3.1.1 Project-based

Back in 1999, performance engineering was a project-related effort. I remember a project at an e-commerce business in Germany. A multi-million-dollar project was launched, and the team faced severe slowness issues during checkout a few weeks before the scheduled launch date. The project lead hired me as their performance expert, equipped me with a load testing tool, and assigned the load testing task. After implementing and executing a few fundamental load tests, I had all the evidence to demonstrate that the new shop could not handle concurrent user traffic. This was no good news because it took several weeks for the developers to fix these issues, and the organization had to postpone their product launches.

3.1.2 Center of Excellence

In the late 90s, organizations streamlined their processes and wiped-out redundancies. The so-called CoE or Center of Excellence optimizes [5] each value component. A CoE performance owns all processes, methods, and tools related to performance engineering. They provided coaching for project teams, maintained the load testing tools, and, in some cases also, supported test implementation and execution.

3.1.3 As a Service

The as-a-service economy has become popular during the Industry 4.0 [8] movement. Driven by cost transparency and efficiency, Performance engineering as a service is often provided by managed service providers or internal teams. They do everything from requirement gathering to project-based load test implementation, execution, and reporting. This model has benefits, such as buying what you need, but disadvantages, such as not going the extra mile when required and slightly higher costs.

3.1.4 Performance as a Value

For performance to be valuable, it must grow and endure. If we look at software development and bug-fixing efforts, we understand that late defect discovery results in high defect costs, delays releases, and negatively impacts end users. By fixing such late-discovered performance issues, you can only heal something already broken but won't create endurable value.

The value of performance requires initial and continuous investments. Like an investor who puts their money into promising assets, the "Performance as a Value" technique follows a risk-based approach. There is no reason to put the same performance investment in all your business applications. Instead, we run a risk rating on applications and their changes, and depending on this rating, we implement mitigating measures. This risk mitigation applies to pre-production performance activities and includes performance monitoring, alerting, and tracing on production.

3.2 Life Cycle Touchpoints

From a software development process perspective, there are several approaches to how enterprises integrate performance engineering activities. The chess grandmaster [6] put it very well when he said, "To improve your game, you must study the endgame before everything else, for whereas the endings can be studied and mastered by themselves, the middle game and the opening must be studied in relation to the endgame." This end-game thinking is often the secret to a successful performance engineering project because we put ourselves, in the end, user's shoes and design a performance approach to validate the system requirements under a realistic production-like setting.

Performance requires early and continuous efforts to keep it at the needed level. The researcher Capers Jones pointed out that defect costs are low in the early stages [1] but up to 640 times higher if discovered in production. In Figure 1, we highlight the dependencies between introducing and resolving defects. When we bring defect resolution closer to coding, the costs could be reduced. At the same time, reputation and business risks increase when performance problems are found late or, in the worst case, in production. Performance must be part of the entire software development lifecycle.

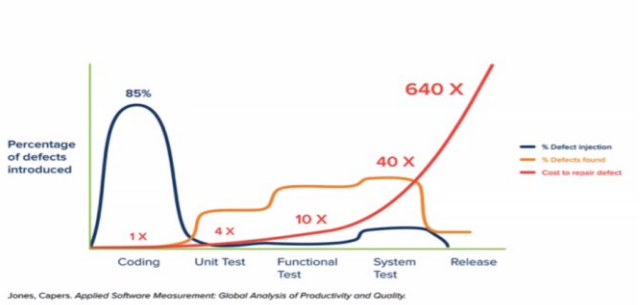

Fig. 1. Statistic from Applied Software Measurement [1] to demonstrate the dependency between defect introduction and defect resolution on software development costs.

3.2.1 Design for Performance

Technology alone is not a guarantee of success [2]. Jim Collins explained the role of technology in successful businesses. It seems the same is true for building fast and reliable business applications. We should not hope that the latest technologies guarantee reliable IT services. Instead, we must lay out realistic performance requirements and make them part of our early software design decisions.

