MIDAS: A Design Quality Assessment Method for Industrial Software

Ganesh Samarthyam, Girish Suryanarayana, Tushar Sharma, Shrinath Gupta
Siemens Corporate Research & Technologies
Siemens Technology & Services Pvt. Ltd.
Bangalore, India
{ganesh.samarthyam, girish.suryanarayana, tushar.sharma, shrinath.gupta}@siemens.com

***Abstract*—Siemens Corporate Development Center Asia Australia (CT DC AA) develops and maintains software applications for the Industry, Energy, Healthcare, and Infrastructure & Cities sectors of Siemens. The critical nature of these applications necessitates a high level of software design quality. A survey of software architects indicated a low level of satisfaction with existing design assessment practices in CT DC AA and highlighted several shortcomings of existing practices. To address this, we have developed a design assessment method called MIDAS (Method for Intensive Design ASsessments). MIDAS is an expert-based method wherein manual assessment of design quality by experts is directed by the systematic application of design analysis tools through the use of a three view-model consisting of design principles, project-specific constraints, and an "ility"-based quality model. In this paper, we describe the motivation for MIDAS, its design, and its application to three projects in CT DC AA. We believe that the insights from our MIDAS experience not only provide useful pointers to other organizations and practitioners looking to assess and improve software design quality but also suggest research questions for the software engineering community to explore.**

***Index Terms*—Software design, software design quality, software design assessment method.**

I. Introduction

Siemens Corporate Development Center Asia Australia (CT DC AA) provides software solutions for the Industry, Energy, Healthcare, and Infrastructure & Cities sectors of Siemens. Applications in these domains often tend to have a large code-base and are critical in nature. Further, some applications even serve as a platform for the development and operation of other applications. Clearly, a high level of software quality is a common and crucial need across all such applications.

According to a study by Capers Jones across five large organizations, the number of software defects that can be traced back to errors in software design was found to be as high as 64% [1]. Further, Boehm observes that "finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase" [2]. Thus, an approach that focuses on assessing and improving the design can not only help avoid certain kinds of bugs but also lower the cost of developing high quality software.

With this in mind, we conducted a survey of CT DC AA architects in order to understand the state of design assessment practices being followed in CT DC AA projects. The survey feedback showed that most architects had a low level of satisfaction with the existing design assessment approaches followed in their projects. Existing tools and methods had a number of shortcomings that made it difficult to understand and address deficiencies existing in the software design. The survey feedback revealed the need for a new design assessment method that would help address these shortcomings.

This paper presents MIDAS (Method for Intensive Design Assessments), a design assessment method that we have developed to address the above need. MIDAS is "goal-oriented" and entails the assessment to be geared towards addressing the specific assessment objectives defined by the project team. MIDAS is an expert-based method wherein manual assessment of design quality by experts is directed by the systematic application of design analysis tools through the use of a three view-model consisting of design principles, project-specific constraints, and an "ility"-based quality model. The operational aspects of MIDAS are mainly derived from EMISQ (Evaluation Method for Internal Software Quality) [3] method which has been successfully applied for code quality assessment in 32 projects across CT DC AA.

MIDAS has been successfully used to steer design assessment in three separate CT DC AA projects. Feedback received from the project stakeholders indicates their overall satisfaction with MIDAS. Project teams have found the results from the MIDAS assessments relevant and helpful. The feedback has also provided us several useful hints which we believe will help us improve the method in the future.

A number of insights have resulted from (a) the practitioners' survey of design assessments, (b) our experience with the development of MIDAS and its application to three projects, and (c) the feedback received from the stakeholders of the 3 projects where MIDAS was applied. We believe these insights provide useful pointers to other organizations that are interested in design assessments and also highlight research directions that need more attention from the community.

The rest of the paper is organized as follows. Section II presents the motivation and background for MIDAS and includes results from the survey of CT DC AA practitioners. Section III describes related work. Section IV describes how MIDAS was developed and Section V documents our experience with applying MIDAS to 3 projects in CT DC AA.

978-1-4673-3076-3/13/$31.00 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA
Software Engineering in Practice

Section VI discusses the key insights resulting from our MIDAS experience and its relevance to the rest of the community. Section VII presents the conclusions and future directions.

II. Motivation and Background

This section presents a brief organizational background of CT DC AA in order to set the context for the need of an industry-centric design assessment methodology. Next, it defines the term "design quality" as it applies to this work. Finally, it describes the survey that we carried out in CT DC AA to understand the existing design assessment practices in our organizations. The results of the survey played a key role in the design of MIDAS.

A. Organizational Background

Siemens Corporate Technology Development Center Asia Australia (CT DC AA) develops software for all four sectors of Siemens (Industry, Energy, Healthcare, and Infrastructure & Cities sectors). Many projects are also large in size, often exceeding a million lines of code. Most projects involve writing applications on top of existing frameworks and platforms. Additionally, many projects such as those in the healthcare and energy domain are critical in nature. Given the nature of software being developed at CT DC AA, it is important for the organization to explicitly focus on systematic design assessments in order to discover and fix bugs early on in the software development life cycle. To assess the existing design assessment practices being followed in CT DC AA, we conducted a survey of architects in CT DC AA. The next section provides an overview of this survey and summarizes the survey results.

B. Software Design Quality

Software quality has been described in various ways in literature. In order to better quantify quality, many models have been developed to measure software quality using metrics, quality attributes etc. [15]. Further, product quality, design quality and code quality all have unique connotations. In the context of MIDAS, we express design quality using the quality attributes of "understandability", "effectiveness", "flexibility", "extendibility", "reusability", and "functionality" as defined by the QMOOD quality model [15] (which is adapted from ISO 9126 [21] for design quality).

