An Empirical Analysis of Approach-Based Metrics Model for Architectural Erosion Detection

AHMED BAABAD[1], HAZURA BINTI ZULZALIL[1], SA'ADAH HASSAN[1], and SALMI BINTI BAHAROM[1]

[1]Universiti Putra Malaysia

March 22, 2023

Abstract

Software architecture determines success or failure in the domain of software development and design. As a system evolves, software architecture erodes. This phenomenon is called architectural erosion. Several studies that focused on various approaches to the problem of architectural erosion have been conducted. As a direct consequence of this, the metrics strategy has emerged as the most widely used solution for architectural erosion. However, providing a comprehensive perception of the elements required to evaluate the phenomenon of architectural erosion with an acceptable level of quality is a challenge. The primary goals of this research, which drew from the prior literature about identifying architectural erosion, were to (1) determine whether various adopted measures approaches determine architectural erosion in order to develop a formal model, and (2) evaluate the construct reliability and construct validity of the model. This research presents a model based on the chosen measures approaches for identifying architectural degradation. This model can be used as the cornerstone for a formal definition of general approaches and adopted metrics. Data was collected from 130 software engineering professionals with experience in architecture erosion and software metrics via a questionnaire-based survey. Structural equation Modelling (SEM) was used to analyse construct reliability, construct validity, and research hypotheses. The results demonstrate a substantial association between all metrics approach classes and architectural erosion, except for architectural complexity and architectural technical debt. Both researchers and practitioners can significantly benefit from this model's empirical assessment and evaluation, which includes a valuable information in this context.

An Empirical Analysis of Approach-Based Metrics Model for Architectural Erosion Detection

AHMED BAABAD [1,2], HAZURA BINTI ZULZALIL [1] , SA'ADAH HASSAN[1] ,

AND SALMI BINTI BAHAROM[1]

[1] Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia[2] Department of Management Information Systems, Administrative Sciences, Hadhramout University, Mukalla, Yemen Corresponding author: Hazura Binti Zulzalil (hazura@upm.edu.my)

Abstract

Software architecture determines success or failure in the domain of software development and design. As a system evolves, software architecture erodes. This phenomenon is called architectural erosion. Several studies that focused on various approaches to the problem of architectural erosion have been conducted. As a direct consequence of this, the metrics strategy has emerged as the most widely used solution for architectural erosion. However, providing a comprehensive perception of the elements required to evaluate the phenomenon of architectural erosion with an acceptable level of quality is a challenge. The primary goals of this research, which drew from the prior literature about identifying architectural erosion, were to (1) determine whether

various adopted measures approaches determine architectural erosion in order to develop a formal model, and (2) evaluate the construct reliability and construct validity of the model. This research presents a model based on the chosen measures approaches for identifying architectural degradation. This model can be used as the cornerstone for a formal definition of general approaches and adopted metrics. Data was collected from 130 software engineering professionals with experience in architecture erosion and software metrics via a questionnaire-based survey. Structural equation Modelling (SEM) was used to analyse construct reliability, construct validity, and research hypotheses. The results demonstrate a substantial association between all metrics approach classes and architectural erosion, except for architectural complexity and architectural technical debt. Both researchers and practitioners can significantly benefit from this model's empirical assessment and evaluation, which includes a valuable information in this context.

Key words: software architecture; architectural erosion; software metrics; measures; quality attributes; empirical analysis

1. Introduction

In the software engineering community, the past decade has seen a rise in focus on software architecture (SA). SA plays a significant role in a stage of software development in overall, particularly in research and industry [1] as well as software development[2]. SA is regarded as the fundamental structure block of every system, since it is the significant and crucial factor in defining, succeeding, and developing systems design [3]along with quality standards [4], making it one of the most essential problems in software design and development today. Software engineering should place a significant emphasis on SA, since the decisions that are made at the beginning of the development process have a direct bearing on the quality and the level of success of the products that are produced. Even though SA achievements are tremendous, some challenges remain. Hence, SA is eroded over time[5, 6] due to variety of factors: time pressure, adding , accumulating architectural debt, fixing bugs, design decisions, code complexity, inconsistent requirements, an unintended cyclic dependency among components, and new features, as well as technical requirements for changes (i.e., programming languages, hardware, new platforms, and operating system), All of these architectural issues may manifest unnoticed and undiscovered for a long time until they grow and become difficult to maintain. Evidently, this is not a new issue; rather, it has a lengthy history in the field of software engineering. Interestingly, the deterioration of the architecture causes the system to change at a higher frequency, with greater difficulty, and greater complexity than it did previously. [7]. This phenomenon is commonly referred to as Architectural Erosion (AE)[8-13], Software Architecture Degradation (SAD)[14-17], Architectural Decay (AD)[18-20]. According to a survey of the studies conducted by the researchers working in this field, the term Architecture Erosion (AE) or Architectural Erosion (AE) is the one that is used in most of their studies. As a result, we have decided to utilize this term in our own study as well.

In this regard, a variety of potential solutions to the problem of AE have been suggested in [17]. As a consequence, the metrics strategy is the one that is utilized the most frequently and has the highest level of success among the various accessible alternatives[17, 21]. Software metrics are collected at numerous points in the software development lifecycle to improve and track the progress of various software engineering products and processes[22]. The fundamental logic is that ”you cannot control what you cannot measure” [23]. In addition, many systems lack an explicit and precise description of prescriptive architecture in practice. A study [24] found that just 5% of open-source projects record software architecture. This means architectural specifications must be generated from scratch. This can be difficult for systems with hundreds of thousands of lines of code. Furthermore, the original developers of a system, who knew its architecture, are often no longer available. So, sometimes creating an architectural specification is impossible. In the event, the architectural documentation is missing, the code is frequently the primary source of information concerning potential architectural violations [15, 25]. As a result, the metrics play a major role in identifying whether there are problems with the code or the architecture.

Despite what was mentioned in empirical studies regarding the adoption of the metrics approach, and that it is currently being widely used in providing proposed solutions to determine architectural erosion, evaluating the architectures of software implemented, and recognizing indicators of architectural deviations;

however, there is a significant lack of knowledge regarding the provision of a comprehensive perception of the elements required to evaluate the phenomenon of architectural erosion. Additionally, there is a need a model for selecting the appropriateness of the adopted measures approaches and the reliance on an appropriate classification that clarify ambiguous definitions regrading metrics, their adopted approaches, and its categories, as well as investigate complicated, inaccurate, or incomplete approaches of the adopted metrics. The certain shortcomings and obstacles regarding architectural erosion have already provided the substantial motivation for researchers to formulate an initial metrics approaches model for architectural erosion based on academic-related literature[13].

In this paper, a model for identifying architectural erosion based on the adopted metrics approaches is presented. This model can serve as the basis for a formal definition of generic approaches and adopted metrics. A model is based on systems monolithic architecture, which includes (1) generic approaches that are used for software metrics in the context of identifying architectural erosion (AE), (2) classification of adopted metrics and placement under the appropriate approach in order to recognise the AE, and (3) essential quality attributes regarding the identification of the AE. This model will investigate the empirical evidence concerning the relationship between the classification of metrics approaches and architectural erosion. This investigation will be based on an assessment of the reliability and validity of the model that was developed in collaboration with experts and related professionals. Additionally, empirical analysis will be utilised in order to determine how various classifications of different metrics approaches can identify the AE.

This paper presents a significant contribution to the field by introducing three new folds: First, A broad categorization for the purpose of determining architectural erosion, containing particular and consistent metrics that can be used to investigate and assess a wide range of approaches relevant to this context. Second, a deeper understanding along with interesting and novel insights into the potential identification of various types of metrics and their approaches to tackle architectural erosion from several perspectives. This can help researchers and developers devise or integrate solutions and approaches that play a prominent role in reducing architectural erosion based on criteria of convergent measures and achieving performance in a more effective and efficient manner. This can also be of assistance to researchers in the process of creating new solutions and approaches. Third, this model has been subjected to both an empirical evaluation and an evaluation from the perspective of software engineering professionals, yielding a wealth of information (such as erosion classifications, approach metrics, and quality attributes) that can serve as guidelines or references for researchers and practitioners for tackling the problem of architectural erosion.

The remaining parts of this paper are structured as described below. Section 2 outlines the related work. Section 3 proposes the research model and hypotheses. Section 4 describes research methodology. Section 5 presents data analysis and results. Section 6 explains discussion of obtained results. section 7 illustrates implications for research and practice. Section 8 clarify threats to Validity of the study. Section 9 concludes the paper and provides directions for future work.

2. Related Work

Despite studies relevant to our work, no model development based on these criteria and approaches has been adopted and validated. However, we will review the literature on those approaches that have been proposed to determine architectural erosion, based on measures that are key indicators in this context.

2.1 Architectural change approach

Changes at different levels of abstraction and from multiple architectural views can induce architectural degradation and instability. Investigating if software system maturity reduces architectural instability and discovered unstable software components based on project design instability and project call instability measures [26]. Unintentional architectural changes, decay, and the presence of vulnerabilities, as well as the identification and tracking of architectural decay across the evolution history of a software system, all require a reliable determination and understanding of architectural change based on architecture-to-architecture and cluster coverage measures[27-30].

Software architecture degrades when changes violate design-time architectural intents. Erosion makes maintenance harder and slows software evolution. In study [31] that used change dispersion, relationship-based similarity, architectural stability, and Move-Join (MoJo) measures to identify architecture degradation. System architecture changes can have unforeseen repercussions. So, study[32] investigated a metric for class fault-proneness based on structural distance. Concerning co-changes between modules,[19] Authors represented cross-module co-changes and inner-module co-changes metrics to identify modifications that occur simultaneously within or across modules. [33] proposed a metric-based, multidimensional approach for analysing the structural stability of system (package abstractness (A) and instability (I)) measures.

2.2 Historical data revision approach

Historical data revision analyses project files for bug and change prone throughout maintenance, revealing serious architectural flaws. Detecting architecture anti-patterns [34] and measuring and predicting architecture quality, and identifying architecture issues[35, 36] by analysing revision history based on bug change frequency, bug churn, change frequency, ticket frequency, and pair change frequency measures.

To forecast architectural decay from evolutionary history,[19] proposed multiple architectural perspectives, including architectural quality prediction models using an effective set of prediction metrics (i.e., number of changes and number of co-changed files). In study [15] highlighted several measures to indicate software architecture degradation, such as number of commits. The severity metric of a code smells based on the change history of a software system was proposed to rank symptoms of architectural problems[37].

2.3 Architectural cohesion approach

Architectural cohesion refers to the degree to which a module's elements are functionally connected. Inconsistencies between intended architecture as detected as symptoms of architectural degradation are investigated by many source code metrics such as lack of method cohesion metric [15] and lack of cohesion metric for packages[31, 38]. Dependency optimization-based metrics are used to identify source code anomalies and architectural erosion through package cohesiveness quality metric to examine internal package dependencies [39] and tight class cohesion[40]. Correlations and interactions between architecture changes and decay [27] are significant concept to measure actual cluster interactions to possible cluster interactions using the ratio of cohesive interactions metric.

2.4 Architectural modularization approach

The term "architecture modularization" is used to describe the degree to which a system or piece of software is divided into independent modules that have little effect on one another when modified. In studies[39, 41], proposed package quality measures (inter-package modularization dependencies, inter-package modularization connections, inter-package modularization cyclic dependencies, and inter-package modularization cyclic connections) over a wide range of software quality, including vulnerability detection, fault-proneness, and violation of coding standard. For the purpose of figuring out if the quality of software architecture is sufficient and if it is degrading over time. Decoupling Level (DL) is a metric introduced by studies[42, 43] to assess how well a software system is separated into independent modules. On the other hand, a modularization quality metric has been presented to measure the entire system in order to simplify the system's understanding and restrict certain types of changes to particular modules [27].

2.5 Architectural technical debt approach

In the context of the software development lifecycle, the term "architectural technical debt" refers to designs that are suboptimal, incomplete, immature, or otherwise lacking in appropriate artifacts. Since software architecture deterioration causes inconsistency between a system's implementation and major design decisions, [15]presented several metrics to detect architectural discrepancy (e.g., Sqale index and Sqale debt ratio).

Minimizing architecture debt is anticipated to make software less costly and more adaptable to change via decoupling level and propagation cost measures [42], evaluation of architectural violations based on architectural debt index [44], and analysis system coupling for numerous components in each system to find defect-related activity [45, 46], which are directly related to software architecture to be maintained and evolved.

2.6 Software architecture size approach

The term "software architecture size" relates to the estimation of the size of a system or component to determine software productivity, anticipate fault locations in testing, and plan for maintenance. It refers to the overall, data, or functional measure of the system.

To characterize the evolution or degradation of architecture in terms of size, highlight architecture and code-level problems that might affect system change costs, and reduce architectural model understandability which may lead to erosion, several size-related aspects have been proposed [15, 31, 47-49], including the number of lines of code, the number of statements, the number of calls, and the number of comments, number of connectors, number of components and class elegance.

2.7 Architectural complexity approach

Architectural complexity relates to a system's structure, stored knowledge on how it operates, and makeup, which may be difficult to understand or solve. To ensure that systems maintain their stability and can continue to deliver the necessary functionalities as they evolve, it is crucial to understand the pattern of software architecture change as the systems evolves over time [33, 50]. Thus, structural complexity and the complexity of a defect have been proposed based on analysis of structural over-complexity [51], cognitive complexity [37], the excessive complexity[33], weighted method per class, cyclomatic complexity[15, 52], and the defect persistence[53] measures to identify a defect and architectural problems.

