WHITEPAPER

SOFTWARE PRODUCT HEALTH ASSISTANT

From Complexity to Clarity: Understanding your Software Product's Health

Software Product Health Assistant

Fraunhofer IEM

ABOUT SPHA

Unleash your data's **full potential** by integrating it with SPHA.

Calculate a customizable and transparent health score based on a hierarchy of KPIs.

SPHA is a **fully automated** tool suite that assesses and communicates all aspects of software product quality. It does so by combining data about your projects from sources like ticketing systems, and static analysis tools.

A Competitive Advantage:
On the Path Towards Resilient Software

SPHA assists your teams in **identifying potential improvements** in their software and aligning it with your company's development processes.

01

To stay competitive in today's digital product landscape, it is essential to build resilient, high-quality software efficiently. Our goal is to assist product teams in achieving this objective while also helping stakeholders understand ongoing tasks and challenges. Achieving this goal is complicated by the increasing complexity of software systems as well as the faster development and deployment times. To keep up with this new speed and complexity and still come to realistic conclusions about software products it is necessary to define an automated and metric based approach.This requires collecting and evaluating data from various analyses, like linters, static and dynamic analysis tools, secret scanners, etc. and deriving an overall management summary. Until now this has been a complex, cumbersome, and often manual task, typically only performed late in the product development process as part of the product release.

Our **Software Product Health Assistant (SPHA)** alleviates this burden by automating the data-driven assessment of the software product health. Software product health is an overarching indicator that summarizes the condition of a software product based on the aggregation of data artifacts generated throughout its development process. Depending on the availability of the data, this software product's health assessment can be generated at every stage of the development process, providing timely feedback with low effort.

SPHA evaluates the health of software products through a hierarchy of key performance indicators (KPIs), which are calculated using a wide range of data sources across the software development process. These sources include code platforms like GitHub and GitLab, static application security testing (SAST) tools, and project management tools like Jira. By incorporating data from these and additional sources and not limiting ourselves to code metrics, SPHA provides a comprehensive and thorough assessment of the product's health.

Scan to go to the GitHub Repository!

Software Product Health

The **health score** of a product is calculated based on a weighted hierarchy of KPIs. This score can be customized for each product by modifying either the hierarchical structure or the respective weights assigned to the KPIs.

The six key aspects of **Software Product Health**

Within our KPI hierarchy, we differentiate between **six different aspects** that collectively assess the software product health:

1. Security: Security KPIs quantify a product's security risk based on available data, such as those provided by SAST tools. The risk is assessed by identifying known vulnerabilities in the product's dependencies, analyzing findings of established code-scanning tools, and information from further data sources.

2. Internal Quality: Internal quality KPIs quantify the quality of a product from an organization's internal perspective, that is, from the perspective of the product's developers and maintainers. These KPIs include and combine quality dimensions like maintainability, comprehensibility, code quality, and testability.

3. External Quality: External quality KPIs quantify the externally observable quality of a product by its users. Note that if your product is a software library or framework, the users are also developers. These KPIs include and combine quality dimensions like usability, reliability, performance, and fulfillment of functional requirements.

4. Sustainability: Sustainability KPIs quantify the sustainability of both the product and its development process. These KPIs primarily focus on the economic aspects of sustainability, such as code maintainability and the agility of the development process. They also incorporate environmental and social factors, like employee turnover rates or compliance with green coding practices.

5. Process Compliance: Process compliance KPIs quantify compliance with company-specific development process policies. These policies can be generally applicable paradigms, like protecting your release branch or organization-specific policies, whose compliance is monitored using specific KPIs.

6. Process Traceability: Process traceability KPIs are especially important for external parties interested in understanding how your product is developed. These KPIs highlight the transparency of your development process. For example, these KPIs assess the availability of data needed for detailed analysis of the preceding software product health aspects or whether the creation and contents of all used artifacts can be tracked.

Get started now:
SPHA GitHub action

All SPHA related code is **freely** available on GitHub.

02

As a straightforward starting point for SPHA, we have created a dedicated set of GitHub actions and reusable workflows designed to run SPHA in your CI/CD pipeline.

Our reusable workflows are particularly suitable for newer projects, as they come with two static analysis tools included by default. These tools are the osv-scanner, which checks a project's dependencies for known vulnerabilities, and trufflehog, which searches for leaked secrets. Both tools run in parallel within your CI/CD pipeline. Their results are first transformed by SPHA's transform action into raw value KPIs, which SPHA can then process further.

Once both tool executions are completed, SPHA's calculate step processes the generated raw value KPIs and computes the overall KPI hierarchy. As a result, our GitHub action uploads all tool results along with the KPI result hierarchy to the CI/CD system. If executed on a pull request, it generates a comment that textually summarizes SPHA's results for the analyzed branch. Moreover, we calculate the impact of the pull request on the target branch by determining and displaying the difference in health score before and after the pull request is merged.

To maximize SPHA's effect on your organization, it is essential to understand that every organization and product has unique requirements. Therefore, to provide real benefit to your organization, it is necessary to utilize SPHA's customizability and fit it to your needs.