C. Survey of Design Assessment Practices in CT DC AA

Our survey targeted architects since they are the stakeholders most concerned about the quality of software design. The survey questionnaire consisted of a total of 16 questions divided across three categories. The first category contained questions aimed at eliciting the background of the architects and their projects. This allowed us to check if the survey responses had representation from all four sectors of Siemens, and from projects using different programming platforms, with different sizes, and in different stages of software development life cycle.

The second category contained questions that were aimed at educing (a) the level of importance given to design quality in CT DC AA projects, (b) the various design aspects that impact design quality in their projects, (c) the current design assessment methods and tools being followed or used, and (d) whether the respondents were satisfied with the results of those assessments. The responses were used to understand the existing awareness of design quality and concepts, the existing state of design assessment practices, and to gauge the extent of satisfaction with existing assessment practices.

The third category contained questions pertaining to the shortcomings of existing design assessment methods and tools that have been perceived by the practitioners. This included questions about whether and to what extent the results from these methods and tools were being used to improve the software design. The following sub-sections summarize the results from our survey.

1) Background of Architects and their Projects: The survey questionnaire was sent to a company-wide mailing list of software architects and we received 39 responses. The responders were spread across all four Siemens sectors and their average experience in the software industry was 11.6 years. Their projects varied in the choice of programming platforms and the stage of software development life cycle. The average age of oldest component across the projects was 8.5 years with minimum 0 years and maximum 20 years. This implies that we had respondents from projects ranging from new development to long-term maintenance.

2) Current State of Design Assessment Practices: All survey respondents agreed that design quality was important. The aspects they felt were most important to achieve good design quality in their projects were adherence to -

- design principles including coarse-grained principles such as abstraction and fine-grained principles such as Single Responsibility Principle (importance of design principles received rating of 4.0 on a scale of 1 to 5, with 5 being the highest)
- project-specific constraints such as those arising from the domain or use of a particular framework or technology (importance of project-specific constraints received rating of 3.4 on a scale of 1 to 5, with 5 being the highest)

Additionally, 81% of the respondents could see a relation between design flaws and actual product defects found in the field. This implies that being able to discover design flaws and address them is very important for achieving good software product quality. However, when asked to rate their satisfaction with the design assessment practices being followed in their projects, approximately 59% of the respondents were only "somewhat satisfied", clearly highlighting a strong need for improving the existing practices.

3) Shortcomings of Existing Design Assessment Practices: A number of reasons (some of them surprising) emerged for the low level of satisfaction with current design assessment practices. Below, we list some of the key reasons (with percentage response in parenthesis).

- Lack of exposure to design assessment tools and techniques *relevant* for their context (70%).

- Design analysis tools or assessment techniques suffer from numerous limitations. The important ones are:
  - Tools report a large number of violations (45%). To sift through these to identify the critical issues is effort intensive and time consuming.
  - Tools report large number of false positives (54%). It requires considerable manual effort to filter them out.
  - Tools do not consider project-specific constraints, resulting in both false positives as well as false negatives. Tools are known to report issues that are not relevant for the project (72%); tools also fail to find design problems relevant for the project (75%).
  - Projects that use only manual review for design assessments find it to be effort intensive and time-consuming (59%).
- A number of factors act as deterrents in addressing issues reported by design assessments or tools. The major ones are:
  - Difficulty in predicting the impact of changes if refactoring were to be performed (44%); specifically making changes to stable legacy components is perceived risky (63%).
  - Difficulty in convincing higher management about the need for refactoring (38%).

D. Insights from the Survey

The following insights emerged from the survey results.

- **Insight 1:** Since adherence to project-specific constraints and design principles is important for good design quality, a design assessment method that explicitly focuses on them will provide results that are more relevant and helpful to the stakeholders.
- **Insight 2:** To address the lack of exposure to design assessment tools and techniques relevant for project contexts, there is a need for a comprehensive knowledge base that a design assessment method should rely on.
- **Insight 3:** To harness the benefits of both design analysis tools and manual reviews and counter their deficiencies (e.g. large number of false positives reported by tools, extensive effort required for manual review, etc.), a design assessment method should synergize the use of tools and manual reviews.
- **Insight 4:** To allow stakeholders to make informed decisions about what changes to adopt, refactoring suggestions for reported issues should be accompanied with change impact analysis.
- **Insight 5:** The design assessment method should report design issues in a manner so as to serve as a strong evidence to support refactoring.

Our objective was to use the above insights to explore the development of a design assessment method that would address the needs of CT DC AA projects. Towards this end, we surveyed a number of assessment methodologies documented in the literature. These are described in the next section.

III. Related Work

Assessment methods documented in literature can be divided into the following four types based on the assessment scope. In the following sub-sections, we delve into each type and discuss whether and to what extent these methodologies can be used to realize the insights from our survey.

A. Product Quality Assessments

A number of methods for assessing product quality have been described in literature including [4][5][6][7]. Most approaches derive from ISO 14598 [8] process along with ISO 9126 quality model [21] (both now super-ceded by SQuARE standard [9]). A number of concepts that have been incorporated into other assessment methods have their roots in ISO 14598. These include use of a quality model, explicit determination of assessment objective, and producing an evaluation plan and sharing it with stakeholders. We believe these are useful concepts that can be incorporated in our design assessment method. However, assessments methods for product quality are ill-suited to be used as they are for detailed design assessments because they primarily focus on externally perceived quality of the software whereas detailed design assessments need an explicit focus on internal quality.

