition of measures, which call not for absolute but only for relative improvement numbers.
3. *Defining a baseline:* In order to evaluate improvements resulting from a new technology, we need a baseline for comparison. Identifying an appropriate source for collecting baseline data is one of the most essential but also challenging tasks, since in large projects with one or two dozen CSs, researchers with an empirical background are usually not involved deeply enough in each CS context to guide and support baseline data collection. On the other hand, for the people responsible

for a CS, it is typically not clear where baseline data can be obtained, what the advantages and limitations of the different sources of baseline data are, and how they can deal with confidentiality issues if sensitive company data are involved. In our approach, we support CS providers by introducing the concept of *internal baselines* to deal with sensible company data, providing dedicated guidelines for baselining that show possible *sources of baselines*, and explaining the specific *advantages and drawbacks* of different baselining solutions.

4 Approach and Application

In the following, we will first sketch the overall approach, then briefly introduce the context in which we are currently evaluating the approach, and finally present the three major components of the approach. In its general procedure, our approach follows the quality improvement paradigm [1] used (at least implicitly) in most CS research and refined by related guidelines: At the beginning, the overall project- and CS-specific goals are consolidated, documented, and quantified via measures (characterize and set goals); then, the studies design is concretized, sources of baseline data are defined, and measures are operationalized via measurement plans (plan); next, baselining is performed, the CSs are conducted, and measurement data is collected (execute); finally, the collected data is analyzed, aggregated, and reported (analyze and package).

4.1 Application Context: MBAT

We applied the approach to the ARTEMIS project MBAT[1] (Combined Model-based Analysis and Testing of Embedded Systems). MBAT aims at facilitating a new leading-edge Reference Technology Platform in the European Union for the efficient validation and verification of software-intensive technical systems by using model-based technologies, focusing primarily on the transportation domain. The 3-year project is supported by 38 European partners ranging from large companies from the automotive, rail, and aerospace domains, via innovative tool vendors to leading-edge research institutes. The project with its structure, tasks, and activities is industrially driven by more than 20 case studies.

4.2 Goal Alignment and Measurement Plans

In order to identify relevant measures in a goal-oriented way and assure that the collected measurement data can be interpreted and aggregated across the boundaries of a single evaluation, we adapted GQM+Strategies (GQM+) [2], which was originally developed for clarifying and harmonizing goals and strategies across different levels of an organization with goal and strategy graphs and support monitoring the success or failure of strategies and goals by means of goal-oriented measurement.

[1] MBAT project website: www.mbat-artemis.eu

In our case of multiple CSs, where we apply GQM+ to a large-scale research project, we introduce four major layers, one with the CS provider's high-level goals and strategies, one with the overall (research) project goals, one with refinements of these through sub-goals, including corresponding strategies, and one with the CS-specific implementation of the addressed strategies and measures. Fig.1 illustrates these layers.

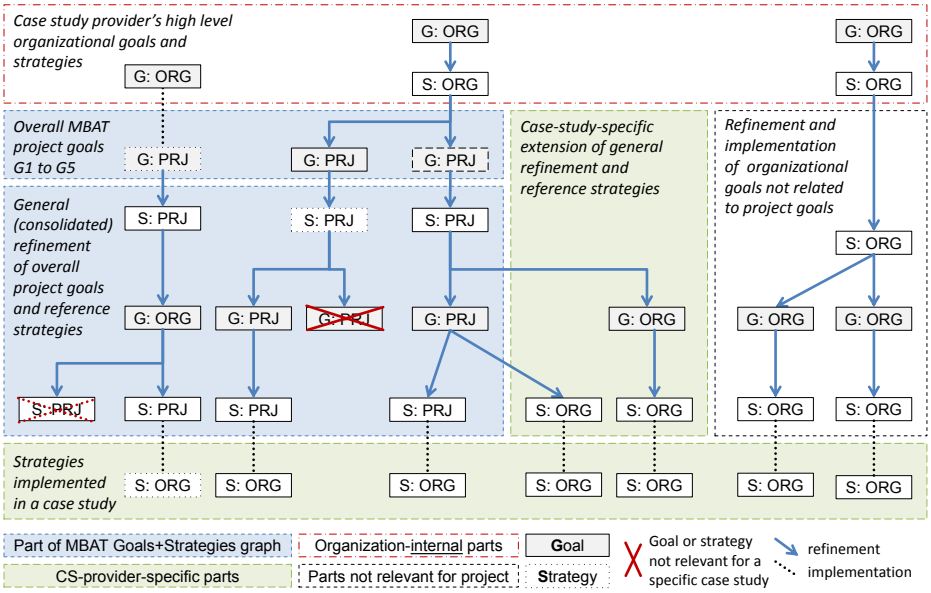

Fig. 1. Relationships between high-level organizational, general project, and case-study-specific goals and strategies