For the definition of KPI hierarchies, we have adapted a process including guiding questions based on the "business understanding" and "data understanding" steps of the well established cross-industry standard process for data mining (CRISP-DM)[1]. When defining a customized KPI hierarchy for your organization, you should follow our process shown in Figure 1.

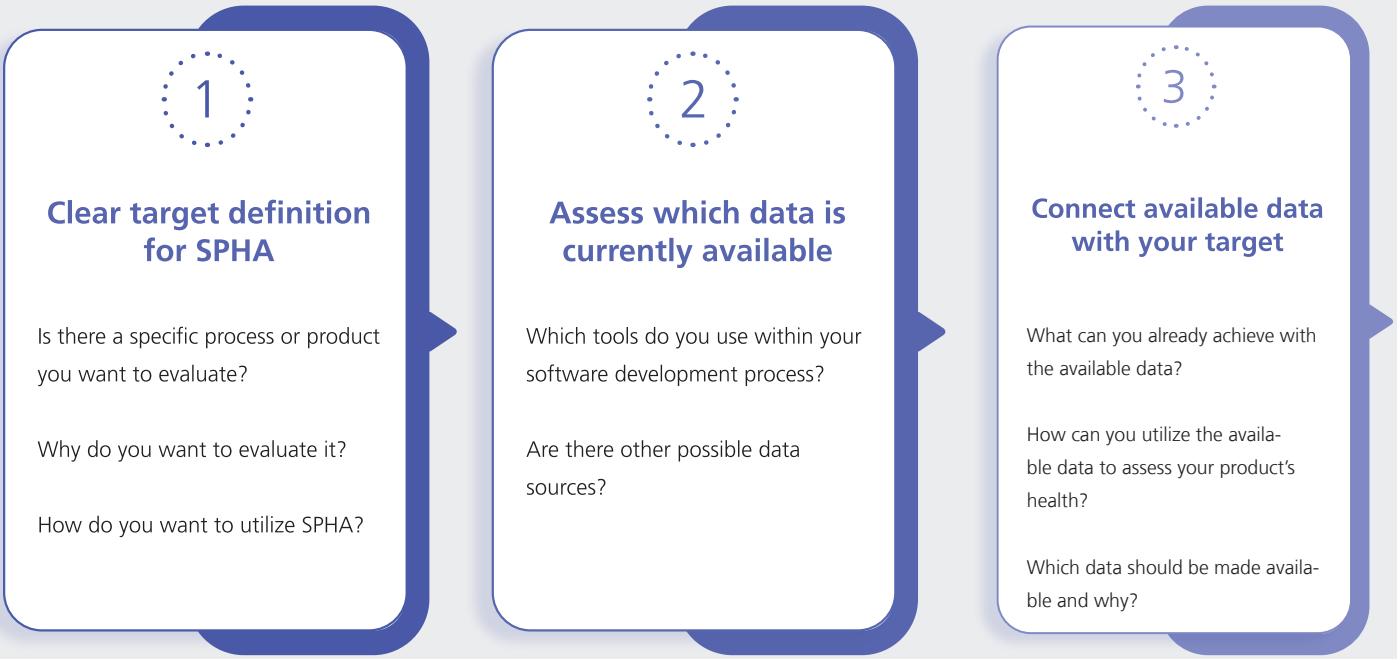

Figure 1: Process for the definition of a customized KPI hierarchy.

[1] Wirth, R., & Hipp, J. (2000, April). CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining (Vol. 1, pp. 29-39).

