Architectural Tactics in Software Architecture: A Systematic Mapping Study

Gastón Márquez[a,*], Hernán Astudillo[b], Rick Kazman[c]

[a] *Department of Electronics and Informatics, Universidad Técnica Federico Santa María, Concepción, Chile.*
[b] *Department of Informatics, Universidad Técnica Federico Santa María, Santiago, Chile.*
[c] *Department of Information Technology Management, University of Hawaii, Honolulu, Hawaii, USA.*

Abstract

Architectural tactics are a key abstraction of software architecture, and support the systematic design and analysis of software architectures to satisfy quality attributes. Since originally proposed in 2003, architectural tactics have been extended and adapted to address additional quality attributes and newer kinds of systems, making quite hard for researchers and practitioners to master this growing body of specialized knowledge. This paper presents the design, execution and results of a systematic mapping study of architectural tactics in software architecture literature. The study found 552 studies in well-known digital libraries, of which 79 were selected and 12 more were added with snowballing, giving a total of 91 primary studies. Key findings are: (i) little rigor has been used to characterize and define architectural tactics; (ii) most architectural tactics proposed in the literature do not conform to the original definition; and (iii) there is little industrial evidence about the use of architectural tactics. This study organizes and summarizes the scientific literature to date about architectural tactics, identifies research opportunities, and argues for the need of more systematic definition and description of tactics.

Keywords: Architectural tactics, systematic mapping study, software architecture, quality attributes

1. Introduction

Software architecture is the discipline that structures every phase of a software project, serving as the blueprint and defining the tasks that must be performed by design and implementation teams [1] [2]. A key in designing software architectures is the satisfaction of quality attribute requirements (QAs) [2]. QAs impact crucial aspects of the system, like run-time behavior, robustness, security, and user experience. Although the various QA communities have developed their own vocabularies, scenarios have become an accepted method to specify QAs [3].

*Corresponding author
Email address: gaston.marquez@usm.cl (Gastón Márquez)

One strategy proposed to represent the fundamental design decisions for achieving QAs are *architectural tactics*. As defined in [4], [5], [1] and [2], architectural tactics are the key design decisions that influence the control of a quality attribute. Architectural tactics influence the system's response to a specific stimulus that is important to the achievement of a QA. Architectural tactics have arisen from the collected experience of architects over the decades. As such, they are a foundation of knowledge, providing a systematic set of architectural design decisions.

The importance of architectural tactics lies in the fact that they are primitive solutions that sustain architectural patterns obtained in software architecture. In this regard, most architectural patterns consist of (are constructed from) several different architectural tactics. Harrison et al. [6] address the relationship between architectural patterns and tactics in more detail. They propose a model that shows how architectural patterns, quality attributes, and tactics relate to each other and how they relate to the overall architecture. Additionally, the model provides the instance for discussing the detailed ways in which implementations of tactics affect the architectural patterns used. The model also describes an architectural pattern as a solution to an architectural problem, often described as a set of architectural concerns. The architectural pattern satisfies multiple architectural concerns but can also have side effects on other architectural concerns. An architectural pattern can have different variants, and the variant used is often based on the tactics employed. For this reason, architectural tactics play a fundamental role because *patterns package tactics* [2].

Interest in architectural tactics has grown significantly over the years in the software architecture community. In this regard, the Software Engineering Institute (SEI) has been relevant when it comes to the investigation of architectural tactics. Initially discussed in technical reports [4], architectural tactics were introduced to support understanding the relationship between quality attribute requirements and architecture design. SEI introduced the concept of the general scenario as a precise and independent mechanism for specifying quality attribute requirements [7]. In this context, the idea of architectural tactics emerged, whose initial objective was to represent the characterization of architecture decisions that are used to achieve a quality attribute. Additionally, architecture tactics were used to satisfy response measures based on quality attributes concerning quality models.

On the other hand, the interest in architectural tactics has also yielded that the original definition of architectural tactics has had to evolve in order to address design decisions in other emerging domains such as cloud [8], cyber-foraging [9] and cyber-security [10]. Nevertheless, this evolution of architectural tactics has implied that information about how they are identified, the mechanisms for describing them and the data sources used to recognize them is increasingly unknown. This causes a lack of explicit knowledge about the fundamentals of architectural tactics in these emerging domains, which limits the systematic replication of the characterization of new architectural tactics to address design decisions in modern systems. The definition and description of architectural tactics taxonomies presented in [2] address seven quality attributes; however, there is little research on which quality attributes have been addressed by architectural tactics research. This implies that it is unclear whether the architectural tactics community has maintained an interest in these seven quality attributes or has addressed others. Moreover, there is not enough research that systematically compiles updates or new proposals for architectural tactics taxonomies.

For this reason, this paper defines a systematic mapping study (SMS) to gather primary studies to describe and illustrate the body of knowledge generated by architectural tactics. We have investigated several perspectives on architectural tactics: quality attributes, techniques and methods for defining them, data sources for recognizing them, criteria for characterizing them, and new or updated architectural tactics taxonomies. Our main contribution is the systematic study showing the state of the art regarding architectural tactics in software architecture research.

This remainder of this paper is structured as follows: Section 2 describes the background; Section 3 describes the systematic mapping design to conduct our study; Section 4 details the study results; Section 5 discusses key findings; Section 6 addresses the threats to validity; Section 7 discusses related studies; and Section 8 summarizes and concludes. Additionally, we have created a repository [11] containing the Open Science material of our study, which corresponds to (i) the search protocol used in our study, (ii) the tables and figures of the article, and (iii) the metadata obtained from the primary studies.

2. Background

Initially introduced in [1] and refined in [2], an architectural tactic (henceforth just "tactic") is a design decision that influences the achievement of a specific quality attribute response. Quality attribute requirements specify system responses that, in turn, are critical to the achievement of the system's business or mission objectives. The initial research describes taxonomies of tactics on the following quality attributes: security, availability, performance, modifiability, interoperability, usability and testability [2].

The literature offers several definitions of tactics:

- An architectural tactic is a *design decision* that helps achieve a *specific quality-attribute-response*, and that is motivated by a quality-attribute analysis model [12].
- An architectural tactic is a *means of satisfying a quality-attribute-response measure* (such as average latency or mean time to failure) by manipulating some aspect of quality attribute model (such as performance queuing models or reliability Markov models) through *architectural design decisions* [4].
- Tactics identify and codify the *underlying primitives of patterns* to solve the problem of the intractable number of patterns existing [13].
- Tactics are *"architectural building blocks"* from which architecture patterns are created [1].
- An architectural tactic is a *design decision* that influences the *control of a quality attribute response*. Each architectural tactic is a *design option* for the architect [2].

On the one hand, definitions point to tactics as design decisions aimed at satisfying quality attributes. In this regard, stimuli or models can express these attributes. Since the design of a system is a collection of decisions, some of those decisions may help to control quality attribute responses and affect the response of a system to some stimulus. On the other hand, the definitions also consider tactics as part of patterns, i.e., atomic elements of a pattern's structure.

Often, the appropriate application of tactics depends on context, which is represented by a general scenario with six essential parts to contextualize quality attribute requirements [2] (see Table 1).

Table 1: General scenario description

Item	Description	Example
Source of stimulus	This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.	Internal AAL system software.
Stimulus	The stimulus is a condition that needs to be considered when it arrives at a system.	Data Receiver component crashes.
Environment	The stimulus occurs within certain conditions. For example, the system may be in a normal state, or in an overload condition, or in startup, or may be offline when the stimulus occurs.	Runtime and high request overhead.
Artifact	Some artifacts are stimulated. This may be the whole system or some pieces of it.	Data Receiver component and internal processes.
Response	The response is the activity undertaken after the arrival of the stimulus.	Component fully operational with no data loss.
Response measure	When the response occurs, it must be measurable so that the requirement can be tested.	Within 60 seconds.