B. Architecture Assessments

A number of architecture assessment methods exist, such as ATAM and SAAM [10]. While most architecture assessment methods focus on quality attribute characterization, there are also methods that point out violations of architectural principles, guidelines, and constraints. In light of *Insight 1*, checking adherence to principles and constraints is a valuable concept that can be incorporated in our design assessment method. However, a main shortcoming of these architecture assessment methods is that they primarily rely on manual effort for the generation and evaluation of scenarios for assessments, and lack tool support [10]. This makes them ill-suited to be used as is for detailed design assessments because in their current form, they cannot leverage the extensive tool support (that exists for design analysis) that can help reduce the manual effort.

C. Detailed Design Assessments

Most of the work (such as [11] [12] [13] [14] [15]) in the area of evaluating design quality are characterized by automatic evaluation using metrics as the basis. These approaches follow FCM (Factor-Criteria-Metrics) model. However, they suffer from the obscure mapping of quality criteria onto metrics and the poor ability to map quality problems to causes [16] [4]. In contrast, we needed an assessment method that maps quality criteria to design violations and thus allows the stakeholders to better trace quality problems to causes. We also needed an assessment method that explicitly focuses on principles and constraints which these above methods lack.

There are methods such as the one described by Marinescu [16] that consider conformance with established design principles, rules, guidelines or heuristics. However, they fail to consider constraints which play a key role in software design. There also exist some methods such as the one described by

Tervonen [17] which uses a Goal-Rule-Checklist-Metrics model and employs inspections to evaluate design quality. However, this work also does not explicitly consider design principles or constraints, though rules or checklists may cover some of them implicitly. Further, only metric tools are considered in this work without exploiting the power of COTS/FOSS design analyzers.

D. Code Assessments

There are many documented code assessment methods [18] [19] [20]. A considerable number of these methods use or derive from ISO 9126 model [21] and therefore also have several concepts from the standard incorporated into these methods. While many methods primarily rely on tool-based analyses, there are methods such as EMISQ that use a mix of manual review and tool-based analysis for evaluating code quality. Many methods such as EMISQ also provide a prioritized list of suggestions for reported changes to help address the critical issues first. Clearly, such concepts are very useful even for design assessment methods. However, a main shortcoming of these code assessment methods is that design principles and constraints that play a key role in affecting the detailed design quality are not accounted for in code assessment methods. This makes them ill-suited in their current form for detailed design assessments.

Our survey of related work reveals that there are many useful elements that can be borrowed from existing assessment methods for our purposes. However, in light of the requirements that emerged from our survey feedback, none of them could be adopted in their current form. Further, our literature survey shows that barring a few exceptions (such as [12] [4]), most assessment methods have not been validated in an industrial context limiting the number of candidate methods to choose from. These reasons motivated the need for developing a new detailed design assessment method for the CT DC AA context.

IV. MIDAS

Based on our survey of assessment approaches, we identified several key concepts that would be helpful in industrial assessments and incorporated them into our new detailed design assessment method called Method for Intensive Design ASessments (MIDAS). Specifically, we were drawn to a number of concepts and practices in EMISQ. While these are stated in the relevant context during the MIDAS description later in this section, it is important to first discuss the motivation for adopting EMISQ as the underlying operational basis for MIDAS.

The Evaluation Method for Internal Software Quality (EMISQ)[3] is an expert-based method for assessing code quality. It is founded on a controlled application of static code analyzers using a quality model. EMISQ has been used successfully for code assessments in 32 projects (differing in programming languages, size, life-cycle stage, domain, etc.) across CT DC AA. A noteworthy aspect of these assessments is that they have been completed within the planned time period demonstrating the operational feasibility of the method in real-world context. This is in contrast to a number of other assessment methods in literature that have not been validated in an industrial context. Additionally, since many project teams in CT DC AA are already familiar with EMISQ, it would become easier for them to use a new design assessment method if it were to be operationally based on EMISQ (see Section IV.C). The rest of this section describes the three-view model which forms the core of MIDAS, the other salient characteristics of MIDAS, and the process steps of a MIDAS assessment.

A. Three-View Model for Design Problems

Most assessment methods rely primarily on an "ility"-based quality model to show the impact of a design problem on quality attributes. This shows an incomplete view of design problems and also does not provide suggestions to address design problems.

Towards addressing these shortcomings, we build upon *Insight 1* that a design assessment method that explicitly focuses on the violations of design principles and project-specific constraints provides more relevant and helpful results to stakeholders. These principles and constraints can be leveraged to:

a) Detect their violations in the design
b) Cast results of design analyzers in the form of violations of principles or constraints
c) Guide towards a solution to address the violations

We combine the principles and constraints with a quality model to provide a unique three-view model for design problems in MIDAS.

1) "Ility"-Based View: In this view, a design problem is viewed in terms of its impact on one or more design quality attributes defined as part of a quality model. Even though MIDAS is a generic method that can accommodate any suitable quality model, by default, we use the high-level attributes from QMOOD quality model [15] (see Section II.B).

2) Design Principles-Based View: This view shows the causal relationship between a design problem and possible violations of design principles. In the absence of a comprehensive ontology of design principles in literature, we have created a layered organization of well-known design principles. Fine-grained principles (mainly from [26]) such as Acyclic Dependencies Principle (ADP), Single Responsibility Principle (SRP), and Liskov's Substitution Principle (LSP) [27] constitute the lowest layer. These are the principles to which the design problems are mapped. The middle layer consists of principles including (derived from various sources such as [27] [28]) - Classification, Aggregation, Grouping, Coherence, Hiding, Decomposition, Locality, Ordering, Factoring, Layering, Generalization, and Substitutability. The middle layer principles help map principles from the bottom-most layer to those in the top layer. The top-most layer consists of the high-level principles (from [29]) - Abstraction, Encapsulation, Modularity, and Hierarchy.