In the first step, the general overall goals are defined based on the project objectives, which can usually be extracted from the project proposal. However, these goals should be made more concrete with more specific sub-goals and strategies, which should obviously include the application of techniques or tools developed in the research project. This refinement needs to be done jointly between measurement and domain experts since the project proposal is usually not detailed enough and contains gaps and ambiguities regarding goals and strategies that need to be resolved. Typical cases are that strategies are mentioned without a clear link to the goal they address or vice versa – that goals are stated without any explanation of how exactly they are to be achieved.

Using a questionnaire, the initially refined goal and strategy graphs are evaluated by all CS providers to check for completeness, identify dispensable parts, and assure a common understanding in the consortium. The feedback is integrated into a consolidated version of the graphs. Fig. 2 presents an example of such a consolidated goal and strategy graph developed in MBAT.

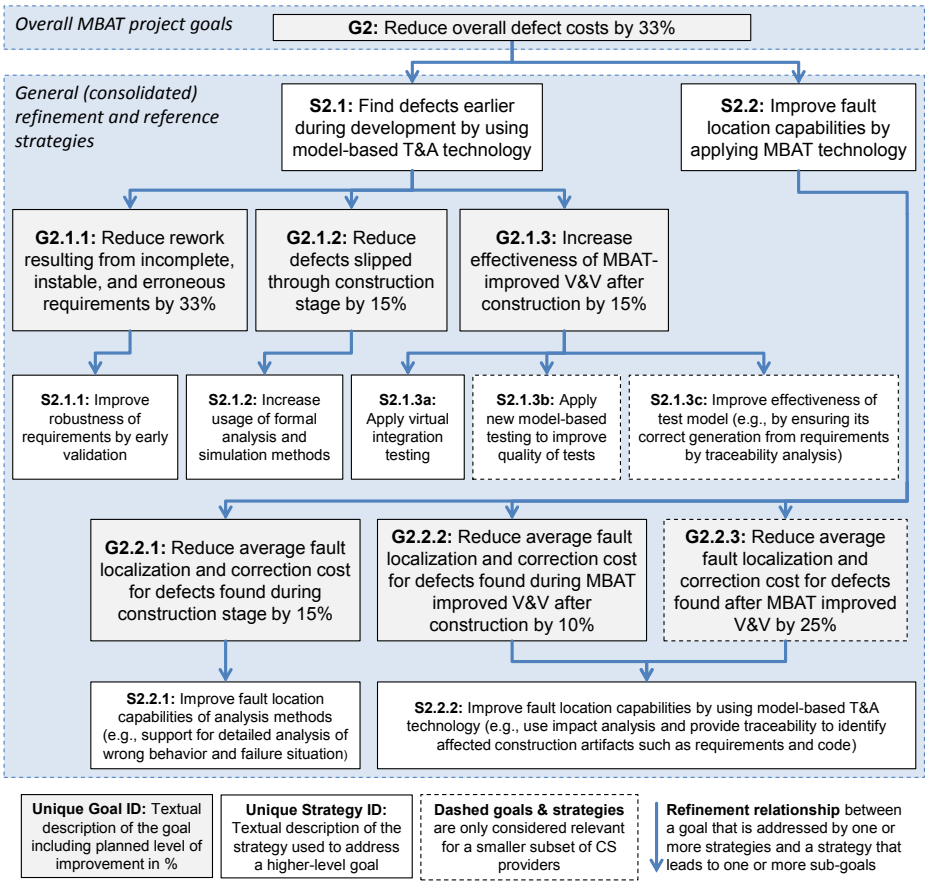

Fig. 2. General refinement of overall MBAT project goal G2

Not every general project goal has the same relevance for each CS provider. Moreover, in different CSs, different combinations of strategies (e.g., selections of technologies) can be used to reach the same project goal. Therefore, in the next step, the GQM+ approach is applied for each CS to identify a CS-specific subset of consolidated goals and align them with the company's *high-level organizational goals*. This allows the CS providers to show the relevance of the general project goals internally and motivate the collection of data. It also means that a CS provider will typically not collect measurement data for all but only for a selection of goals defined in the general Goals+Strategies graphs (which is illustrated in Fig. 1 by the crossed-out goals and strategies).