3.2.2 Coding for Performance

When developers know performance requirements such as request volumes or response time expectations, they can integrate lazy loading, resilience, or caching concepts in their software components. At the same time, developers should be motivated to implement unit tests to validate the performance as part of their build processes. The benefit of these practices is that they identify and fix performance problems earlier and at a lower effort.

3.2.3 Testing for Performance

Performance must be validated, release by release. Any change comes with a performance risk. We can ignore or mitigate these

risks by running a predefined set of performance tests and validating our performance quality gates. This continuous performance validation creates a quality-first mindset and avoids expensive surprises when we deploy our new features to the user community.

These days, performance validations are highly automated and fully integrated into the deployment process. When performance testing is a lean and continuous process, we detect problems much earlier. Ideally, performance testing-related activities are scalable self-service, and we have performance-trending capabilities and dashboards in place. The chart in Figure 2 outlines how continuous performance testing supports early component-level performance tests as well as integration and end-to-end performance tests.

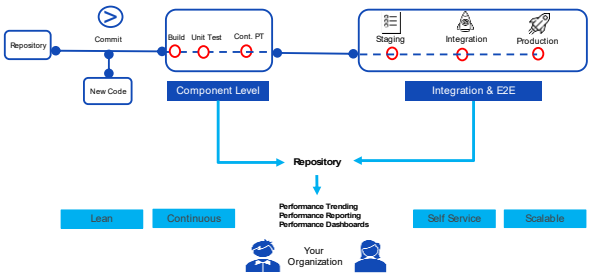

Fig. 2. A visualization of how performance testing can become a continuous process.

3.2.4 Operations for Performance

In our digitized world, IT services must sustain significant variations in user and data volumes. The expectations for fast and reliable business services grow, and when customers get frustrated due to performance issues, we see a loss in sales. It's more important than ever to identify the root cause of degradations and implement the remediations fast by keeping the mean time to repair (MTTR) low.

4 ESTABLISH A PERFORMANCE ENGINEERING CULTURE IN A LARGE BANK IN EUROPE

In 2019, performance engineering was absent from a large European bank; their customers complained, and regulatory agencies were on the doorstep to review how this bank ensures that only validated software is deployed to production. The Board launched a program to modernize their testing activities, including identifying and remedying gaps. Within a few days, it was evident that performance engineering was one of their challenges, and the team hired me to make performance a matter for everyone. In this section, I explain how we've established a culture of performance discipline in this organization.

4.1 Culture of Performance Discipline

Performance requires awareness and commitment from management from the first steps to the entire length of this investment. We tried to get everyone settled in and explained why, what, and how we planned to implement performance engineering. The following sections outline how we've implemented performance as a shared responsibility. From leadership to business and technical roles, performance became a matter for everyone.

4.1.1 Performance for Leadership

Empowered by line managers, the QA, Project, and performance lead conducted performance risk assessments. Depending on the outcome of these risk ratings, they engaged performance experts to validate their performance requirements.

The leadership team's role is to remind everyone involved in the software development process about the importance of reliability and performance. Any enterprise application changes are on the daily agenda. The leadership team provides the framework and rules for performance risk assessments. Supported by a Performance expert, they can review their risk assessment and plan meaningful performance tests to mitigate identified performance risks.

Fig. 3. The Leadership's roles and responsibilities to make performance a continuous process.

4.1.2 Performance for Business

Performance requires end-game thinking, so we've made business service owners, business QA, and testers responsible for performance requirements and their validation. The transformation at the business level, from what to build to how to build it, created much better awareness for performance considerations.

The focus changed from testing functionality to testing how the end customers will use the product. For each release performance requirement, risk assessment and load and performance testing became a fundamental, planned discipline, as outlined in Figure 4.