2.8 Architectural bad smells approach

Architectural bad smells are architectural decisions that negatively affect system quality and may refer the architectural degradation. Measures for architectural elements [54], architectural concern-based metrics [54, 55], and architecture sensitive metrics [56] have all been proposed as heuristics to help in the prioritisation of key code anomalies. In addition, in study [57] that performed strategies based on metrics for identifying architecturally significant smells.

Moreover, architectural anomalies can be detected through analysis a correlation and co-occurrences between architectural and design smells[58] or among each other [59] based several metrics. A tool introduced by studies [60, 61]for the detection of architectural smell is based on a metrics engine that computes all Martin metrics [62].

2.9 Architectural dependency coupling approach

Dependency coupling in architecture is the relationship between abstraction-level entities within a module that is dependent on another module, whether that dependency is direct or indirect. Analysis of the dependency between conceptual architecture on the one hand and architecture changes, erosion, the presence of vulnerabilities, problem violations, and defect-related activity on the other was one of most significant motivations and areas of interest for researchers in many studies.[15, 27, 31, 35, 39, 45, 61, 63-66], which adopted coupling metrics between models, objects, packages, and classes to determine the state of the system architecture in terms of evolution or degradation. For example, coupling metrics such as coupling between objects, afferent coupling, efferent coupling, data abstraction coupling, number of coupled classes, sum of coupling, module dependency strength (MDS), and dependency frequency.

In study [54], proposed metrics for architectural elements such as external fan-out, external fan-in, and architectural element locality using architecture-sensitive strategies to detect code anomalies. In other words, investigating architectural decay based on architectural defects, architectural smells, and modularization quality using measuring the dependencies between modules [19], as well as expanding DSL-based architecture abstraction approach has been recommended to analyse the understandability of the generated architectural models to identify architecture erosion based on incoming and outgoing interdependence metrics [48].

Based on the approaches that have been reviewed, it can be concluded that there is a wide range of approaches that have been proposed for determining architectural erosion. Each approach focuses on different key indicators and metrics that are considered important for detecting and quantifying architectural erosion.

Overall, these approaches offer different perspectives on architectural erosion and provide a range of metrics that can be used to quantify and detect erosion in software systems.

3. Research Model and Hypotheses

The initial development of a software metrics model for analysing architectural erosion was conducted using systematic mapping[13]. Meanwhile, there is a major disparity and deficiency in realizing and assessing the metrics criteria that fundamentally drive and identify architectural erosion; Therefore, the research hypotheses were developed on the basis of (1) the literature review that was presented in this study and (2) the results of a previously conducted systematic mapping study, which demonstrated the significance of taking metrics into consideration when determining the presence of architectural erosion. In this study, measurement is conceptualized as an essential task in the process of software measurement and quality attributes. Based on an evaluation of two criteria for the primary studies (i.e., the study objective and the approach adopted to address architectural erosion), the methods of architectural erosion analysis have been categorized into nine broad categories that can be taken into account when assessing architectural erosion: Architectural change (ACH), Historical data revision (HDR), Architectural dependency coupling (ADC), Architectural bad smells (ABS), Architectural cohesion (ACO), software architecture size (SAZ), Architectural technical debt(ATD), Architectural complexity (ACP), and Architecture modularization(AM). A complete description of the measurement items used in this research can be found in the Supplementary Materials.

These classifications reflect the authors' diverse strategies for tackling the issue of architectural erosion through analysis of the mechanisms of adopted metric approaches that have been proven to handle this issue. Most of the approaches and methods proposed by researchers to address the phenomenon of erosion fall into the categories of architectural dependency coupling analysis, architectural bad smells, and architectural change because of the large number of metrics included in these groups and the large number of studies based on the analysis of relationships and dependencies among architecture artifacts of a given system.

A research model, shown in Figure 1, was constructed based on the aforementioned literature analysis and the facts, as well as the results of our own pr-conducted studies [13, 17]. Considering this, two-tailed hypotheses that have been developed as follows:

Hypothesis 1 (H1) . Architectural change significantly identifies Architectural Erosion.

Hypothesis 2 (H2) . Historical data revision significantly identifies Architectural Erosion.

Hypothesis 3 (H3) . Architectural dependency coupling significantly identifies Architectural Erosion.

Hypothesis 4 (H4) . Architectural bad smells significantly identify Architectural Erosion.

Hypothesis 5 (H5) . Architectural cohesion significantly identifies Architectural Erosion.

Hypothesis 3 (H6) . software architecture size significantly identifies Architectural Erosion.

Hypothesis 4 (H7) . Architectural technical debt significantly identifies Architectural Erosion.

Hypothesis 5 (H8) . Architectural complexity significantly identifies Architectural Erosion.

Hypothesis 5 (H9) . Architecture modularization significantly identifies Architectural Erosion.

4 . Research Methodology

It is possible to solve a research problem using models, procedures, and methods. Research methodology outlines the process by which hypotheses are produced and how they are tested [67]. It is important to know the purpose of a study's design in order to effectively conduct research. Therefore, this methodology includes literature analysis, instrument validity and reliability, sampling and data collection, data preparation, measurement, and structural model assessment. This section provides a detailed explanation of the procedures and analysis that were used to fulfil the study goals.

4.1 Literature Analysis

The initial step in the study process was to identify the problem that needed to be addressed by looking at the challenges and gaps left by prior studies. A literature review was used to gather the necessary information. In our previous research, we conducted a review of the literature on software architecture degradation and the metrics used to measure architectural erosion.

To create a proposed model for architectural erosion contexts, a systematic mapping study (SMS) was performed [13]. The study encompassed 92 metrics and 10 quality attributes. Each of the measures studied was allocated to an approach of architectural erosion (historical data revision, architectural bad smell, architectural dependency coupling, architectural cohesion, architectural change, architectural technical debt, architectural complexity, architecture modularization, and software architecture size). The prior approaches were deduced based on the most widespread analysis of proposals for architectural degradation.

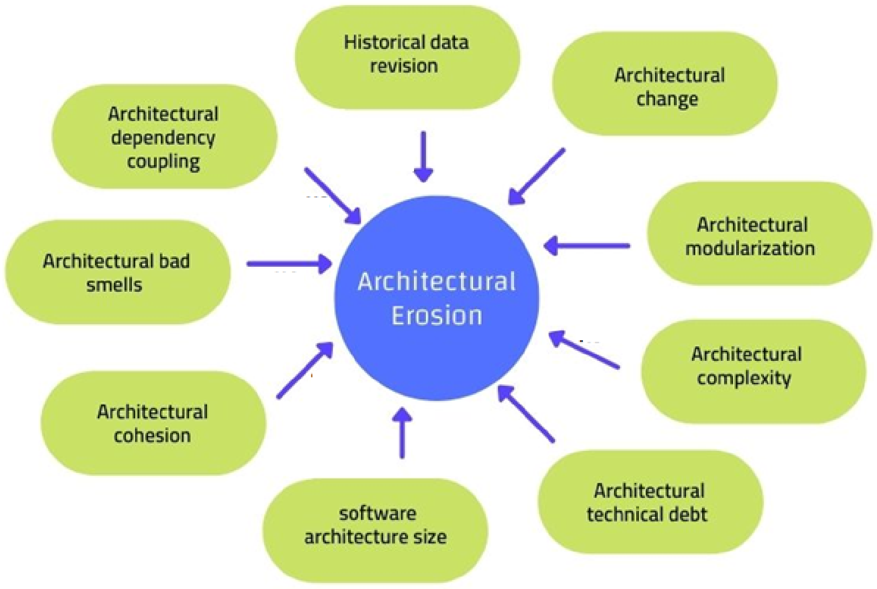

Figure 1. Proposed Model

4.2 Instrument validity and reliability

The quality of quantitative measurement could be judged by the validity and reliability of the data being collected. An instrument's reliability is demonstrated by its ability to consistently deliver the same results when tested in a variety of scenarios. In terms of validity, one definition states that it is when "what the instrument claims to measure is what it is measuring."[68]. As a result, the study instrument's validity and reliability were evaluated. Instrument validity and reliability testing are described in detail in the following sections. In the supplementary materials, you will find a summary of the survey measuring items that were utilized in this study.

4.2.1 Content validity

Analysis of a scale of content validity is critical for researchers who want to improve the instrument's construct validity, which is why content validation should be a primary concern[69]. Content validity focuses on the extent to which an item's sample of relevant factors adequately represents the construct of interest[70]. In general, the content validity of a model or instrument is determined by the analysis of its concept by a panel of experts [71, 72]. In the research, a questionnaire was developed and distributed to researchers and practitioners with prior experience in the context of architectural erosion. These researchers were selected through systematic studies of their published works and their relevance to the study, based on their interest in the study.

According to study [73], the model should be evaluated by at least three experts; however, the presence of more than ten experts is unnecessary [74]. Many academics believe that the model should be validated by at least five experts in order to be considered reliable [75]. In this study, 30 emails were sent to experts, but only 8 responded and filled out a questionnaire.

Diverse perspectives existed regarding the appropriate number of measurement scales, such as even-numbered (2, 4, 6, 8, or 10) or odd-numbered (3, 5, 7, or 9) scales [76]. Odd-numbered scale, numbers middle represents neutral, unbiased, not sure, don't know, or not applicable. In contrast, the even-numbered scale represents positive or negative responses. Odd-numbered scale can adjust the model more accurately than even-numbered scale [77, 78]. This study employed a five-point Likert scale to measure response appropriateness and intelligibility. Therefore, this study's questionnaire used a 5-point scale ranging from 1-strongly disagree to 5-strongly agree to reflect experts' opinions.

This study used a content validity ratio (CVR) model that was based on[79] to measure the individual scale items and Judgment calculation means [78]. The model is feasible due to the study's number of panellists, which is fewer than ten[80]. The formula could be written as follows:

$CVR = (Ne – N/2)/(N/2)$

Where

Ne = the proportion of experts who rated the item as a 4 or 5 on a 5-point scale.

N = the total number of experts.

For the eight panellists in content validation, CVR value was set at 0.75. Table 1 demonstrates component CVR values.

Table 1: minimum CVR value based on study [79]

Numbers of panellists	Min CVR value
5	0.99
6	0.99
7	0.99
8	**0.75**
9	0.78
10	0.62
11	0.59
12	0.56
13	0.54
14	0.51
15	0.49
20	0.42
25	0.37
30	0.33
35	0.31
40	0.29

For the purpose of computing the mean for each item, the values reported in the questionnaire were converted as follows [78]:

Strongly Agree (5) or Agree (4) - was replaced by 2

Neither disagree or agree (3) - was replaced by 1

Strongly Disagree (1) or Disagree (2) - was replaced by 0

Table 2: summary of criteria for finalizing the items

Criteria	Status
If CVR [?] 0.75 only for panellist =8	Accept
0 [?] CVR [?] 0.75 AND Mean [?] 1.5	Accept
If CVR [?] 0 AND Mean [?] 1.5	Reject

If the expert mean value of an item's assessment is more than 1.5, it is accepted for final items. As a result, the mean is more likely to be "Strongly Agree" or "Agree" than the expert judgement value of "0" or "No Idea". As an alternative, if the mean judgement of experts falls below 1.5, the item would be rejected since experts found the item to be unsuitable for measuring the construct in question. However, the CVR must be equal to or higher than 1.5 for the items to be accepted. Table 2 shows the finalizing criteria for the measures.

4.2.2 Reliability study

A pilot study was done before main data collection to validate and test research instruments. It's the initial step in designing a survey questionnaire. It helps validate the research instrument and reduce errors [81]. Most scholars recommend a sample size of 20–40 [82, 83]. Thirty respondents from software engineering research groups, familiar with architecture erosion metrics, were purposively sampled.

The pilot study's data was analysed using SPSS. Cronbach's alpha internal consistency test was used to assess the survey's measurement items for reliability. From 0 to 1, which indicates a high level of reliability. Values more than 0.9 are excellent; values between 0.8 and 0.9 are good; values between 0.7 and 0.8 are acceptable; values below 0.6 are considered poor [84]. Three criteria were used to assess the model's reliability:

1. Each construct's Cronbach's alpha value must be at least 0.70.
2. Corrected item-total correlation should be > 0.2.
3. If an item is eliminated, Cronbach's alpha must be lower than Cronbach's alpha for the construct.

4.3 Sampling and Data Collection

Data collection is one of the key elements of model evaluation. Data collection entails gathering helpful ideas and information about research issues from target respondents [85]. Numerous methods to data collecting are described in literature, including face-to-face, telephonic, email, and use of online media[86]. Email and various forms of online media were used in the collection of data for this study. When methods are combined, both time and cost could be saved. In this research, a questionnaire was used to collect data which is realistic and convenient tools for data collection [87].

Given the impossibility of obtaining an accurate representation of the entire population, a representative sample was selected. This study used a technique known as ”purposive sampling.” The study's main emphasis is on the researcher's strategic use of chance to find potentially helpful respondents [74]. Therefore, in this research, software engineering professionals who are related to the area (e.g., software metrics, software architecture, architectural decay/degradation/erosion, software quality, architectural smell/code smell, architectural technical debt, and software maintenance and evolution) were selected as respondents. Also, in perspective industrial, Product Manager, System Architect, IT Consultant, Software Engineering Manager/Lead, and Automation Technical Leader, who have wide knowledge on software architecture erosion. Table 3 outlines respondent demographics.