To illustrate, Figure 1 shows the general scenario for availability. This describes the dimensions of availability-relevant requirements that must be considered in the design of an architecture. Other general scenarios have been described for the QAs of deployability, energy efficiency, integrability, modifiability, performance, safety, security, testability, and usability [2] [14]. Each of these QAs has its own set of tactics.

2.1. Illustrative example

Let us consider the following quality attribute requirement for an Ambient-Assisted Living (AAL) system: *The system must be 99.9999% available to alert family members and emergency units if an older adult falls inside the home.* Although this requirement is relevant multiple quality attributes (such as interoperability and performance), we address availability in this illustrative example. One can infer, from this requirement, that the medical devices that monitor the elderly and the system components associated with such monitoring should be fault-tolerant. Analyzing the risk surrounding this requirement in the AAL system's architecture, an architect might consider the Data Receiver component (the component that receives data from the medical devices) as one of the critical components. To better probe the design of this component the architect considers some relevant scenarios.

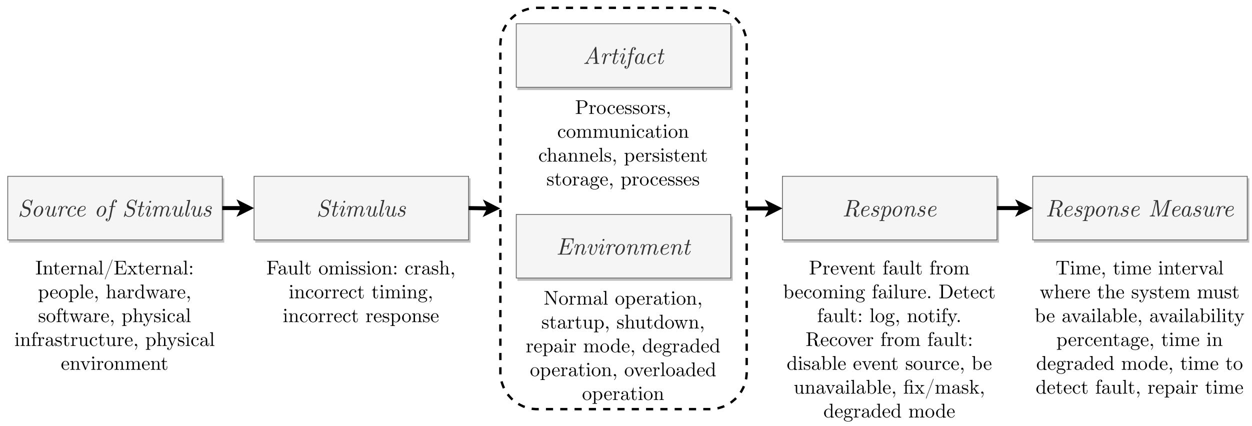

Figure 1: General scenario for availability described in [2]

One such scenario is related to when the Data Receiver component crashes (the *stimulus*). The taxonomy of availability tactics (see Figure 2) suggests that the architect should make three decisions with respect to how the system will *respond* to address this *fault*: (1) how to *detect* it, (2) how to *prevent* to it, or (3) how to *recover* from it.

Each category of the availability tactics taxonomy describes a set of tactics (complete detail of availability tactics can be consulted in [1] and [2]). Each of these tactics is a design option for the architect. The architect can use these to choose among and evaluate design alternatives to decide how to address the stimulus affecting the Data Receiver component. For this example, the architect decides on tactics to detect and recover from failures. Consequently, Table 1, in the "Example" column, describes the specific scenario for this illustrative example.

This scenario could be achieved by a number of tactics-based design choices. First, the fault (crash) needs to be detected. This detection could be achieved by having the Data Receiver component issue periodic *heartbeats* and *monitoring* this component. Additional design choices might be made to recover from a crash, such as using a *spare* (to replace the failed component) and *scaling restart* (to reintroduce a previously failed component back into service). Thus, the set of availability tactics provides a kind of vocabulary and a checklist for design and analysis of the scenario described in Table 1.

3. Systematic study process

This section describes the process conducted in this SMS (see Figure 3). Inspired by the systematic literature mapping process proposed by [15], our process mainly includes activities that focus on (i) search and selection of studies, (ii) data extraction and (iii) synthesis and classification of studies. Additionally, we include a snowballing process [16] to identify further studies.

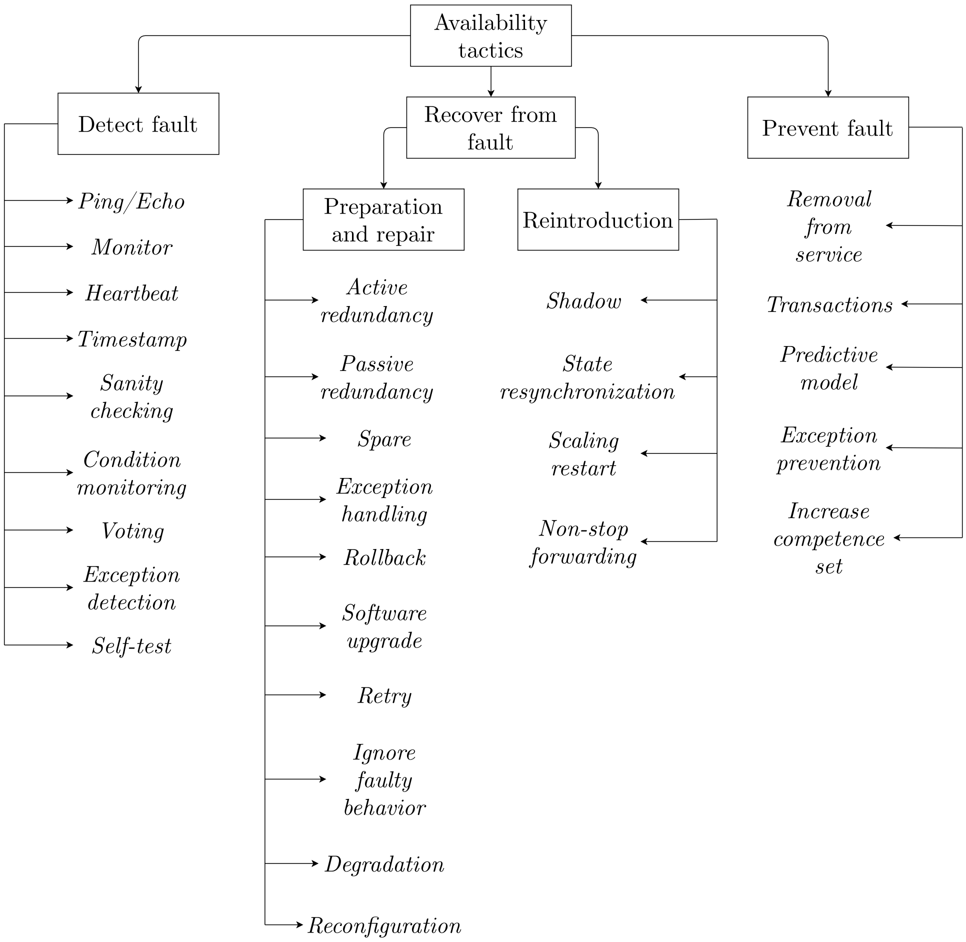

Figure 2: Availability tactics (taken from [2])

3.1. Research objective

The study objective is to identify, analyze, evaluate, and interpret the body of knowledge on architectural tactics. We focused on conducting a comprehensive review of academic studies to characterize the contributions of tactics in the discipline of software architecture.