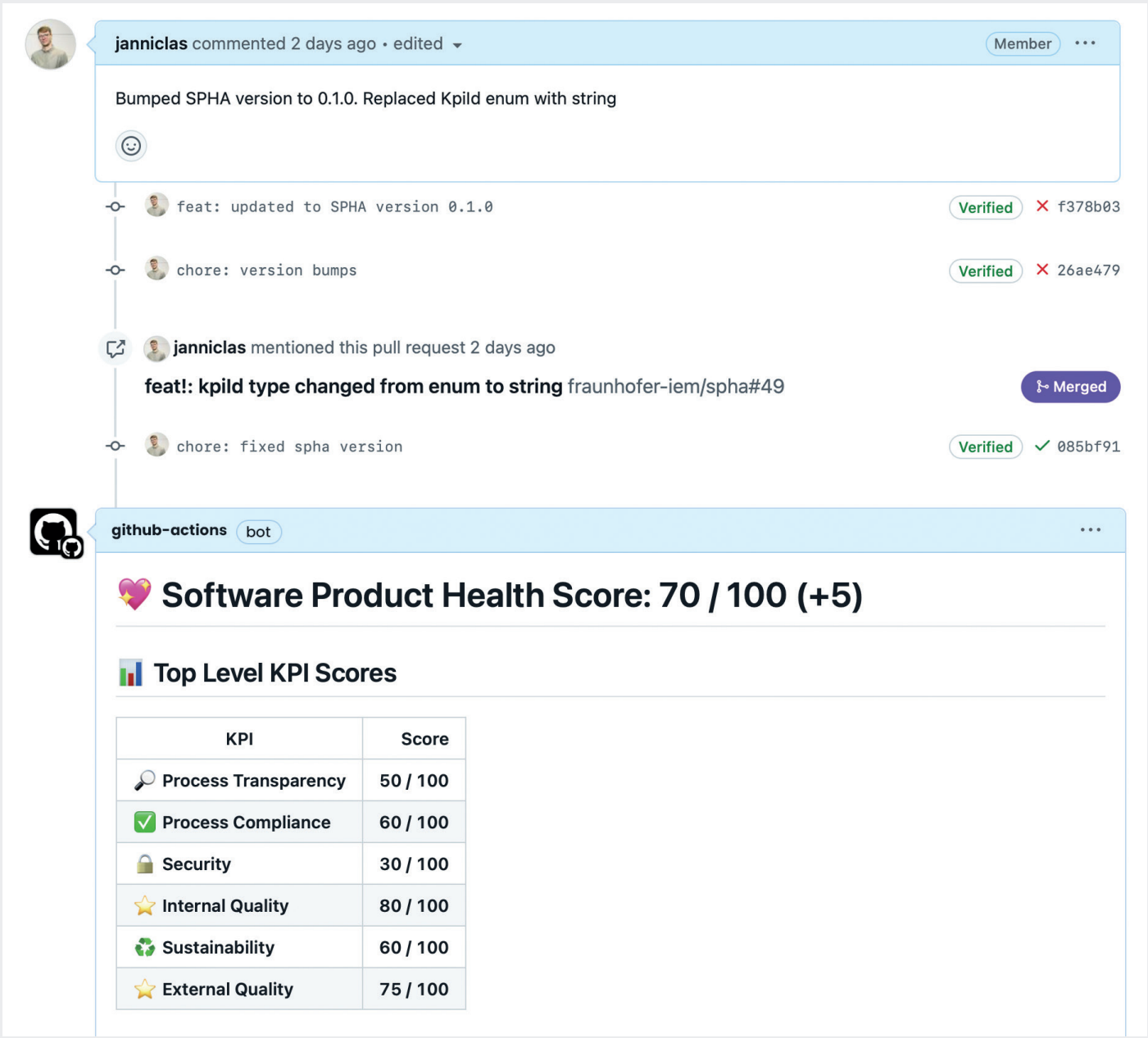

Figure 2: Automatically generated summary of SPHA's results as a comment on GitHub.

To the Github repository
https://github.com/fraunhofer-iem/spha

SPHA
and its use cases

SPHA helps to secure the software supply chain by evaluating third-party dependencies and supports compliance with evolving security regulations through **tailored** assessments.

03

SPHA is built on a domain-independent model for KPI-based process management[2], making it applicable to a variety of software development use cases. Currently, we are focusing on two main use cases: automated product assessment and continuous process improvement.

However, one of SPHA's key strengths is its full customizability, allowing it to reflect your product and organizational needs. This flexibility extends to various aspects, such as data sources and the KPI hierarchy, enabling you to tailor SPHA to fit your specific use case.

3.1. Transparency as a first step toward securing your Software Supply Chain

The software supply chain includes all the code, tools, and technology necessary for developing, building, releasing, and distributing a software product. It is crucial for companies to secure this supply chain. Third-party dependencies within your product can be a decisive security factor in the software supply chain. Therefore, it is imperative to have a sound understanding of your external dependencies.

When adding a dependency to your product, your team needs to consider the security and long-term maintenance of the third-party code being integrated. This means that your team must either be prepared to fix and maintain the dependency if its current maintainers abandon it or be ready to replace it in the future. To avoid these challenges, it is essential to choose well-maintained dependencies with a strong track record of addressing vulnerabilities and bugs as well as a high code quality. This approach contributes to a more secure software supply chain.

However, evaluating third-party code can be complex and time-consuming, even for experienced developers. By analyzing your dependencies, SPHA can assist your developers in making the right decisions more efficiently. We have a set of KPIs solely dedicated to third-party dependencies. For example, we check for known vulnerabilities in the used dependency versions, analyze previously found vulnerabilities and the time it took to fix them. Furthermore, we estimate the technical lag of your third-party dependencies, e.g., by calculating their libyears[3], which measure the age of a dependency as the difference between the release date of the currently used version and the newest version.

3.1.1. Taking it one Step Further: A SPHA-powered dependency management artifactory

In the previous section, we have described how to analyze dependencies from the outside by tracking known vulnerabilities or calculating technical lag. However, we can take things one step further by evaluating your product's third-party dependencies with SPHA's general-purpose KPI hierarchy. The general-purpose hierarchy covers various software product health aspects and is meant to be used on projects without further configuration or finetuning to provide a general overview of their health.

[2] Wohlers, B., Strüwer, J., Schreckenberg, F., Barczewicz, F., Dziwok, S. (2022). A Domain-independent Model for KPI-based Process Management. In: Beata Mrugalska (eds) Production Management and Process Control. AHFE (2022) International Conference. AHFE Open Access, vol 36. AHFE International, USA.