An online questionnaire was designed to gather survey data, using Smart Survey tool for ease of response. It was determined that Google scholar would be the most effective tool for finding relevant researchers for the survey. Additionally, authors of studies relevant to our research were invited to participate in this survey. LinkedIn was chosen as the best way to recruit software engineers especially who are working in industrial companies worldwide to participate in the survey. Moreover, Online questionnaire was shared in social media in LinkedIn and Facebook groups which are concerned with software engineering and development.

The size of the sample matters a great deal when conducting statistical analysis. Thus, the choice of sample size should be made considering the specific statistical method that will be employed to analyse the data collection. In this study, statistical methods such as structural equation modelling (SEM) and SPSS were used for analysis. There is no consensus regarding the exact sample size for SEM, and researchers have various opinions on the matter. In order for a SEM estimation to be reliable and valid, an adequate sample size is required. According to[88], a sufficient sample size for SEM requires at least five respondents for each construct and a total of no fewer than 100 persons for the entire set of data. Almost any SEM may be unworkable with less than 100 cases unless a very simple model is tested[89]. The model is considered simple because it lacks the complex variables that are associated with multi-ranking, and the sample of the target population is restricted to specific experts around world, online questionnaires were distributed to get as many as possible responses. and ultimately, 130 completed questionnaires were returned.

Table 3. Profiles of the respondents' demographics that were collected (n = 130)

demographic Variable	Category	Frequency (n)	Percentage (%)
Field	academic	89	68.5
	Industry	18	13.8
	academic /industry	23	17.7
Current position	Researcher	89	68.5
	Software development engineer	11	8.5
	Researcher / Software development engineer (both)	22	16.9
	Others	8	6.1
Year of experience	Over 15	70	53.8
	11- 15	22	16.9
	6-10	24	18.5
	3-5	8	6.2
	1-2	6	4.6

4.4 Data Preparation

The responses were pre-processed after the data had been collected. Therefore, the steps involved in each stage of data pre-processing are outlined in the following points, as well as a complete description of data investigation and descriptive statistics for constructs and their items in this research can be found in the Supplementary Materials.

4.4.1 Missing Data

Incomplete data gathering is a problem [90]. In addition, missing data affects statistical tests that examine the correlations between variables [72]. Data was collected online in this study. All questionnaire responses are required and responders can't skip any. Statistical Package for Social Sciences (SPSS) was used to examine the raw data. Thus, no data was lost.

4.4.2 Outliers

Another challenge that can arise when using statistical methods is that of outliers. An observation that stands out as being dramatically dissimilar to other value observations is known as an outlier[72, 89]. There was no univariate outlier in this study due to the Likert Scale 5-scale range being employed for responses. Outliers were done by SPSS, and no outliers were found.

4.4.2 Normality

To determine the shape of the sample distribution, data normality testing was employed. According to [91] study , the appropriate range for skewness is between -2 and +2, whereas the appropriate range for kurtosis is between -7 and +7[92]. In addition, some researchers, a score of +1 to -1 for skewness and +3 to -3 for kurtosis is considered to be normal value[93]. In this study, the normality of the data was evaluated using SPSS to identify the range of skewness and kurtosis, ensuring that the prescribed limit was not exceeded.

4.5 Measurement Model Assessment

The term ”measurement models” refers to the models, either implicit or explicit, that relate the latent variable to its indicators. The evaluation of the PLS-SEM model begins with a look at the measurement models. Using PLS-SEM estimations, researchers can determine the reliability and validity of the construct measurements. Specifically, multivariate measurement is the use of multiple variables (i.e., multi-items) to assess a construct [94]. The PLS algorithm was employed in Smart PLS to assess the measurement model. Factor Loadings, internal consistency, convergent, and discriminant validity were used to evaluate the measurement model[94].

4.5.1 Internal Consistency Reliability

Internal consistency reliability is often the first criterion to be assessed. To measure internal consistency, Cronbach’s alpha is used, which traditional criterion and estimates the reliability based on intercorrelations between observed indicator variables and their correlations. Due to Cronbach’s alpha’s restrictions, a new measure of internal consistency reliability, known as composite reliability (CR), is more technically acceptable [94, 95]. Internal consistency reliability is indicated by an a-value of 0.7 or above[84, 96]. Thus, any a-value that is more than 0.70 is regarded as reliable and acceptable.

4.5.2 Convergent validity

The degree to which one measure correlates positively with other available measures of the same construct is what is meant by the term ”convergent validity.” [72, 89, 94]. Average variance extracted (AVE) is a common measure of concept convergent validity. This criterion is the grand mean of the construct’s squared loadings (i.e., the sum of the squared loadings divided by the number of indicators). The following formula is used to determine the AVE [94]:

AVE= $\frac{\sum x^2}{N}$

where [?] x^2 s is the sum of all squared standardized factor loadings and n represents the items number. An appropriate value is more than 0.5, which indicates that latent variables have convergent validity[94, 97].

4.5.3 Discriminant validity

The degree to which a construct is empirically distinguishable from other constructs is known as discriminant validity. A construct’s ability to distinguish itself from others in the model is an important step in the process of proving discriminant validity[72, 89, 94].

We assessed at the discriminant validity based on HTMT approach to determine a threshold that has been used in most of the research. Technically, HTMT estimates the correlation between two constructs if they were perfectly quantified (i.e., if they were perfectly reliable). Disattenuated correlation refers to real correlation. A correlation near to 1 shows a lack of discriminant validity [94]. According to previous research and the findings of those studies,[98] determined a 0.90 threshold for conceptually similar path model structures. Above 0.90 HTMT shows lack of discriminant validity.

4.6 Structural Model Assessment

The primary purpose of the structural model is to investigate the relationship between the independent and dependent variables. This model also makes it possible to ascertain the impact that the independent factors (exogenous variables) have on the variables that are being studied (endogenous variables) [72]. A structural model is a perspective of a system that focuses on the structure of the objects, including the relationships between those objects, their characteristics, and the activities those attributes they perform[99]. Through investigating their relationship within a structural model, we can infer the effect of the independent factors on the dependent variable. Following are the key considerations that were taken into account in the evaluation of the structural model: 1) Structural Model Path Coefficients (Hypotheses Testing), 2) Coefficient of Determination (R^2 Value), 3) Effect Size F^2, 4) Predictive Relevance Q2 5) Goodness of Fit of the Model - GoF.

4.6.1 Structural Model Path Coefficients (Hypotheses Testing)

It is possible to identify the individual impact of an exogenous variable on an endogenous variable by carrying out a slope test. In order to demonstrate that the exogenous variables have a significant influence on the endogenous variables, the p-value should be less than 0.05, and the t-value should be more than the critical value (C.R), which should be greater than 1.96 [94, 100].

4.6.2 Coefficient of Determination ($\mathbf{R^2}$ Value)

The coefficient of determination (R2 value) is the standard method for assessing the quality of a structural model. In order to evaluate how well a model can predict future values for a specific endogenous construct, researchers use a coefficient defined as the square of the correlation between those values and the model's predictions. According to study [101], values of R^2 that are greater than 0.67 are considered to be high; values that fall within the range of 0.33 to 0.67 are considered to be moderate; values that fall within the range of 0.19 to 0.33 are considered to be low; and anyR^2 values that are less than 0.19 are deemed unsatisfactory.

4.6.3 Assessment of Effect Size ($\mathbf{F^2}$)

During the evaluation of the structural model, effect size (F^2), measured by Cohen's, is used to evaluate the impact of a given construct on an endogenous variable (the independent variable). According to the guidelines for evaluating F^2, the value above 0.35 represents large; whereas F^2 ranging from 0.15 to 0.35 are considered medium effect size; whereas F^2ranging from 0.02 to 0.15 are considered small effect size. The absence of an effect can be inferred from effect size values that are lower than 0.02 [102].

4.6.4 Assessment of Predictive Relevance ($\mathbf{Q^2}$)

In addition to assessing the magnitude of the R2 values as a measure of prediction accuracy, researchers should also evaluate Stone- Geisser's $\mathbf{Q^2}$ value [103, 104]. This measure is an indicator of the model's predictive strength or predictive relevance when it is applied to data that is not contained within the sample. We used blindfolding procedures to achieve cross-validity redundancy [102] to assess the model's predictive usefulness. For a cretin's endogenous constructs, the model can be predictively useful if Q^2 is greater than zero, as shown by [94].

4.6.5 Goodness of Fit of the Model - GoF

As an overall measure of how well a model fits the data, the goodness-of-fit model (GoF), has been developed for use with PLS-SEM. GoF, as described by [105], is the geometric mean of AVE and the average of R^2 of the endogenous variable. The goal of GoF is to provide how suitability of the study model at two levels—the measurement level and the structural level—with focus on the model's performance as a whole [106, 107]. The calculation formula of GoF is as follow:

GoF= $\sqrt{(R^2 * AVG)}$

According to [108], a GoF of 0.36 or higher is deemed to be large, while a GoF of 0.25 to 0.36 is considered to be medium, and a GoF of 0.1 to 0.25 is deemed to be small, and a GoF of less than 0.1 is deemed to be a "No fit."

5. Data Analysis and Results

The results will be provided in this section in such a way that each section of the methodology contains analyses of the results referred to, such that the results are sequenced in a sequential series since each subsequent section depends on the preceding section, and so on.

5.1 Content validity

We conducted evaluation round for content validity between Dec 2021 and Mar 2022, based on the literature, the approach defines the construct clearly and precisely, and provides a novel taxonomy of the construct for this research. Therefore, round for content validity and questionnaires are provided in Appendices A–B.

Out of the 41 experts identified, only 8 of them responded to the questionnaire and completed the content validity survey. Table 4 contains the basic research-related information of the content validity experts.

Table 4 summarizes the experts' basic knowledge in content validity.

No	Designation field	Current profession	Experience
1	academic	Researcher	> 15
2	academic /industry	Researcher / Software development engineer	> 15
3	academic	Researcher	11-15
4	academic	Researcher	> 15
5	academic /industry	Researcher / Software development engineer	3-5
6	academic	Researcher	> 15
7	academic /industry	Researcher / Software development engineer	> 15
8	academic /industry	Researcher / Software development engineer	> 15

A large number of items from previous studies were included, totalling 102 items spread across 10 different constructs. Therefore, these items will need to be reconsidered by experts through one of two concepts: either rejecting this item altogether or combining it with another item that has a similar concept based on the comment that was provided by experts.

In Table 5 which was used to present the findings of CVR and Mean for the content validity, indicates significant points in accepting, rejecting, or merging items according to the consensus reached by the eight experts. In the historical data revision construct, combining HDR1 with HDR4 made sense since the two items complement each other and are generally accepted in terms of CRV and mean value. Regarding HDR 3 and HDR10 items, they were rejected due to the CVR value being less than 0 (-0.25, -1.00) and the Mean being less than 1.5 (1.3, 0.63) respectively. The CVR value for the ABS8 and ABS10 items in the architectural bad smell construct was found to be less than 0 (-0.50 and -0.25), and the Mean value was found to be less than 1.5 (1.00 and 1.25) respectively, hence these items were not accepted.

Table 5. CVR and Mean results for the content validity

Construct	Item	CVR	Mean	Status	Construct	Item	CVR	Mean	St
Historical Data Revision (HDR)	HDR1	1.00	2.00	Accept	Architecture Modularization (AM)	AM1	-0.25	1.38	Re
	HDR2	1.00	2.00	Accept		AM2	0.75	1.88	Ac
	HDR3	-0.25	1.13	Reject		AM3	0.50	1.75	Ac
	HDR4	0.50	1.63	Accept		AM4	0.25	1.63	Ac
	HDR5	0.75	1.88	Accept		AM5	0.25	1.63	Ac
	HDR6	1.00	2.00	Accept		AM6	-0.50	1.00	Re
	HDR7	0.25	1.50	Accept		AM7	-0.25	1.13	Re
	HDR8	0.75	1.88	Accept		AM8	-0.25	1.25	Re

Construct	**Item**	**CVR**	**Mean**	**Status**	**Construct**	**Item**	**CVR**	**Mean**	**St**
	HDR9	1.00	2.00	Accept		AM9	0.00	1.38	Re
	HDR10	-1.00	0.63	Reject		AM10	0.25	1.63	Ac
Architectural bad smell (ABS)	ABS1	0.75	1.88	Accept	Architectural change (ACH)	ACH1	0.75	1.88	Ac
	ABS2	0.75	1.88	Accept		ACH2	0.25	1.50	Ac
	ABS3	0.75	1.75	Accept		ACH3	0.75	1.88	Ac
	ABS4	1.00	2.00	Accept		ACH4	0.50	1.75	Ac
	ABS5	0.75	1.88	Accept		ACH5	0.25	1.63	Ac
	ABS6	0.75	1.75	Accept		ACH6	0.50	1.75	Ac
	ABS7	0.25	1.50	Accept		ACH7	-0.25	1.25	Re
	ABS8	-0.50	1.00	Reject		ACH8	0.00	1.38	Re
	ABS9	0.50	1.75	Accept		ACH9	0.00	1.38	Re
	ABS10	-0.25	1.25	Reject		ACH10	0.00	1.38	Re
	ABS11	0.25	1.63	Accept		ACH11	-0.25	1.13	Re
Architectural Depen-dency Coupling (ADC)	ADC1	0.50	1.75	Accept		ACH12	0.75	1.88	Ac
	ADC2	0.50	1.75	Accept		ACH13	0.25	1.63	Ac
	ADC3	1.00	2.00	Accept	Architectural Technical Debt (ATD)	ATD1	0.75	1.88	Ac
	ADC4	1.00	2.00	Accept		ATD2	0.75	1.88	Ac
	ADC5	0.25	1.63	Accept		ATD3	0.50	1.63	Ac
	ADC6	1.00	2.00	Accept		ATD4	0.75	1.88	Ac
	ADC7	0.50	1.63	Accept		ATD5	1.00	2.00	Ac
	ADC8	0.25	1.63	Accept	Architectural Cohesion (ACO)	ACO1	0.25	1.50	Ac
	ADC9	-0.25	1.25	Reject		ACO2	1.00	2.00	Ac
	ADC10	-0.25	1.25	Reject		ACO3	0.25	1.50	Ac
	ADC11	0.75	1.88	Accept		ACO4	-0.25	1.00	Re
	ADC12	-0.25	1.25	Reject		ACO5	0.25	1.63	Ac
	ADC13	-0.25	1.25	Reject	Software Architec-ture Size (SAZ)	SAZ1	1.00	2.00	Ac
	ADC14	-0.25	1.25	Reject		SAZ2	0.50	1.63	Ac
	ADC15	0.25	1.63	Accept		SAZ3	1.00	2.00	Ac
	ADC16	0.25	1.63	Accept		SAZ4	0.75	1.88	Ac
	ADC17	0.50	1.75	Accept		SAZ5	-0.50	0.75	Re
	ADC18	0.50	1.75	Accept		SAZ6	-0.50	1.00	Re
	ADC19	0.25	1.38	Reject		SAZ7	0.50	1.63	Ac
	ADC20	-0.50	1.13	Reject		SAZ8	-0.25	1.00	Re
	ADC21	0.50	1.75	Accept		SAZ9	-0.50	0.75	Re