3.2. Research questions

To illustrate the contribution of tactics, the study addresses five research questions:

Research question 1 (RQ1)
Which quality attributes have been addressed by tactics research?

Objective: This research question aims to explore the quality attributes that have been studied by tactics research studies and describe the role and purpose of tactics in the studies in order to illustrate their contribution.

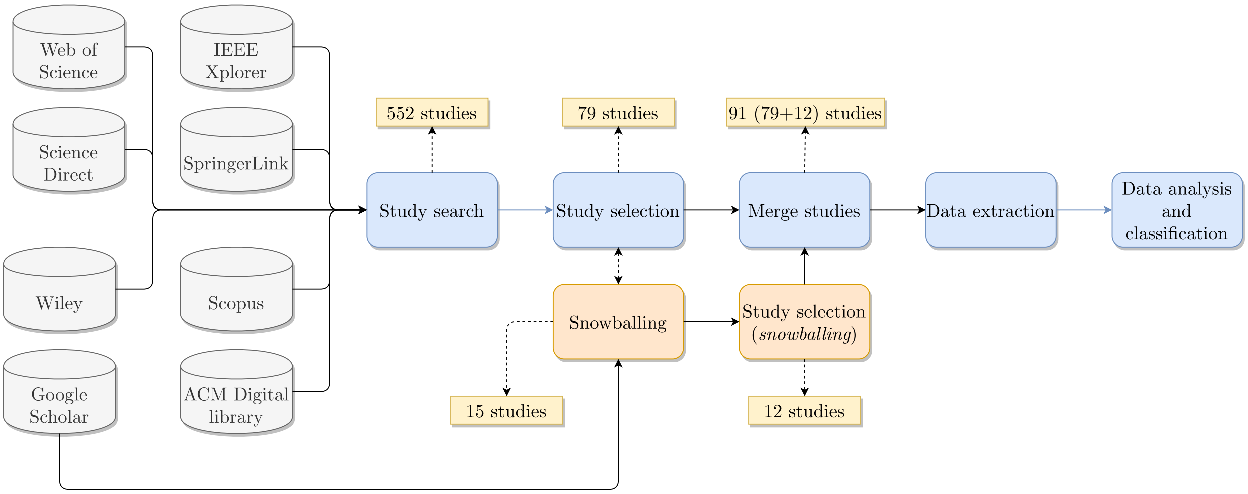

Figure 3: Design and execution summary of the systematic mapping study

Research question 2 (RQ2)

Which techniques have been proposed to identify tactics?

Objective: This research question intends to show and describe the main techniques primary studies use to identify tactics. Additionally, this research question aims to discuss why the studies use the identified techniques.

Research question 3 (RQ3)

Which kinds of data sources are used to recognize tactics?

Objective: The purpose of this research question is to identify and describe the most popular data sources used by studies to recognize tactics in order to analyze the motivation and rationale of studies to use the identified data sources.

Research question 4 (RQ4)

What mechanisms are used to describe tactics?

Objective: This research question aims to identify and describe the mechanisms used by primary studies to describe tactics, as well as the variables, rationale, or aspects used by researchers for the description.

Research question 5 (RQ5)
Which taxonomies of tactics have been proposed or updated?

Objective: Based on the taxonomies of tactics originally described in [1], the purpose of this research is to detail which taxonomies have been updated or proposed in research studies, as well as the corresponding motivation for this.

3.3. Study search

We defined the search string using the P.I.C.O. (Population, Intervention, Comparison, Outcomes) approach proposed by Kitchenham *et al.* [17]. As we are conducting an SMS, we focused only on Population and Intervention, as suggested by Petersen *et al.* [15] [18].

- *Population*: Studies related to software architecture.
- *Intervention*: Architectural tactics.

For each strategy, we defined keywords and defined the search string, which is ("software" OR "architecture") AND "architectural" AND "tactic*". Having defined the search string, we searched for primary studies in electronic databases (see Table 2) and we focused on the title and keywords of each paper. For each database consulted, the search string was adapted using the specific standards of each database. For each study, we reviewed the title and abstract in order to understand the proposal of each paper. The review period began in September 2020 and ended in August 2021. Finally, in this step, we collected 552 papers.

Table 2: Databases consulted

Name	URL
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
SpringerLink	https://link.springer.com
Scopus	https://www.scopus.com
ACM Library	https://dl.acm.org
Web of Science	http://login.webofknowledge.com
ScienceDirect	https://www.sciencedirect.com
Wiley	https://onlinelibrary.wiley.com

Once the first set of primary studies was obtained, we stored the papers using Reference Management Software[1] to support the search for primary studies.

[1]For this SMS, we used Mendeley (`https://www.mendeley.com`)

3.4. Study selection

To select the articles, we executed the following filters:

- *First filter*: We scrutinized for the keywords of the search string in each article abstract. If the keywords were not found in the abstract, the article was omitted. In this filter, we also removed duplicate articles. After applying the first filter, we obtained 223 papers.

- *Second filter*: We applied the inclusion and exclusion criteria. Any irrelevant articles were omitted at this stage. We defined the following inclusion and exclusion criteria:

  - Inclusion criteria
    - The study should focus on tactics in software architecture research.
    - The study should be subject to peer review.
    - The study must be written in English.
  - Exclusion criteria
    - Secondary studies
    - Studies that used "tactics" as technological innovation strategies, rules, algorithms and marketing strategy.
    - Grey literature.
    - Short studies ($<$ 4 pages).
    - Studies in poster format, tutorials, editorials, etc.

  One author executed the inclusion and exclusion criteria. Subsequently, two authors reviewed the final set of primary studies to mitigate any potential bias, and hence a threat to validity. After applying the second filter, we obtained 109 papers.

- *Third filter*: From the remaining set of articles we read the abstract again, the introduction and conclusion. In this step, we omitted those articles whose abstract does not appropriately represent what was described in the introduction and conclusion.

After applying all the filters, we obtained 79 papers.

3.5. Snowballing process

To explore more primary studies, we executed a snowballing method [16]. This method is a non-probabilistic (non-random) sampling used when the information in the samples (primary studies) is difficult to find. The main characteristic of snowballing is the use of initial primary studies to generate additional studies. For this SMS, we used this method in order to increase the search scope for primary studies. We performed backward and forward snowballing procedures (i.e. references and citations), using Google Scholar[2]. This step yield twelve additional papers to the study.

[2] https://scholar.google.com

3.6. Data extraction

To extract the primary studies data, we created a template to organize the information (see Table 3). One author conducted the data extraction, and a second author verified the extracted information.

Table 3: Data extraction template

Item	Data item	Description	RQ
I1	ID	Unique study identifier	Demographics
I2	Authors	Name of the authors.	
I3	Title	Study title.	
I4	Venue	Name of the journal, conference, workshop, symposium or book where the primary study was published.	
I5	Type venue	Categorization of the venue: journal, conference, workshop, symposium or book chapter.	
I6	Year	Publication year.	
I7	Research type	Classification of studies by research type (see Section 3.7 for more details).	
I8	Study contribution	Classification of studies by contribution type (see Section 3.7 for more details).	
I9	Quality attributes	Description of the quality attributes addressed by the primary studies.	RQ1
I10	Technique/method	Detail of the techniques and methods used to identify architectural tactics.	RQ2
I11	Data source	Description of the data source used by the primary studies to recognize tactics.	RQ3
I12	Criteria	Identification of criteria to characterize tactics.	RQ4
I13	Taxonomy	Identification of new or updated tactics taxonomies.	RQ5

3.7. Data analysis and classification

The information extracted from the primary studies was grouped, extracted and tabulated in the template described in Table 3. Items I1 through I8 correspond to the demographic data from the primary studies. Item I7 categorizes the studies based on the research type. For this classification, we used the proposal of Wieringa et al. [19], which classifies studies based on the following categories:

- *Evaluation research*: This type of study deals with investigating a practical problem or the implementation of a technique in practice.
- *Proposal of solution*: These studies propose a solution or technique and argue about its relevance, without exhaustive validation.
- *Validation research*: These studies investigate the properties of a proposed solution that has not yet been implemented in practice.