In the next step, specific measurement plans using abstract measures have to be defined for each CS. Fig. 3 illustrates the template used based on an actual measurement plan developed in the MBAT project.

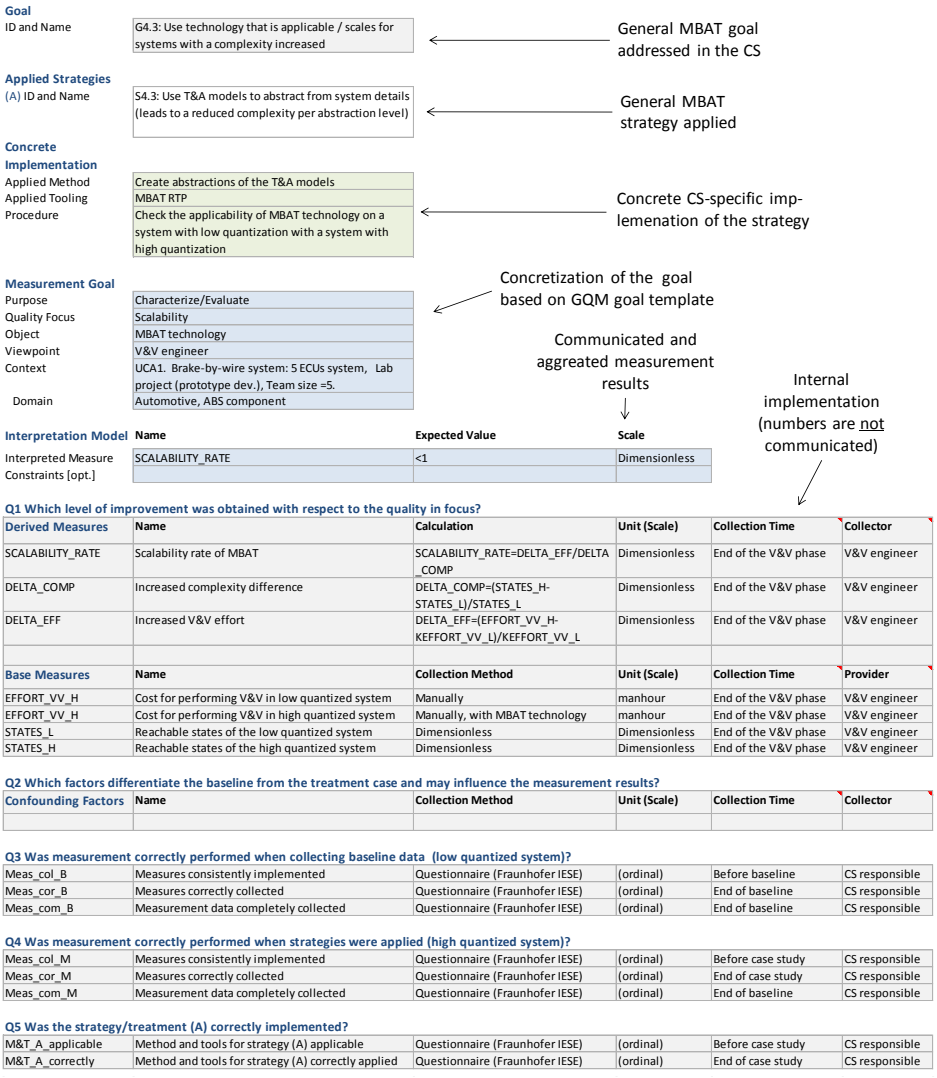

Fig. 3. Example of case study specific measurement plan at Volvo (G4.3)

Application experience in MBAT: Based on the business needs and objectives stated in the project proposal, we extracted five overall project goals for MBAT and developed a Goals+Strategies graph refining them:

- **G1** Reduce overall verification and validation costs by 20%;
- **G2** Reduce overall defect costs by 33%;
- **G3** Reduce total cost of ownership for development platform by 15%;
- **G4** Provide high quality in the face of increased complexity by 30%;
- **G5** Reduce time to market for embedded systems products by 20%.

