Tailoring the NFR Framework for Measuring Software Ecosystems Health

Simone da Silva Amorim[1], Sandro Santos Andrade[1], John D. McGregor[2], Eduardo Santana de Almeida[3], and Christina von Flach G. Chavez[3]

[1] Federal Institute of Education, Science and Technology of Bahia, Bahia, Brazil
simone.amorim@ifba.edu.br, sandroandrade@ifba.edu.br
[2] Clemson University, South Carolina, USA
johnmc@cs.clemson.edu
[3] Federal University of Bahia, Bahia, Brazil
esa@dcc.ufba.br, flach@ufba.br

Abstract. A healthy software ecosystem is capable of maintaining productivity and attractiveness, even in the face of problems and disruptions. Some studies have used software metrics to measure the health of a software ecosystem; however, there is little agreement on how to measure those aspects related to software architecture nor how to weigh their influence on the health state of the ecosystem. This paper introduces an approach to measuring and assessing the state of a software ecosystem's health that is aware of the architectural practices used. The approach uses a variation of the *softgoal interdependence graph* belonging to NFR (Non-Functional Requirements) framework on goal modeling. The key idea is to model and estimate influences of architectural practices on the health indicators. This research carried out an exploratory case study in the KDE ecosystem. KDE architectural practices were identified and analyzed with the support of our proposed practice-aware approach. The findings present the measurable influences that can be used to support decision-making processes related to architectural practices.

Keywords: Software Ecosystems · Software Architecture · Non-Functional Requirements · NFR framework · Health Measures.

1 Introduction

The quality of a software architecture, which will be used as the basis for many different systems, is critical to the success of those systems. In recent years organizations have been joining together to form communities and ecosystems to face problems and achieve common goals. The engagement in a software ecosystem allows participants to share problems, solutions and benefits among members of the community. Changes to the utility of a software architecture emerge much like the health of a person changes in response to changes in the environment.

A software ecosystem "refers to a collection of software products that have some given degree of symbiotic relationships" [14]. Organizations can take advantage of group work and launch new products faster than if they were working individually. Immersed into an environment of collaboration but also of competition, many participants have grown quickly and kept their success for several years [1, 2]. This success can be represented partially by the health state of the software ecosystem.

The health of a software ecosystem has been defined by Iansiti and Levien as "the growing and continuity of the software ecosystem remaining variable and productive over time" [9]. Awareness of the health state of a software ecosystem is relevant to support decision-making in different areas such as business processes, software design, and social interactions. Stakeholders can decide to enter or leave the ecosystem, or to keep their participation. Iansiti and Levien proposed three health indicators, robustness, productivity, and niche creation to be used in health evaluation of software ecosystems [9].

The health evaluation of a software ecosystem is not a trivial activity and faces complex and challenging tasks. There are obstacles in defining which metrics can be measured, how to work with highly abstract metric, and so on. Most of existing proposals [9, 12, 8] propose metrics to gather values that may relate to or impact health indicators.

Previously we proposed an approach to evaluating the health of open source software ecosystems based on the identification of technical, social and business practices [4] adopted by the software ecosystem and the assessment of their influence on health indicators [9]. This work proposed a variation of the *NFR Framework* [15] [5] that introduced the concept of *practices* [11]. The *NFR Framework* is a process-oriented framework that supports a qualitative treatment of Non-Functional Requirements (NFRs) and relates them to design decisions taken during software development, with the support of Softgoal Interdependency Graphs (SIG) [7]. Our approach *Health Evaluation for Software Ecosystems/-Software Architecture (Heval/SA)* uses a variation of the softgoal interdependence graph belonging to NFR framework. We introduced *SIG with Practices*, hereafter SIG-P, that models influences of architectural practices on the health indicators. Our ongoing work *Heval/SA* is initially composed by for two steps: Softgoal Designing and Influence Measuring. Practices used in the KDE ecosystem were collected from different sources, and preliminary versions of SIG-P were drawn for each architectural key area.

Aiming to support a quantitative way to estimate the degree of influence of architectural practices on the health indicators, we also adapted the idea of measuring on SIG-P levels proposed by Subramanian *et al.* [17]. We conducted a case study to validate our approach with the KDE ecosystem. Forty four architectural practices were collected after interviews with KDE developers and organized around 3 keys areas related to software architecture. KDE architectural practices were represented with SIG-Ps. They supported the measures achieved for the KDE architectural health.