3) Constraint-Based View: The third view is a constraint based view and shows the relation of a design problem to one or more possible violations of project-specific constraints. These constraints include Language and Platform Constraints,

Framework Constraints, Domain Constraints, Architectural Constraints, Hardware Constraints, and Process Constraints.

We give a few examples below to demonstrate how design problems can be viewed using the three-view model.

- "Cyclic dependencies between classes" is a well-known design problem. The "-ility" based view will show that this problem impacts Flexibility, Extendibility, and Reusability. Further, the design problem indicates a violation of Acylic Dependencies Principle [26] (at the lowest layer of design principle view) which in turn maps to Ordering [30] and Modularity [29] (belonging to middle and top layer of design principle view respectively). The design principle view also suggests the high-level refactoring required for eliminating it. In this case, the dependency order should be made acyclic towards improving modularity.
- "Use of recursion" is a design problem in embedded systems where recursion is prohibited. The "ility" based view will show the problem to impact Effectiveness, and Functionality. In addition, this problem would be viewed as a violation of Domain Constraint in the constraint-based view. Hence, the recommended refactoring step would be to use iteration instead.
- According to .NET framework guidelines [31], a class that holds any "unmanaged resources" should implement the *IDisposable* interface and define a *Dispose* method for releasing the resources (i.e., class must implement Dispose pattern). A design problem occurs when a class that must implement this Dispose pattern does not implement it. This design problem can be viewed as a violation of the Language and Platform constraint.
- Consider a GUI application that is supposed to follow the MVC pattern [32]. However, if the implementation does not strictly follow the separation between Model, View, and Controller roles, then it is a design problem. This problem impacts quality attributes such as Flexibility, and Extendibility. The design problem could also be viewed as a violation of the principle of Decomposition [23] and Modularity [29]. In addition, the design problem indicates a violation of Architectural Constraint imposed by the MVC style.

These examples also illustrate how modeling design problems in multiple views help answer the questions "*why should one care about a design problem?*" and "*how should this problem be addressed?*" It would be rather difficult to answer these questions if an "ility"-based view alone were to be used.

B. Salient Characteristics of MIDAS

1) "Goal-Oriented" Assessments: MIDAS assessments are focused towards answering business or technical questions relating to design quality. These questions can vary depending upon the specific needs of the project. For instance, a project may require a Go/No-Go decision for taking up a major refactoring activity or may want an estimation of current status of design quality or want to know the priority of components to take up refactoring.

Most design assessments offer tailoring of quality models (a.k.a. "quality profiles") for specific quality-centric needs of the project. However, this customization is insufficient for addressing other needs of the project. Our EMISQ experience has indicated that in addition to quality model tailoring, a "goal-directed" assessment makes the assessments more relevant to the project needs. MIDAS, therefore, explicitly elicits goals at the onset of a design assessment and structures the rest of the assessment towards addressing that goal.

2) Synergy of Manual Review and Tool Analysis: Insight 3 recognized that a judicious mix of manual review and tool-based analysis can help harness their benefits while countering their deficiencies. Our EMISQ experience also lends support to this approach. Hence, MIDAS adopts this strategy.

In MIDAS, manual assessment is directed towards analyzing the results of design analysis tools. Faced with a large design artifact, experts get a better understanding of design problems when they examine tool results. Further, experts can help reduce "false positives" to a large extent, prioritize the design problems, and annotate the design problems with refactoring suggestions thus rendering the tool results more useful for the project team.

MIDAS assessments are aided by the SPQR tool [25] (which was developed in the context of EMISQ) which helps analyze the tool-based results, document the findings, and generate a report. This reduces the manual effort required in generating the detailed tool reports containing a prioritized and annotated set of design problems. It should be noted that these reports form a subset of the Design Quality Report which is described in Section V.B.4. Further, as per *Insight 4*, change impact analysis should accompany refactoring suggestions. Since manually analyzing change impact of a large set of refactoring suggestions in a large code base is a herculean task, suitable tools for impact analysis can be used [24].

Insight 2 identified the need for a comprehensive knowledge base. In MIDAS, this knowledge base is available in two forms. The first is in the form of experts who bring their knowledge and experience to assessments. The second is in the form of design rules and metrics available in existing design analysis tools. Towards the end of a MIDAS assessment, experts can leverage the experience gained during the assessment to provide feedback to the project stakeholders on tools and techniques that could be adopted by the project team for regular use.

3) Stakeholder-Centric Reporting: The end result of a MIDAS assessment is a Design Quality Report (DQR). As per Insight 5, assessments should report issues in a manner so as to aid decision making. Based on this and borrowing from our EMISQ experience, DQR presents the results of design assessments in the following three forms to cater to different stakeholder needs.

- For managers – a management summary to aid decision making. This summary provides explicit answers to the assessment goals and suggests preventive and corrective measures. The summary

optionally includes a SWOT (Strengths-Weaknesses-Opportunities- Threats) analysis of the design.

- For architects – a technical summary that provides a high-level view of the problems inherent in the design (in terms of violations of principles and constraints), their impact on design quality, and relevant strategies to address those problems. It also includes relevant design analysis tools or techniques that could be adopted by the project team.
- For developers – a detailed list of prioritized design problems along with refactoring suggestions. This part of the DQR is automatically generated using the SPQR tool [25].

C. MIDAS Process Steps

In this sub-section, we describe the process steps of a MIDAS assessment. The process steps are based on EMISQ [3], ISO 14598 [30], and W-Process [4]. Broadly, the process constitutes four steps (see Fig. 1) which are described below.

