Stefan Toth, Stefan Zörner

Team-Driven Architecture Evaluation with LASR

Accepted version of the paper presented at the
IEEE 23rd International Conference on Software Architecture, Amsterdam 2026

Winner of the Distinguished Paper Award (Software Architecture in Practice), June 25, 2026, sponsored by the Network Institute, Vrije Universiteit Amsterdam

Accepted version. To appear in: 2026 IEEE 23rd International Conference on Software Architecture Companion (ICSA-C), Amsterdam, Netherlands, 2026.

Team-Driven Architecture Evaluation with LASR

Stefan Toth
embarc GmbH
Vienna, Austria
email: stefan.toth@embarc.de

Stefan Zörner
embarc GmbH
Hamburg, Germany
email: stefan.zoerner@embarc.de

***Abstract* – Software architecture evaluation methods such as ATAM have proven effective but many require external facilitation, substantial preparation, follow-up work or do not fit fast-paced industrial development efforts. This paper presents LASR, the Lightweight Approach for Software Reviews that enables development teams to run systematic, quality-focused architecture evaluations in a single half-day session. LASR is self-contained and team-driven, allowing 2-4 developers to perform the review without separate evaluation support. We position LASR among existing architecture evaluation approaches and report on its industrial application across different domains. Based on data from 75 independently conducted reviews, we discuss the method's effectiveness, typical usage patterns, and practical limitations. The results indicate that LASR delivers actionable, repeatable insights with modest effort and offers a pragmatic addition to the current spectrum of architecture evaluation methods.**

Keywords – software architecture, architecture evaluation, lightweight methods, risk-based review, industrial case study

I. Introduction

Software architecture reviews are an established means of identifying quality risks and design shortcomings in software systems. Well known methods such as the Architecture Tradeoff Analysis Method (ATAM) have demonstrated substantial value in systematically assessing design decisions and quality attributes [2]. As traditional methods are often perceived as heavyweight, numerous lightweight variants have been proposed [10][11][12][13][14]. Common ideas of weight-reduction include using smaller groups of participants or putting higher trust in experts reviewing the system. Still some challenges remain, as evaluation methods often seem hard to facilitate, require external participation, need extensive preparation or produce results that demand additional synthesis before they become actionable.

To address this gap, we developed LASR – the Lightweight Approach for Software Reviews [20]. LASR enables development teams to conduct systematic, quality-focused architecture reviews within a single half-day session. The method is lightweight, self-contained (as it integrates all necessary activities and artifacts into one cohesive session) and team-driven (because it can be performed entirely by the team responsible for the system, without the need for external reviewers or facilitators). This makes LASR well suited for recurring use within agile development cycles, where maintaining architectural awareness must coexist with frequent delivery and limited coordination overhead.

LASR was not conceived as a theoretical framework but emerged from extensive industrial review practice. The recurring patterns and risks observed in architecture evaluations with various methods were distilled into the LASR risk categories that form the basis of the method. Evaluation results are condensed in a management-compatible format, that was previously used in many reports of bigger, ATAM-based evaluations.

Since its introduction, LASR has been applied across multiple industries, including automotive, finance, energy, logistics, and embedded systems. Teams of different sizes and compositions have used the method to reflect on architectural quality and identify architectural risks. The majority of these applications have been carried out independently and offer great insight into the methods applicability.

This paper situates LASR among established evaluation methods, defines the steps of the LASR approach and reports on its industrial use. Based on the findings during its application, the paper reflects on the lessons learned and trade-offs observed. A comprehensive description of LASR is available as a practice-oriented book [20].

II. Background and Related Work

Software architecture evaluation has emerged as a critical activity in the software engineering lifecycle, serving to assess whether an architecture satisfies stakeholder expectations, identify potential risks before they incur substantial costs, and verify that quality requirements are addressed in the design [3]. The importance of architecture evaluation is underscored by research indicating that system maintenance and evolution account for the biggest part of total lifecycle costs for software systems [24][25][26], making early identification of architectural limitations essential for risk mitigation.

Over the past three decades, a diverse ecosystem of architecture evaluation methods has developed, ranging from comprehensive approaches designed for detailed analysis to lightweight methods tailored for agile and industrial contexts. These methods differ substantially in their underlying assumptions, stakeholder involvement, process complexity, analysis techniques, and supported quality attributes.