The initially developed Goals+Strategies graphs including the identified goals, strategies, and abstract measures were sent to all CS providers as an Excel-based questionnaire asking

- which of the goals they plan to evaluate quantitatively in their CS,
- which of the strategies they plan to apply to reach these goals,
- which additional goals and strategies not mentioned yet are relevant for their CS,
- open questions and issues regarding goals, strategies, and abstract measures.

Based on the results, we identified 21 goals and strategies addressed only few or none of the CS. In addition, we obtained estimates for the expected relative improvements, which were also included in the consolidated goals. All 16 identified yet missing goals and strategies were integrated into the graphs. Ambiguous or difficult-to-understand descriptions were revised. Moreover, the survey results indicate that a CS addresses an average of 14 of the 29 goals. The resulting graph for G2 is shown in Fig. 2.

A two-day workshop format was used to teach the CS providers the fundamentals of GQM+ and goal-oriented measurement. The participants learned how to align their organizational high-level goals with the general MBAT project goals and specified them in CS-specific measurement plans. The workshop used a mix of presentations and exercise where the actual goals of the CS providers were addressed in moderated groups. The measurement plans were finalized by the CS provider offline with the support of a moderated online interest group and individual support by experts for conducting empirical studies. In order to illustrate the content of such measurement plans, the plan implementing G4.3 at Volvo is presented in Fig. 3.

4.3 Internal Baselining

In order to empirically evaluate the effect of a new technology in a CS, measurement results gained during the CS (when the new technology is applied in a particular project) are usually compared to so-called baseline data [5]. These baseline data are intended to represent the "current" status before the new technology was applied. For instance, if we collect data in a CS about the average effort for defining a test case because we introduced a new technology to reduce this effort, we also need to know the average effort it took to define a test case before we introduced the new technology. If we do not have a baseline value for this measure, we cannot judge whether the new technology reduced the average effort or not.

The software engineering literature (e.g., [9][5][18]) commonly mentions three means to get baseline data: use data from a sister project, use historical company data, or split the studied project into components with and without application of the evaluated technology. These three options span the dimensions of time (sequential: historical project vs. concurrent: sister or split project) and locality (inside CS project: split project vs. outside CS project: historical or sister project). Therefore, we extend these three possibilities by a fourth option not mentioned before: taking data from an earlier release of the CS project for baselining (sequential/inside). In the following, we will discuss each baseline source regarding its strengths and weaknesses, taking as a basis

the three baselining assumptions by Jung et al. [7], which we extended in this work with a fourth one that we called "comparable context" to get a total of *four properties of good baselines*[2]:

- *Absence of treatment:* Because we want to evaluate the effect of our treatment (i.e., the new or modified technology in our case study), we obviously need to collect baseline data in a situation where the new or modified technology does not affect the measurement results.
- *Consistent operationalization:* Baseline data should be collected using the same operationalization used to collect the data during the case study in which the new or modified technology is applied. Using a different kind of operationalization makes the data potentially incomparable (e.g., if effort numbers are collected for the baseline with overtime, but without overtime during the case study).
- *Comparable context:* Baseline data should be collected in a context as comparable as possible to the context in which the application of the new or modified technology takes place. The objective is to reduce the effect of confounding factors such as time pressure or experience that influence the measured performance.
- *Multiple measurements:* In the best case, a baseline should consist of more than one measurement point. The reason is that a natural and uncontrollable fluctuation in the measured value that is not the effect of the newly introduced or modified technology can only be determined if multiple measurement points are available.

Additionally, we discuss the baselines in the context of three potential obstacles hindering the usage of a specific baseline source: Differences regarding the *period needed* to collect the required data, whether an *intervention* in non-CS projects is required to collect the data, and on which *level of detail* data have to be collected (e.g., total effort vs. effort per module), which affects the effort and ease of data collection.