- *Philosophical papers*: This type of study outlines a new way of looking at things, a new conceptual framework, etc.

- *Opinion papers*: These studies emphasize the authors' opinion about what is right or wrong about something, how something should be done, etc.

- *Personal experience papers*: These studies emphasize the *what* rather than the *why*; may relate to one or more projects, but part of the author's personal experience.

According to [15],the classification proposed by Wieringa et al. describes the research facet of primary studies, reflecting the research approach used in the papers. Furthermore, this classification is easy to interpret and use without evaluating each paper in particular.

Regarding I8, we followed Kuhrmann et al. [20] (inspired by Shaw's classification of research results [21]) to classify the contributions of primary studies as follow:

- *Model*: Representation of a reality observed by concepts related to tactics.

- *Theory*: Construct of cause-effect relationships.

- *Framework*: Framework or method related to tactics.

- *Guidelines*: List of suggestions regarding the use of tactics.

- *Lessons learned*: Set of outputs from results obtained in empirical studies related to tactics.

- *Advice*: Recommendations about the use or experiences of tactics

- *Tool*: A tool that uses tactics for different purposes

For items I9 through I12, we classified the primary study data by identifying the primary studies' research themes. These themes were identified through the guidelines proposed by Braun et al. [22] to conduct thematic analysis in documents. Thematic analysis corresponds to a method of searching for repeated patterns of meanings over a data set (e.g., text). It is a method adaptable to the context and allows for collaborative discussion to identify themes. A *theme* captures something important about the data concerning the research topic (in our study, tactics). The steps executed to identify themes are as follows:

- *Search of themes*: This step aims to identify the main features that allow answering the research questions. The features are characterized in concrete sentences.

- *Theme review*: Two authors and one contributor review the identified topics. This review is focused on verifying whether the theme effectively characterizes an answer to a research question. More precisely, this step checks that the identified topic is unambiguous and precise, as far as possible.

- *Define and name the themes*: Once the themes are reviewed, this step names and defines the themes.

The theme identification method was used to tabulate I10, I11, and I12. Regarding I13, the identification of new or updated taxonomies is performed through the analysis of each selected study.

4. Results

This section describes the results of the SMS, detailing first the primary studies' demographics and then answering each research in each subsection.

4.1. Demographics

The study identified 91 primary studies, published in several venues and years. Three-fifths (57.1%) of primary studies appeared in conference proceedings (see Figure 4 and Table 4), with 2015 having the highest number (see Figure 5). One-fourth (27.5%) of primary studies have been published in journals, beginning in 2009 and remaining constant until 2021, minus 2011; the Journal of Software and Systems (JSS) leads with most 7 publications. The remainder of primary studies were published in workshops (6.6%), symposiums (7.7%), and book chapters (1.1%).

Table 4: Top 5 Conferences

Conference	Acronym	Publications
International Conference on Software Architecture	ICSA	6
European Conference on Software Architecture	ECSA	6
International Conference Series on the Quality of Software Architectures	QoSA	5
International Conference on Software Engineering and Knowledge Engineering	SEKE	4
International Conference on Software Engineering	ICSE	3

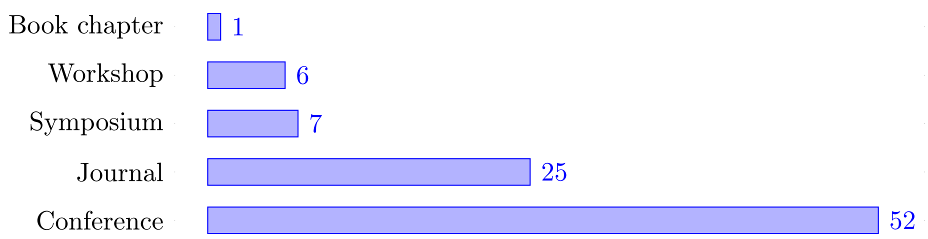

Figure 4: Distribution of publication type

From a historical point of view, we have covered a range of 16 years (we have excluded the years 2006 and 2007 as we did not find primary studies in those years) based on the years of publication (see Figure 5). Although the fundamentals of tactics were set in SEI technical reports between 2000 and 2002, the first external publication that introduced tactics dates to 2003, in the International Software Requirements to Architectures Workshop (STRAW). Since then, publications have remained relatively constant over the years, excepting 2006 and 2007. The years 2012, 2014, 2015 and 2019 have had the most significant numbers of publications. Overall, these publication trends demonstrate consistent interest in tactics research.

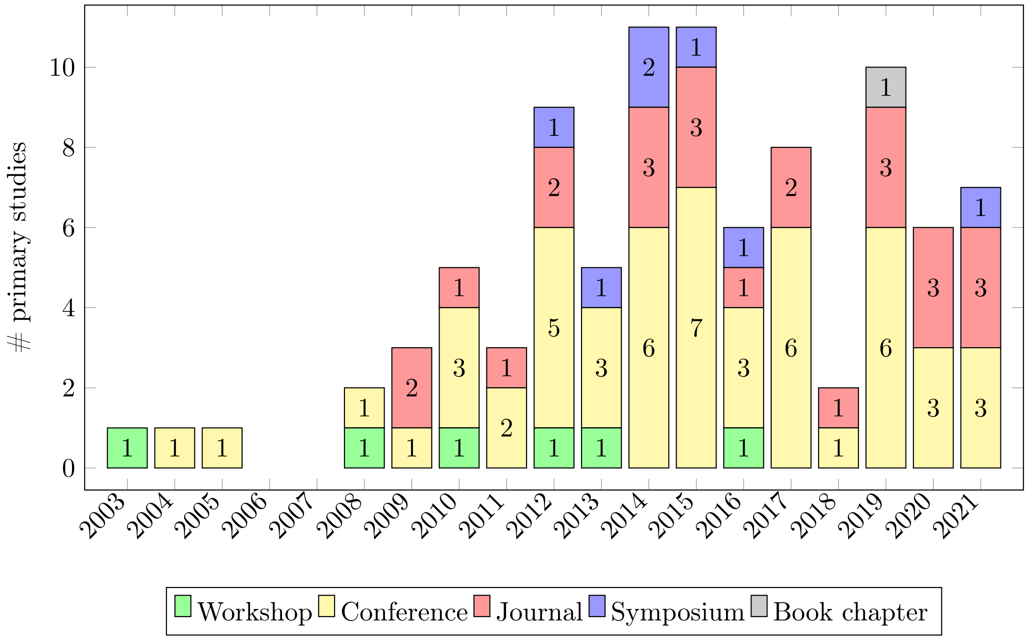

Figure 5: Number of papers per year and publication type

The first primary studies were published in workshops and conferences, but in 2009 started to appear in journals. This interest in publishing papers in journals has remained constant since then, with at least one publication per year on the topic.

Figure 6 shows the studies' distribution of *research type*:

- Almost half (47.3%) are *solution proposals*, which describe how to use tactics to address challenges in architectural design and decision making.