A. Scenario-Based Evaluation Methods

SAAM pioneered the scenario-based evaluation idea back in 1994 [1]. Its successor, the ATAM, expanded the approach to include all relevant quality attributes of the system under review, therefore making tradeoffs visible [2]. Both methods were defined by the Software Engineering Institute of the Carnegie Mellon University (SEI). Over the years, several derivates and extensions emerged, including lightweight derivates (LAAM [17]), and methods that found specific focus by comparing architectural alternatives (SACAM [18]), quantifying economic impact (Cost Benefit Analysis Method - CBAM [15]) or supporting the design process (Active Review for Intermediate Designs - ARID [4]). Furthermore Quality Attribute Workshops (QAW) cover the gathering of quality requirements with stakeholders in more detail [21],

while the Mini-QAW describes a lightweight derivate [22]. Non-SEI methods include PASA [16], CPASA [19] and ALMA [7] that focus on specific quality-attributes or the SBAR (Scenario-Based Architecture Reengineering) that integrates scenarios with simulation and mathematical modeling [8].

Although practical shortcuts exist, many of these methods are deemed heavyweight. ATAM, for example, requires extensive preparation, multiple days of intensive stakeholder engagement during the review and external facilitators with specialized expertise [6][17]. The overhead of preparation, coordination, and post-review results communication creates friction with modern development practices and limits adoption in resource-constrained teams [6][9].

B. *Lightweight Approaches and Persistent Limitations*

Responding to these barriers, lightweight methods have emerged. While LAAAM (Lightweight Architecture Alternative Assessment Method [13]) stays relatively close to the scenario-based roots of architecture evaluation, other methods use different approaches. Decision-Centric Architecture Reviews (DCAR) put decisions in the center of attention, following a mapping and documentation path, to identify risks [10]. Pattern-Based Architecture Reviews (PBAR) take advantage of known pros and cons of used patterns to get fast results [11]. The Tiny Architecture Review Approach (TARA) limits the group of participants and takes advantage of expert [14]. And the Decision and Scenario-based architecture Evaluation (DASE) seeks middle ground between comprehensive and minimal approaches [12].

These methods reduce review costs, as they reduce analysis times and the need of active stakeholder participation. At the same time, review results are still fine grained and need further work to be summarized or made actionable. Furthermore, most lightweight methods retain external facilitation requirements or demand significant overhead for preparation or result communication.

C. *Positioning of Self-Contained Team-Driven Evaluation*

Extending the family of lightweight architectural reviews, team-driven approaches embed risk exploration and result documentation into a single, self-contained session. Without the benefit of "external" impulses, these evaluations normally rely on broad participation and diverse experiences that reviewers bring to the table. As an example, the Pre-Mortem approach [5] can quickly discover risks with architecture-specific concern framing. It takes advantage of the teams system and context knowledge, framing the evaluation session as a "game" where every doubt is welcome. Risk Storming is another team-driven approach that identifies risks based on software architecture diagrams [23].

Though delivering fast results, team-driven approaches seem random and not very "scientific". The results are very dependent on the team conducting the evaluation. While these methods overcome several drawbacks of other lightweight methods, new limitations arise.

III. The LASR Method

A. *Origins and Rationale*

The LASR approach is the result of extensive review practice and practical limitations for these reviews in industry settings. Looking back at over 100 architecture evaluations in different industry sectors, empirical analysis showed recurring architectural issues. By feeding these recurring issues into a Pre Mortem inspired review setting, we discovered that smaller groups of three to four people were able to generate meaningful insights – evaluation results were less random compared to vanilla Pre Mortem workshops and the identified risks covered a broad range of architectural aspects.

To focus the evaluation approach and contextualize the review, LASR draws inspiration from ATAM and uses quality attributes and quality statements to develop an evaluation benchmark. The identified risks and problems are quantified and associated with the most important quality attributes to allow for a condensed review result, that goes beyond fine-grained review findings and is directly communicable to other stakeholders.

LASR addresses a critical gap in the architecture review landscape. While comprehensive methods like ATAM excel in high-stakes scenarios with significant uncertainty, complex stakeholder dynamics, and substantial financial implications, they require experienced facilitators and considerable time investment. LASR distills the essential elements of proven practices into a format that teams can execute independently, producing structured review outcomes in as little as two to three hours.

B. *Design Objectives*

LASR was designed with the following key objectives:

1. **Quality-focused:** Architecture evaluation should focus on quality goals and associated risks. The method steers groups to important quality aspects and puts findings into perspective.
2. **Self-contained:** Preparation or post-session work should be minimal. The method relies on materials and meeting setup but has no requirements for prepared inputs and delivers ready-to-use results.
3. **Team-driven:** Software development teams should be able to conduct the evaluation by themselves. The method works with 2-4 members of a development team acting as reviewers. External facilitation is optional.
4. **Time-bounded:** First evaluation results should be available within 2-3 hours. This includes a condensed results overview that can be understood by stakeholders.
5. **Interruptible:** Each step produces usable artifacts with immediate value. Later steps refine them, but no mandatory final "sense-making" step is required.