[3] J. Cox, E. Bouwers, M. van Eekelen and J. Visser, „Measuring Dependency Freshness in Software Systems," 2015 IEEE/ACM 37th IEEE International Conference on Software. Engineering, Florence, Italy, 2015, pp. 109-118, doi: 10.1109/ICSE.2015.140.

We developed such a general-purpose KPI hierarchy and implemented it on the Open CoDE platform.

Open CoDE

Open CoDE is a platform that includes a version control system accessible to all developers working in public administration throughout Germany. Our system utilizes the general-purpose KPI hierarchy to automatically evaluate projects on the platform in the different aspects of software product health. This provides developers with decision support and potential users with clear guidance, thereby building trust and encouraging the reuse of software components among administration, industry, and society. The same concept can be applied to a project's existing dependencies.

The primary benefit of using SPHA with this general-purpose KPI hierarchy to assess third-party dependencies is that it allows developers to monitor the detailed status of each dependency in all covered software health aspects consistently. This enables them to determine when maintenance is required or when it may be time to replace a dependency. This information can be aggregated and provided to all teams in a central artifactory.

3.2. Automated assessment through continuous measurements and transparency

The success of DevOps has shown organizations the value of having self-sufficient teams that embrace a culture of continuous learning to create the best products for their customers. These teams use short feedback loops to gather data, often during operations or through dedicated experiments, which helps them identify areas for improvement in their products. Just as these teams enhance their products based on customer feedback, we aim to use SPHA to help refine your development processes and assist them in building secure, high-quality software.

In the following, we will explore how SPHA can help you measure and enhance the security of your software products. Assessing software security is often challenging due to the numerous factors involved. This complexity is evident in the variety of software security maturity and capability models, as well as secure software development processes available. SPHA's flexible KPI hierarchy can be tailored to evaluate a product's compliance with an existing security process or assist in establishing one. Your teams can collaboratively commit security activities and objectives, using SPHA to automatically verify adherence to these goals. This is a bottom-up approach in which SPHA supports teams in tracking their self-defined security best practices.

Additionally, SPHA can facilitate and guide both internal and external review processes, especially those related to software releases. This is a top-down approach, which can be beneficial for companies developing software for critical infrastructures, which have stringent security requirements defined by external entities.

However, with the introduction of new legislation, such as the EU Cyber Resilience Act[4] and the Securing Open Source Software Act of 2022[5], all software development companies are now facing increased scrutiny regarding software security. This makes SPHA a perfect fit for your teams and auditors to monitor your product's status in relation to the evolving security requirements.

As a central collection and processing point for all relevant information regarding the development process, including test and tool results, SPHA can also be utilized to prepare audits. This allows both legal compliance checks and specific certifications to be simplified by providing all relevant information in a transparent, easy-to-understand report, generated by SPHA.

[4]https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act
[5]https://www.congress.gov/bill/117th-congress/senate-bill/4913

Maximal flexibility:
the customizable KPI hierarchy

SPHA's **core strength** is its customizability that enables us to fit it to every organization.

04

SPHA's core library automates the calculation of the project's health score based on a given KPI hierarchy and project-specific raw values from various data sources. Figure 1 shows a simplified example of a KPI hierarchy. Each node in the hierarchy has a KPI calculation strategy that defines how to compute its score on a scale from 0 (worst) to 100 (best), factoring in values from its child nodes. The edges connecting the nodes have weights that modify the influence of the connected node in the calculation. The values of the leaf nodes are derived from so-called raw values, which come from tool results or other automatically processable data sources. SPHA has a set of predefined tool adapters that transform given tool results into raw values, which SPHA's core can then process. Additionally, SPHA implements a plugin system to enable easily adding new tool adapters.

4.1. KPI Calculation Strategies

The following provides an overview of SPHA's KPI calculation strategies and their application using the exemplary KPI hierarchy shown in Figure 3.