- Two-fifths (39.6%) are *evaluation research*, which focuses primarily on evaluating tactics-based techniques or methods in various research contexts.

- The reminder are *philosophical papers* (new proposals to structure and categorize tactics, and mainly in security) or *experience papers* (which probe the contribution of tactics for designing software architectures).

Figure 7 shows the studies' *contribution type*. Almost half (48.4%4) are *frameworks*, which include tactics in the set of techniques they use to address some specific engineering problem (for topics including secure architectures, traceability, satisfying requirements, and

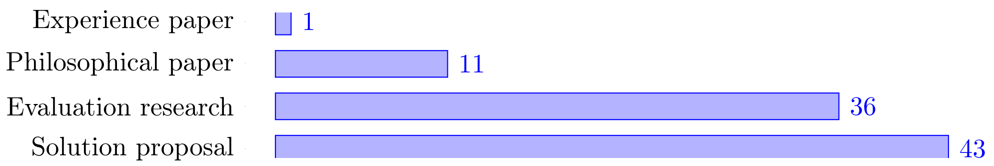

Figure 6: Distribution of research type

selecting software components). Two minor but important subgroups are *models* (16.5%), which allow to represent tactics knowledge by either refining previously described tactics taxonomies or by proposing new taxonomies in other disciplines (including cyber-foraging, safety, and data-intensive systems); and *guidelines* (15.4%), which describe key findings obtained in empirical studies. The remaining studies describe *lessons learned* (7.7%) of actual use of tactics to solve design problems; formalize tactics theories (6.6%) through various methods (such as Z specifications); contribute *tools* to identify tactics in source code; or give *advice* on the use of tactics to design software architectures.

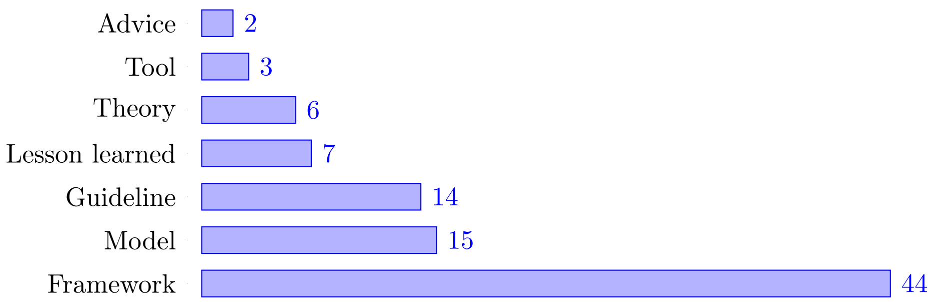

Figure 7: Distribution of contribution type

Figure 8 shows the crossing of *research type* versus *contribution type*. As seen above, almost half of studies are *solution proposals*; within this, the largest combination of the whole dataset (an overwhelming 33 studies of 91) are *solution proposals* that contribute *frameworks*; solution proposals of *models* are a much smaller seven studies. The two-fifths of studies doing *evaluation research* are distributed in all categories of research type, with *guidelines* (13 studies) being the largest subset (and the second largest kind of study in the whole dataset). The *philosophical papers* tend to contribute a *theory* (5 studies out of 11). And perhaps naturally, the only *experience paper* contributes *lessons learned*. We did not find studies with research type *opinion paper*.

Figure 9 describes the *types of validation* used in the primary studies. The most used empirical validation method (almost-half, 46.2%) are *case studies*. One-fourth (25.3%) do not specify their type of validation (if any); indeed, most of them focus on describing a proposal. One-sixth (16.5%) use experiments; most focus on validating techniques and algorithms to identify and characterize tactics in code, and others validate secure software architecture designs and selection of security design decisions. One-ninth (11%) do not attempt validation, but use illustrative examples to explain their ideas in predetermined

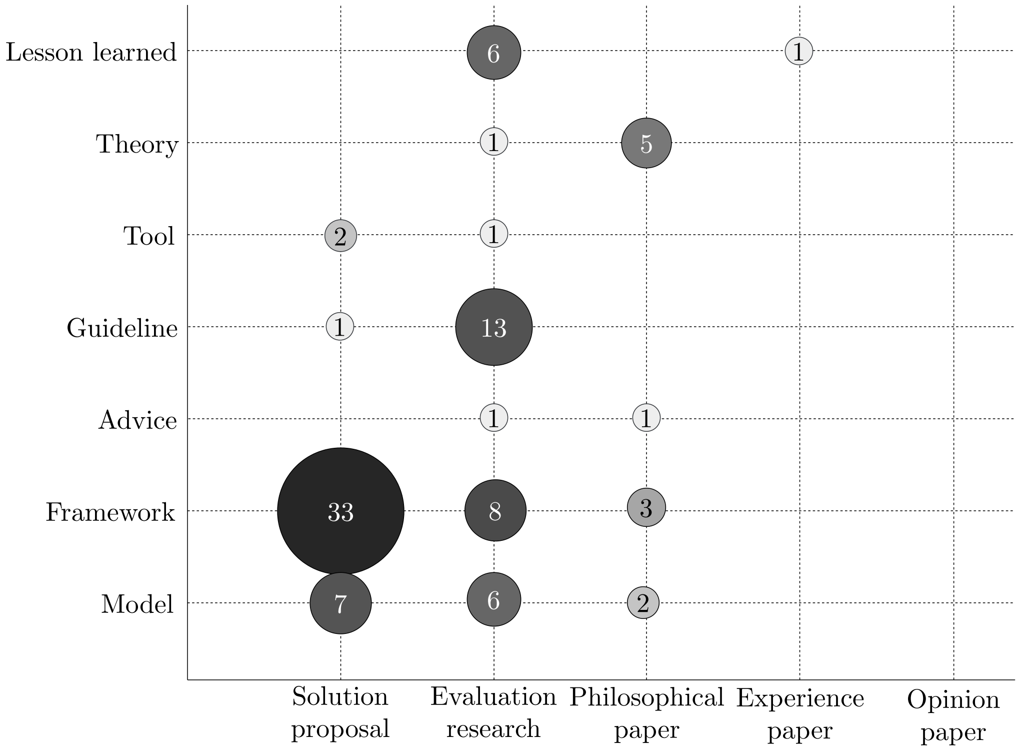

Figure 8: Map of research and contribution types

systems and environments. Finally, one study uses interviews to validate its proposals.

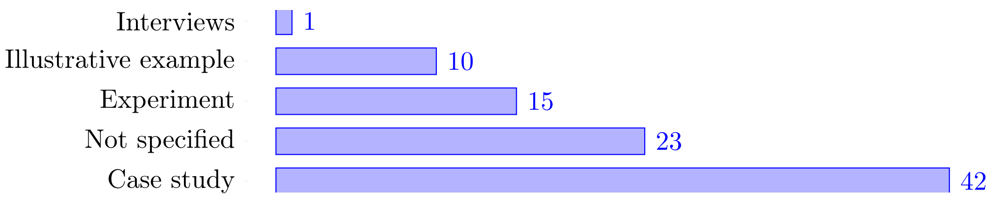

Figure 9: Distribution of empirical strategies used by primary studies

4.2. RQ1: Quality attributes

Table 5 describes the quality attributes that have been addressed by the primary studies with their corresponding description and distribution of studies. Some of these quality attributes (such as security, fault tolerance, availability, performance, adaptability, reliability and modifiability) may be found in the ISO/IEC 25010 quality model. The rest of the quality attributes have been recognized as quality attributes to evaluate software operational aspects (safety, deployability, and scalability) or as means to evaluate service reliability (dependability).