C. *Method Structure*

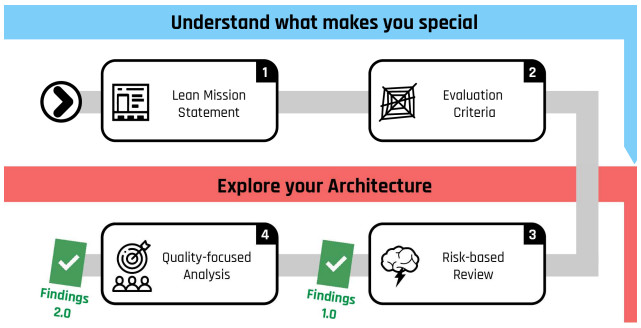

Fig. 1. LASR method – two core activities and four steps

Figure 1 provides an overview of LASR. LASR consists of two key activities, highlighted in blue and red respectively: "Understand what makes you special" and "Explore your Architecture". They serve as overarching categories and group two steps each (e.g., "Evaluation Criteria"). Steps always produce intermediate findings and follow a defined method that guides reviewers toward this result. All activities and steps are carried out by the same group of people, normally members of the development team. In steps 1 and two, the participation of a product owner can be valuable. We refer to all LASR participants as "reviewers". They contribute equally to the review's findings.

The method overview also shows review findings in green. After completing each step in "Explore your Architecture", they are available in versions 1.0 and 2.0 and include a "LASR Results Diagram" that makes findings understandable to stakeholders. Nevertheless, LASR is primarily designed for development teams and architecture-minded developers and does not produce "polished" reports for management roles.

Step 1: Lean Mission Statement

Reviewers distill what is unique about the system by formulating a lean mission statement.

First the system under review is defined by scoping it and defining which areas of a software solution might be excluded from the review intentionally (Framing). Then the essence of the business context is captured by developing "claims". In an active exercise reviewers use the landing page metaphor: Which claims would a homepage of the system under review make? What are the most remarkable features and qualities, unique selling points, differentiators from competitors, or critical improvements compared to previous versions? What are the main benefits and why is the company spending money to build the system?

To focus the findings, the set of claims is narrowed down to around 7 ± 2 key points. This list serves as the Lean Mission Statement. Instead of using presentations, the reviewers are actively participating in this step and open their minds for creating the review benchmark in step 2.

Step 2: Evaluation Criteria

A good evaluation benchmark ensures that the assessment stays focused on the aspects that are most critical to the system's success. Building on the mission statement from step 1, LASR identifies the 3-5 key quality goals most relevant to the system under review.

To start this step, reviewers randomly select quality attributes as their starting Top-5 from a predefined pool. Then the remaining quality attributes are evaluated one by one, treating each as a 'challenger' to the current selection. For each challenger, the reviewers discuss why it should replace any of the existing Top 5 quality goals. After several rounds of challenging and replacement, a set of quality attributes remains as the most important quality goals of the system under review. Furthermore, the reasoning for favoring certain quality attributes over other quality attributes, makes it possible to characterize the chosen qualities.

The result of this step is reminiscent of Mini-QAW "quality attribute web" [22] and includes:

- A set of 3-5 key quality goals. They form the axes of a spider web diagram that is later used to condense the findings of LASR.
- Important "quality statements" describing essential aspects of these quality attributes. They are similar to quality scenarios in ATAM, but less formal.
- The expected "quality level" for each quality attribute as a review benchmark. Figure 2 illustrates how this quality level is set, both in comparison with other attributes in focus and relative to comparable systems known to the reviewers.

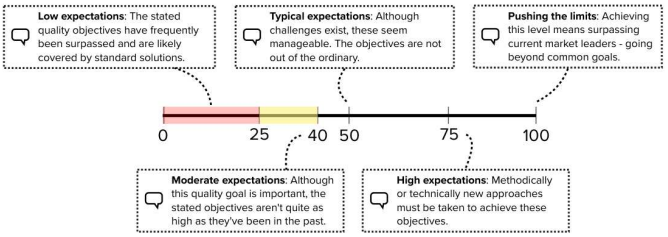

Fig. 2. Reference to find quailty levels as a review benchmark

Step 3: Risk-based Review

The third LASR step is loosely based on Pre-Mortem workshops: Reviewers put themselves in a possible future, looking back at a failed system, asking "what went wrong?" The search for future problems (which from today's perspective are just risks), is not completely unguided in LASR. Reviewers are supported by 32 "standard risks", that are empirically extracted from 100+ industry reviews. Standard risks are organized into 8 risk categories, with 4 standard risks in each. Here is a rundown of these categories:

- **Software Solution**: Covers issues with unsuitable technologies, overly complex frameworks, immature libraries, or problematic architectural patterns.
- **Skill and Experience**: Encompasses missing knowledge, siloed expertise, or insufficient learning opportunities within the team.
- **Objectives and Expectations**: Addresses unrealistic deadlines, misaligned expectations, or poor customer communication.
- **External Systems and Platforms**: Focuses on unstable third-party systems, integration challenges, or licensing issues.
- **Legacy Systems and Technical Debt**: Highlights fragile code, missing tests, outdated documentation, or knowledge gaps.
- **Organization and Processes**: Examines restrictive processes, organizational boundaries, office politics, or unclear decision authority.
- **Operations and Deployment**: Looks at deployment bottlenecks, low automation, inadequate feedback loops, or missing operational tools.
- **Soft Factors**: Considers team conflicts, communication breakdowns, discipline issues, or cultural misalignments.

Based on predefined standard risks within these categories reviewers derive specific risks for their context and system and write them down. The standard risks help reviewers break out of their usual thought patterns and enable smaller teams to brainstorm efficiently without getting stuck.

Each identified specific risk is estimated in the probability of occurrence and the extent of damage it could inflict on the systems success – both in three step: low (L), medium (M) and high (H). Based on the outcome, a gap estimation is assigned: L/H and H/L risks make for a 4% gap, M/M risks are estimated with an 8% gap, M/H and H/M with a 12% gap and H/H risks with a 20% gap each. The specific percentage numbers (4/8/12/20) are derived from experience and work in most cases to show meaningful gaps with some spread. Risk-storming uses a similar scale to evaluate risks [23]. The gap is then applied to the quality goal that is most affected (or multiple goals if necessary) and shown on a spider web diagram that is named "LASR result diagram" from here on. An example is shown in Figure 3.

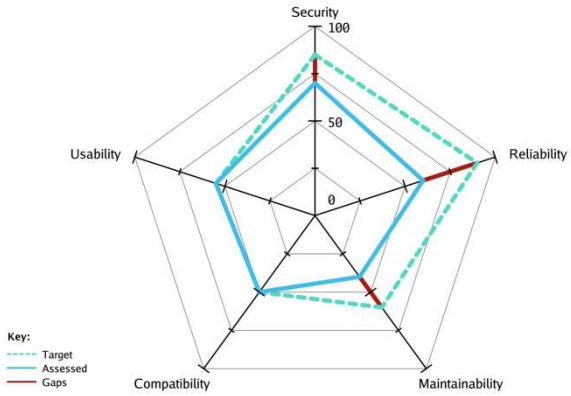

Fig. 3. Example of a LASR result diagram – The dashed green line shows the desired quality levels as identified in step 2. The dark red sections of the axis show estimated gaps after step 3, which leads to the inner solid blue line that shows the current quality level of the system under review.

Step 4: Quality-focused Analysis

The Risk-based Review from step 3 is a first assessment, but the actual scale of problems or quality achievement might still be incorrectly represented, as gaps where estimated in a quite „mechanical" way. However, the result triggers something important: A gut feeling that reviewers can use to identify which axes of the LASR result diagram deserve a more focused, detailed look.

First the reviewers identify those quality goal axes where the gap in the LASR result diagram seems to be too large, too small, or is simply controversial. Then they perform an in-depth analysis:

1. Refine the goal axis with quality statements noted in step 2 and brainstorm additional details.
2. Gather the risks that were associated with the quality goal during step 3.
3. Identify new risks by talking about specific quality statements
4. Examine the risks found and analyze their impact on the most important quality statements.
5. Conclude the analysis by assessing strengths and weaknesses of the solution (supporting either the quality statements or risks found) and updating the gap value by consensus.

The outcome of the Quality-focused Analysis looks identical to that of the Base Review, but it's been selectively questioned and better understood. Compared to the Base Review, it may contain larger or smaller gaps and might even show goal axes that exceed requirements.

D. Typical Outputs

A LASR Review generates several outputs over its four steps. Step 1 produces a mission statement of the system under review. In step 2 a list of the most important quality attributes is defined and detailed with quality statements. Step 3 generates the LASR result diagram showing desired quality levels and assessed gaps in a spider web diagram. It also lists the risks that led to these gaps. Step 4 further refines quality statements and risks. It identifies important architectural approaches as strengths or weaknesses of the system under review.

IV. Industrial Application of LASR

LASR is already broadly used in software development efforts across different industries (see Figure 4). This section shows some details of real-life applications and gives an overview of the broader usage, as well as the methods reception in the industry.