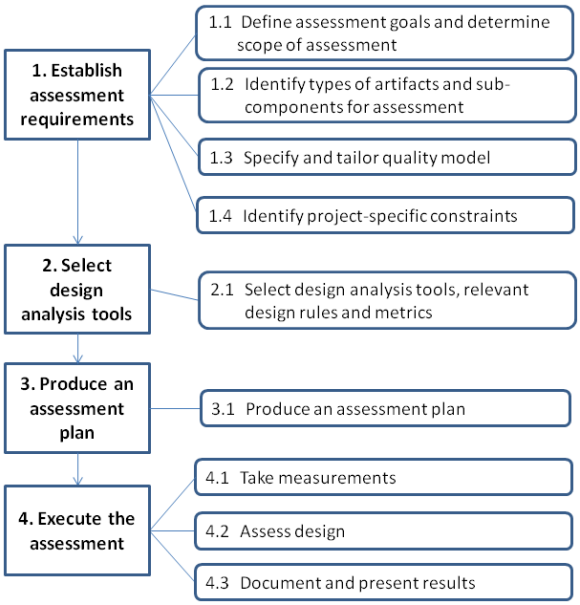

Fig. 1. Process steps for a MIDAS assessment

1) Establish Assessment Requirements:

a) Define Assessment Goals and Determine Scope of Assessment: In this step, relevant stakeholders for the MIDAS assessment are first identified. These typically include the experts who conduct the assessment, project managers, and project architects. Next, roles and responsibilities for each stakeholder are clearly defined. This is followed by a brainstorming session among the stakeholders to define feasible assessment goals considering the business needs of the project, business and technical roadmap of the project, and the technical problems being faced by the project team. Assessment goals can be specified in technical, business, or quality terms.

The assessment goals are then prioritized in agreement with the project stakeholders. The experts estimate the time and effort required to achieve the prioritized assessment goals. This step concludes with the agreement on the prioritized goals and the time/effort required to achieve the same.

b) Identify Types of Artifacts and Sub-components for Assessment: Depending on the assessment goals and the project context, appropriate artifacts (e.g. source code, design diagrams, etc.) need to be selected for assessment. In case the project is large, it is infeasible to evaluate the design of the whole software system. Hence, relevant sub-systems, sub-components, or modules are identified based on a suitable criterion (criticality, size, technical problems faced, etc.).

c) Specify and Tailor the Quality Model: In this step, a suitable quality model is selected for use. MIDAS by default employs QMOOD quality model; however, any other suitable quality model could be used based on the organizational context and assessment goals. The selected quality model is tailored to produce the project specific "quality profile" for the assessment.

d) Identify Project-Specific Constraints: Stakeholders identify all project-specific constraints relevant for answering the assessment goals in this step.

2) Select Design Analysis Tools, Relevant Design Rules and Metrics: In this step, experts select applicable design analysis tools (COTS/FOSS), relevant design rules, and metrics based on the assessment goals, the quality profile, and the project-specific constraints.

3) Produce an Assessment Plan: Experts create an assessment plan which includes timelines, milestones, and deliverables. In addition, it addresses operational aspects such as establishing a "single point of contact" for the project team and the expert team for interaction and clarifications, and planning for availability of resources (such as the architect for any clarifications).

4) Execute the Assessment:

a) Take Measurements: In this step, design analyzers are configured and executed on the selected project artifacts and the generated output is gathered for future use. The end result of this process step is the generated reports for further processing by the SPQR tool.

b) Assess Design: In this critical step, detailed manual assessment is performed by experts. The SPQR tool processes the output generated by design analyzers. The experts use the SPQR tool to categorize the reported design problems based on their impact on quality attributes, violation of design principles, and/or violation of constraints while keeping the project context and assessment goals in mind. Further, using the SPQR tool, experts thoroughly check the reported categorization of design problems, fine-tune the categorization and prioritization, discard problems that are false positives, and annotate the remaining problems with refactoring suggestions.

c) Document and Present Results: Experts document the manual review findings in the DQR. During the assessment, experts share their findings with the project stakeholders at planned intervals via intermediate DQR reports. This helps inform the project stakeholders about early results. It also helps the experts to elicit feedback from the project stakeholders

which can be used to make needed course corrections. At the conclusion of the assessment a final DQR is shared with the project team. Optionally, a presentation is made to the project team to describe the outcome of the assessment.

It should be pointed out that the process steps reuse existing relevant templates (such as template for quality model tailoring, template for deriving and capturing evaluation goals, evaluation plan, and feedback form) from the EMISQ method.

V. Evaluation

We have used MIDAS to conduct design assessments for three projects in CT DC AA. In the following sections, we describe these projects, their objectives behind design assessments, how MIDAS was used to address these objectives, and the feedback received from the project teams post-assessments. We also highlight key insights gained from the assessments and the feedback from the project teams.

A. Project-1

The first project where MIDAS assessment was applied was a GUI editor meant for creating customized UI controls/components for automation & drives domain. There were around 40 controls which were written in C# (.NET platform). The source code size was 125k eLOC. The GUI editor followed MVC (Model-View-Controller) style [32].

1) The Goal of the Design Assessment: The project team was aware of the poor design quality of their software and was interested in improving the design quality. Specifically, they needed help to not just identify refactoring candidates, but also suggest potential solutions in the context of their project constraints. An example of such a project constraint was that the public interfaces should not be modified since there were already many applications that made use of the UI components for development.

2) MIDAS Assessment: Towards achieving this objective, we first selected the following tools: FxCop [33], Gendarme [34], Doxygen [35], NDepend [36], Understand [37], and Clocksharp [38]. Following that, starting from the design rules in these tools, we further tailored the rule sets in these tools towards meeting the assessment objective.