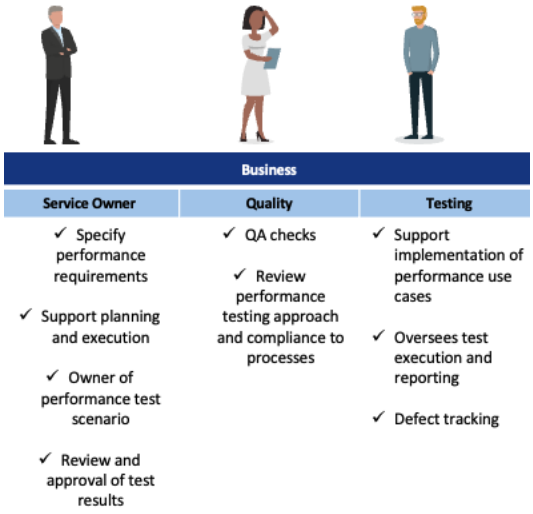

Fig. 4. The business teams' roles and responsibilities to make performance a continuous process.

4.1.3 Performance for Technical Experts

Business teams got support from technical services owners and developers. The technical service owner had a shared responsibility for performance, as outlined in Figure 5, and worked with developers on fixing identified performance issues. Once we've created this common understanding for performance, we looked at the processes in this bank. We intended to make performance a continuous activity without building large additional teams.

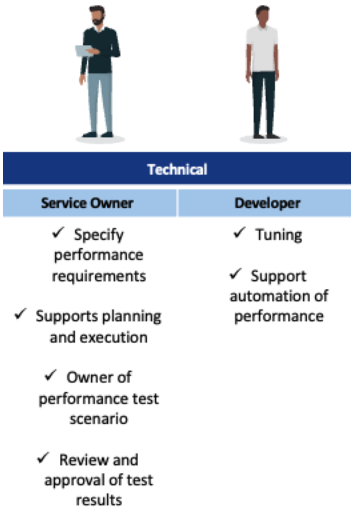

Fig. 5. The technical teams' roles and responsibilities to make performance a continuous process.

During the first few months, we focused on building a small core team for performance engineering. This self-motivated team designed a performance engineering framework as laid out in the performance as a value section 3.1.4 to clarify a few general rules, such as who is in charge, what the rules are, and how to validate the performance requirements. At the same time, we deployed a test lab, installed performance monitoring and load injection tools, and educated the teams on how to use our performance framework. In this project, our performance framework consisted of the following tooling:

- Maturity Assessment: Gobenchmark
- Load injection: Gatling and LoadRunner
- Script development: InteliJ
- CICD: Jenkins
- Version Control: Git
- Monitoring: Prometheus and Dynatrace
- Workload modeling: Performance Toolbox
- Reporting: Confluence
- Performance Board and Defect Tracking: Jira

4.2 Automation of Performance

Manual performance test execution and analysis is time-consuming and prone to human errors. Thanks to continuous integration solutions such as Jenkins and plugins from Dynatrace, we automated test executions and configured performance quality gates. As mentioned earlier, late detection of performance problems is expensive. By increasing the performance engineering maturity, we empowered this organization to capture the true value of performance. Figure 6 outlines the impact of the performance maturity level on organizations. When they improve their practices, they detect and solve performance problems earlier and reduce defect costs as mentioned in section 3.2.

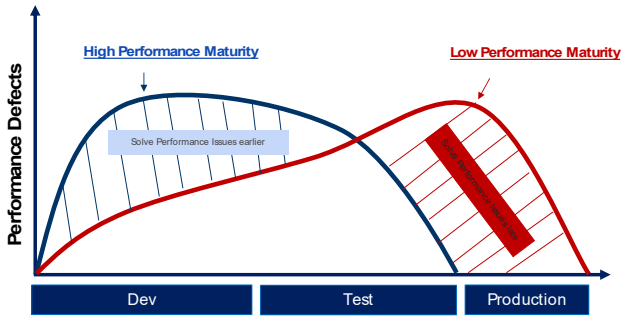

Fig. 6. A low-performance maturity level results in more performance problems detected in production and higher performance defect costs.

Looking back on this three-year project, I realize it was an incredible time. We established a culture of performance and prevented hundreds of performance issues from getting flawed to production.

4.3 Challenges in this Performance Engineering Project

We've created a culture of performance in this organization, which is the most important one from my perspective. If performance is in everyone's mind, the team will consider it during the entire software development process.

The five challenges below need to be solved from a technical standpoint.