This paper is organized as follows. Section 2 presents related work. Section 3 presents background on architectural health. Our approach is described in Section 4. Section 5 presents the methodology. Results are described in Section 6. Lessons learned are presented on Section 7. Section 8 describes the limitations of the approach used. Finally, Section 9 provides some concluding remarks and future work.

2 Related Work

This section presents related work that addresses the health evaluation of software ecosystems and different perspectives of applications for the *NFR Framework*.

Starting with Health of Software Ecosystems, Jansen's work [12] introduced the Open Source Ecosystem Health Operationalization (OSEHO) framework, a new approach to measuring health using the three health indicators proposed by Iansiti and Levien [9]. Jansen collected a set of measures from literature and refined to choose relevant metrics to his model. Jansen's work [12] covered large aspects of the health of software ecosystems, but also had problems to handle metrics that could not be extracted effectively. All projects reported on lack of data, abstract metrics that could not be measured, and metrics with ambiguous definitions. His work stumbled on the difficult of obtaining practical data to work with several metrics, posing questions on how to get effective data and how to use data productively.

An approach to improving and preserving the health of hardware-dependent software ecosystems with a governance model was proposed by Wnuk [19] to investigate the governance model activities focusing on the health indicators. For each indicator, governance activities were evaluated depending on to what degree the activity was performed as: yes, no or partially. This way, they gathered the degree of contribution of these activities to achieve health. Their model showed relationships between activities and health indicators.

Lastly, Franco-Bedoya *et al.* [8] proposed a quality model to evaluate important quality characteristics as the health of open source software ecosystems. From a systematic literature review, they classified and reorganized a set of 68 measures to incorporate them into the QuESo model. Their model considered grade quality characteristics and measures to achieve the health state. Similarly to Jansen's work [12], they reported that unavailability of some measure values hindered the creation of a comprehensive quality assessment process.

Our work shares some aspects with related work on health evaluation: focus on the health of open source software ecosystems, use of health indicators, and the concern with handling the influence of practices on these indicators. However, our work differs from those in the literature and provides a process-oriented approach to health evaluation based on the *NFR Framework* [15].

Considering related work to NFR Framework, in 1992, Mylopoulos, Chung and Nixon introduced the *NFR Framework* [15] as a process-oriented framework that provided a qualitative treatment of non-functional requirements, and related them to design decisions taken during the software development process. These design decisions could affect positively or negatively NFRs [15, 5]. They introduced the concept of "softgoal" to denote an ill-defined goal, and introduced the *Softgoal Interdependency Graph* (SIG) to represent whether a design decision contributed positively or negatively to satisfying a particular goal or requirement, under the architect or developer's perspective. Other authors have adapted and discovered new uses for the NFR Framework [17, 18, 16, 13].

Subramanian *et al.* introduced the Process-Oriented Metrics for Architecture Evolvability (POMSAE) approach to develop architectural evolvability metrics and analyze the cause of strengths and weaknesses for the metrics [17]. Their work developed a qualitative framework to generate quantitative numbers for the NFRs. The *POMSAE framework* encompassed a set of softgoals, methods, correlation rules, and metrification schemes to map labels to numbers. Subsequently, Subramanian *et al.* used the *NFR Framework* to quantitatively evaluate the safety and security properties for cyberphysical systems [18], and proposed the degree of satisfaction as a measure for the satisfaction of stakeholders with respect to requirements for safety and security.

Ruiz-López *et al.* proposed a pattern-based approach to clarify and capture NFRs such as usability or privacy for ubiquitous computing [16].

Finally, Mehta *et al.* applied the *NFR Framework* to analyze smartphone applications. They proposed a goal-oriented approach to explore and select alternatives to satisfy NFRs specially for the particular characteristics of smartphones [13].

Most of the related work described above focus on different uses for the *NFR Framework* and benefit from the support given by the *NFR Framework* to mapping and analyzing influences in diverse situations. Our work adapts the original *NFR Framework* [15, 5] and part of the *POMSAE Framework* [17] and focuses on qualitative and quantitative analysis with the support of SIG-P.