Construct	Item	CVR	Mean	Status	Construct	Item	CVR	Mean	St
	ADC22	-0.50	1.00	Reject	Architectural Erosion (AE)	AE1	0.75	1.88	Ac
Architectural Complex-ity (ACP)	ACP1	0.75	1.88	Accept		AE2	0.75	1.88	Ac
	ACP2	0.25	1.63	Accept		AE3	0.50	1.63	Ac
	ACP3	0.50	1.63	Accept		AE4	0.50	1.63	Ac
	ACP4	0.25	1.50	Accept		AE5	0.25	1.50	Ac
	ACP5	0.25	1.63	Accept		AE6	1.00	2.00	Ac
	ACP6	1.00	2.00	Accept		AE7	0.75	1.88	Ac
	ACP7	0.50	1.63	Accept		AE8	1.00	2.00	Ac
						AE9	1.00	2.00	Ac
						AE10	1.00	2.00	Ac

Concerning the architectural dependency coupling construct which has 22 constructs, the greatest number of any construct, ADC1, ADC3, and ADC5 items were merged into a single item, as well ADC2, ADC4, and ADC6 were merged into a single item because of how well their complementary concepts meshed and each of which may be considered valid. On the other hand, the values of the items ADC9, ADC10, ADC12, ADC13, ADC14, ADC19, ADC20, and ADC22 showed that the CVR value was found to be less than 0 (-0.25, -0.25, -0.25, -0.25, -0.25, -0.25, -0.50, -0.50) and the Mean value was found to be less than 1.5 (1.25, 1.25, 1.25, 1.25, 1.25, 1.38, 1.13, 1.00) respectively. Therefore, they were removed.

Regarding architecture modularization construct, AM1, AM6, AM7, AM8, AM9 items provided CVR values (-0.25, -0.50, -0.25, -0.25, 0.00) were less than 0 respectively. However, last value of AM9 item which is 0.00 was acceptable; on the contrary, AM9 item did not meet Mean value which is greater or equal to 1.5 as illustrated in Table 2, whereas Mean obtained value of AM9 item is 1.38. In addition, the Mean values of the rest items were also less than 1.50 (1.38, 1.00, 1.13,1.25, and 1.38) respectively. As a result, these items were removed from this construct.

Concerning architectural change construct, ACH7 and ACH11 items achieved CVR values were less than 0 which are -0.25, -0.25 respectively. Moreover, Mean values found to be less than 1.50 which are 1.25, 1.13 respectively. In same context, ACH8, ACH9, and ACH8 items achieved all CVR values are acceptable which are 0; however, all Mean values of stated items found to be less than 1.50 which are 1.38, 1.38, and 1.38 respectively. Conclusively, all stated items were removed as demonstrated in Table 5.

With regards to architectural cohesion and software architecture size constructs, ACO4 item was removed from architectural cohesion since it did not meet the CVR and Mean values found to be less than 0 and 1.5 respectively. While SAZ5, SAZ6, SAZ8, and SAZ9 items were removed from software architecture size for the same reason mentioned above. In relation to the architectural erosion construct, all items were accepted; however, AE7, AE8, AE9, and AE10 items were combined within the AE6 item because they are considered sub-items or have the same concept. This decision was made based on the suggestions and comments of experts in the questionnaire to achieve content validity.

Table 6. outlines adopted items of final version for content validity model

Construct	Original items	Deleted/merged items	Final version
Historical data revision	10	3	7
Architectural bad smell	11	2	9
Architectural dependency coupling	22	12	10

Construct	**Original items**	**Deleted/merged items**	**Final version**
Architectural complexity	7	-	7
Architecture modularization	10	5	5
Architectural change	13	5	8
Architectural technical debt	5	-	5
Architectural cohesion	5	1	4
Software architecture size	9	4	5
Architectural erosion	10	4	6

In its final version, the software metrics model for architectural erosion quality consisted of 66 items, grouped into 10 constructs: 7 items for Historical data revision, 9 items for Architectural bad smell, 10 items for Architectural dependency coupling, 7 items for Architectural complexity, 5 items for Architecture modularization, 8 items for Architectural change, 5 items for Architectural technical debt, 4 items for Architectural cohesion, 5 items for Software architecture size, and 6 items for Architectural erosion as illustrated in Table 6.

5.2 Internal consistency reliability

Based on the findings from the content validity analysis, the model was subjected to additional testing to determine its internal consistency reliability using a five-point Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). In this study, a selection of profiles was provided, each of which included a designation, the participant's current profession, and familiar experience with architectural erosion and its measures. The survey was completed by a representative sample of 30 professionals in the field of software engineering; these professionals are directly related to architectural erosion and its measures.

Most of the respondents were from academic field (N=22, 73%) while remaining respondents were distributed between industry and combination of the academic and industry fields simultaneously (N =4, N=4, 13%, 14%) respectively. The current profession of majority of respondents was researcher (N =22, 73%), followed by software development engineer (N= 3, 10%), software development engineer and researcher at same time (N =3, 10%), product manager (N =1, 4%), and industrial researcher (N =1, 3%). Most respondents had an excellent experience of architectural erosion and its metrics, which is more than 15 years (N =14, 47%), followed by between 11 -15 and between 6-10 years (N =7, 23%, N =7, 23%), while between 3-5 and 1-2 years (N =1, 4%, N= 2, 3%) respectively.

Table 7 presents a summary of the results of the reliability tests conducted on the validated model, including the Cronbach's alpha for each construct, the corrected item-total correlation, and the Cronbach's alpha coefficients if an item was removed. The results of the reliability analysis showed that the internal consistency of the model was satisfactory. The values of Cronbach's alpha computed for the following constructs: Historical data revision (N=7), Architectural bad smell (N=9), Architectural dependency coupling (N=10), Architectural complexity (N=7), Architecture modularization (N=5), Architectural change (N=8), Architectural technical debt (N=5), Architectural cohesion (N=4), and Software architecture size (N=5), as well as architectural erosion (N=6), were 0.874, 0.922, 0.887, 0.854, 0.870, 0.859, 0.942, 0.834, 0.941, and 0.840 respectively. The corrected item-total correlation coefficients for Historical data revision items ranged from 0.532 (HDR1) to 0.805 (HDR7); Architectural bad smell items ranged from 0.443 (ABS1) to 0.892 (ABS8); Architectural dependency coupling items ranged from 0.467(ADC 10) to 0.690 (ADC7); Architectural complexity items ranged from 0.427 (APC7) to 0.779 (ACP 3); Architecture modularization items ranged from 0.665 (AM2) to 0.699 (AM4); Architectural change items ranged from 0.440 (ACH5) to 0.742 (ACH1); Architectural technical debt items ranged from 0.780(ATD4) to 0.897 (ATD3); Architectural cohesion items ranged from 0.586 (ACO4) to 0.753 (ACO3); Software architecture size items ranged from 0.788(SAZ3) to 0.940(SAZ4); Architectural erosion items ranged from 0.513 (AE6) to 0.688(AE1).

Table 7 shows the reliability test results for the validated model.

Construct	**Item**	**Cronbach's Alpha if item deleted**	**Corrected item-total correlation**	**Construct**	**Item**	**Cronbach's Alpha if item deleted**	**Corrected item-total correlation**
Historical Data Revision (HDR) (**0.874**)	HDR1	0.876	0.532	Architecture Modularization (AM) (**0.870**)	AM1	0.850	0.667
	HDR2	0.874	0.540		AM2	0.852	0.665
	HDR3	0.857	0.651		AM3	0.843	0.690
	HDR4	0.851	0.689		AM4	0.843	0.699
	HDR5	0.849	0.725		AM5	0.824	0.782
	HDR6	0.846	0.754	Architectural change (ACH) (**0.859**)	ACH1	0.825	0.742
	HDR7	0.842	0.805		ACH2	0.842	0.609
Architectural bad smell (ABS) (**0.922**)	ABS1	0.928	0.443		ACH3	0.836	0.655
	ABS2	0.921	0.626		ACH4	0.854	0.492
	ABS3	0.908	0.799		ACH5	0.859	0.440
	ABS4	0.911	0.749		ACH6	0.836	0.658
	ABS5	0.912	0.740		ACH7	0.839	0.633
	ABS6	0.914	0.697		ACH8	0.843	0.604
	ABS7	0.912	0.740	Architectural Technical Debt (ATD) (**0.942**)	ATD1	0.928	0.849
	ABS8	0.902	0.892		ATD2	0.924	0.870
	ABS9	0.905	0.834		ATD3	0.919	0.897
Architectural Dependency Coupling (ADC) (**0.887**)	ADC1	0.884	0.515		ATD4	0.940	0.780
	ADC2	0.881	0.556		ATD5	0.932	0.826
	ADC3	0.877	0.603	Architectural Cohesion (ACO) (**0.834**)	ACO1	0.775	0.696
	ADC4	0.880	0.559		ACO2	0.789	0.671
	ADC5	0.879	0.585		ACO3	0.748	0.753
	ADC6	0.874	0.656		ACO4	0.832	0.586
	ADC7	0.872	0.690	Software Architecture Size (SAZ) (**0.941**)	SAZ1	0.925	0.855
	ADC8	0.872	0.679		SAZ2	0.935	0.800

Construct	Item	Cronbach's Alpha if item deleted	Corrected item-total correlation	Construct	Item	Cronbach's Alpha if item deleted	Corrected item-total correlation
	ADC9	0.873	0.677		SAZ3	0.938	0.788
	ADC10	0.885	0.467		SAZ4	0.910	0.940
Architectural complexity (ACP) (**0.854**)	ACP1	0.808	0.781		SAZ5	0.930	0.830
	ACP2	0.832	0.638	Architectural Erosion (AE) (**0.840**)	AE1	0.800	0.688
	ACP3	0.809	0.779		AE2	0.827	0.549
	ACP4	0.828	0.677		AE3	0.800	0.686
	ACP5	0.853	0.499		AE4	0.817	0.604
	ACP6	0.843	0.552		AE5	0.803	0.672
	ACP7	0.858	0.427		AE6	0.835	0.513

5.3 Measurement Model Assessment

The measurement model is the starting point for evaluating a research model because it determines whether each construct being measured is being measured correctly.

Before examining the measurement model, we considered to rule out the common method bias (CMB). VIF values greater than 5 indicate not only severe collinearity but also model contamination with common method bias[94]. The maximum VIF value for any predictor construct was 3.402, as reported in Table 9, ruling out the CMB in our data. As a result, collinearity among predictor constructs is not a critical issue in the measurement model, and we can proceed with the analysis of the findings report.

The measure is said to be reliable when its factor loadings (FL) are above 0.50 [94]. [109] provided a range of factor loadings quality i.e., excellent (0.71), very good (0.63), good (0.55), fair (0.45) to poor (<0.32). It was feasible to either remove one of the factor loadings or put a limit on both[110, 111]. In order to improve construct definition in relation to high factor loadings and uncorrelated items, the deletion procedure was chosen as the recommended option. [111]. Most of our scale factor' loadings were above 0.53 (see Table 8 and Figure 2), so base for the reliability of our measures was established.

All the constructs' internal consistency reliability was calculated using composite reliability (CR) and Cronbach's alpha (CA). As can be observed in Table 8, the threshold for reliability of the measure is > 0.7 scores of CA for each of the construct[94], thereby estimations met the criteria very well. However, owing to the underestimation problem with CA there is a need of greater estimation of true reliability [94]. Therefore, CR was measured to evaluate the reliability. As shown in Table 8, the model met adequately the acceptable values of CR i.e., > 0.7 for confirmatory purposes.

Table 8 presents the results of the reliability, validity, and quality of measurement assessment for the model.