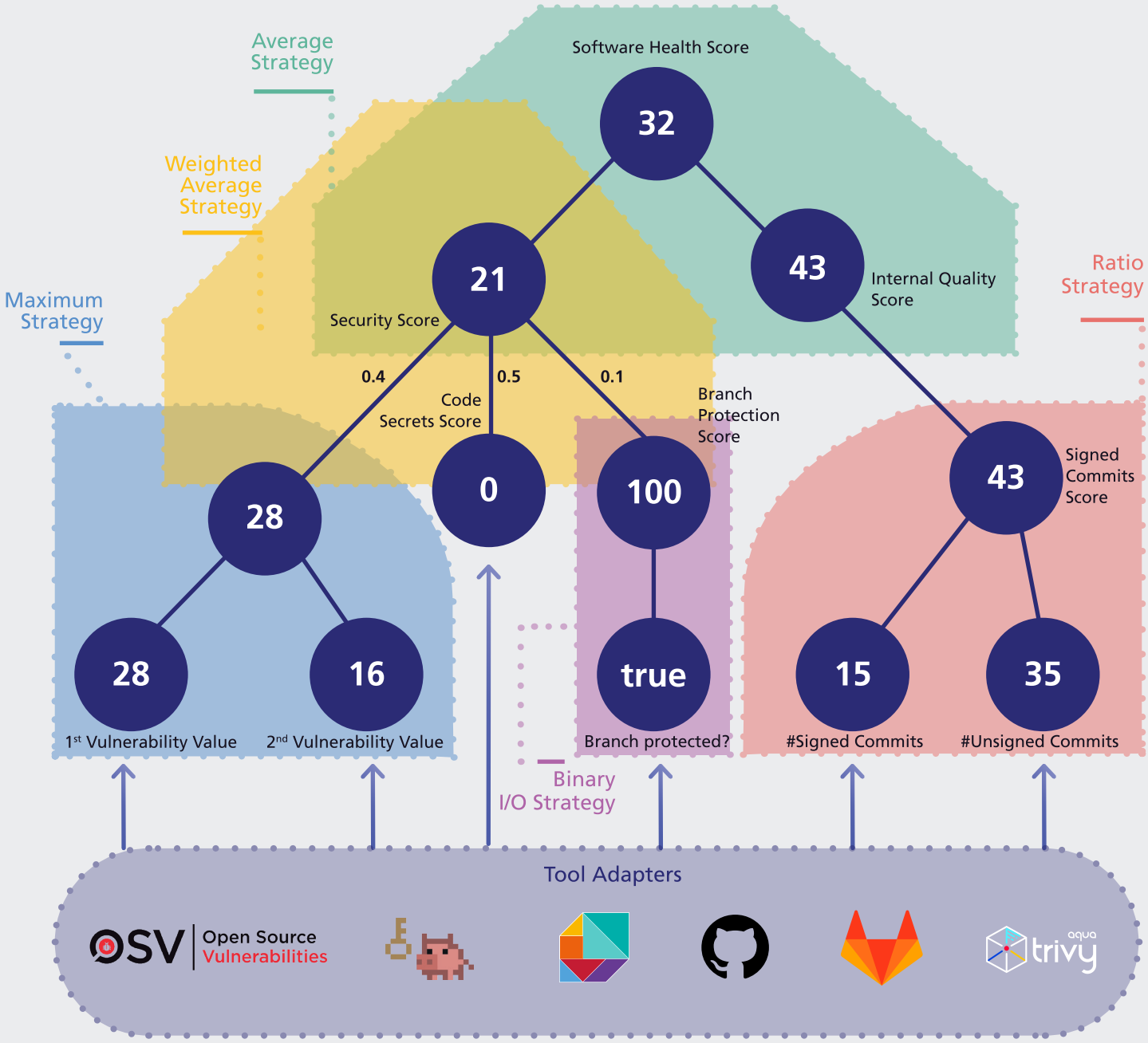

Figure 3: Example KPI hierarchy including raw values, highlighting different calculation strategies.

Strategy	Description
Average	Determines the KPI value as the arithmetic mean of all inputs. **Formula:** $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ Example(s): Calculation of the Software Health Score $\frac{21+43}{2} = 32$
Weighted Average	Determines the KPI value as the weighted sum of all inputs. Note that the sum of weights should always be 1. **Formula:** $\text{Score} = \sum_{\forall i \in \text{input}} \text{value}_i \times \text{weight}_i$ Example(s): Calculation of the Security Score as $0.4 \times 28 + 0.5 \times 0 + 0.1 \times 100 = 21.2 \approx 21$
Weighted Maximum/Minimum	Determines the KPI value as the highest/lowest value of all chosen weighted inputs and propagates the non-weighted input. Note that the weights must be defined beforehand and that their sum should always be 1. **Formular:** $\text{Score} = \max(\forall \in \text{input*weight})$ $\text{Score} = \min(\forall \in \text{input*weight})$ Example(s): $\max([(28, 0.4), (16, 0.6)]) = \max(28*0.4 = 11.2, 16*0.6 = 9.6) =$ 11.2 propagate 28 as KPI score
Maximum/Minimum	Determines the KPI value as the highest/lowest value of all inputs. **Formular:** $\text{Score} = \max(\forall \in \text{input})$ $\text{Score} = \min(\forall \in \text{input})$ Example(s): Calculation of the Vulnerability Score as $\max([28, 16]) = 28$ $\min([28, 16]) = 16$