1. Isolation of services and applications under test: Smaller test stages and reduced capacity on 3rd party services impacted performance test results.
2. Test data management: Performance tests generate huge data volumes. Assignment of test data sets and housekeeping is highly recommended.
3. Technical debts: The developers' workload is high, and they cannot include performance defects in their monthly sprints.
4. Workload models: New products result in difficult-to-predict transaction patterns. Workload modeling should be executed frequently in production to gain new insights for subsequent performance sprints.
5. Performance Engineering skills: To learn this science, more than introductory courses are required. Universities could integrate performance engineering lectures into their syllabus to educate the next generation of performance engineers.

5 PERFORMANCE ENGINEERING MATURITY

On the one hand, every organization could find its way to develop and operate reliable applications. On the other hand, we could share good practices and increase the chances that everyone would implement better business applications.

Based on our experience, one of the significant challenges is convincing leadership and creating awareness for performance engineering. To solve this business **problem,** we've invented [9] the performance engineering maturity knowledge model and implemented its algorithms in the Gobenchmark platform. At Gobenchmark, we combine human-AI-powered knowledge with qualitative analysis, making the unmeasurable measurable and bringing flexibility to changes in markets, customers, and technologies. Figure 7 outlines the core elements of Gobenchmark, which are:

- Advice from industry experts.
- Framework-based Analysis.
- Rating and comparison.
- Remediation.

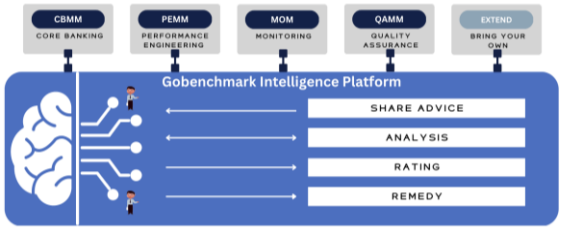

Fig. 7. The *Gobenchmark* platform, including its knowledge models and core features Advice, Analysis, Rating, *and Remedy.*

5.1 Advice from industry experts

Classic maturity models often fail because they do not adapt to changes. Collecting the latest information about methods and tools is crucial to avoid outdated knowledge. In Gobenchmark, we created a share advice catalog [6], allowing every industry professional to share good practices and hints about their solutions and how they rate their practices and tools. Furthermore, we store such advice in a flexible and reusable format to ensure that the built-in AI can utilize this information when creating recommendations for a customer's remediation plan.

5.2 Framework Analysis

At Gobenchmark, we have implemented framework-based analysis because it generates descriptive and explanatory conclusions. The interviewee walks through 27 questions structured in domains and practices. Each practice can be answered by choosing Always, Often, Rarely, or Never. After completing the assessment, Gobenchmark will calculate and present the performance engineering maturity score. Such ratings are easy to understand and allow a comparison to peers or industry standards. Gaps can be identified by showing how teams or organizations are rated, and remediation actions can be derived.

5.3 Rating and Comparison

We have no rating for the performance maturity of business applications or organizations. A high CMM level is no indicator of performant, secure, and well-designed IT services. Nevertheless, the outcome of our framework analysis in Gobenchmark can be transformed to a rating from C- to A and indicates how organizations or teams are adopting industry best practices.

By seeing the rating, we can identify blind spots, compare businesses to their peers, and create a remediation plan. In Gobenchmark, we show the rating immediately after the framework analysis, which creates essential benefits:

- We understand gaps much faster.
- We can focus our efforts on critical blind spots.
- We have everything we need to show a comparison to industry standards and peers.
- We can build the remediation plan based on identified gaps expressed by lower scores.

The performance engineering benchmark in our Gobenchmark platform is dynamic and will be re-calculated month by month.

5.4 Remediation

For performance engineering to work, it must take us on a journey where we learn concepts as we do things. Seeing gaps expressed by a rating does not solve these problems. If we leave organizations alone to solve these shortcomings, they might run into further issues, such as going in the wrong direction.

The AI-powered brain of Gobenchmark provides the expected guidance. It analyzes a customer's assessment results, incorporates knowledge from industry experts, and creates a remediation plan

that shows how organizations can reach the next level by improving their practices and methods and using better tools.