3 Architectural Health of Software Ecosystems

Years ago, Iansiti and Levien introduced the concept of health of software ecosystems [9]. Since then, several researchers have developed efforts to achieve forms to measure this health. In previous work, we proposed the idea to study one part of this health – the portion concerning the impact of the software architecture on the health. With this in mind, we advocate that the *architectural health of software ecosystems* "is composed of the parts of the ecosystem health influenced by architectural issues"[3]. This definition is supported by the belief that the creation and maintenance of the software architecture *"weigh"* directly on the behavior of the health indicators.

In addition, we observed that the software architecture is designed under several influence factors such as: requirements, time, goals, experience of the architect, resources, and so on. Leading with these factors, architects develop a series of practices to maintain the architecture. They provide a picture of a specific aspect of software development describing outcomes, and how to achieve them. Moreover, these practices face specific parts of the problem inside established limits, allowing to measure the success of their application [11]. Even more, the use of a practice causes impact in different parts of the ecosystem, including contributions positive or negative to the state of the health indicators[3].

Table 1. Types of Non-Functional Requirements from ISO/IEC FDIS 25010

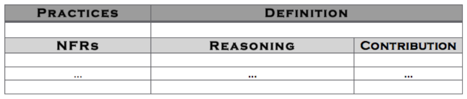
Types of NFR	Description
Functional Suitability	Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions
Performance Efficiency	Performance relative to the amount of resources used under stated conditions
Compatibility	Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment
Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use
Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period
Security	Degree to which a product or system protects information and data so that persons or other products or systems have the data access appropriate to their types and levels of authorization
Maintainability	Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers
Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

Based on these observations, we organized these architectural practices into key areas to help development and foster comprehension of their effects. The architectural key areas are: (i) Architectural Knowledge; (ii) External Management; (iii) Choice of Technology; (iv) Resources Management; (v) Design-Making; (vi) Quality Management, and (vii) Change Management.

4 Health Evaluation for Software Ecosystems

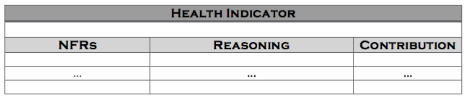
The *NFR Framework-with-Practices* [4] is an extension to the *NFR Framework* [15, 5, 6], a goal-oriented framework used to represent and analyze NFRs.

The *NFR Framework-with-Practices* supports health evaluation of software ecosystems based on the practices they use. Such evaluation relies on SIG-P models [4], that is, SIG models [6] extended with practices – that represent the influence of practices on non-functional requirements of concern and health indicators [9]. The use of practices should be properly assessed by means of measures of their successful application [11]. Our *NFR Framework-with-Practices* represents: (i) the influence of a practice on every NFR of concern, and (ii) the influence of a NFR on each softgoal on the top level of the SIG-P. In the context of software ecosystems, softgoals are productivity, robustness, and niche creation, the three ecosystem health indicators [9].



Practices	Definition	
NFRs	Reasoning	Contribution
...	...	...

(a) Form A



Health Indicator		
NFRs	Reasoning	Contribution
...	...	...

(b) Form B

Fig. 1. Forms A and B

The present work introduces the *Heval/SA*. It brings a refined version of our first study [4]. In this version, SIG-P uses non-functional requirements defined and elaborated by the International Standard ISO/IEC FDIS 25010[4]. Such standard defines a quality model encompassing eight important characteristics to software products and computer systems, that "provide consistent terminology for specifying, measuring and evaluating system and software product quality" [10]. These eight characteristics stand as the non-functional requirements of concern used by our approach (Table 1).

To evaluate the architectural health of a software ecosystem with the support of two steps of the *Heval/SA*, an architect or experienced developer must be designated to play the role of *evaluator*. She is expected to perform two steps: Softgoal Designing and Influence Measuring. Softgoal Designing are composed by two activities: (i) Analysis of architectural practices and (ii) Representation with SIG-P graphs. Influence Measuring is composed by measurement activity.

4.1 Softgoal Designing - Analysis of Architectural Practices

Analysis is concerned with examining carefully the architectural practices to identify their effects on the ecosystem health. First, the evaluator analyses the influences, documenting their reasoning and values for their contribution. Starting with the practices implemented in an ecosystem under evaluation, the evaluator should fill `Form A` (Figure 1(a)) to register the influence of each practice on every NFR of concern (Table 1). The influence of a practice on a NFR can be positive (+) or negative (-), and is labeled with one of three values: 0 - no influence, 1 - partial influence and 2 - fully influence.