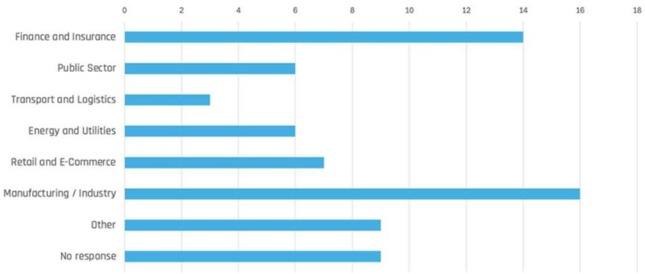

Fig. 4. Applications of LASR in different industries

We analyzed 75 LASR reviews conducted without facilitation or support from us, the authors of the method. We conducted 18 detailed interviews with reviewers that used the method for the first time and gathered insights from 57 respondents to an online survey.

A. Case Study Artifacts

The flow and feel of LASR as an evaluation approach is best illustrated with examples. The following artifacts originate from different LASR workshops and show how the described steps and activities can be implemented in industry contexts.

The purpose of the **Lean Mission Statement** (step 1), is to frame the reviewers for the upcoming benchmark definition. They should get a feeling for the customer(s), the software product and the most important traits of this product. The form in which the lean mission statement is delivered is therefore secondary, but the step is easier, when groups start of by defining marketing claims for the system and then vote which ones are most important. The whole step usually takes 15-30 minutes.

Here are important claims of the mobile instant massager Threema to give a vivid example of what the result of this step can look like:

- **Privacy-first**: End-to-end encryption, no phone number required
- **Fully anonymous**: No personal data collection
- **Swiss-made security**: Complies with strict Swiss data protection laws
- **Decentralized storage**: No central user database
- **Cross-platform**: Available on iOS, Android, and desktop
- **Threema Work**: Secure enterprise messaging with business features

When defining the **evaluation criteria** (step 2), the team chooses the top 3-5 quality attributes by making pairwise comparisons of quality attributes and discussing the priority of these attributes. The arguments for or against a quality attribute lead to quality statements – Table I shows an example outcome of this step from another LASR-workshop that focused on an online shop system of a big German retailer.

TABLE I. TOP QUALITY ATTRIBUTES AND QUALITY STATEMENTS

Top Quality Aspects				
Reliability	***Maintainability***	***Operability***	***Security***	***Scalability***
Disruption-free shopping experience	Quick reaction to change requests	Deep insight if necessary - transparency	No data breaches (external impact)	Quick reaction to high request volume
24/7 shopping is possible	Fast extension with 3rd party solutions	Low effort for dev teams	Avoidance of fraud (manipulation)	Cost-efficient handling of load-spikes
Site stability generates customer trust	Easy version updates and security patches	Clear responsibilies for all roles involved	Security of Checkout-Data	minimal manual effort
	Easy to analyze in case of failure	Available, good contact persons	Detect and defend against attacks	
	Good time to marketfor important features		Prevent competitors from gaining insights	

The quality statements are not necessarily measurable at this stage, but illustrate what aspects of the quality attribute are important. The quality attributes used, were a superset of ISO 25010:2023 quality attributes, augmented with attributes that often come up in the domain at hand. For Webshops this can be scalability, operability, cost efficience or auditability (all of which are not top level attributes in the ISO norm).

The characterizing of the top quality attributes and the definition of their target levels (see figure 2) takes about 30 minutes in most contexts.

The **Risk Based Review** (step 3) takes standard risks as an inspiration and reviewers come up with specific risks for their project or software product. An example risk, as formulated in this step is: "A developer accidentally deletes an index which makes the data sources unreachable". The standard risk that triggered this risk-idea was *Lack of operational concepts*, which sits inside the *Operations and Deployment* risk category. It reads: „Are monitoring, capacity planning, backup, disaster recovery, (security) alerting, ... backed by sound concepts? Are these concepts well supported by tools and widely known or adopted?". The risk was estimated in the probability of occurrence and the extent of damage it could inflict on the systems success and landed in the low likelihood, high impact category (L/H – 4% gap extimation).

Figure 5 shows the risk distribution of a real LASR workshop. The darker sticky note in the upper left corner is the example from above. The numbers are standard LASR-gaps (in percent) and were discussed in section C - step 3.