Strategy	Description
Binary – I/O	Determines the KPI value as the weighted sum of all inputs. Note that the input is assumed to be binary – that is, with only two values (true or false resp. 100 or 0). **Formular:** $\text{Score} = \begin{cases} \text{true} = 100 \\ \text{false} = 0 \end{cases}$ Example: Calculation of the Security Score as $\text{true} = 100$
Binary – AND	Determines the KPI value based on the logic operator AND for multiple binary inputs. If all inputs show the value "true" or 100, "true" or 100 is passed on to the higher-level node. Note that the input is asNote that each input is assumed to be binary – that is, with only two values (true or false resp. 100 or 0). **Formular:** $\text{Score} = \wedge\forall \in \text{input}$
Binary – OR	Determines the KPI value based on the logic operator OR for multiple binary inputs. If at least one of the inputs shows the value "true" or 100, "true" or 100 is passed on to the higher-level node. Note that each input is assumed to be binary – that is, with only two values (true or false resp. 100 or 0). **Formular:** $\text{Score} = \wedge\forall \in \text{input}$
Binary – XOR	Determines the KPI value based on the logic operator XOR for exactly two binary inputs. Forwards "true" or 100 to the higher-level node if only one of the two inputs has this value. Note that each input is assumed to be binary – that is, with only two values (true or false resp. 100 or 0). **Formular:** $\text{Score} = (i_1 \wedge \neg i_2) \vee (\neg i_1 \wedge i_2)$

Strategy	Description
Ratio	Determines the KPI value as the weighted sum of all inputs. Note that the KPI can only be calculated for exactly two inputs and that the denominator must be larger or equal to the numerator to provide a value in the range {0, ..., 100} **Formular:** $\text{Score} = \begin{cases} \frac{\text{numerator}}{\text{denominator}} \times 100, \text{ if denominator} > 0 \\ 0, \text{else} \end{cases}$ Example: Calculation of the Signed Commits Score as $\frac{15}{35} \times 100 = 42{,}9 \approx 43$
Weighted Ratio	Determines the KPI value as the weighted ratio between numerator and denominator.Note that the input Note that the KPI can only be calculated for exactly two inputs and that the denominator must be larger or equal to the numerator to provide a value in the range {0, ..., 100} **Formular:** $\text{Score} = \begin{cases} \frac{\text{numerator*weight}}{\text{denominator*weight}} \times 100, \text{ if denominator} > \\ 0, \text{else} \end{cases}$ Example: Note that the KPI can only be calculated for exactly two inputs and that the denominator must be larger or equal to the numerator to provide a value in the range {0, ..., 100} $\frac{15*0.3}{35*0.7} \times 100 = 16{,}9 \approx 17$

4.2. KPI Hierarchy Serialization

The KPI hierarchy can be serialized and stored using a publicly available JSON schema. Figure 3 shows an example of a serialized KPI hierarchy. By storing the serialized hierarchy alongside the code in the project's version control system, we can track possible changes to our calculations. This can be helpful, e.g., to compare historical data for your project.

Users > janinedeniseredecker > Desktop > {} JSONSchema > ...

```
{ "rootNode": {
    "kpiId": "ROOT",
    "kpiStrategyId": "AGGREGATION_STRATEGY",
    "edges": [
      {
        "target": {
          "kpiId": "SECURITY",
          "kpiStrategyId": "AGGREGATION_STRATEGY",
          "edges": []
        },
        "weight": 0.4
      },
      {
        "target": {
          "kpiId": "PROCESS_COMPLIANCE",
          "kpiStrategyId": "AGGREGATION_STRATEGY",
          "edges": []
        },
        "weight": 0.3
      },
      {
        "target": {
          "kpiId": "INTERNAL_QUALITY",
          "kpiStrategyId": "AGGREGATION_STRATEGY",
          "edges": []
        },
        "weight": 0.3
      }
    ]
  },
  "schemaVersion": "1.0.0" }
```

Figure 4: JSON Schema of a serialized KPI hierarchy.

Architecture
& Deployment

SPHA can be deployed both as a **command line tool** and as a **standalone service** to support tools that are not part of your CI/CD.

05

We currently support two deployment scenarios for SPHA.

The first scenario involves using SPHA as a command-line tool. This tool can be run manually on your development machine for quick product checks or integrated into a build pipeline to run automatically with every pull request.

The second scenario involves deploying SPHA as a service within your organization's infrastructure. This setup allows for advanced use cases, such as trend analysis for each project, integration with external data sources like ticketing systems or network storage, and the use of server-based analysis tools.

5.1. Command Line Tool

SPHA can run as a standalone command-line tool, locally on your machine or inside your project's pipeline, as outlined in Figure 4. SPHA's command-line tool accepts a KPI hierarchy and tool results as input. It uses SPHA's core library to transform the given tool results into raw value KPIs and then calculate the project's health score using the KPI hierarchy and the raw value KPIs. The command line tool's output is a JSON file containing a serialized KPI hierarchy in which every node is annotated with its calculated score. We provide a GitHub action to easily integrate SPHA's command line tool into an existing build pipeline.