- *Historical project baseline:* One option for establishing a baseline is to use data collected for one or more past projects as a baseline. The advantage is that the data were typically collected in the past and must only be extracted in order to be used as a baseline. The major disadvantage is that past project data were usually not collected using the defined measures to be collected during the CS (*consistent operationalization*). Moreover, it is hard to assure that the projects for which the required data are available are sufficiently similar to the CS project to draw reliable conclusions (*comparable context*). The second point can be partially addressed by identifying potential confounding factors other than the new or modified technology that may influence the measured properties as well as analyzing the variation in the measured values for the different applications of the "current" baseline technology in the past projects.
- *Sister project baseline:* An option that addresses the most critical limitation of historical project baselines (i.e., having *consistent operationalization*) is to use one or more similar concurrent projects instead of past projects to get the required

[2] The comparability of baseline and treatment context was most likely not considered in Jung et al. since they assumed in their understanding of baselining a given stable context.

baseline data. Because a sister project runs in parallel to the CS, the same measures as in the CS can be collected, assuring consistent operationalization. However, this means additional measurement effort in the sister projects. A challenge kept from the historical project baseline is to identify one or more sister projects that are sufficiently similar to the CS project to allow drawing reliable conclusions (*comparable context*). As mentioned for the historical project baseline, this can be partially addressed by identifying and controlling influencing factors other than the new technology.

- *Inter-release baseline:* An inter-release baseline does not use historical or sister projects to define a baseline, but rather uses different releases of the same (longer running) project to first define the baseline and then conduct the CS. The major advantage is that the context in different releases of the same project is usually much more stable than between different projects (e.g., similar team, work environment, processes). Thus, inter-release baselines address the major challenge of sister project baselines (*comparable context*). Moreover, data only need to be collected in one project. On the other hand, sufficient time should be planned for collecting the baseline data and then conducting the CS, since these activities cannot be parallelized.
- *Split-project baseline:* A split-project baseline is the most elaborate way to evaluate the effect of a new technology in a real-world context. The activities affected by introducing the new technology are identified and the people who are to perform these activities are randomly assigned to one of two groups. The members of the baseline group apply the "current" baseline technology, whereas the members of the treatment group apply the new technology. For each group, performance is measured separately. For instance, if a new MBAT technique is to be evaluated against the existing manual derivation of test cases, it has to be possible to differentiate between defects found in the product parts tested in a model-based manner and the parts tested with manually defined test cases. Also, effort data has to be collected separately for the parts tested in a model-based manner and those tested manually (*granularity in data collection*).

Table 1 summarizes the discussion by presenting the result of a small survey conducted among six researchers with experience in technology evaluation. Each participant rated each baseline data source regarding each aspect on an ordinal scale (-,o,+,++). With the exception of "Granularity required in data collection", which had to be excluded from the analysis due to an ambiguous formulation in the survey, the table presents the median of their rating.

Table 1. Strengths and weaknesses of baseline data sources

Type of Baseline / Data Source	Historical Project	Sister Project	Inter-Release	Split-Project
Feasibility of baseline data collection				
Period needed to collect required data	++	O	O	+
Intervention in non-case study projects	+	O	+	+
Granularity required in data collection	+	+	+	O
Accuracy of technology evaluation				
Absence of treatment	++	+	+	O
Consistent operationalization	-	+	+	++
Comparable context	-	O	+	++
Multiple measurements	++	O	+	++

Our approach introduces the concept of *internal baselines*. This means that a baseline is defined individually for each CS provider. Baseline data stay inside the organization that applies the technology and collects the data, meaning that baseline data do not need to be communicated to other project partners nor to organizations outside the project. Instead of reporting that 2.5h (*baseline*) were needed with the old technique and 1.5h with the new one to write an average test case, for example, a CS provider only reports the 40% reduction in time (*relative improvement*) and potential confounding factors in the baseline and CS.

Application experience in MBAT: Based upon the feedback, CS providers in MBAT considered the overview of baseline sources including strengths and weaknesses as useful when selecting an appropriate baselining approach. However, it is difficult to evaluate whether this will ultimately also increase the availability of reliable baseline data.

5 Conclusion and Future Work

This paper motivates the need for more multiple case studies in the context of technology evaluation for which specific guidelines are still missing. *Major challenges* identified by a focus group were presented and an approach was proposed to address them. In particular, we showed how a simplified version of GQM+Strategies can be instrumented and applied to consolidate general project goals and align them with CS-specific goals. Moreover, we address typical confidentiality issues in our approach by communicating and aggregating only relative measurement results on the project level in combination with our concept of *internal baselines*. Finally, typical baseline approaches are extended by *inter-release baselines* and presented together with an evaluation of their advantages and disadvantages using expert-based evaluation.