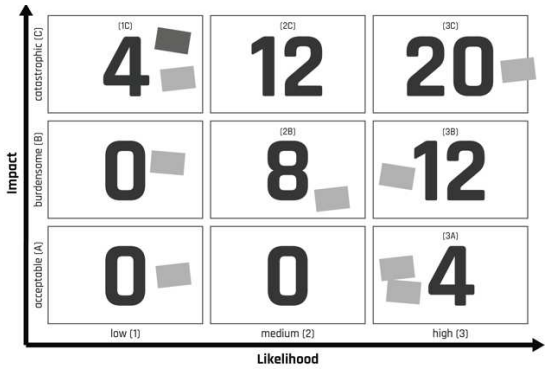

Fig. 5. Risk distribution after LASR step 3

To map the estimated gaps to the right axis in the LASR Result Diagram, the reviewers map the main impact of the risk to the most important quality attributes. In this case the risk of accidential index deletion can be associated with the "Operability" or "Reliability" attribute. After discussion the group decides good operability concepts would be sufficient to mitigate the problem, so they add the gap to the Operability axis of the LASR Result Diagram. It is possible to show gaps on multiple axes if necessary. Step 3 takes from 45-60 minutes.

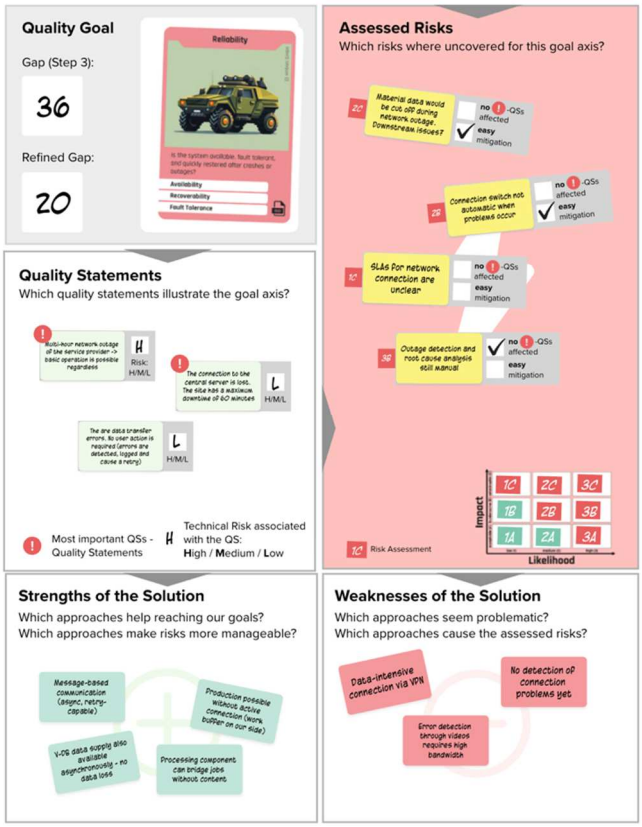

Fig. 6. LASR template for step 4

The **Quality Focused Analysis** (step 4) is optional, but gives more depth to certain quality areas. It takes 30-45 minutes per axis. The reviewers use a more qualitative analysis approach to improve on the first evaluation result that is available after step 3. They typically choose 2-3 quality aspects to discuss in depth. Figure 6 shows a template that can be used during this deep-dive discussion. It is filled with examples from a LASR workshop in the manufacturing domain.

B. Cross-Case Observations

By analyzing the independent usage of LASR, we can gather some insights into the effectiveness of the method, it's reception in the industry and the achievement of its design objectives.

The systems examined in the reviews varied considerably in their level of maturity. Twelve systems were not yet in production or available on the market, indicating that the review was conducted early in their life cycle. At the other end of the spectrum, six systems have been in operation for ten years or more, some even over 20 years. LASR results were deemed valuable in all age classes, which suggests that the method is applicable throughout the lifecycle of software.

LASR is *lightweight* in practice: The median review in our dataset took 3-4 hours. When interpreting the data, we must consider, that many respondents were first-time reviewers. Also, some reviewing groups seem to be bigger than originally intended by the method. That is most probably why some respondents reported review durations of up to 2 days. Out of all analyzed LASR reviews, 15.38% took over a single day. These longer reviews where predominantly reported by bigger reviewing teams, most being in the 7+ reviewer category. Respondents often mentioned soft factors as the reason for these big reviewing teams, like including the whole team or involving people that want to participate. The method itself does not profit from this increase of peoplepower, as Table II shows (at least when the *quantity* of risks is taken as a measure).