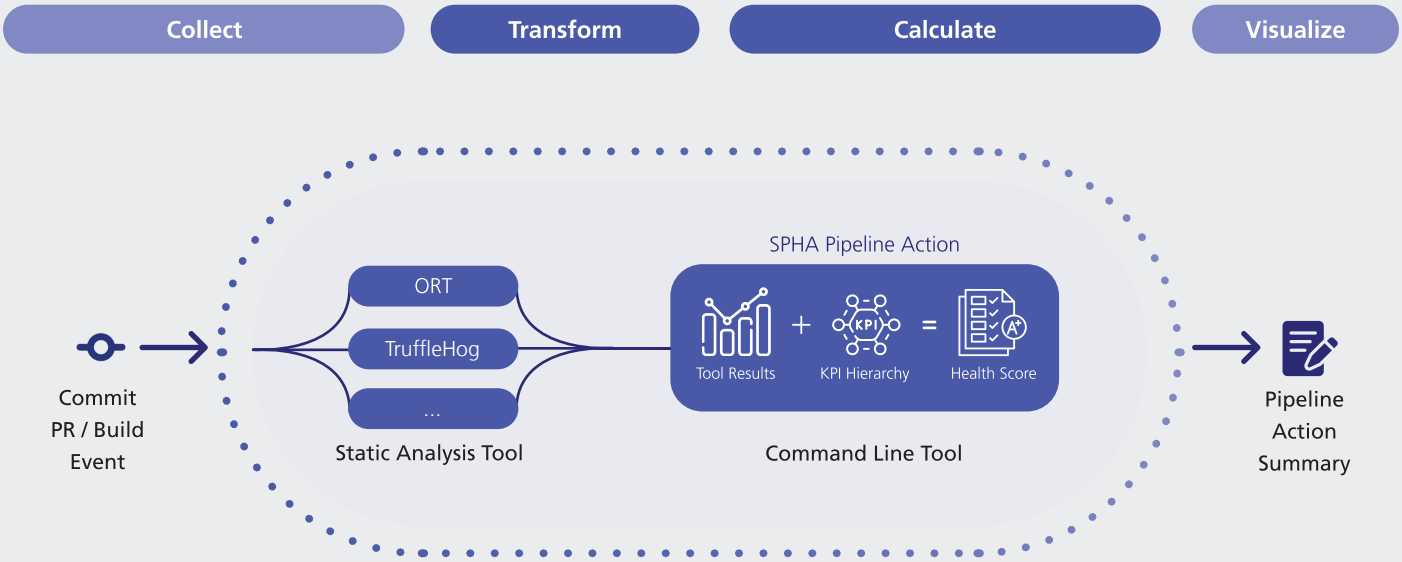

Figure 5: SPHA command line tool inside a build pipeline.

GitHub GitLab CI/CD

5.2. Service

SPHA can be deployed and run as a standalone service connected to your CI/CD pipeline. This deployment setup is similar to the one we have successfully used to deploy SPHA on Open CoDE. As a service, SPHA can utilize information about the project that is not directly generated or accessible within the build pipeline. For example, many SAST tools operate as standalone services rather than being executed directly in the build pipeline. This limits SPHA's ability to access and process their results directly during the pipeline execution. Consequently, to synchronize the calculated KPIs with the results from external tools, SPHA must also function outside the build pipeline.

In this configuration, the build pipeline sends a notification to the SPHA backend. This notification contains information to identify the project, and any results generated within the build pipeline. The SPHA backend then checks registered SAST services to determine if they are analyzing the specified project and retrieves their results. Similarly, SPHA can query and utilize results from other external services, such as ticketing systems or network storage.

Based on the collected data, the SPHA backend uses the same core library to calculate the project's health score and stores it in a database. Once the health score is calculated, SPHA updates the pull request check, if applicable. This setup allows us to create the most accurate representation of a product's current health, with all relevant information available alongside the code. Additionally, we provide dedicated visualizations for the calculated KPIs, ensuring a clear understanding of the results for all stakeholders.