Construct	Item	Factor Loading (FL)	Cronbach's Alpha (CA)	Composite Reliability (CR)	AVE	V
HDR	HDR1	0.71	0.855	0.888	0.533	1
	HDR2	0.60				
	HDR3	0.69				
	HDR4	0.73				

Construct	Item	Factor Loading (FL)	Cronbach's Alpha (CA)	Composite Reliability (CR)	AVE	V
	HDR5	0.78				
	HDR6	0.76				
	HDR7	0.81				
ABS	ABS1	0.58	0.876	0.901	0.507	3
	ABS2	0.69				
	ABS3	0.70				
	ABS4	0.76				
	ABS5	0.59				
	ABS6	0.73				
	ABS7	0.78				
	ABS8	0.74				
	ABS9	0.81				
ADC	ADC1	0.53	0.891	0.911	0.512	1
	ADC2	0.76				
	ADC3	0.74				
	ADC4	0.72				
	ADC5	0.58				
	ADC6	0.71				
	ADC7	0.70				
	ADC8	0.80				
	ADC9	0.90				
	ADC10	0.66				
ACP	ACP1	0.81	0.859	0.887	0.531	1
	ACP2	0.69				
	ACP3	0.76				
	ACP4	0.75				
	ACP5	0.72				
	ACP6	0.74				
	ACP7	0.61				
AM	AM1	0.74	0.804	0.864	0.560	3
	AM2	0.75				
	AM3	0.76				
	AM4	0.74				
	AM5	0.76				
ACH	ACH1	0.71	0.862	0.892	0.509	3
	ACH2	0.70				
	ACH3	0.74				
	ACH4	0.67				
	ACH5	0.66				
	ACH6	0.74				
	ACH7	0.76				
	ACH8	0.72				
ATD	ATD1	0.90	0.882	0.893	0.627	1
	ATD2	0.81				
	ATD3	0.75				
	ATD4	0.73				
	ATD5	0.77				
ACO	ACO1	0.76	0.798	0.868	0.623	2
	ACO2	0.83				
	ACO3	0.83				

Construct	Item	Factor Loading (FL)	Cronbach’s Alpha (CA)	Composite Reliability (CR)	AVE	V
	ACO4	0.72				
SAZ	SAZ1	0.85	0.896	0.923	0.707	1
	SAZ2	0.84				
	SAZ3	0.83				
	SAZ4	0.79				
	SAZ5	0.89				
AE	AE1	0.75	0.859	0.895	0.587	1
	AE2	0.73				
	AE3	0.75				
	AE4	0.76				
	AE5	0.81				
	AE6	0.78				

In terms of calculating construct convergent validity, average variance explained (AVE) values are used using previous formula. The AVE results were found to range from 0.507 to 0.707 as summarized in Table 8. The AVE results for each individual indicate that all AVE values are greater than the minimum threshold value (>0.5). In addition, all constructs satisfy the AVE criteria, indicating convergent validity.

We reached to the conclusion that discriminant validity is based on < 0.85, which is a threshold that is used in most studies. Table 9 demonstrates that most of the HTMT values are below 0.85, with a few exceptions that are still below 0.90, such as the value for AE which is 0.889. This value is considered acceptable, although it is on the higher side of the liberal threshold [94]. With regards to computing quality of the measurement model by employing the values of communality (H2), all those values were positive for all blocks (as shown in Table 8), ensuring the predictive validity quality of the measurement model.

Table 9 shows the results of the discriminant validity analysis using HTMT ratios.

Item	ABS	ACH	ACO	ACP	ADC	AE	AM	ATD	HDR	SAZ
ABS										
ACH	0.812									
ACO	0.588	0.832								
ACP	0.528	0.417	0.377							
ADC	0.204	0.308	0.128	0.274						
AE	0.872	0.843	0.758	0.352	0.34					
AM	0.868	0.664	0.631	0.511	0.179	0.889				
ATD	0.139	0.184	0.285	0.19	0.108	0.157	0.136			
HDR	0.567	0.575	0.405	0.588	0.141	0.402	0.522	0.177		
SAZ	0.504	0.519	0.553	0.182	0.246	0.739	0.659	0.099	0.382	

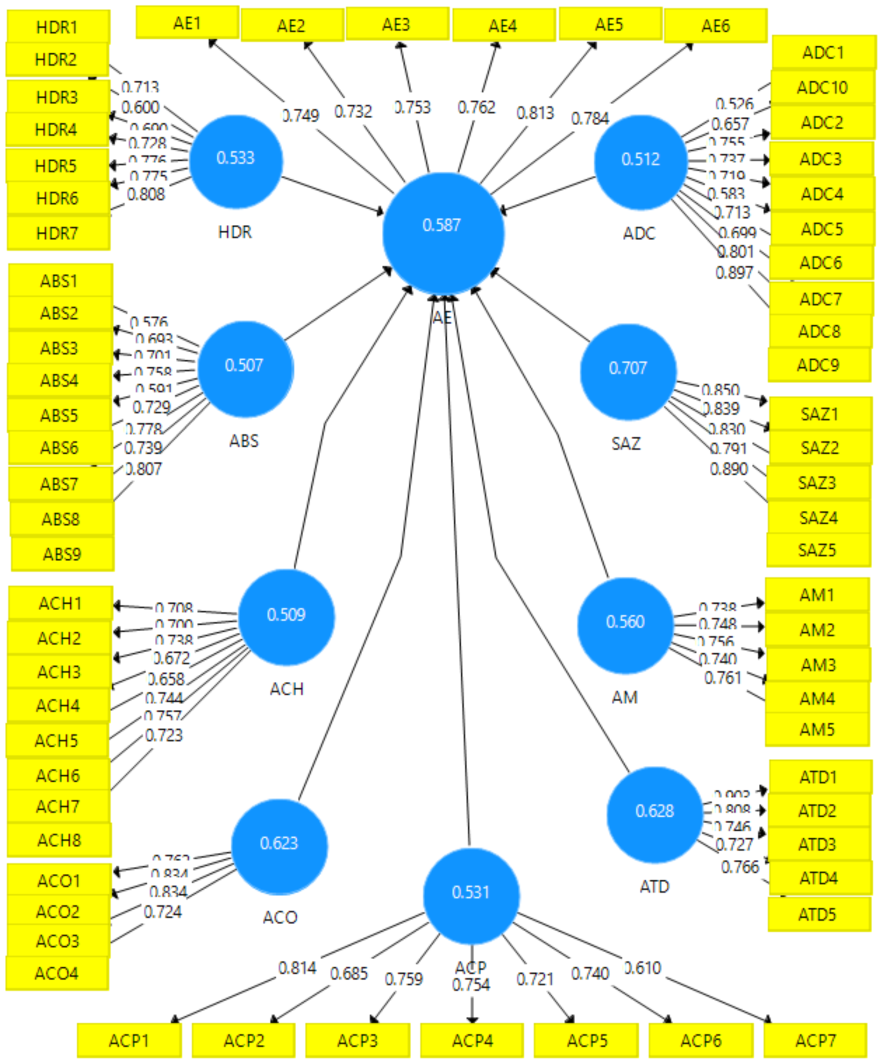

Figure 2. Measurement model of architectural erosion

5.4 Structural Model Assessment

The assessment of the structural model was critical in continuation with the first step of the research model evaluation. A structural model shows the relationship between constructs and related theories based on existing literature. According to study [94] a large number of characteristics were necessary to evaluate the structural model. This model has only direct effect hypothesis only to be tested i.e., direct effect relationships.

5.4.1 Hypothesis Testing

The hypothesis on the direct effect was tested using bias-corrected 95% confidence intervals, and the impact of the independent constructs on AE was determined using the Partial Least Squares Structural Equation Modelling (PLS-SEM) technique. The results of the hypothesis testing and their accompanying interpretations can be viewed in Table 10.

Hypothesis 1 (H1) proposed that ACH significantly identifies an occurrence of the AE, results revealed that ACH had a statistically significant impact on AE ($B = 0.172$, $t = 2.866$, $p = 0.004$), hence H1 was supported; Hypothesis 2 (H2) postulated that ACO significantly identifies an occurrence of the AE, results showed that

ACO had a statistically significant impact on AE (B = 0.119, t = 2.142, p = 0.033), so H2 was supported; Hypothesis 3 (H3) anticipated that ADC significantly identifies an occurrence of the AE, analysis findings showed that ADC had also a statistically significant impact on AE (B = 0.229, t = 4.23, $p < 0.001$), so H3 was also supported; Hypothesis 4 (H4) predicted that ACP significantly identifies an occurrence of the AE, results of the analysis revealed that ACP had not a statistically significant impact on AE (B =0 .063, t = 1.067, p = 0.286), therefore H4 was not supported; Hypothesis 5 (H5) forecasted that ATD significantly identifies an occurrence of the AE, results exhibited that ATD had not a statistically significant impact AE (B = 0.05, t = 0.869, p = 0.385), so H5 was also not supported; Hypothesis 6 (H6) proposed that AM significantly identifies an occurrence of the AE, findings revealed that AM had a statistically significant impact on AE (B = 0.379, t = 6.263, $p < 0.001$), so H6 was approved; Hypothesis 7 (H7) projected that ABS significantly identifies an occurrence of the AE, findings showed that ABS had a statistically significant impact on AE (B = 0.215, t = 3.356, p = 0.001), so H7 was supported; Hypothesis 8 (H8) anticipated that HDR significantly identifies an occurrence of the AE, data analysis revealed that HDR a statistically significant impact on AE (B = 0.152, t = 3.313, p = 0.001), so H8 was also supported; Hypothesis 9 (H9) estimated that SAZ significantly identifies an occurrence of the AE, findings exhibited that SAZ a statistically significant impact on AE (B = -0.196, t = 4.11, $p < 0.001$), so H9 was approved.

Table 10. Findings of the hypothesis tests.

Hypo	Path	St.d Beta	Std. Error	T-value	P-value	Decision
H1	ACH AE	0.172	0.065	2.866	0.004	Supported**
H2	ACO AE	0.119	0.057	2.142	0.033	Supported*
H3	ADC AE	0.229	0.054	4.23	0.000	Supported**
H4	ACP AE	0.063	0.060	1.067	0.286	Not Supported
H5	ATD AE	0.05	0.058	0.869	0.385	Not Supported
H6	AM AE	0.379	0.068	6.263	0.000	Supported**
H7	ABS AE	0.215	0.066	3.356	0.001	Supported**
H8	HDR AE	0.152	0.050	3.313	0.001	Supported**
H9	SAZ AE	-0.196	0.047	4.11	0.000	Supported**

Significant level at p** <0.01, p*<0.05

Based on the empirical evidence, it can be concluded that ACH, ACO, ADC, AM, ABS, HDR, and SAZ have the most significant influence on the identification of the AE construct, as shown in Figure 3. However, the effect of ACP and ATD is not statistically significant, as indicated by the non-significant p-value. Therefore, it is recommended to remove ACP and ATD from the structural model.

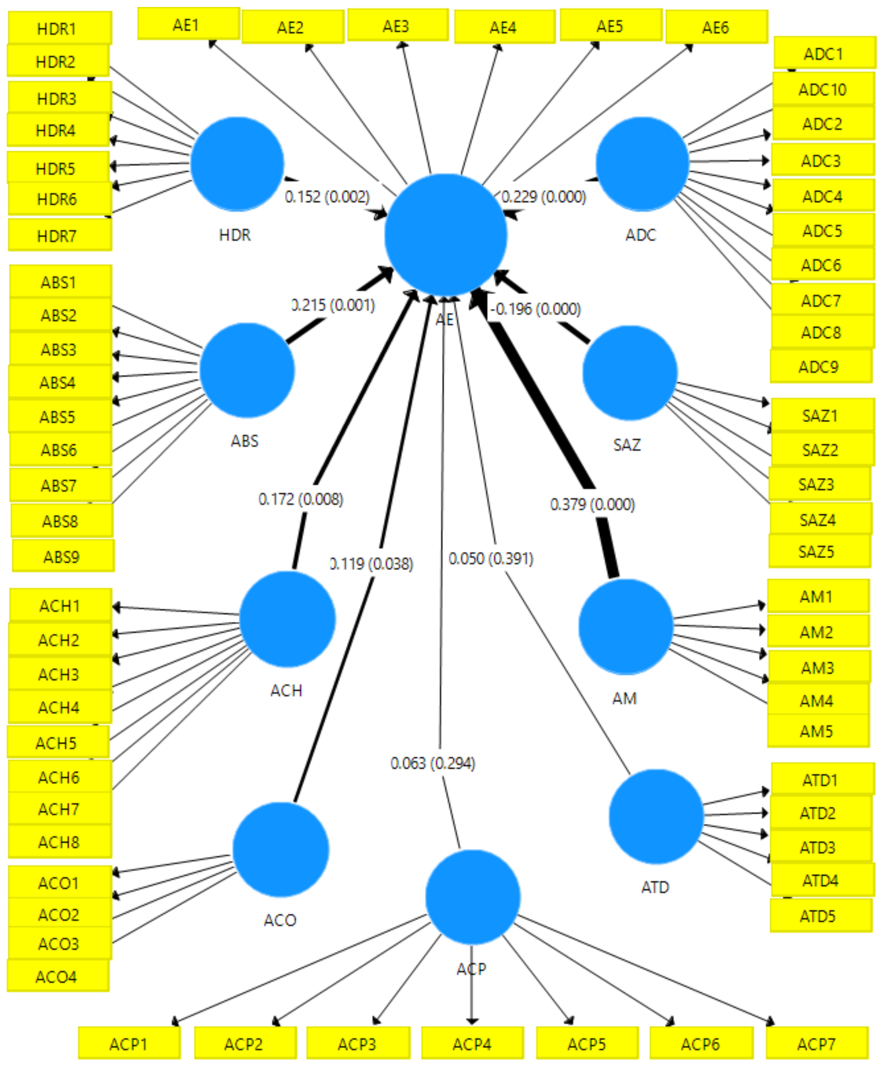

Figure 3. The findings of the structural model's evaluation for architectural erosion

5.4.2 Coefficient of Determination ($\mathbf{R^2}$)

Model accuracy is measured by R^2. This metric illustrates how much endogenous (dependent) variance is accounted for by exogenous constructs. In order to calculate the value of R2 that is associated with the dependent constructions, Smart PLS was utilised. According to the findings, the total R^2 predicted for CCCU was 0.819, which indicates that approximately 81.9% of the variance in identifying AE is explained by its independent components (i.e., ACH, ACO, ADC, AM, HDR, ABS, ACP, ATD, and SAZ). As a result, the value of R^2 is regarded to be high because it is greater than 0.67, as suggested by[101].