Currently, we are applying the approach in a large-scale project with more than 20 CSs. In this paper, first but promising results regarding applicability and usefulness are reported; however, after project completion in 2014, we plan to provide more elaborate results.

Acknowledgments

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n° 269335 and from the German Federal Ministry of Education and Research (BMBF). We would also like to thank all empirical experts for their survey and workshop participation and Andreas Jedlitschka, Adam Trendowicz, Liliana Guzman, and Sonnhild Namingha for their support and initial review of the paper.

References

1. Basili, V. 1984. Quantitative Evaluation of Software Engineering Methodology. *Pan Pacific Computer, Proc. of 1st Conf.* 1984.
2. Basili, V. et al. 2010. Linking Software Development and Business Strategy through Measurement. *Computer*. 2010, 43, 57-65.
3. Ciolkowski, M., Heidrich, J., Simon, F., and Radicke, M. 2008. Empirical results from using custom-made software project control centers in industrial environments. *Empirical software engineering and measurement, Proc. of the 2nd Int. Symp.* 2008, 243-252.
4. Davis, A., Dieste, O., Hickey, A., Juristo, N., and Moreno, A. 2006. Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review. *Requirements Engineering, Proc. of 14th IEEE Int. Conf.* 2006, 179-188.
5. Fenton, N. and Pfleeger, S. 1997. *Software metrics – a rigorous and practical approach*; *2nd edition*, PWS publishing group, Boston.
6. Greene, D. and David, J. 1984. A research design for generalizing from multiple case studies. *Evaluation and Program Planning*. 1984, 7, 73-85.
7. Jung, J., Nunnenmacher, S., Ciolkowski, M., 2012. A Constructive Approach towards Defining Baselines in Empirical Software Engineering. *Public Fraunhofer IESE report*.
8. Kitchenham, B., et al. 2010. Systematic literature reviews in software engineering -- A tertiary study. *Information and Software Technology*. 2010, 52, 792-805.
9. Kitchenham, B., Pickard, L., and Pfleeger, S. 1995. Case studies for method and tool evaluation. *Software, IEEE*. 1995, 12, 52-62.
10. Kontio, J., Lehtola, L. and Bragge, J. 2004. Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences, *Empirical Software Engineering, Proc. of the Int. Symp.* 2004, 271-280.
11. Lindstrom, D. 1993. Five Ways to Destroy a Development Project. *Software, IEEE*. 1993, 10, 55-58.
12. McLeod, L., MacDonell, S., and Doolin, B. 2011. Qualitative research on software development: a longitudinal case study methodology. *Empirical Software Engineering*. 2011, 16, 430-459.

13. Mohagheghi, P. and Conradi, R. 2007. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*. 2007, 12, 471-516.
14. Mohagheghi, P., Gilani, W., Stefanescu, A., and Fernandez, M. 2012. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*. 2012, online first.
15. Neto, A. et al. 2008. Improving Evidence about Software Technologies: A Look at Model-Based Testing. *Software, IEEE*. 2008, 25, 10-13.
16. Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering *Empirical Software Engineering*. 2009, 14, 131-164.
17. Santos, P. and Travassos, G. 2009. Action research use in software engineering: An initial survey. *Empirical Software Engineering and Measurement, Proc. of 3rd Int. Symp.* 2009, 414-417.
18. Verner, J. et al. 2009. Guidelines for industrially-based multiple case studies in software engineering. *Research Challenges in Information Science, Proc. of 3rd Int. Conf.* 2009, 313-324.
19. Williams, L., Krebs, W., Layman, L., Antón, A, and Abrahamsson, P. 2004. Toward a framework for evaluating extreme programming, *Empirical Assessment in Software Engineering. Proc. of 8th Int. Conf.* 2004, 11-20.
20. Williams, L., Layman, L., and Abrahamsson P. 2005. On establishing the essential components of a technology-dependent framework: a strawman framework for industrial case study-based research. *ACM SIGSOFT Software Engineering Notes*. 2005, 30, 1-5.