Secondly, an analysis of the influences of NFR on each softgoal is performed. The evaluator should fill `Form B` (Figure 1(b)) for each health indicator / softgoal, registering the reasoning and values for their contribution. The influence of a NFR on a softgoal can be positive (+) or negative (-), and labeled with one out of five values: 0 - no influence, (+) HELP the influence, (++) MAKE the influence, (-) HURT the influence, and (–) BREAK the influence.

The output of analysis is a set of forms registering reasonings and values of contribution to measure the degree of the effects of the architectural practices on health indicators. Based on these forms, the evaluator can draw the corresponding SIG-P for the ecosystem.

[4] Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software quality models

4.2 Softgoal Designing - Representation with SIG-P Graphs

Representation is concerned with the creation of a SIG-P graph.

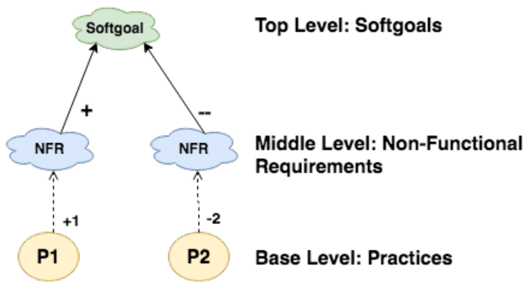


Fig. 2. SIG-P Graph

Figure 2 presents the SIG-P Model. The base level consists of graph elements that represent the architectural practices. Two letters are used to designate the key area associated with each architectural practice (for instance, AK for Architectural Knowledge) and a number identifies each practice.

The middle level consists of graph elements that represent NFRs. They have associated labels named after each NFR of concern. Relations between base level and middle level elements, called *influence links*, are represented by dashed arrows. Each influence link is labeled with the assigned value for the influence of a practice on a NFR.

The top level represents the three health indicators as softgoals, each one named after the corresponding indicator. Relations between middle level and top level elements are represented by solid arrows, also called influence links. These influence links are also labeled with a value that represents the degree of influence of each NFR on a health indicators.

Based on the information provided in forms A and B, the evaluator can draw the SIG-P graphs (Figure 2). There is an associated graph for each key area. To facilitate visualization, influence links with value zero (no influence) are not exhibited.

4.3 Influence Measuring - Measurement

Measurement is concerned with calculating, for each key area, the *value of influence* of architectural practices on NFRs, and the value of influence of NFRs on health indicators. The value of influence for each key area is determined based on the values informed on forms A (figure 1(a)) and B (figure 1(b)).

$$infP = [\sum_{k=1}^{N} vP_k)/(2 * N)] * 100 \tag{1}$$

Formula 1 is used to calculate $infP$, the value of influence of a set of architectural practices P on a NFR. Each vP_k (k = 1, 2, ..., N) represents the assigned values of influence of each practice, and N represents the size of set P for a key area. Since 2 is the maximum absolute value assigned to some vP_k, $2 * N$ stands for the maximum value for the set of practices P. We calculate $infP$ as the proportion of the sum of assigned values of influence with relation to the maximum value for the set of practices P in the key area.

The value of influence of a set of non-functional requirements NFR on a softgoal S is estimated based on the values of influence calculated for each nfr that belongs to NFR.

$$Vnfr = (Vanfr * infP)/100 \tag{2}$$

First, Formula 2 is used to estimate the value of influence of a non-functional requirement nfr on a softgoal S. We assume that the value of influence for a non-functional requirement nfr ($Vnfr$) is the relative amount of the assigned value ($Vanfr$) to the value of influence of practices ($infP$). For example, if the assigned value was +1, and the value of influence of practices was 50%, the value of influence of a non-functional requirement nfr is 50% of +1.

$$Vhea = [\sum_{k=1}^{Nnfrs} Vnfr_k/(2 * Nnfrs)] * 100 \tag{3}$$

Formula 3 is used to calculate $Vhea$, a value that represents the overall influence of a set of non-functional requirements NFR on a softgoal S. The value of $Vhea$ is estimated based on the values of influence calculated for each nfr in NFR. We use a percentage for each representative influence to calculate $Vhea$. Each $Vnfr_k$ (k = 1, 2, ..., $Nnfrs$) represents the assigned value of influence of each NFR on a softgoal. Then we calculate the proportion of the sum of assigned values of influence of NFRs ($Vnfr$) with relation the maximum value for the total number of NFRs.