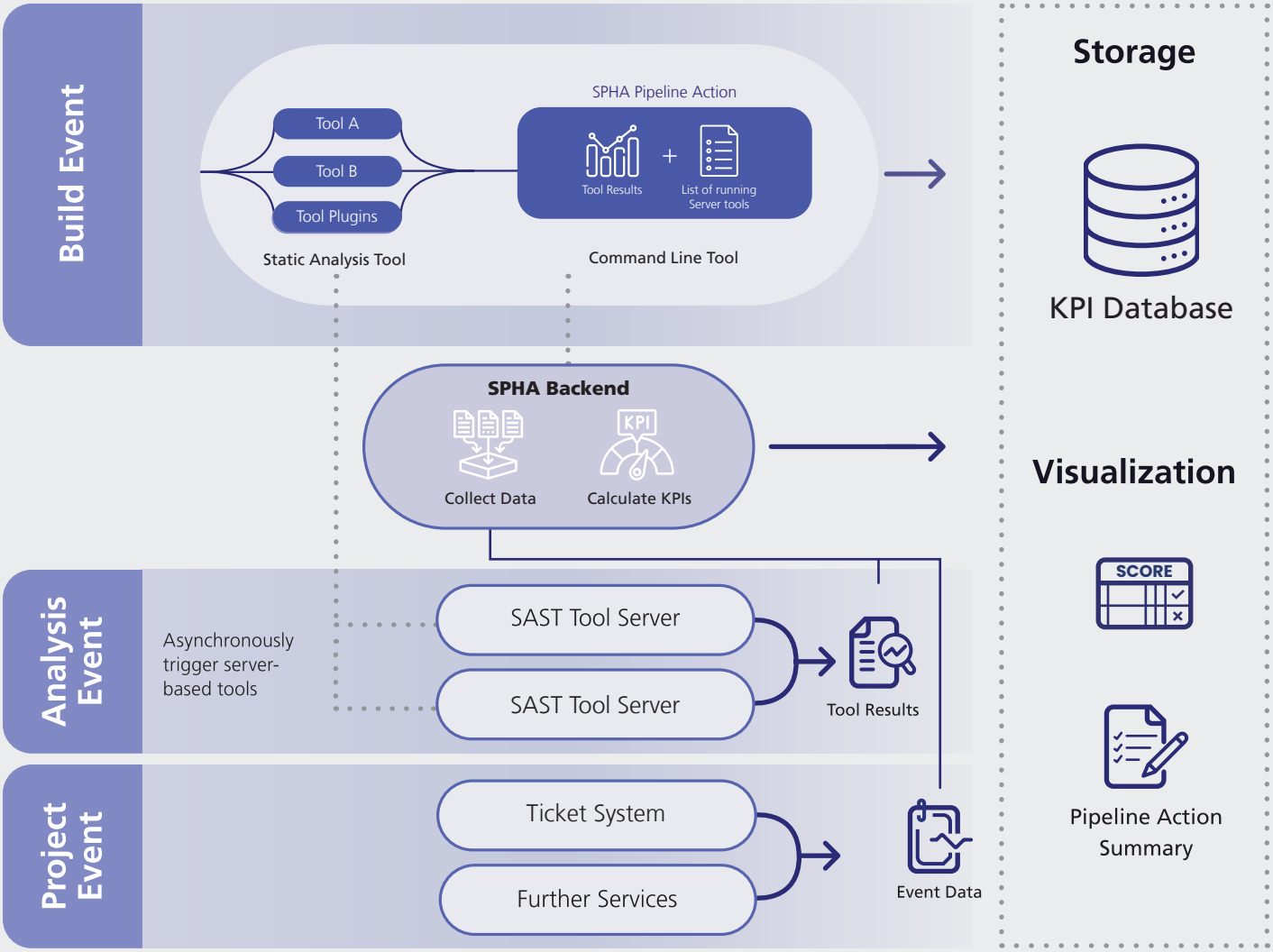

Figure 6: SPHA as a service.

Summary

SPHA can be deployed both as a **command line tool** and as a **standalone** service to support tools that are not part of your CI/CD.

06

The Software Product Health Assistant (SPHA) simplifies the assessment of software product health by automating the collection and evaluation of data from various sources, such as static analysis tools and project management systems. Its main strength lies in a customizable hierarchy of key performance indicators (KPIs), which allows organizations to tailor the tool to their specific needs.

SPHA provides a comprehensive view of software health across multiple dimensions, including security, quality, sustainability, and compliance. This empowers teams to make informed decisions that enhance their software development processes while aligning with business objectives and regulatory requirements.

By enabling continuous and automated health assessments throughout the software lifecycle, SPHA ensures that teams receive relevant, actionable insights that drive improvements in software quality. This flexibility helps organizations strengthen their software supply chain, while addressing both internal and external quality concerns. In today's competitive digital landscape, SPHA's comprehensive approach offers a significant advantage in maintaining resilient, high-quality software.

Benefits of SPHA

Leverage SPHA's customizability by defining a KPI Hierarchy that aligns with your specific **software development goals and processes**. Integrating SPHA into existing CI/CD pipelines will **maximize** its impact.

Implement SPHA as a **continuous assessment tool** to provide ongoing insights into your product's health. This will facilitate proactive management of software quality and security.

Utilize SPHA to ensure compliance with evolving security standards and laws, such as the **Cyber Resilience Act (CRA)** or **Digital Operational Resilience Act (DORA).**

Explore SPHA's potential beyond basic software product health assessment by using it to **drive continuous process improvement** and foster a culture of transparency and accountability within your development teams.

Imprint

Fraunhofer Institute for
Mechatronic Systems Design IEM
Zukunftsmeile 1
33102 Paderborn

Project Website
www.software-product.health

Publisher
Fraunhofer Institute for
Mechatronic Systems Design IEM
Zukunftsmeile 1
33102 Paderborn

Picture Credits
Cover picture: Adobe Stock / arhendrix
Page 13: GitHub / GitLab / OSV /
TruffleHog / ORT / aqua trivy Logo
Page 19: GitHub / GitLab Logo

Authors
Jan-Niclas Strüwer, Benedict Wohlers,
Hutomo Saleh, Matthias Becker, Eric Bodden

Auflage: 200 Stück

Design
Janine Denise Redecker

Contact

Jan-Niclas Strüwer

Business Developer SPHA &
Research Associate

jan-niclas.struewer@iem.fraunhofer.de

@struewer

Benedict Wohlers

Group Lead &
Research Associate

benedict.wohlers@iem.fraunhofer.de

@benedictwohlers

Software Product Health Assistant

Fraunhofer IEM