Figure 10 describes the purpose of using tactics for each quality attribute identified in the research question. This taxonomy summarises how primary studies use tactics to address

Table 5: Quality attributes addressed by tactics research

QA	Description	# of studies
Adaptability	Adaptability controls how easy it is to change the system if requirements have changed [23].	1
Dependability	Property of a system that delivers services at a specified reliability level and the system's ability to avoid failures that are serious and numerous [24].	1
Reliability	The degree to which a system, product or component performs specified functions under specified conditions for a specified period of time [25].	1
Modifiability	The degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality [25].	1
Interoperability	The ability of systems to share data and enable the exchange of information and knowledge between them [2].	1
Deployability	The time to get code into production after a commit [26].	2
Scalability	This quality attribute represents a system's ability to handle an increasing amount of work, or its potential to be expanded to accommodate growth [27].	3
Performance	Performance concerns itself with a software system's ability to meet timing requirements [2].	4
Safety	Attention to safety is required at each step of the software development process, identifying which functions are critical to the system's safe functioning and tracing those functions down into the modules that support them [28].	4
Availability	Characteristic of architectures that measures the degree to which system resources are available for use by end-users over a given time period [25].	4
Fault tolerance	This quality attribute is related to a system's ability to continue to function continuously in the event of faults [25].	5
Security	The degree to which a product or system protects information and data so that individuals or other systems have the appropriate degree of access to data according to their types and levels of authorization [25].	18

their research objectives. Figure 10 illustrates that, in general, tactics can be applied to support myriad research approaches. For instance, with regard to security, fault tolerance and availability, tactics are essential to define quality attribute requirements. Although the original definition of tactics aims at satisfying quality attributes, primary studies have made a significant effort to expand the boundaries of tactics to other research approaches.

In the following sections, we further discuss how primary studies use tactics for each quality attribute.

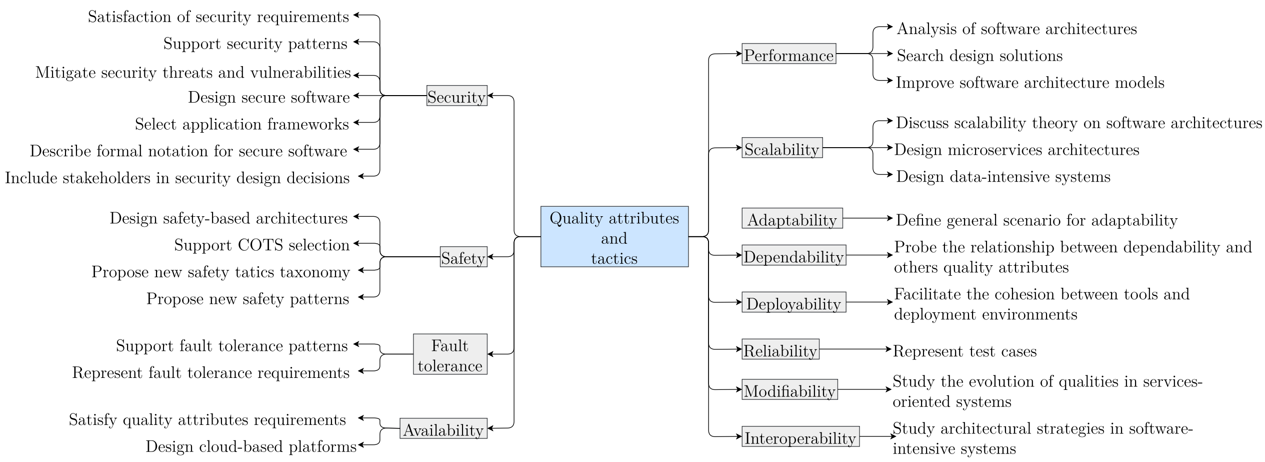

Figure 10: Taxonomy of purposes to achieve quality attributes using tactics

4.2.1. Security

The first contributions on security tactics are focused on the development of attribute-driven architectures (S6) and the satisfaction of security requirements (S8). These papers introduce empirical studies describing the importance of using security tactics to address aspects of architectural design as well as quality requirements analysis to satisfy stakeholder needs. S10 was the first to define a methodology to identify security tactics based on security patterns. This study is one of the first to discuss the intrinsic relationship between tactics and architecture patterns. Another study that follows the same research focus as S10 is S55. This study conducts an experiment regarding the mitigation of security threats using security tactics and patterns.

Primary studies have also used tactics to design secure software (S36, S41, S43), to investigate the contribution of security tactics in open source software projects (S52), to select application frameworks (S73) and to mitigate threats in cyber-physical systems (S74). These studies argue that security tactics' contribution is related to the analysis to address security issues in the design and evaluation of architectures. Furthermore, S63 and S78 expanded the boundaries of security tactics to understand and mitigate vulnerabilities in software. These studies combine vulnerability databases (such as [29]) with the security tactics specification to define techniques to detect, characterize and evaluate vulnerabilities in software.

Another interesting aspect of security is that some researchers (S12) have proposed a formal notation to describe security tactics. In this study, the authors attempt to represent security tactics specifications using formal languages to address security objectives on a more abstract level. More precisely, they focus on representing tactics related to authentication and authorization. Additionally, this paper is the first to contribute to the definition of the theoretical body of knowledge on security in software architectures.

From a business point of view, S67 proposes a collaborative approach to include stakeholders in security design decisions. In this study, the authors suggest transforming the descriptions of security tactics into cards that stakeholders can use to reach consensus in the security decision. Inspired by the Planning Poker technique [30], the authors' proposal suggests making security design decisions by considering all stakeholders' opinions involved in security decisions.

4.2.2. Safety

S2 uses the methods proposed by the SEI to define tactics for safety. The main focus of this paper is to explore methods for designing safe software architectures. S7 addressed safety tactics from a control system perspective; this paper investigated the use of safety tactics to support Commercial Off-The-Shelf (COTS) acquisition in systems composed of intelligent re-configurable hardware. S30 proposed a refinement to the safety tactics catalogue proposed by S2. This refinement used the IEC 61508 safety standard as inspiration to propose a new taxonomy. The authors manually identify architectural methods from the standard and mapped them to safety tactics. Later, the same authors proposed safety patterns in S77. These patterns were obtained based on the STRIDE (**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privilege) security method.

4.2.3. Fault-tolerance

S4 introduced this quality attribute into tactics research by studying architectural patterns and tactics for fault-tolerance. The authors analyzed fault-tolerance measures and their relationship to architectural patterns. For each pattern, the authors examined several fault-tolerance tactics and investigate how each tactic would be realized in each pattern. The authors also addressed which parts of the pattern structure would change to implement the tactic and how they would change.

S8 addressed fault-tolerance requirements by applying requirement analysis techniques, such as the NFR framework [31] and fault-tolerance tactics. The authors' goal in using the NFR framework was to represent tactics through soft-goals. In this way, trade-offs that may arise when assessing fault-tolerance concerns can be identified at an early stage. S11 addressed the impact of fault-tolerance tactics on architectural patterns. The authors study the usefulness of fault-tolerance tactics in using architectural patterns as a design mechanism. S42 proposed to model fault-tolerance tactics using aspect-oriented modeling. The purpose of using aspect-oriented modeling is to represent cross-cutting fault-tolerance concerns as reusable aspects with dependent attributes. In this way, the authors aimed to integrate dependability analysis in the early stages of software development.