5 Case Study Design

The goal of this research is to provide a new approach to measure the influence of architectural practices on health indicators of software ecosystems. The proposed approach can be used to evaluate the health status of open source software ecosystems. It does so by capturing some architectural practices of the KDE ecosystem and applying the *Heval/SA* presented in Section 4. Practices were gathered by interviews with some KDE developers. Following, we detail the research methodology.

5.1 Data Collection

To gather architectural practices from KDE ecosystem, we conducted interviews with three KDE members: two of them were software architects / experienced

Table 2. Architectural Practices of Architectural Knowledge

Identity	Architectural Knowledge
AK1	Architect leaders provide some tutorials and/or video courses for the technology available
AK2	Build maps of modules and runtime elements during face-to-face meetings
AK3	Create personal blogs and/or wikis to inform about the development and architectural issues
AK4	During code review, provide feedback information about architecture, good practices to code, doing refactoring and show the best way to solve the problems
AK5	Document APIs daily
AK6	Provide mentoring programs to training new developers
AK7	Keep different architects with different levels of specialty to spread the knowledge in the community
AK8	Provide code recommendations, defining a standard in the community
AK9	Provide a development manual for newcomers to know how to start contributing
AK10	Keep a register of meetings available to community to know all decisions of the meeting

developers and one was a newcomer developer. The experienced architects have worked in different KDE projects over the years, transitioning between roles such as developer, architect, committer, maintainer, or board participant. Based on their expertise, we were able to collect architectural practices for the whole ecosystem. In addition, the interview with the newcomer developer brought useful information about training and architectural practices used, from a different perspective.

The interviews were conducted using Skype[5], and simultaneously, the interviewees had access to the text of all questions. First, there was a short text introducing the context and objectives of the interview. The text also provided information about use of data and confidentiality of participants.

The interview scope covered important topics of Software Architecture. The set of questions encompassed 7 key areas related to architectural health, as defined in our previous work [3]. The interview was semi-structured and had 30 open-ended questions. The first 5 questions intended to collect the profile of the interviewees. The other aimed at understanding which daily architectural practices were used in the KDE ecosystem.

5.2 Data Analysis

The interviews were audio-recorded and later they were analyzed by the authors. Forty four architectural practices were identified and organized with respect to 7 key areas of architectural health. Each practice was classified and associated to one key area. For example, the practice "Provide mentoring programs to training new developers" was assigned to Architectural Knowledge area. Table 2 shows the set of practices related to the Architectural Knowledge key area.

[5] www.skype.com

After cataloging architectural practices, we analyzed their influence on each NFR to build the SIG-P. Due to space limitation, we present the results for 3 out of 7 key areas analyzed: Architectural Knowledge, Choice of Technology and Quality Management. For each architectural practice, we analyzed its influence for the 8 NFRs presented previously.

After classifying influences from practices to NFRs, we examined influences of NFRs on health indicators. Another form was filled out following the model presented in Figure 1(b).

Influences from practices to NFRs and from NFRs to health indicators served as input to shape SIG-P graphs. The reasonings and values of contribution for these influences were reviewed by the authors. For each filled out form, one author created the form and the others inspected the results. All disagreements were discussed by the group to achieve a final result. The questions of the interview, the list of practices, and the forms filled out by authors are available at http://www.professores.ifba.edu.br/simoneamorim/measures/.

6 Case Study Results

We applied our Heval/SA to evaluate the health of the KDE ecosystem based on architectural practices of three key areas: Architectural Knowledge, Choice of Technology, and Quality Management. For each key area, we have analyzed KDE architectural practices, we filled out form A (Figure 1(a)) to relate architectural practices to NFRs and form B (Figure 1(b)) to relate NFRs to health indicators. After analysis, three instances of SIG-P were created, one for each key area.

Figure 5 presents a SIG-P for Architectural Knowledge, with 10 architectural practices linked to one or more non-functional requirements, which in turn, are connected to health indicators.