These issues were further filtered manually by experts based on project constraints, and refactoring candidates were identified. In consultation with project architects, experts identified refactoring steps for these refactoring candidates. These steps were formulated in such a manner so as to enable easy understanding of the underlying cause of the problem (as the violation of design principles, constraints, or both), easy understanding of how the proposed refactoring step would remedy the situation, and easy practical adoption of the steps.

A detailed report containing the results of the design assessment and tool generated reports were shared with the project team, and a presentation was made to the project stakeholders summarizing the findings. The assessment was completed by two design experts over a period of two months (requiring total 4 person months for completing the review).

3) Feedback from the Project Team & Lessons Learned: The feedback received from the project team clearly shows that the project team was satisfied with the results of the MIDAS assessment. The project team felt that the refactoring suggestions provided by MIDAS were very relevant and useful to their context and enabled them in meeting the objective of design improvement. Specifically, they felt that some of the refactoring suggestions were very good and helped them better restructure the class hierarchies in their design.

An interesting facet that was revealed during this assessment is the importance of eliciting the assessment goal clearly. Even though the DQR prioritized the refactoring suggestions, the project team further decided to divide these suggestions on the basis of the effort required and the risk involved for implementing those suggestions. Since this further classification was not specified at the onset of the assessment, the DQR did not include such a categorization. This experience made us realize that at the onset of the assessment, the stakeholders may not realize what they actually want or may not be able to effectively articulate their needs. This points to the importance of brainstorming sessions and the focus required during of the sessions with the stakeholders.

As part of the feedback, the project team also made a special mention of the rich knowledge and experience as well as the enthusiasm of the experts involved in the MIDAS assessment. This indicated that the background and experience of experts plays a critical role in the overall success of a MIDAS assessment.

B. Project-2

The second project where MIDAS was applied was also a GUI editor for to assist in the configuration of intelligent electronic devices. This editor can be thought of as a Form designer of Visual Studio but for intelligent electronic devices. This editor followed the MVC architectural style, was developed using C# .NET, and had a size of 32k eLoC.

1) The Goal of the Design Assessment: The project team was facing difficulties in enhancing the existing features of the editor. They suspected that this was due to the poor design quality. So, their objective behind the assessment was to (a) discover the general design issues that were making it difficult to perform changes, and (b) understand the possible refactoring solutions that could help address those design issues.

2) MIDAS Assessment: The assessment method used here is similar to that used in Project-1. This includes the tools that were used, the general process of identification and filtering of issues by experts and identification of refactoring solutions, and the generation and presentation of reports. Like the previous assessment, the assessment here also involved 2 experts for 2 months.

However, there was a significant difference in the assessment methods followed for Project-1 and Project-2. This stemmed primarily from the difference in objectives. Specifically, since the project team was interested in knowing about the design issues affecting the quality of their design, the assessment for Project-2 specifically focused on reporting issues in terms of violation of design principles in the technical and management summaries.

3) Feedback from the Project Team & Lessons Learned: The project team reported their satisfaction with the design quality issues and the suggested refactoring solutions that were documented in the DQR. It is interesting to note that in addition to the violations of well-known design principles, the lack of conformance of the low level design to the overarching architectural style was also reported. The project team found this non-conformance especially revealing. The project team also mentioned that they found the categorization of issues as reported in the DQR useful in determining the most critical issues that needed to be addressed.

The project team felt that the assessment time was long and suggested that it be shortened in the future. However, they also added that the periodic delivery of intermediate DQRs greatly helped in receiving early inputs.

C. Project-3

The third project where MIDAS was applied was to a healthcare application that aims to provide an integrated workflow solution for delivery of particle beams and their recording and verification. This application was based on C# .NET and had a size of 17.5k eLoC.

1) The Goal of the Design Assessment: The project team suspected that certain platform constraints were not being followed properly in the design. For instance, they wanted to know whether C#.NET standard patterns [31] related to resource acquisition and release (such as use of Dispose pattern and 'using' keyword) were consistently followed in the design. Another example is whether guidelines suggested by the .NET framework for exception handling (such as a method should not raise an exception type that is too general or that is reserved by the runtime) are being consistently followed in the design. Additionally, the project team desired to know possible refactoring solutions keeping the .NET specific framework constraints in mind.

2) MIDAS Assessment: The assessment method followed for Project-3 was similar to the method used in Project-1 and Project-2 and hence is not described in detail here. The main difference in this assessment was the focus on .NET framework-specific constraints due to the assessment objectives. This required tailoring of tool rule-sets relating to guidelines suggested by the .NET framework. Due to non-availability of experts and time constraints, this assessment involved only 1 expert for 2 months.

3) Feedback from the Project Team & Lessons Learned: A number of platform constraint violations were reported in the DQR as a result of the MIDAS assessment. The project team found these reported issues very useful and they incorporated the suggestions for the reported issues in the ongoing release itself. In order to prevent the re-occurrence of such issues in the future, they also decided to share the findings of the DQR with the entire project team and planned a series of trainings to increase the awareness of the team about the best practices and guidelines that were suggested in the DQR.

The project team pointed out two concerns in the context of the assessment. The first was the low degree of manual review that was performed during the assessment. This was expected since only one expert was involved in the assessment process; however, it clearly underlines the important role that experts play in the MIDAS method. The second issue was the lack of effective and powerful tools (that would report more relevant issues, report lesser number of false positives, etc.) due to which a larger investment was needed in manual reviews.

VI. Discussion

This section discusses the broader impact of the work described in this paper. We believe the ramifications of our work can be divided along two dimensions. The first relates to the lessons we have learned from our MIDAS experience that can provide useful pointers for other organizations and practitioners who are interested in software design assessments. The second points to key research questions in the context of design assessments that we believe would be of interest to the wider software engineering community.