5.4.3 Assessment of Effect Size ($\mathbf{F^2}$)

In accordance with the classification approach proposed by[102] and the approach proposed by [94], Table 11 shows that, in the context of the constructs impacting the identification of AE. ADC and AM were ascertained to have a medium effect size. Concurrently, it was determined that the effect sizes for ACH, ACO, ABS, HDR, and SAZ were all small, while the remaining constructs which are ACP and ATD, had a no effect size.

Table 11. The results of effect (F^2)

Construct	F^2	Effect size status
ACH	0.048	Small
ACO	0.034	Small
ADC	0.189	Medium
ACP	0.013	No effect
ATD	0.012	No effect
AM	0.240	Medium
ABS	0.075	Small
HDR	0.076	Small
SAZ	0.117	Small

5.4.4 Assessment of Predictive Relevance (Q^2)

If the value of Q2 is more than zero, the model is said to have the potential to have predictive relevance for a cretin endogenous component, as stated by [94]. The value of Q2 corresponding to the endogenous constructs (i.e., AE) is 0.446, thus, this demonstrates that the model possesses the required predictive relevance.

5.4.5 Goodness of Fit of the Model - GoF

Combining effect size and convergent validity is how the GOF is measured[105] and the allowable range for this measurement is 0 to 1. GOF value was well-acceptable, whereas obtained value is 0.693. This value is considered to be large as stated by [108]when above of 0.36. Consequently, the overall fitness of the structural model was demonstrated to be confirmed by the findings of this investigation. Moreover, multicollinearity for the structural model was also assessed. VIF values beyond 5 are indication of collinearity[94], inner VIF value which is 3.402 showed that structural model was well fitted. Figures in the supplementary materials provide a detailed structural model with or without architectural technical debt and complexity constructs.

6. Discussion

In this section, the most important results obtained will be dealt with, analysed and discussed, and the implications of the importance of the results will be clarified in terms of their division. In addition, some recommendations will be presented in order to identify future research directions in relation to this domain.

6.1 Instrument validity and reliability

After the preliminary work on the software metrics model for architectural erosion that was published in [13].This idea was originally created using a set of 92 metrics representing 10 quality factors for monitoring architectural erosion. Each metrics practice was assigned to one of the measures approaches (10, 11, 22, 7, 10, 13, 5, 5, and 9 items for Historical data revision, Architectural bad smell, Architectural dependency coupling, Architectural complexity, Architecture modularization, and Architectural change, Architectural technical Debt, Architectural cohesion, and Software architecture size respectively.

Following a strict methodology, including a content validity evaluation process and a reliability analysis, the model was fine-tuned. Experts provided feedback during the content validity round, suggesting answers and comments that would help refine the language used and explicitly declare the convergence of some metrics and their refinement into a single concept that explains the extent of its strong significance through the various definitions of some items. This is what provides a considerable explanation for the integration of many convergent items in an abstract idea to recognize a concept that aims for the coherence of a unified definition. It should be noted that many of the metrics items were proposed by several researchers. These same items are used in more than one study, each time with a concept that is quite different from the one used in the other study. As a consequence of this, a large number of metrics were commonly used among the researchers based on the unified understanding of the study's authors. Thus, this model provides an initial contributory perception that can be utilized to establish and refine the metrics of unified significance under the classification or appropriate approach appropriate approach for identifying architectural erosion.

In historical data revision construct, HDR1 and HDR4 items were combined to form one coherent concept, while the HDR3 and HDR10 items were deleted because the required criteria were not met. It is also noticeable in architectural dependence coupling construct stands out because it has the most items of any of the others. However, six of these items are so similar in definition—each one converges with another in the same concept—that they could be reduced to only three. On the other hand, eight other items of this construct were eliminated since they did not fulfil the requirements. A similar observation holds correct when we examined the architectural erosion construct, we found five items—AE6, AE7, AE8, AE9, and AE10—present a convergent concept that is essentially one. This is probably since past research and expert judgement have concluded that all of these things represent only a single concept. This led to their consolidation into a single concept, which satisfies all the necessary criteria. The rest of the other constructs are only included in the removal of some items due to the failure to meet these criteria. Notably, the content validation was adopted in the first round and subsequently passed to the same experts, with no changes made to the path taken in the adopted round. It would be inappropriate to proceed with another round using the same items and evaluation criteria [112].

As soon as the agreement was established on the model's content validity, an additional round was included to investigate the items' and construct's internal consistency reliability. Researchers with backgrounds in software engineering were asked to evaluate items on the constructs again, using a 5-point Likert scale. In this context, the reliability, as determined by Cronbach's alpha, is evidence that the items represent the constructs in a consistent manner. All the items per each construct indicate none of them significantly reduced the value of the alpha coefficient if they were removed from the construct, except for HDR1, ABS1, and ACP7 items; their values increased from 0.874 to 0.876, 0.922 to 0.928, and 0.854 to 0.858 respectively. However, increasing the value of these items is a minor change that will have no effect on the items and their constructs, especially since these values represent the study's initial sample. Furthermore, even though these are the least closely related items, their corrected item-total correlation is high, and these values are greater than the specified value of 0.2. Therefore, the resulting value indicates that the model has high reliability.

6.2 Measurement Model Assessment

When conducting an analysis of a research model, the first thing that has to be done is to take into consideration the measurement model assessment. This is because the measurement model is the one that decides whether or not each construct being tested is being correctly assessed. In this respect, the findings indicated that the quality of the predictive validity of the measurement model was ensured, which was based on the criteria that were decided to be investigated in order to evaluate this model. This explains that examining the content in terms of formulating items and categorizing them within the appropriate construct and reconsidering the concepts in a comprehensive and investigative manner, in addition to the consistency of reliability, has provided a strong indication of the coherence and ensuring the predictive power of the model in terms of constructs and their items.

In connection with the results obtained, the model was passed on the criteria, as the measuring factor loading should be at least 0.50 as stated by [94, 109] in order to be reliable, most of our scale factor loadings were above 0.53, thereby, every item must have at least this value to be considered for preservation, hence it's important that this number be as high as possible. Composite reliability (CR) revealed that internal consistency reliability the threshold for the reliability of the measure is > 0.7 which is a greater estimation of true reliability than Cronbach's alpha (CA), and whether the values are Composite reliability (CR) or for Cronbach's alpha (CA), they all achieve the required value, thereby, it could be that the concept of reliability to present as an indication that has a significant impact when two terms work with each other to produce same results with high considerable. Furthermore, all constructs and items that meet AVE criteria and HTMT values for predictability in the measurement model were shown to have evidence of convergent and discriminant validity.

6.3 Structural Model Assessment

In this perspective, hypotheses were established to test the possibility of a relationship between classifications of various metrics approach and the architectural erosion. The hypotheses were empirically tested to determine whether the categorization of metrics approaches significantly identifies architectural erosion. The results of the evaluation of the structural model show that all classifications of various metrics approach significantly identify the architectural erosion except for the architectural complexity (ACP) and architectural technical debt (ATD).

We observed empirical evidence for ACH significantly identifies an occurrence of the AE based on obtained result (B = 0.172, t = 2.866, $p < 0.004$, F2 = 0.048), hypothesis H1 can be accepted. This demonstrates that ACH is considered to be a crucial stage in the process of architectural tactics and knowledge analysis [113], which aims to predict problems in a software system using metrics that can identify architectural deterioration [19, 114]. In addition to this, assessing architectural change can provide clues as to the presence of flaws [32], as well as information regarding the connection between architecture change, deterioration, and the manifestation of vulnerabilities [27]. In same context, as a consequence of the fact that the ACO and ADC significantly detect an occurrence of the AE (B =0.119, t = 2.142, p <0.033, F2 =.034; B = 0.229, t = 4.23, $p < 0.001$, F^2= 0.189 respectively), hypothesis 2 (H2) and hypothesis 3 (H3) were able to be accepted as valid hypotheses. This agrees with the findings of the architectural cohesion and coupling analysis [15, 31], and it is a major approach to identifying architectural inconsistency by analysing system releases based on the degree of coherence and dependency coupling between the modules, packages, or classes. Focusing on hypothesis 6 (H6), introduced satisfied results in terms of identifying an occurrence of the AE (B = 0.379, t = 6.263, p < 001, F2 = 0.238). Therefore, the findings of the correlation between AM and the emergence of AE are consistent with the observations by[115], besides, modularization metrics provide a better representation picture for fault prediction, design flaw detection, identifying source code anomalies and architectural degradation[116] , as well as improving architecture[117] with regard to the analysis of faults and changes that could be isolated and separated. Moreover, the results regarding the relationship between ABS and the occurrence of AE (B = 0.215, t = 3.356, p 0.001, F2 = 0.075) are consistent with the analysis and examination of the various study on architectural smells and their relationship to determine the presence of architectural erosion[55, 56, 59, 60, 118] and instability in order to identify hidden defects in software architecture [119]. Concerning hypothesis 8 (H8), it was determined that statistically indicates an occurrence of the AE (B = 0.152, t = 3.313, $p < 0.001$, F2 = .074).This provides significant evidence supporting the hypothesis, which is in line with research showing that historical metrics may automatically discover violations of design principles[34], high-performance forecasting of low architectural quality (i.e., architectural erosion) in architectural modules, and even in the case of rapid decay [19], estimation of severity based on analysis of change over time [37], as well as historical metrics for assessing and forecasting architecture quality condition [36].

Specifically, hypothesis 9 (H9) exhibited SAZ findings in terms of identification of AE (B = -0.196, t = 4.11, p <0.00, F2 = 0.117) which significantly negative impact. Mostly since the hypotheses were looking at both sides in terms of determining the erosion, rather than the effect. The appearance of size metrics with a negative determination does not necessarily imply that using such metrics would be useless. As a result, many studies [15, 31, 47, 50]have pointed out the significance of studying the metrics of architecture size from this perspective, suggesting that these metrics may be reached in the study of the size of architecture with other measurements of another classification (e.g., cohesion or coupling metrics) or analysis of negative side for erosion in terms of effect.

According to the empirical results of hypotheses 4 (H4) and (H5), the findings showed no significant identification on AE (B = 0.063, t = 1.067, p <0.286, F2 = 0.013; B = 0.05, t = 0.869, $p < 0.385$, F2 = 0.012) respectively. Hence, the hypothesis (H5) which is ATD was not supported. [15] indicated that ATD metrics are not appropriate to identify architectural inconsistencies. In contrast, [120] indicated that bad code smells contribute to technical debt. Code smell correlates more with technical debt than architectural degradation. Few studies include architectural degradation and code smell, suggesting it's linked to technical debt. Therefore, this observation demands further investigation and consideration.

The quality of the output of the structural model suggests that the predictive quality, which was measured by Q2, R2, and goodness of fit (GOF), is well-acceptable. Furthermore, the values of Q2, R2, and GOF are regarded as high, which confirmed the overall fitness of our structural model. This reveals a general observation of these results that all classifications of metrics approaches have a statistically significant for identifying architectural erosion, except for the ATD and ACP approaches, which were not statistically supported. In the same line, it would be wonderful to see an initiative done to empirically analyse this relation within the context of an independent study. This would be a terrific step forward. As a consequence of this, the study model can be utilized for the purpose of recognizing the context of architectural erosion.

7. Implications for research and practice

The primary objective of this study was to develop a model that would customize and categorize metrics used in previous studies to identify architectural erosion. The validity and reliability of the model were empirically evaluated to prove its effectiveness and improvement. This study aimed to address the research gap and make a valuable contribution to the literature for future researchers pursuing similar goals.

As a matter of fact, the model provides a unique broad classification based on various metrics approaches to address architectural erosion for academic research from different aspects (e.g., varieties of measurement, classifications of metrics approaches, and quality attributes of monitoring architectural erosion). This model provides also researchers with methods for assessing architectural erosion from multiple angles, such as the combination of these classifications or relevant metrics to propose solutions approach in a different manner that could have considerable efficiency rather than ones.

In addition, the model highlights to researchers and practitioners the ambiguity in directly investigating erosion in terms of the approach that is taken to handle this phenomenon. In anticipation of additional research attention in this field, the improved model will also be an effective and useful tool for classifying the current and forthcoming metrics for addressing and recognizing architectural erosion. This is because the developed model will consider the effects of architectural erosion in a more comprehensive manner.

In this respect, the empirical findings of this study will assist software developers and architects in understanding the relationship between architectural erosion and software metrics. Additionally, the developed model primarily focuses on the most classifications of measure approaches and most-related quality attributes architectural degradation, which makes it easier for identifying erosion to conduct control assessment. This contrasts with evaluating all the approaches of various metrics with mapping all quality attributes.

8. Threats to Validity of the study

This section discusses the four key threats to the validity which are described [121, 122] in this study and the mitigation proposed to deal with them are below.