Each SIG-P served as input to measuring the values of influences for the corresponding key area. From the base level of a SIG-P, the influence of the practices on NFRs is calculated, using formula 1. Figure 4 shows the results of influence for practices of Architectural Knowledge considering each NFR.

To estimate the influence of NFRs on softgoals, we applied formula 2 and formula 3. Figure 3 presents the resulting values for three key areas. Due to space limitation, SIG-P graphs and tables with the results of the calculations, Choice of Technology and Quality Management, are available only at our website: http://www.professores.ifba.edu.br/simoneamorim/measures/.

7 Lessons Learned

Our study evaluates the architectural health of the KDE ecosystem. Applying our *Heval/SA*, we performed qualitative and quantitative evaluations for influences on health indicators. Regarding the values obtained for influences on the NFRs for Architectural Knowledge key area, we can observe that the majority of practices impact on `Maintainability` and `Security`. See figure 3. Besides,

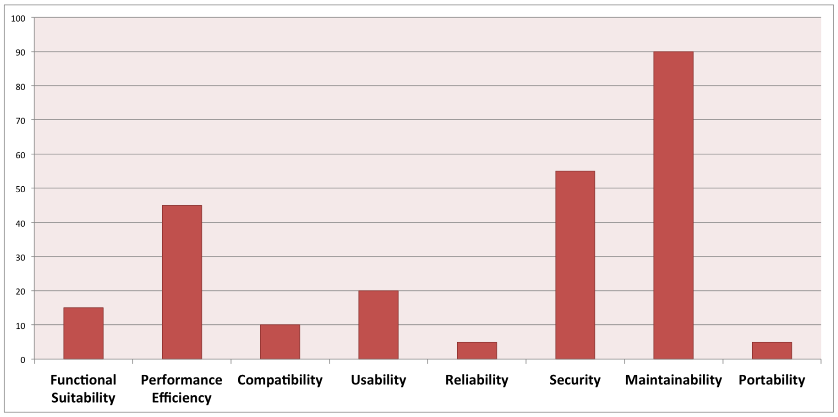


Fig. 3. Influence on NFRs

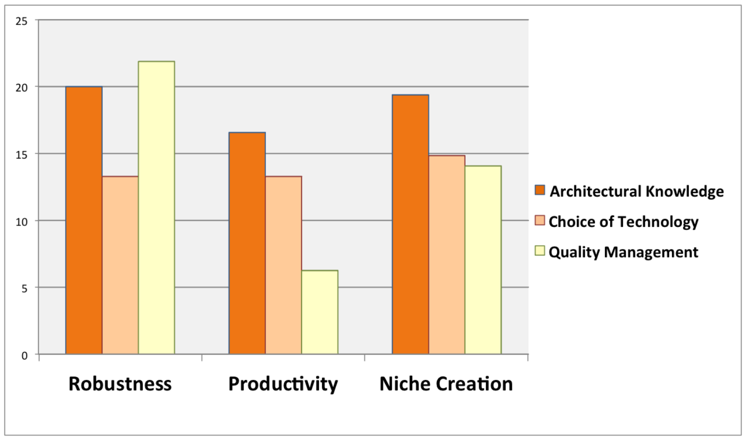


Fig. 4. Influence on Softgoals

a low percentage of these practices impacts on `Reliability` and `Portability`. These values detect the need to invest more in practices to increase the influence rate for NFRs with low impact. A healthy ecosystem should have positive influences for all NFRs.

Moreover, analyzing the top level of the SIG-P, in figure 5, we observed that `Functional Suitability` and `Usability` influence only one indicator and `Portability` has a major number of negative influence on health indicators. Considering the values calculated, negative influences are not perceived, but the results in general reveal a low percentage of influence, less than 50%. This signalizes the need to rethink about dynamic interactions among of the practices to enhance these numbers positively. In addition, the figure 4 shows key areas having a stronger influence on the `Robustness` indicator. Our approach allows