4.2.4. Availability

S5 and S6 discussed how to approach NFRs in software architecture by representing availability tactics as feature models. In the case studies addressed by these primary studies, the authors modeled availability tactics using the Role-Based Metamodeling language (RBML). RBML defines the solution domain of a role-based design pattern at the metamodel level [32]. Furthermore, RBML supports the development of precise specifications that can be used to develop pattern tools. Along the same lines, S35 also used feature models for availability tactics in a study related to a cloud platform design. To expand the scope of availability tactics, S76 identified five availability tactics from an analysis of application framework documentation in the context of microservices. These availability tactics were: (i) providing fallbacks, (ii) preventing single dependencies, (iii) asynchronous messaging, (iv) set timeouts, and (v) self-preservation.

4.2.5. Performance

Like the studies addressing availability tactics, S5, S6 and S35 address the representation of performance tactics through feature models. This representation is claimed to aid in the design and analysis of software architectures. S15 uses performance tactics as a technique to way to search for design solutions. In this study, the authors propose PerOpteryx, an approach to improve software architecture modeling using meta-heuristics guided by tactics. The research problem addressed by the authors of this study is that most evaluation tools are only able to determine specific values of quality attributes for a given architectural model. Therefore, any improvement of the architectural model becomes a manual exercise for the architect. Due to the ample design space of non-trivial systems and the many degrees of freedom, improving the architecture is a tedious task. This implies that an isolated improvement of a single quality attribute may result in the degradation of other quality attributes, which is difficult for software architects to determine and quantify manually. Therefore, the use of tactics (in this case, performance tactics) helps the software architect contextualize the design space of architecture to reduce biases and errors in architecture improvement.

4.2.6. Scalability

S19 introduced the first set of tactics for scalability. In this study, the authors discussed scalability theory to propose a taxonomy of scalability tactics. S72 discussed the role of scalability in the context of microservice architectures. In this study, the authors suggested five scalability tactics for designing microservices-based systems, which are: (i) data store separation, (ii) build separation, (iii) container deployment, (iv) network location, and (v) balancing scale. S79 addresses the role of scalability in data-intensive systems. The authors reviewed the theory of scalability to propose a taxonomy of scalability tactics for data-intensive systems.

4.2.7. Adaptability

S46 addresses the contribution of tactics in the context of mobile development. This primary study defines a general scenario for adaptability and then proposes a set of adaptability tactics. The general scenario for adaptability is as follows:

- *Source*: Changes in the resource environment.
- *Stimuli*: Resource disappear; resource appear; resource changes quality of service.
- *Artefact*: System.
- *Environment*: Few resources; low quality of resources; plenty of resources; high quality of resources.

- *Response*: Processes stimuli; change resource dependencies, change level of service.
- *Response Measure*: Time with a degraded service level; quantification of degraded service level (throughput, latency, accuracy, etc.); time interval between degraded service level; amount of spatial areas with degraded service levels

Additionally, the authors validated the proposed tactics for adaptability in a case study with master's students. The adaptability tactics are as follows: (i) resource selection, (ii) resource prediction, (iii) increase resources, (iv) mask variability, (v) resource fusion, and (vi) domain modeling.

4.2.8. Dependability

S14 describes a study that uses tactics to probe the relationship between dependability and other quality attributes in embedded systems. The authors propose a set of tactics that address dependability concerns. The authors argue that the use of these tactics enables the investigation of the relationships between dependability and other quality attributes of embedded systems.

4.2.9. Deployability

S32 discusses the relationship between software architecture and agile methodologies. This study proposes three tactics for deployability to facilitate the cohesion between tools and deployment environments. These deployability tactics are as follows: (i) Parameterization, (ii) Self-monitoring, and (iii) Self-initiated version update. S39 explores a software architecture's contribution to achieving continuous delivery and deployment objectives.

4.2.10. Reliability

S33 gathers a group of tactics (such as voting, heartbeat, and state resynchronization) and defines them as reliability tactics. It then represents them through sequence diagrams to provide test cases. These test cases are oriented to test reliability and safety concerns.

4.2.11. Modifiability

S69 uses modifiability tactics to study the evolution of qualities in service-oriented systems. In this study, the authors classified 15 modifiability tactics into three categories, which are as follows: principles for both SOA and microservices, design patterns for SOA and microservices. The results obtained describe that SOA and microservices have several beneficial properties for modifiability. This implies that there is a wide variety of patterns for the concrete realizations of the tactics identified in the study.

4.2.12. Interoperability

S88 executed an online survey to investigate how architectural strategies to promote interoperability of software-intensive systems have been used in practice. The survey results reveal that tactics are the least used architecture strategies by practitioners. This is because practitioners do not have enough information to identify the impact of using tactics.

Key findings of RQ1
• The quality attribute that has yielded the most research is security. • For primary studies, tactics are not only design decisions to achieve a quality attribute; they are also used to address other software architecture topics such as requirements description, application framework selection, proposing/refining patterns, among others. • In terms of quality attributes, tactics have been used mainly to design architectures, support patterns and represent requirements.

4.3. RQ2: Identification of tactics

Table 6 shows that 65 of the primary studies (71% of the total) do not explicitly describe what techniques or methods were used to identify tactics. In these papers, the authors describe tactics, but there is inadequate clarity about the means used to identify or characterize the tactics.

Table 6: Techniques to identify tactics

Technique	Description	# of studies
Multifacetic	This definition concentrates on techniques described by the primary studies that do not resemble the other techniques identified, for example, analysis of architecture and design patterns, surveys, consensus analysis, etc.	3
Text analysis	This technique uses text processing techniques to identify, analyze and report tactics within empirically collected data.	4
Manual mapping	This technique's primary mechanism is to manually analyze project documentation.	10
Code analysis	This technique consists of the (systematic or semi-systematic) exploration of source code to extract information about software design in order to identify tactics.	10
Not described	There is insufficient information provided by the primary study describing the method used to identify tactics.	65

In the following sections, we detail the proposals of the primary studies to identify tactics.

4.3.1. Multifacetic

In S20, the authors propose a more rigorous and replicable method for creating and reviewing tactics; they identify three approaches to identifying tactics. The first is to derive new tactics from existing ones. The second is to decompose an existing architectural pattern into its constituent tactics. And the third is to extract tactics that have been misidentified as patterns. The authors use a variant of the Wideband Delphi approach to identify and review tactics. The approach consists of the following steps:

1. Coordinator presents each expert with a specification and an estimation form.

2. Coordinator calls for a group meeting in which the experts discuss estimation issues with the coordinator and each other.
3. Experts fill out forms anonymously.
4. Coordinator prepares and distributes a summary of the estimates.
5. Coordinator calls for a group meeting, specifically focusing on having the experts discuss points where their estimates vary widely.
6. Experts fill out forms, again anonymously, and steps 4. to 6. are iterated for as many rounds as appropriate.

S10 points out that the number of tactics discovered thus far is insufficient to cover all the important aspects of architectural decision-making. In turn, the authors mention that tactics could be created from scratch, but it would be more efficient and trustworthy if tactics could be extracted from proven sources.

One possible source is any pattern that consist of tactics. Therefore, the authors propose to examine architectural patterns to verify if they satisfy these conditions for identifying tactics: (i) atomicity, (ii) force limitation, (iii) problem-specificity, (iv) completeness, and (v) trade-offs between forces.

S2 is one of the pioneers in using surveys to identify tactics. The authors surveyed studies on software safety design used in research and practice. The selection of studies was restricted by considering their appropriateness for architectural design. However, one of the obstacles in obtaining tactics from these sources is the complicated relationship between safety techniques and tactics. For example, one safety technique can implement multiple tactics affecting multiple quality attributes; and a safety technique may group mechanisms that are unrelated to quality attributes. Therefore, the authors mention that using surveys in domains not directly related to software architecture design requires more exhaustive tactic elicitation.