TABLE II. GROUP SIZES AND THEIR EFFICIENCY

Number of Reviewers	*Avg. Time Spent*	*Avg. Number of Risks Found*	*Avg. Efficiency*[a]
1-2	4.82	5.91	1.23 risks / h
3-4	4.15	7.11	1.71 risks / h
5-6	5.17	7.87	1.52 risks / h
7+	7.16	9.55	1.33 risks / h

[a.] Risks found per hour (not considering personnel expenditure)

The data indicates a "sweet spot" at the 3-4 reviewers range. Groups of this size uncover the most risks per hour and are most efficient with the LASR approach, even when personnel costs of bigger reviewing groups are *not* taken into account. On average LASR Reviews uncovered 9.10 risks, when at least step 3 of the method was reached.

The *quality* of the identified risks was not part of our survey and is hard to evaluate objectively. Larger teams could produce fewer but higher-quality risks. As a proxy, we considered how teams used the findings: across all group sizes, risks appeared to surface new areas of concern, as only 4% of respondents reported *no* further use of the findings. LASR also yields *actionable results*, with 72% taking next steps without delay. We found no correlation between result usage and participant count, suggesting that the identified risks were equally useful in all group settings. Future work could examine this with a more detailed approach.

LASR seems to be *interruptible*, as intended. Although almost all reviewers found the review results helpful and took action based on the results, only a fourth of the respondents carried out all 4 steps of the LASR method. The most common exit seems to be after step 3 of the method, when reviewers have uncovered the first risks. About half of the reviewers move on directly from there or intend to do so soon. But 48.6% take the findings 1.0, that are available after step 3 and don't take specific deep dives into quality aspects (step 4).

V. CHALLENGES AND CONCLUSIONS

This paper introduced the LASR evaluation method, outlined it's purpose and design goals and summarized findings from independent usage data that was gathered. Although the evaluation of the method wouldn't meet higher scientific study standards, it documents industrial use and provides insights into practice.

Though the method is widely adopted and deemed useful, there are some limitations that should be taken into account. Being team-driven, there is a lack of external viewpoints and impulses. This can be fixed by including external reviewers but that makes LASR a little more heavyweight. Step 1 typically is a little more time consuming when including people who don't know the software product and the group of reviews grows with external participation.

Moreover, LASR skips the alignment of architectural viewpoints, as it presumes participants to be knowledgeable of the system and its current design. In some cases, this may lead to intense discussions during risk assessment (LASR Step 3), were not only risks, but also architectural details and misconceptions about them are addressed. If participants have divergent views, even the scope of the review or the system boundaries may need some late refinement. In complex systems and bigger development efforts LASR therefore needs proper facilitation or even a preparation step where the system under review is consolidated.

Another challenge is that raised problems or mental reservations are not wrapped or sugarcoated in LASR. Risks are always directly addressed and conflicts during estimations and prioritization are not facilitated. This can lead to suboptimal experiences in situations where conflicts and political topics are present. We would recommend evaluations with more analytical depth and facilitation structure – like ATAM – in these environments.

LASR was designed to work well in setups with 2-4 people. As we saw with independent usage, 5-6 people reviews are common and still show reasonable effectiveness. Bigger groups have diminishing returns though and at some point it is a questionable decision to throw too much resources at a method that is thought to be "quick and fun". Solutions of multiple groups doing LASR Basic Evaluations in parallel worked, but are not in the sweet spot of the method. Effectiveness drops and the merging of multiple results proves to be complicated.

In summary, there are limitations to the LASR evaluation method and the independent practice diverges from the methods core ideas sometimes. Nevertheless, LASR seems to be successful in most of its set goals and is a valuable addition

to existing evaluation approaches. It works as a team-driven approach with minimal facilitation requirements, is self-contained with very low effort preparation and follow-up work. It is a lightweight method, even compared to other methods that claim to be lightweight and seems to generate new and actionable insights in almost all cases.

VI. Acknoledgements

We like to thank all members of the LASR Community who participated in individual interviews and the survey. The insights into independent practice were very valuable. This paper was written by hand (the traditional way). The usage of Large Language Models was limited to grammar checks (Open AI ChatGPT 5.1).

VII. References

[1] R. Kazman, L. Bass, G. Abowd, and M. Webb, ªSAAM: A Method for Analyzing the Properties of Software Architectures,º Proc. 16th Int'l Conf. Software Eng., pp. 81-90, 1994.

[2] R. Kazman, M. Klein, M. Barbacci, H. Lipson, T. Longstaff, and S.J. Carriere, The Architecture Tradeoff Analysis Method, Proc. Fourth Int'l Conf. Eng. of Complex Computer Systems (ICECCS '98), Aug. 1998

[3] L. Dobrica and E. Niemelä, "A Survey on Software Architecture Analysis Methods" *IEEE TSE*, vol. 28, no. 7, 2002.

[4] P.C. Clements, "Active Reviews for Intermediate Designs," Tech. Report CMU/SEI-2000-TN-009, Carnegie Mellon University, 2000.