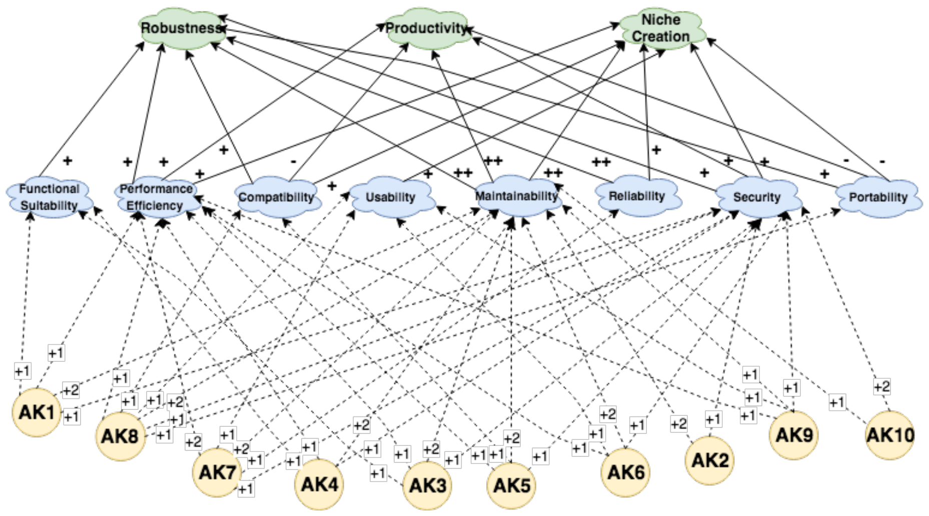


Fig. 5. SIG-P for Architectural Knowledge

analyzing variations in the results. An evaluator can try to assign different values applying the approach to observe different results. After, he can visualize different options of ways to follow in his decision-taking process. This way, our approach provides support to the judgment of possible actions.

The study further presents a set of benefits such as: (i) Catalog of architectural practices - from interviews we got several practices used by KDE, but which also can be used by majority of open sources software ecosystems; (ii) Capture tacit knowledge - got the wisdom that is not registered formally though actions of developers; (iii) Measurable influences - a quantitative way to perceive the influence of practices; and (iv) Graph-based perception for influences - a clear way to observe and identify the impact of the influences.

In the context of software ecosystem health several approaches offered mechanisms to extract metrics. However, some of them faced some problems as the fact of having abstract metrics that are difficult to get in practice. Others should analyze the nature of the metric, if it is harmful or helpful [12]. Our approach intends to be as practical as possible, being easy and useful. The qualitative analysis depends on the background of the evaluator, but the quantitative analysis is simple to apply. Through the use of proper practices, the approach also contributes to building not only “good” architectures that work well at the moment but creating “healthy” architectures that will contribute to the continuity of the ecosystem increasing new projects, activities, and developers.

Notwithstanding the benefits promoted by our approach, we report some restrictions. These results not determine what practices are more important. Furthermore, the approach does not provide a schema to compare results among ecosystems of similar size and in the same domain. For this, we will need additional deeply studies.

8 Threats to Validity

We have performed our case study with scientific rigor, but the process can have suffered from some limitations. There is a potential bias considering the interviews, where the number of respondents or their level of experience cannot be enough to cover all KDE architectural areas. There are also limitations regarding practices not captured during the interview process or some misunderstanding by the interviewer. To reduce this risk, the practices were inspected by other authors.

Additionally, there is a risk of bias of the researchers. The analysis process is influenced by the vision of the researchers. The judgment is in accordance to his/her architectural background. To reduce this bias, all results of the analyzing process were discussed by the authors. Finally, the last threat is whether the conclusions can be generalized to different organizations. In our work, we only collected KDE practices, but we believe to have captured many characteristics that are general to the architectural process for open source ecosystems. This way, further studies are needed for other ecosystems.

9 Conclusion

In this paper, we have described a case study for measuring the health of an open source software ecosystem. The basis for the analysis was a set of architectural practices collected from KDE ecosystem. These practices were gathered from a series of interviews with relevant members of this ecosystem. Regarding the practices collected, we applied our *Heval/SA* approach to build SIG-P graphs and calculate the level of influence of architectural practices on health indicators.

The results demonstrated that the influence of architectural practices is concentrated on `Maintainability`, and a low number of the studied practices influences `Reliability` and `Portability`. Besides, general findings for influences considering three key areas show a low percentage of influence on the health indicators. These outcomes emphasize the need for analyzing practices in detail to enhance their contribution to achieve a health state. This case study is a first step to apply our approach to measure the architectural health of software ecosystems. We intend to conduct additional case studies with other open source software ecosystems in the future to verify our outcomes.

Acknowledgments: This work was partially supported by PRPGI/IFBA.