A. Lessons Learned

MIDAS has been designed primarily to fulfill the needs of CT DC AA. Thus, an organization that plans to use MIDAS as a basis for design assessments should carefully modify it to suit their organizational context. We draw upon the feedback from the evaluation of MIDAS to present the following key aspects that we believe impacts its adoption in other industrial contexts.

- MIDAS is a design assessment method applicable to a range of projects differing in aspects such as the language used, the project size, and the domain. For instance, though the three projects where MIDAS was used were C# projects, the method is language agnostic. By choosing the right experts and tools, it can be made applicable to projects based on other languages. Further, out of the three projects where MIDAS was used, two were small sized (<=100k eLOC) and one was moderately sized (>100k eLOC and <=500k eLOC). Additionally, MIDAS explicitly identifies sub-components for evaluation as a process step (see Figure 1) which allows assessments for large projects to be handled practically. Finally, the three projects where MIDAS was applied spanned the domains of healthcare and industrial automation. Based on these instances, we believe MIDAS is applicable for different kinds of projects across varying domains in other organizations as well.
- Feedback from MIDAS evaluations shows that the third-party experts played a critical role in making design assessments effective. We believe this is because the experts are highly competent in not only software design but also in conducting assessments. Additionally, these experts do not belong to the project teams which allow them to have a fresh and independent view. Thus, an organization that plans to adopt MIDAS or a similar assessment method should invest in creating a separate independent team of assessment experts with the required competencies.

- Depending upon the organizational and project contexts, the right mix of expert review and tool-based analysis should be used in MIDAS assessments. For instance, if the nature of a project or organization constrains the availability of tools for assessments, more effort will be needed for manual reviews.
- The assessment goals differed across the three evaluations; still, the stakeholders were satisfied with the MIDAS assessment results. This is primarily because MIDAS employs a "goal-centric" approach. We believe, therefore, that organizations that use such a goal-centric approach in their assessment methods will not only benefit from relevance of the assessment results to their project contexts but also allow the use of that method across a broad range of projects. Further, it is important to clearly elucidate the goals before embarking on an assessment to ensure satisfaction of project teams.
- While we used the QMOOD [15] quality model in MIDAS, an organization can choose to select an appropriate quality model to fit their needs. This resonates with a recent study that shows that 70% of companies use custom quality models [22].
- Feedback from our MIDAS evaluation shows that using principles and constraints to report violations are very useful. We believe, therefore, that other organizations will benefit by enhancing their existing design assessment methods by explicitly considering principles and constraints for evaluation and reporting.

B. Research Directions for the Community

Based on the results from our survey that guided the development of MIDAS and the feedback from the MIDAS evaluations, as well as a study of related work in the area of design assessments, we believe the following key research directions emerge for future exploration.

First, as indicated in our survey results, project-specific constraints play a key role in design quality. Hence, MIDAS explicitly captures constraints to evaluate the conformance of design to those constraints. While we have seen benefits arising due to this, our literature survey has revealed the lack of a design assessment method that explicitly considers constraints. A research question that emerges thus is whether a readily usable reference model of constraints can be created for assessing design quality.

Second, unlike product quality which has ISO 9126 a widely accepted reference model, there is no reference model for design quality. With the realization of the value of design assessments and their role in evaluating and improving design quality, we anticipate the need for a design quality reference model in the future.

VII. Conclusions & Future Work

This paper described our work with MIDAS, a method for assessing the quality of detailed design for projects in CT DC AA. The motivation for MIDAS stemmed from a survey of software architects in CT DC AA which revealed several shortcomings of the existing design assessment practices in the organization. MIDAS builds upon concepts and best practices from other assessment approaches and uses EMISQ, a successful and prevalent code assessment method in CT DC AA as its operational basis. MIDAS was applied to assess the software design quality of three projects in CT DC AA. Feedback from the project stakeholders reveals a high-level of satisfaction with MIDAS and pointers for future improvement.

The main contribution of this paper is the definition and validation of a novel three-view model for representing design problems in which each design problem is viewed as a violation of design principle(s) or project-specific constraint(s) in addition to its impact on design quality. Additional contributions are presented in the form of key insights that have resulted from our survey of CT DC AA architects and the feedback obtained from the stakeholders of the assessed projects. Further, the results from our survey and the lessons learned from the application of MIDAS have been leveraged to identify important considerations for the adoption of MIDAS in a broader context, and to motivate key research directions for the research community.

Buoyed by the successful application of MIDAS in three projects, in the future, we plan to conduct MIDAS assessments in a larger scale within CT DC AA. Concurrently, we also plan to augment and categorize the current knowledge of design principles and project-specific constraints in order to provide a richer representation for expressing their violation in MIDAS. Finally, in order to cater to the increasing number of projects adopting agile-based methodologies within our organization, we plan to investigate how MIDAS can be integrated into agile-based software development life cycle models.

Acknowledgment

We thank K. Ravikanth and C. Körner from Siemens Corporate Research and Technologies for the helpful suggestions. We also thank the survey responders and the project teams involved in the evaluation of MIDAS for providing their feedback.

References

[1] C. Jones, Software Quality in 2012: A Survey of the State of the Art, http://sqgne.org/presentations/2012-13/Jones-Sep-2012.pdf.

[2] B. W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[3] R. Plösch, H. Gruber, A. Hentschel, C. Körner, G. Pomberger, S. Schiffer, M. Saft, and S. Storck, "The EMISQ Method and its Tool Support -- Expert-based Evaluation of Internal Software Quality", Innovations in Systems and Software Engineering, Vol. 4, No. 1, pp. 3--15, 2008.