8.1 Threats to external Validity

The ability of the researchers to generalize the findings of academic research to applications in the industry is one of the factors that contribute to the study's external validity [121]. The number of respondents in this study should be deemed a sufficient sample size for the SEM analysis [123]. Although this number of people that participated in the survey could lead to potential criticism of this study for having limited generalizability, but the model is simple and the sample of the target population is restricted to specific experts around the world, as well as, target respondents have a variety of academic and industrial backgrounds and experiences from different countries.

8.2 Threats to internal Validity

Internal validity lies in the extent to which the independent factors that influence or cause a change in the dependent factors are determined[122]. One of the most important limitations in this study is the extent to which different metrics approaches (independent variables) are selected within an appropriate classification and the extent to which it is able to determine architectural erosion. It is acknowledged that there may be alternative approaches for assessing architectural erosion, however, this study is limited to metric-based approaches that have been demonstrated to be the most popular solutions in the field. The findings of relevant studies were analysed to categorize these approaches into relevant classifications and provide a clear understanding of architectural erosion.

8.3 Threats to construct Validity

The concept of construct validity is concerned with the degree to which there exists a gap between the theoretical depiction of a concept and its practical use [124]. One of the most significant limitations that poses a threat to the construct's validity and reliability is the self-selection bias for respondents chosen. In current investigation, the potential for concerns was addressed by identifying the most suitable experts and engaging with them through a comprehensive mapping study process. The goal was to present the actual picture of the concepts in a manner that was correct and accurate based on knowledge and experience that the experts possess in the relevant field of work. Another threat to this validity is the collection of metrics and their association within specific approaches categories. To mitigate this threat, the metrics used in the study are widely used and well established in the literature. Thus, they accurately represent the concepts they propose to measure. Furthermore, the specified metrics represent all the concepts and approaches to these categories, as well as experts, agreed these categories at the stage of the content validity and reliability.

8.4 Threats to conclusion validity

The ability to draw a correct and legitimate conclusion about a connection between independent variables provided by the experiment (dependent variable) resulting from it is what is being evaluated by the concept of conclusion validity [122]. According to[94], VIF values should be less than 5, so we excluded common method bias (CMB) by ensuring that no predictor variable had a VIF of more than 3.402 as shown in Table 8. Another limitation of this study is the use of immature subjects, which may represent a significant threat to conclusion validity. However, our study's representative sample was comprised of professionals working on this topic, whether they belong to academic or industry fields.

9. Conclusions and Future Research

The comprehension of the significant metrics approaches and practices in the identification of architectural erosion context and the identification of the key quality attributes associated with architecture degradation is an opportunity to deepen knowledge of erosion and generating additional discussions on the topic. Since the primary objective of the study is to develop a model with aim to assess and classify the metrics within the earlier approaches that have been used in studies to recognize architectural degradation. Furthermore, it is possible to consider this work to be the first one of its kind, to the best of the authors' knowledge, to build a theoretical model of metric approaches classification for architectural erosion that has been validated and is reliable. To evaluate this model from an initial conceptualization perspective, instrument content validity and reliability testing were used. A questionnaire-based survey was used to collect data from 130 software engineering professionals with experience in the architecture erosion and software metrics. After several steps are achieved in terms of measurement and structural mode, the needed model fit, reliability, and validity were attained. The findings revealed that there is a significant relationship between all the classifications of metrics approaches and architectural erosion, with the exception of architectural complexity and architectural technical debt.

During this research, various concepts and improvements were considered as potential directions for future work. Specifically, the focus was on established metrics and quality attributes for monitoring architectural erosion, based on systematic mapping studies. This study was limited to the investigation of these approaches. The authors of this work do not intend to imply, however, that these approaches of established metrics and quality attributes of monitoring degradation associated with architectural erosion are unique. As a result, further study is needed to fully broaden the model's scope to include many different perspectives. Second, although this study introduced an empirical report on each classification of metrics approaches on identification of the architectural erosion, it did not consider the identification of each classification of metrics approaches on each quality attribute of architectural erosion. In subsequent research, it may be possible to investigate these relationships, for instance, the measures of architectural complexity approach for identification of the usability (e.g., understandability attribute) (as one of the indicators of monitoring architecture degradation) In addition, despite the fact that the measures of architectural complexity approach (external change) for determining architectural erosion was not statistically supported in this study, it would be interesting to see if there was an initiative to empirically investigate the relationships between the measures of architectural complexity approach and identification of the architectural erosion independently. Third, on the other hand, according to an analysis of the systems utilized in the previous studies, monolithic architecture is one of these measures that is most frequently used. This demonstrates to developers and architects that this model is appropriate for systems based on monolithic architecture, however, the applying this model to another architecture may result in inconsistent outputs (e.g., the microservice architecture). From this point, researchers have a great opportunity to do an additional empirical study based on a combination of system contexts with different architectural patterns, with the goal of finding a solution to this issue.

Acknowledgment

The authors wish to show their appreciation to Universiti Putra Malaysia (UPM) for their assistance in certain aspects of this work. The first author would also like to extend his thanks to the Hadhramout Foundation and Hadhramout University in Yemen for their financial support towards tuition fees.

Reference

1. Clements P, Garlan D, Bass L, et al. Documenting Software Architectures: Views and Beyond. 2002.

2. Shaw M, Clements P. The golden age of software architecture. IEEE Software. 2006;23(2):31-39.

3. Garlan D. Software Architecture: A Roadmap. Proc of the 22nd International Conference on Software Engineering, Future of Software Engineering Track. 2000.

4. Dobrica L, Niemela E. A survey on software architecture analysis methods. IEEE Transactions on Software Engineering. 2002;28(7):638-653.

5. Lehman MM. On understanding laws, evolution, and conservation in the large-program life cycle. Journal of Systems and Software. 1979;1:213-221.

6. Lehman MM. Laws of software evolution revisited. In Montangero C, (Ed). Software Process Technology. Berlin, Heidelberg: Springer Berlin Heidelberg 1996:108-124.

7. Stringfellow C, Amory CD, Potnuri D, et al. Comparison of software architecture reverse engineering methods. Information and Software Technology. 2006;48(7):484-497.

8. Bosch J. Software Architecture: The Next Step. In Oquendo F, Warboys BC, Morrison R, (Eds). Software Architecture. Berlin, Heidelberg: Springer Berlin Heidelberg 2004:194-199.

9. Hochstein L, Lindvall M. Combating architectural degeneration: a survey. Information and Software Technology. 2005;47(10):643-656.

10. Medvidovic N, Egyed A, Grünbacher P. Stemming Architectural Erosion by Coupling Architectural Discovery and Recovery. Proc of the 2nd International Software Requirements to Architectures Workshop. 2003.

11. Parnas DL. Software aging. Proceedings of the 16th international conference on Software engineering. Sorrento, Italy: IEEE Computer Society Press 1994:279–287.

12. Li R, Liang P, Soliman M, et al. Understanding software architecture erosion: A systematic mapping study. Journal of Software: Evolution and Process. 2022;34(3):e2423.

13. Baabad A, Zulzalil HB, Hassan S, et al. Characterizing the Architectural Erosion Metrics: A Systematic Mapping Study. IEEE Access. 2022;10:22915-22940.

14. Perry DE, Wolf AL. Foundations for the study of software architecture. 1992;17(4 %J SIGSOFT Softw. Eng. Notes):40–52.

15. Lenhard J, Blom M, Herold S. Exploring the suitability of source code metrics for indicating architectural inconsistencies. Software Quality Journal. 2019;27(1):241-274.

16. Taylor RN, Medvidovic N, Dashofy E. Software architecture: foundations, theory, and practice: John Wiley & Sons 2009.

17. Baabad A, Zulzalil HB, Hassan S, et al. Software Architecture Degradation in Open Source Software: A Systematic Literature Review. IEEE Access. 2020;8:173681-173709.

18. Riaz M, Sulayman M, Naqvi H. Architectural Decay during Continuous Software Evolution and Impact of 'Design for Change' on Software Architecture 2009.

19. Garcia J, Kouroshfar E, Ghorbani N, et al. Forecasting Architectural Decay From Evolutionary History. IEEE Transactions on Software Engineering. 2022;48(7):2439-2454.

20. Baabad A, Zulzalil HB, Hassan S, et al. A Survey on Characterizing the Empirical Analysis, Proposed Approaches, and Research Trends for Architectural Decay. International Journal of Software Innovation (IJSI). 2022;10(1):1-18.

21. Misra S, Adewumi A, Fernandez-Sanz L, et al. A Suite of Object Oriented Cognitive Complexity Metrics. IEEE Access. 2018;6:8782-8796.

22. Misra S. An Approach for the Empirical Validation of Software Complexity Measures. Acta Polytechnica Hungarica. 2011;8.

23. DeMarco T. Controlling Software Projects: Management, Measurement, and Estimates: Prentice Hall PTR 1986.

24. Ding W, Liang P, Tang A, et al. How Do Open Source Communities Document Software Architecture: An Exploratory Survey. 2014 19th International Conference on Engineering of Complex Computer Systems. Tianjin, China: IEEE 2014:136-145.

25. Lenhard J, Hassan MM, Blom M, et al. Are code smell detection tools suitable for detecting architecture degradation? Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings. Canterbury, United Kingdom: Association for Computing Machinery 2017:138–144.

26. Aversano L, Guardabascio D, Tortorella M. An Empirical Study on the Architecture Instability of Software Projects. International Journal of Software Engineering and Knowledge Engineering. 2019;29(04):515-545.

27. Sejfia A. A Pilot Study on Architecture and Vulnerabilities: Lessons Learned. 2019 IEEE/ACM 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE). Montreal, QC, Canada: IEEE 2019:42-47.

28. Shahbazian A, Nam D, Medvidovic N. Toward predicting architectural significance of implementation issues. Proceedings of the 15th International Conference on Mining Software Repositories. Gothenburg, Sweden: Association for Computing Machinery 2018:215–219.

29. Behnamghader P, Le DM, Garcia J, et al. A large-scale study of architectural evolution in open-source software systems. Empirical Software Engineering. 2017;22(3):1146-1193.

30. Laser MS, Medvidovic N, Le DM, et al. ARCADE: an extensible workbench for architecture recovery, change, and decay evaluation. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Virtual Event, USA: Association for Computing Machinery 2020:1546–1550.

31. Barros MdO, Farzat FdA, Travassos GH. Learning from optimization: A case study with Apache Ant. Information and Software Technology. 2015;57:684-704.

32. Steff M, Russo B. Measuring Architectural Change for Defect Estimation and Localization. 2011 International Symposium on Empirical Software Engineering and Measurement. Banff, AB, Canada: IEEE 2011:225-234.

33. Sangwan RS, Vercellone-Smith P, Neill CJ. Use of a multidimensional approach to study the evolution of software complexity. Innovations in Systems and Software Engineering. 2010;6(4):299-310.

34. Mo R, Cai Y, Kazman R, et al. Architecture Anti-Patterns: Automatically Detectable Violations of Design Principles. IEEE Transactions on Software Engineering. 2021;47(5):1008-1028.

35. Schwanke R, Xiao L, Cai Y. Measuring architecture quality by structure plus history analysis. Proceedings of the 2013 International Conference on Software Engineering. San Francisco, CA, USA: IEEE Press 2013:891–900.

36. Reimanis D, Izurieta C, Luhr R, et al. A replication case study to measure the architectural quality of a commercial system. Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. Torino, Italy: Association for Computing Machinery 2014:Article 31.

37. Singh R, Bindal A, Kumar A. Reducing maintenance efforts of developers by prioritizing different code smells. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2019;8(8S3):2223-2232.

38. Altınışık M, Ersoy E, Sözer H. Evaluating software architecture erosion for PL/SQL programs. Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings. Canterbury, United Kingdom: Association for Computing Machinery 2017:159–165.

39. Mohsin S, Akhtar H, Jalbani., Adil A, et al. Evaluating Dependency based Package-level Metrics for Multi-objective Maintenance Tasks. International Journal of Advanced Computer Science and Applications(IJACSA). 2017;8(10).

40. Macia I, Garcia J, Popescu D, et al. Are automatically-detected code anomalies relevant to architectural modularity? an exploratory analysis of evolving systems. Proceedings of the 11th annual international conference on Aspect-oriented Software Development. Potsdam, Germany: Association for Computing Machinery 2012:167–178.

41. Abdeen H, Ducasse S, Sahraoui H, et al. Automatic Package Coupling and Cycle Minimization. 2009 16th Working Conference on Reverse Engineering. Lille, France: IEEE 2009:103-112.

42. Nayebi M, Cai Y, Kazman R, et al. A longitudinal study of identifying and paying down architecture debt. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice. Montreal, Quebec, Canada: IEEE Press 2019:171–180.

43. Mo R, Cai Y, Kazman R, et al. Decoupling level: a new metric for architectural maintenance complexity. Proceedings of the 38th International Conference on Software Engineering. Austin, Texas: Association for Computing Machinery 2016:499–510.

44. Roveda R, Fontana FA, Pigazzini I, et al. Towards an architectural debt index. Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018. Prague, Czech Republic: IEEE 2018:408-416.

45. MacCormack A, Sturtevant DJ. Technical debt and system architecture: The impact of coupling on defect-related activity. Journal of Systems and Software. 2016;120:170-182.

46. Fontana FA, Pigazzini I. Evaluating the Architectural Debt of IoT Projects. 2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT). Madrid, Spain: IEEE 2021:27-31.

47. Rocha H, Durelli RS, Terra R, et al. DCL 2.0: modular and reusable specification of architectural constraints. Journal of the Brazilian Computer Society. 2017;23(1):12.