References

1. Android's Success: by the Numbers. Information Week (2012), http://www.informationweek.com/mobile/mobile-devices/androids-success-by-the-numbers/d/d-id/1103058
2. Global Hadoop Market – industry analysis, size, share, growth, trends and forecast 2012 – 2018. Transparency Market Research - White Paper (2013), http://www.transparencymarketresearch.com/hadoop-market.html

3. Amorim, S., McGregor, J., Almeida, E., Chavez, C.: Software ecosystems′ architectural health: Another view. In: Proc. of the 5th ICSE Int. Workshop on Software Engineering for SoS and 11th Workshop on Distributed Software Development, Software Ecosystems and SoS. pp. 66–69. SESoS/WDES '17 (2017)
4. Amorim, S., McGregor, J., Almeida, E., Chavez, C.: Understanding the effects of practices on kde ecosystem health. In: Balaguer, F., Di Cosmo, R., Garrido, A., Kon, F., Robles, G., Zacchiroli, S. (eds.) Open Source Systems: Towards Robust Practices. pp. 89–100. Springer International Publishing, Cham (2017)
5. Chung, L.: Representing and Using Non-Functional Requirements: A Process-Oriented Approach. Ph.D. thesis, University of Toronto (1993)
6. Chung, L., Mylopoulos, J., Yu, E.: Non-Functional Requirements in Software Engineering. Int. Series in Software Engineering, Kluwer Academic Publishers (1999)
7. Chung, L., Nixon, B.A.: Dealing with Non-functional Requirements: Three Experimental Studies of a Process-oriented Approach. In: Proc. of the 17th Int. Conference on Software Engineering. pp. 25–37. ICSE'95, ACM, New York, NY, USA (1995). https://doi.org/10.1145/225014.225017
8. Franco-Bedoya, O., Ameller, D., Costal, D., Franch, X.: Measuring the quality of open source software ecosystems using queso. In: Proc. of the 10th Int. Conference on Software Technologies (ICSOFT). p. 3962 (2015)
9. Iansiti, M., Levien, R.: Keystones and Dominators: Framing Operating and Technology Strategy in a Business Ecosystem. Harvard Business School **3**(61) (November 2002)
10. ISO/IEC/FDIS: Standard 25010:2010 Systems and Software Quality Requirements and Evaluation (SQuaRE) (2010)
11. Jacobson, I., Ng, P.W., Spence, I.: Enough of Processes – Let's Do Practices. Journal of Object Technology **6**(6), 41–66 (2007)
12. Jansen, S.: Measuring the health of open source software ecosystems: Beyond the scope of project health. Information and Software Technology **56**, 1508–1519 (November 2014)
13. Mehta, R., Wang, H., Chung, L.: Dealing with NFRs for Smart-Phone Applications: A Goal-Oriented Approach. Software Engineering Research, Management and Applications **430**, 113–125 (2012)
14. Messerschmitt, D., Szyperki, C.: Software Ecosystem - Understanding an Indispensable Technology and Industry. MIT Press Cambridge (2003)
15. Mylopoulos, Chung, L., Nixon, B.: Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on Software Engineering **18**, 483–497 (1992)
16. Ruiz-López, T., Garrido, J.L., Supakkul, S., Chung, L.: A pattern approach to dealing with nfrs in ubiquitous systems. In: Proc. of the CAiSE'13 Forum at the 25th Int. Conf. on Advanced Information Systems Engineering (CAiSE). vol. 998, pp. 25–32 (01 2013)
17. Subramanian, N., Chung, L.: Process-oriented metrics for software architecture evolvability. In: Proc. of the 6th Int. Workshop on Principles of Software Evolution (IWPSE) (September 2003). https://doi.org/10.1109/IWPSE.2003.1231212
18. Subramanian, N., Zalewski, J.: Quantitative Assessment of Safety and Security of System Architectures for Cyberphysical Systems Using the NFR Approach. IEEE Systems Journal **10**, 397–409 (June 2016)
19. Wnuk, K., Manikas, K., Runeson, P., Lantz, M., Weijden, O., Munir, H.: Evaluating the governance model of hardware-dependent software ecosystems - a case study of the axis ecosystem. In: Proceedings of the 4th International Conference on Software Business (ICSOB). pp. 212–226 (June 2014)