[4] T. Punter, R. Kusters, J. Trienekens, T. Bemelmans, and A. Brombacher, "The W-Process for Software Product Evaluation: A Method for Goal-Oriented Implementation of the ISO 14598 Standard", Software Quality Control 12, 2 (June 2004), 137-158.

[5] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidl, A. Goeb, and J. Streit, "The quamoco product quality modelling and assessment approach", in Proceedings of the 2012 International Conference on

Software Engineering (ICSE 2012). IEEE Press, Piscataway, NJ, USA, 1133-1142.

[6] R. G. Dromey, “A model for software product quality”, IEEE Trans. Softw. Eng., 21(2):146--162, 1995.

[7] G. Gousios, D. Spinellis, I. Samoladas and I. Stamelos, “The SQO-OSS quality model: measurement based open source software evaluation”, Proceedings of the International Conference on Open Source Systems, 2008

[8] ISO/IEC 14598-1:1999, Information technology -- Software product evaluation, 1999.

[9] ISO 25000, Software engineering Software product Quality Requirements and Evaluation (SQuaRE) Guide to SQuaRE. 2005.

[10] M. A. Babar, L. Zhu, and R. Jeffery, "A Framework for Classifying and Comparing Software Architecture Evaluation Methods”. Proceedings of the Australian Software Engineering Conference. 2004.

[11] F. Dandashi, “A method for assessing the reusability of object oriented code using a validated set of automated measurements”, In SAC '02: Proceedings of the 2002 ACM symposium on Applied computing, pages 997--1003, New York, USA, 2002.

[12] A. Bergel, J. Laval, and S. Ducasse, “Assessing the quality of your software with MoQam”, 2nd Workshop on FAMIX and Moose in Reengineering, 2008.

[13] L. H. Etzkorn, W. E. Hughes Jr., and C. G. Davis, “Automated reusability quality analysis of OO legacy software”, Information & Software Technology, 43(5):295--308, 2001.

[14] R. Conradi, G. Sindre, and E. A. Karlsson, “The reboot approach to software reuse”, Journal of Systems and Software, 30(3), 1995.

[15] J. Bansiya and C. G. Davis, “A Hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Trans. Softw. Eng. 28, 1 (January 2002), 4-17.

[16] R. Marinescu, “Measurement and Quality in Object-Oriented Design”, Proceedings of the 21st IEEE international Conference on Software Maintenance, ISCM '05, pp. 701-704, 2005.

[17] I. Tervonen, "Support for Quality-Based Design and Inspection," IEEE Software, vol. 13, no. 1, pp. 44-54, Jan. 1996.

[18] Y. Kanellopoulos, I. Heitlager, C. Tjortjis, and J. Visser, “Interpretation of source code clusters in terms of the ISO/IEC-9126 maintainability characteristics”, editors: K. Kontogiannis, C. Tjorjis and A. Winter, in 12th European Conference on Software Maintenance and Reengineering (2008), pp. 63‑72.

[19] D. Spinellis, G. Gousios, V. Karakoidas, and P. Louridas, “Evaluating the Quality of Open Source Software”, Electronic Notes in Theoretical Computer Science, Volume 233, 27 March 2009, Pages 5–28.

[20] D. S. Ducasse, S. Laval, J. Bellingard, F. Vaillergues, P. Balmas, F. Bergel, A. and K. Mordal Manet, “Squale software quality enhancement”, In Proceedings of the European Conference on Software Maintenance and Reengineering, 2009.

[21] ISO 9126: Information Technology – Software Product Quality; 1991.

[22] S. Wagner, K. Lochmann, S. Winter, A. Goeb, M. Kläs, and S. Nunnenmacher, “Software quality in practice - survey results”, https://quamoco.in.tum.de/wordpress/wp-content/uploads/2010/01/Software Quality Models in Practice.pdf.

[23] H. van Vliet, Software Engineering: Principles and Practice (3rd edition), Wiley Publishing. 2008.

[24] M. Acharya and B. Robinson, “Practical change impact analysis based on static program slicing for industrial software systems”, In Proceedings of the 33rd International Conference on Software Engineering (ICSE '11). ACM, New York, NY, USA, 746-755.

[25] H. Gruber , C. Körner, R. Plösch, S. Schiffer, “Tool Support for ISO 14598 based code quality assessments”, Proceedings of the 6th International Conference on Quality of Information and Communications Technology, p.21-29, September 12-14, 2007.

[26] R. C. Martin, "Design principles and design patterns." Object Mentor. 2000.

[27] B. Liskov, “Data Abstraction and Hierarchy,” SIGPLAN Notices, May, 1988.

[28] D. Ross, J. Goodenough, and C. Irvine, “Software engineering: Process, principles, and goals”, Computer, 8:17–27, 1975.

[29] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston, Object-oriented analysis and design with applications (third edition), Addison-Wesley Professional, 2007.

[30] C. Lilienthal, “Architectural complexity of large-scale software systems”, in Proceedings of the 13th European Conference on Software Maintenance and Reengineering. IEEE Computer Society, 2009, pp. 17–26.

[31] K. Cwalina and B. Abrams, Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries (2nd ed.), Addison-Wesley Professional. 2008.

[32] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern Oriented Software Architecture: A System of Patterns, Wiley, New York, 1996.

[33] FxCop, http://msdn.microsoft.com/en-us/library/bb429476.aspx

[34] Gendarme, http://www.mono-project.com/Gendarme

[35] Doxygen, http://www.doxygen.org/

[36] NDepend, http://www.ndepend.com/

[37] Understand, http://www.scitools.com/

[38] Clocksharp, http://www.clocksharp.com/