Our world is changing extremely fast, and we can't expect our current approach to work tomorrow or several weeks ahead. Knowledge from industry experts helps. However, we also see challenges in getting relevant insights from subject matter experts. For this reason, we've integrated large language models to acquire domain and practice-specific advice, which we incorporate into the AI-powered remediation plans to provide better customer mentorship.

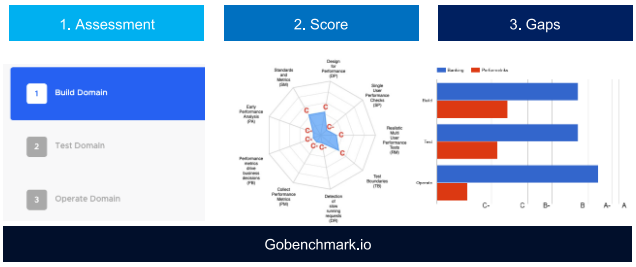

Fig. 8. In three steps from self-assessment through scoring and remediation.

6 OUTLOOK AND CONCLUSION

We believe that reliable business applications must become a commodity and should be achievable by every business. Today, the limiting factor [3] is mainly knowledge and awareness. Not knowing how to integrate performance or observability into value streams can set your business at risk. Organizations might spend too much time reinventing the wheel while their competitors adopt industry standards and dramatically reduce their development efforts.

To adopt performance engineering practices much faster and lower the risks involved, we propose the [9] "Performance Engineering Maturity Model." Figure 8 outlines how we use Gobenchmark in our performance engineering project. The three steps to improve the performance engineering maturity are:

1. Assessment to get guidance on integrating performance practices into organizations' value stream.
2. Scoring to raise awareness and highlight organizations' adoption of industry standards.
3. Remediation to solve technical and methodical gaps much faster and save time by avoiding reinventing the wheel.

The benefits of using Gobenchmark are:

- Safe time because we understand gaps and get a remediation plan within a few minutes.
- Reduce risks because we follow industry best practices.
- Simplify things because you get guidance along the way.
- Avoid DIY (Do it yourself) because Gobenchmark shares practice-proofed methodical and technical insights with us, so we no longer need to reinvent the wheel.
- Reduce costs because a higher maturity level helps our teams avoid expensive reliability issues in the first place.

Read more about the Gobenchmark platform on this page https://gobenchmark.io/.

ACKNOWLEDGMENTS

I thank the performance engineering community for sharing their knowledge. Also, I am very grateful for the hard work of our Performetriks team on developing products such as Gobenchmark to make performance engineering scalable and protect thousands of businesses from learning performance the hard way.

REFERENCES

[1] **Applied Software Measurement**, Capers Jones, 1996.
[2] Jim Collins. Good to Great. 2001
[3] Josef Mayrhofer. 2023. *Human-AI powered Strategies for Better Business Applications.* Performetriks, Minnesota. https://link.springer.com/chapter/10.1007/978-3-031-35734-3_26
[4] M. Woodside, G. Franks, D. C. Petriu. 2007. *The Future of Software Performance Engineering*. IEEE. https://ieeexplore.ieee.org/abstract/document/4221619
[5] Peter J. Pronovost, George J. Ata, Brent Carson, Zachary Gordon, Gabriel A. Smith, Leena Khaitan, and Matthew J. Kraay. 2022. *What Is a Center of Excellence.* Liebertpub. https://www.liebertpub.com/doi/abs/10.1089/pop.2021.0395?journalCode=pop
[6] Josef Mayrhofer: Performetriks.: https://www.performetriks.com/blog/categories/gobenchmark. (2023)
[7] José Raúl Capablanca,Good Reads Quotes, https://www.goodreads.com/quotes/650766-in-order-to-improve-your-game-you-must-study-the .
[8] E.W. Fleisch, Markus; Wortmann, Felix, Geschäftsmodelle im Internet der Dinge. HMD Praxis der Wirtschaftsinformatik. 51(6) (2014) 812-826.
[9] Josef Mayrhofer: Performetriks.: https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/11847597. (2023)