48. Stevanetic S, Haitzer T, Zdun U. Supporting Software Evolution by Integrating DSL-based Architectural Abstraction and Understandability Related Metrics. Proceedings of the 2014 European Conference on Software Architecture Workshops. Vienna, Austria: Association for Computing Machinery 2014:Article 19.

49. Guimaraes E, Garcia A, Cai Y. Exploring Blueprints on the Prioritization of Architecturally Relevant Code Anomalies – A Controlled Experiment. 2014 IEEE 38th Annual Computer Software and Applications Conference. Vasteras, Sweden: IEEE 2014:344-353.

50. Maisikeli SG. Measuring Architectural Stability and Instability in the Evolution of Software Systems. 2018 Fifth HCT Information Technology Trends (ITT) 2018:263-275.

51. Fontana FA, Roveda R, Vittori S, et al. On evaluating the impact of the refactoring of architectural problems on software quality. Proceedings of the Scientific Workshop Proceedings of XP2016. Edinburgh, Scotland, UK: Association for Computing Machinery 2016:Article 21.

52. Fontana FA, Roveda R, Zanoni M, et al. An Experience Report on Detecting and Repairing Software Architecture Erosion. 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA) 2016:21-30.

53. Li Z, Madhavji NH, Murtaza SS, et al. Characteristics of multiple-component defects and architectural hotspots: a large system case study. Empirical Software Engineering. 2011;16(5):667-702.

54. Macia I, Garcia A, Chavez C, et al. Enhancing the detection of code anomalies with architecture-sensitive strategies. Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR. Genova, Italy: IEEE 2013:177-186.

55. Guimaraes E, Garcia A, Figueiredo E, et al. Prioritizing software anomalies with software metrics and architecture blueprints. 2013 5th International Workshop on Modeling in Software Engineering (MiSE). San Francisco, CA, USA: IEEE 2013:82-88.

56. Guimarães E, Garcia A, Cai Y. Architecture-sensitive heuristics for prioritizing critical code anomalies. Proceedings of the 14th International Conference on Modularity. Fort Collins, CO, USA: Association for Computing Machinery 2015:68–80.

57. Ferreira M, Barbosa E, Macia I, et al. Detecting architecturally-relevant code anomalies: a case study of effectiveness and effort. Proceedings of the 29th Annual ACM Symposium on Applied Computing. Gyeongju, Republic of Korea: Association for Computing Machinery 2014:1158–1163.

58. Zhong C, Huang H, Zhang H, et al. Impacts, causes, and solutions of architectural smells in microservices: An industrial investigation. Software: Practice and Experience. 2022;n/a(n/a).

59. Fontana FA, Ferme V, Zanoni M. Towards Assessing Software Architecture Quality by Exploiting Code Smell Relations. 2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics. Florence, Italy: IEEE 2015:1-7.

60. Biaggi A, Fontana FA, Roveda R. An Architectural Smells Detection Tool for C and C++ Projects. 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Prague, Czech Republic: IEEE 2018:417-420.

61. Fontana FA, Pigazzini I, Roveda R, et al. Automatic detection of instability architectural smells. Proceedings - 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016. Raleigh, NC, USA: IEEE 2017:433-437.

62. Martin RC. Agile software development: principles, patterns, and practices. Upper Saddle River, NJ, USA: Prentice Hall PTR 2003.

63. Zapalowski V, Nunes DJ, Nunes I. Understanding architecture non-conformance: why is there a gap between conceptual architectural rules and source code dependencies? Proceedings of the XXXII Brazilian Symposium on Software Engineering. Sao Carlos, Brazil: Association for Computing Machinery 2018:22–31.

64. Capiluppi A, Knowles T. Software Engineering in Practice: Design and Architectures of FLOSS Systems. In Boldyreff C, Crowston K, Lundell B, et al., (Eds). Open Source Ecosystems: Diverse Communities Interacting. Berlin, Heidelberg: Springer Berlin Heidelberg 2009:34-46.

65. Zapalowski V, Nunes I, Nunes DJ. The WGB method to recover implemented architectural rules. Information and Software Technology. 2018;103:125-137.

66. Lindvall M, Tesoriero R, Costa P. Avoiding architectural degeneration: an evaluation process for software architecture. Proceedings Eighth IEEE Symposium on Software Metrics. Ottawa, Ontario, Canada: IEEE 2002:77-86.

67. Little RJA. A Test of Missing Completely at Random for Multivariate Data with Missing Values. Journal of the American Statistical Association. 1988;83(404):1198-1202.

68. Segre S. Contemporary Sociological Thinkers and Theories. United Kingdom: Ashgate Publishing, Ltd 2014.

69. Lohr SL. Sampling Design and Analysis: Chapman & Hall 2022.

70. Navarro Sada, Maldonado A. Research Methods in Education. British Journal of Educational Studies. 2007;4(55):469-470.

71. Bryman ABE. Business Research Methods. 2015.

72. Hair Jr JF AR, Tatham RL, Black WC. Multivariate data analysis: New Jersey: Pearson education. 2015.

73. Lynn MR. Determination and Quantification Of Content Validity. Nursing Research. 1986;35(6).

74. Polit DF, Beck CT. The content validity index: are you sure you know what's being reported? Critique and recommendations. Research in nursing & health. 2006;29(5):489-497.

75. Zamanzadeh V, Ghahramanian A, Rassouli M, et al. Design and Implementation Content Validity Study: Development of an instrument for measuring Patient-Centered Communication. Journal of caring sciences. 2015;4(2):165-178.

76. Brace I. Questionnaire design: How to plan, structure and write survey material for effective market research: Kogan Page Publishers 2018.

77. MacKenzie SB, Podsakoff PM, Podsakoff NP. Construct Measurement and Validation Procedures in MIS and Behavioral Research: Integrating New and Existing Techniques. MIS Quarterly. 2011;35(2):293-334.

78. Teimour A, Narmin Hassanzadeh R, Yahya K, et al. Development and Evaluation of a New Questionnaire for Rating of Cognitive Failures at Work. International Journal of Occupational Hygiene. 2011;3(1).

79. Lawshe CH. A quantitative approach to content validity. Personnel Psychology. 1975;28(4):563-575.

80. Ayre C, Scally AJ. Critical Values for Lawshe's Content Validity Ratio: Revisiting the Original Methods of Calculation. Measurement and Evaluation in Counseling and Development. 2013;47(1):79-86.

81. Rubin A, Bellamy J. Practitioner's guide to using research for evidencebased practice: Malaysia: John Wiley & Sons. 2012.

82. Hertzog MA. Considerations in determining sample size for pilot studies. Research in nursing & health. 2008;31(2):180-191.

83. Sekaran U, Bougie R. Research methods for business: A skill building approach. Chichester, England: john wiley & sons 2016.

84. Sekaran U, Bougie R. Research Methods For Business: A Skill Building Approach. Hoboken: John Wiley & Sons. 2016.

85. Churchill GA, Iacobucci D. Marketing research: methodological foundations: Dryden Press New York 2006.

86. Cooper DR, Schindler PS, Sun J. Business research methods: Mcgraw-hill New York 2006.

87. Kasi PM. Research: What, Why and How?: A Treatise from Researchers to Researchers. United State: AuthorHouse. 2009.

88. Gorsuch R. Factor analysis (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates 1983.

89. Kline RB. Principles and practice of structural equation modeling (Third edition). United State: Guilford publications 2015.

90. Olerup A. Design Approaches: A Comparative Study of Information System Design and Architectural Design. The Computer Journal. 1991;34(3):215-224.

91. Tabachnick BG, Fidell LS, Ullman JB. Using multivariate statistics: pearson Boston, MA 2007.

92. Byrne BM. Structural Equation Modeling with AMOS: Basic Concepts, Applications and Programming (second edition). New York:: Taylor & Francis Group 2010.

93. Teddlie C, Yu F. Mixed Methods Sampling: A Typology With Examples. Journal of Mixed Methods Research. 2007;1(1):77-100.

94. Hair J, Hult GTM, Ringle C, et al. A Primer on Partial Least Squares Structural Equation Modeling (SECOND EDITION). Thousand Oaks: SAGE, London 2016.

95. Werts CE, Linn RL, Jöreskog KG. Intraclass Reliability Estimates: Testing Structural Assumptions. Educational and Psychological Measurement. 1974;34(1):25-33.

96. Hinton PR, McMurray, I., & Brownlow, C. SPSS Explained (2nd ed.). London: Routledge 2014.

97. Höck M, Ringle CM. Strategic networks in the software industry: An empirical analysis of the value continuum. IFSAM VIIIth World Congress 2006.

98. Henseler J, Ringle CM, Sarstedt M. A new criterion for assessing discriminant validity in variance-based structural equation modeling. Journal of the Academy of Marketing Science. 2015;43(1):115-135.

99. Otero C. Software Engineering Design Theory and Practice (1st Edition). New York: CRC Press 2012.

100. Hair J. F, Black W. C, Babin B. J, et al. Multivariate data analysis (Seventh edition Pearson new international). London, UK: Pearson Education Limited. 2014.

101. Chin WW. The partial least squares approach to structural equation modeling. Modern Methods for Business Research. 1998;295(2):295-336.

102. Cohen J. Statistical power analysis for the behavioral sciences. Academic Press: Cambridge, MA, USA 2013.

103. Stone M. Cross-validatory choice and assessment of statistical predictions. Journal of the royal statistical society: Series B (Methodological). 1974;36(2):111-133.

104. Geisser S. A predictive approach to the random effect model. Biometrika. 1974;61(1):101-107.

105. Tenenhaus M, Vinzi VE, Chatelin Y-M, et al. PLS path modeling. Computational Statistics & Data Analysis. 2005;48(1):159-205.

106. Henseler J, Dijkstra TK, Sarstedt M, et al. Common Beliefs and Reality About PLS: Comments on Ronkko and Evermann (2013). Organizational Research Methods. 2014;17(2):182-209.

107. Chin WW. Bootstrap Cross-Validation Indices for PLS Path Model Assessment. In Esposito Vinzi V, Chin WW, Henseler J, et al., (Eds). Handbook of Partial Least Squares: Concepts, Methods and Applications. Berlin, Heidelberg: Springer Berlin Heidelberg 2010:83-97.

108. Wetzels M, Odekerken-Schroder G, Van Oppen C. Using PLS path modeling for assessing hierarchical construct models: Guidelines and empirical illustration. MIS quarterly. 2009:177-195.

109. Andrew L. Comrey HBL. A First Course in Factor Analysis (2nd Edition). New York: Lawrence Erlbaum Associates 1992.

110. Afthanorhan WMABW, Ahmad S, Mamat I. Pooled Confirmatory Factor Analysis (PCFA) Using Structural Equation Modeling on Volunteerism Program: A Step by Step Approach. International Journal of Asian Social Science. 2014;4(5):642-653.

111. Awang Z, Afthanorhan A, Mohamad M, et al. An evaluation of measurement model for medical tourism research: the confirmatory factor analysis approach. International Journal of Tourism Policy. 2015;6(1):29-45.

112. Tojib DR, Sugianto L-F. Content validity of instruments in IS research. Journal of Information Technology Theory and Application (JITTA). 2006;8(3):5.

113. Mondal A, Schneider K, Roy B, et al. A Survey of Software Architectural Change Detection and Categorization Techniques. Journal of Systems and Software. 2022.

114. Le D, Medvidovic N. Architectural-based speculative analysis to predict bugs in a software system. Proceedings of the 38th International Conference on Software Engineering Companion. Austin, Texas: Association for Computing Machinery 2016:807–810.

115. Carvalho LPdS, Novais R, Mendonca M. Investigating the Relationship between Code Smell Agglomerations and Architectural Concerns: Similarities and Dissimilarities from Distributed, Service-Oriented, and Mobile Systems. Proceedings of the VII Brazilian Symposium on Software Components, Architectures, and Reuse. Sao Carlos, Brazil: Association for Computing Machinery 2018:3–12.

116. Shaikh M, Jalbani D, Ansari A, et al. Evaluating Dependency based Package-level Metrics for Multi-objective Maintenance Tasks. International Journal of Advanced Computer Science and Applications. 2017;8(10).

117. Mo R, Cai Y, Kazman R, et al. Decoupling Level: A New Metric for Architectural Maintenance Complexity. 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). Austin, TX, USA: IEEE 2016:499-510.

118. Rizzi L, Fontana FA, Roveda R. Support for architectural smell refactoring. Proceedings of the 2nd International Workshop on Refactoring. Montpellier, France: Association for Computing Machinery 2018:7–10.

119. Fontana FA, Pigazzini I, Roveda R, et al. Automatic Detection of Instability Architectural Smells. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). Raleigh, NC, USA: IEEE 2016:433-437.

120. Das D, Maruf AA, Islam R, et al. Technical debt resulting from architectural degradation and code smells: a systematic mapping study. 2022;21(4 %J SIGAPP Appl. Comput. Rev.):20–36.

121. Wohlin C, Runeson P, Hst M, et al. Experimentation in Software Engineering. Springer Berlin, Heidelberg: Springer Publishing Company, Incorporated 2012.

122. Malhotra R. Empirical research in software engineering: concepts, analysis, and applications: Chapman and Hall/CRC 2016.

123. Lei P-W, Wu Q. Introduction to Structural Equation Modeling: Issues and Practical Considerations. Educational Measurement: Issues and Practice. 2007;26(3):33-43.

124. de Oliveira Barros M, Dias-Neto AC. Threats to validity in search-based software engineering empirical studies. Technical Report TR 0006/2011, UNIRIO—Universidade Federal do Estado do 2011.