[5] Dave Gray, Sunni Brown, and James Macanufo, "Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers" (1st. ed.). O'Reilly Media, Inc., 2010

[6] J. F. Maranzano, S. A. Rozsypal, G. H. Zimmerman, G. W. Warnken, P. E. Wirth and D. M. Weiss, "Architecture reviews: practice and experience," in IEEE Software, vol. 22, no. 2, pp. 34-43, March-April 2005

[7] H. van Vliet, P. Bengtsson, and J. Bosch, "ALMA: Architecture Level Modifiability Analysis," Journal of Systems and Software, vol. 69, 2004.

[8] P. Bengtsson and J. Bosch, "Scenario-based software architecture reengineering," presented at the 5th International Conference on Software Reuse, 1998.

[9] M. Steffens, "Overview of different software architecture review processes". Retrieved from https://steffenmueller4.github.io/

[10] U. van Heesch, V. -P. Eloranta, P. Avgeriou, K. Koskimies and N. Harrison, "Decision-Centric Architecture Reviews," in *IEEE Software*, vol. 31, no. 1, pp. 69-76, Jan.-Feb. 2014

[11] Harrison, Neil & Avgeriou, Paris. (2012). Pattern-Based Architecture Reviews. Software, IEEE. 28. 66 – 71, 2010

[12] Tuovinen, AP., Mäkinen, S., Leppänen, M., Sievi-Korte, O., Lahtinen, S., Männistö, T., Unwasted DASE: Lean Architecture Evaluation, In: Product-Focused Software Process Improvement. PROFES 2017

[13] S.J. Carriere, Lightweight Architecture Alternative Assessment Method, retrieved from: https://web.archive.org/web/20151206225441/http://technogility.sjcarriere.com/2009/05/11/its-pronounced-like-lamb-not-like-lame/.

[14] E. Woods, Industrial architectural assessment using TARA, Journal of Systems and Software, Volume 85, Issue 9, Pages 2034-2047, 2012

[15] R. Kazman, Jai Asundi and M. Klein, "Quantifying the costs and benefits of architectural decisions," *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, Toronto, ON, Canada, 2001

[16] Lloyd G. Williams and Connie U. Smith, PASASM: a method for the performance assessment of software architectures. In Proceedings of the 3rd international workshop on Software and performance (WOSP '02). Association for Computing Machinery, New York, NY, USA, 179–189, 2002

[17] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 4th ed., Addison-Wesley, 2021.

[18] C. Stoermer, F. Bachmann and C. Verhoef. "SACAM: The Software Architecture Comparison Analysis Method.", 2003

[19] R. J. Pooley and A. A. L. Abdullatif, "CPASA: Continuous Performance Assessment of Software Architecture", 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems, Oxford, UK, pp. 79-87, 2010

[20] S. Toth and S. Zörner, "Reviewing Software Systems with the Lightweight Approach for Software Reviews – LASR.", 2024. Retrieved from: https://leanpub.com/reviewing-software-systems

[21] M. Barbacci, R. Ellison, A. Lattanze, J. Stafford, C. Weinstock, and W. Wood, "Quality Attribute Workshops (QAWs), Third Edition," *Carnegie Mellon University, Software Engineering Institute's Digital Library*. Software Engineering Institute, SEI Report CMU/SEI-2003-TR-016, 2003 Retrieved from: https://doi.org/10.1184/R1/6582656.v1

[22] M. Keeling and W. Chaparro, "Facilitating the Mini-Quality Attributes Workshop - A Lightweight, Architecture-Focused Method", SATURN 2014. Retrieved from: https://studylib.net/doc/13086749/facilitating-the-mini-quality-attributes-workshop

[23] S. Brown, "Risk-storming - A visual and collaborative risk identification technique", retrieved from: https://riskstorming.com/.

[24] R. Zarnekow, J. Scheeg, W. Brenner, "Untersuchung der Lebenszykluskosten von IT-Anwendungen", WIRTSCHAFTS-INFORMATIK 46 (2004) 3, S.181–187, retrieved from: https://www.alexandria.unisg.ch/server/api/core/bitstreams/f59694de-1613-4435-b3e3-ea9167730316/content

[25] Barry, E.J., Kemerer, C.F. and Slaughter, S.A. "Toward a Detailed Classification Scheme for Software Maintenance Activities", Proceedings of the 5th Americas Conference on Information Systems, Milwaukee, 13-15 August 1999, 126-128.

[26] F.P. Engelbertink, H.H. Vogt, " How to Save on Software Maintenance Costs", Omnext White Paper, 2010