4.3.2. Text analysis

Text classification offers another way of identifying tactics. S83, for example, proposes the use of language models such as Bidirectional Encoder Representations from Transformers (BERT) [33] to detect tactics in source code through multi-class classifications. BERT is based on the assumption that programmers tend to program similar functions similarly. S87, on the other hand, uses coding question and answer repositories (such as Stack Overflow) to apply text analysis and classification techniques (such as a semi-automatic dictionary-based mining approach) to extract tactics-related posts.

Thematic analysis supports the identification, organization, and analysis of patterns or themes to infer results that aid in the understanding and interpretation of the phenomenon under study [22]. S85 and S91 uses thematic analysis to identify tactics. The authors select this technique because architectural information can be highly dependent on the project's specific characteristics.

4.3.3. Manual mapping

Most primary studies identify tactics through rigorous analysis of different sources of information. Authors who use this technique to identify tactics generally rely on consensus to identify and evaluate tactics. For example, S19 explored research on scalability to identify tactics and architectural patterns. The authors review various sources related to scalability studies to identify techniques that can be mapped to tactics. S30 reviewed safety standards for detecting and mapping tactics. The authors used the IEC 61508 standard to identify tactics. S48 investigates the architectural security tactics proposed in [2] and compares them with information security theory to refine the set of security tactics.

Studies such as S38 and S45 explore systemic properties, such as Energy Efficiency and Cyber-Foraging, to define quality attribute scenarios and identify tactics. S86 conducts a systematic study based on academic studies in the Internet of Things (IoT) discipline. From the collected documentation, the authors make a significant effort to identify tactics for IoT-related quality attributes related to security, scalability and performance. Studies propose to expand the boundaries of tactics foundations from software to systems to explore other properties that characterize systems.

Some primary studies also use the experience gained in real-world projects to identify tactics. In S39, the authors use interviews and project documentation to identify deployability objectives, design decisions and deployability tactics. Also, in S50, the authors use experience from projects related to Big Data as a Service (BDaaS) to extract data and map it to DevOps tactics. In this same line, S72 and S76 combine pattern language descriptions for scalability and availability in microservices with open source project documentation to identify tactics.

4.3.4. Code analysis

A project's source code provides insights that can be used in different ways to identify tactics. Primary studies using this option have explored various strategies to discover, characterize and/or recover tactics, which are: (i) use of topics models, (ii) exhaustive code analysis, (iii) development of custom tools, (iv) predictive models and (v) machine learning techniques. In the following sections, we discuss each of these strategies.

Topic models. Topic modeling has gained prominence over the past decade as a technique to analyze and summarize large corpus of textual data. S62 proposes a multifaceted approach to use latent topics to predict the use of tactics in code. This approach involves two phases: Training & Experimentation (Phase I) and Application (Phase II). In Phase I, the approach reviews open-source repositories to create a code crawler to classify and analyze topics. Phase II uses all the information obtained from Phase I to create predictive models and identify tactics in source code.

Exhaustive code analysis. Comprehensive code analysis is another technique for identifying tactics. S52 addresses the situation where an architect claims the use of a secure design by employing some tactic, but the source code does not support the claim. To do this, the authors explore an architect's intention to use security tactics. The authors then attempt to retrieve evidence of efforts to implement the design in the source code through the use of analysis tools to identify keywords that characterize tactics in source code.

Development of custom tools. S31 presents Archie, an Eclipse plug-in to maintain architectural qualities in design and code despite long-term maintenance and evolution activities. Archie detects tactics in a project's source code, builds traceability links between tactics and code, and then uses these links to monitor the environment for significant architectural changes and to keep developers informed of underlying design decisions and their associated justifications.

Predictive models. In S22 the authors describe a process for mining tactics and design patterns to build decision trees and tactics reference models. The approach incorporates a catalog of user customizable pattern definitions, supports pattern variations and alternatives, and applies multiple search technologies in order to execute a custom matching process. Additionally, this process uses the tactic detection algorithm [34] [35] to identify tactics in the source code. On the other hand, S26 proposes predictive models that capture the relationships between thematic domains and the use of specific tactics. Based on an extensive analysis of over 1000 open-source systems, the authors identify significant correlations between domain issues and tactics and use that information to build a predictor to generate recommendations related to tactics. It is important to mention that their model uses topic modeling techniques, such as Latent Dirichlet Allocation (LDA) [36].

Machine learning techniques. The use of machine learning techniques provides a novel perspective to identify tactics. These techniques focus on training a neural network to "educate" the machine to perform a task requiring intelligence [37]. S24 and S28 introduce the use of Machine Learning to identify tactics. These studies address traceability approaches, such as tactic Traceability Pattern (tTP) to build models for recognizing tactics. S24 and S28 serve as a foundation for the approach described in S58, where the authors use diverse machine learning techniques, such as support vector machine, classification by decision tree, Bayesian Logistic Regressions, AdaBoost, SLIPPER and Bagging, to classify and train code segments to produce a set of indicator terms that are considered representative of each tactic type. S63 uses the same techniques as S58 to identify vulnerabilities in security tactics code. The authors reported that 30% of the vulnerabilities found in code correspond to security tactics code. Finally, S66 makes tactic identification operational through the ArchEngine tool (ARCHitecture search ENGINE). This tool automates all the approaches described in S24, S28 and S58.

Key findings of RQ2

- The analysis of project documentation is the most common technique to identify tactics.
- Machine Learning techniques such as Decision Tree, Support Vector Machine, and AdaBoost emerge as an alternative to identify and classify tactics in source code.
- Natural Language Processing techniques such as Latent Dirichlet Allocation are used to discover tactic-related topics in source code.

4.4. RQ3: Kinds of data sources

4.4.1. Overview

We identified 7 kinds of data sources in primary studies (see Table 7). 63 studies (approx. 69% of the total) do not explicitly describe which kind of sources they used to identify tactics. This situation occurs because these studies use predefined tactics (primarily those published in [2]) for other research purposes. In the following sections, we further discuss the data sources identified.

Table 7: Data sources identified in primary studies

Data sources	Description	# of studies
Standards	This data source is related to the definition of quality attribute standards (e.g., ISO 25000).	1
Experts	Source of knowledge generated by experts' practical and industrial experience in software development.	1
Patterns	Source based on descriptions and implementations of design/architectural patterns.	1
Design decisions	This source is based on the architectural knowledge created during the development of software architectures.	3
Web repository	This source is related to web communities where practitioners interact through questions and answers about software architecture issues.	3
Documentation	Sources related to documentation of projects and technologies.	7
Source code	This data source corresponds to the relationships among classes, implementation of specific methods, invocation of packages, and application frameworks realized in code.	12
Not described	There is insufficient information provided by the primary study describing the data source used to identify tactics.	63

4.4.2. Standards

Another source used to recognize tactics is the description of standards. S30 uses the description of the IEC 61508 standard as a source. This standard is related to the functional safety of electrical, electronic and programmable electronic equipment. It is a publication of the IEC[3]. Its main objective is to help individual industries to develop supplementary standards, specifically designed for industries that employ the original IEC 61508 standard.

4.4.3. Experts

There is no doubt that experience is one of the most important sources for identifying tactics. The fundamentals of tactics can be extracted based on interviews with experts. From the data obtained in these interviews, it was possible to propose the first taxonomies of tactics [4]. Using the same methodology, S20 used experts as a source for recognizing and comparing tactics. In that study, the authors proposed an approach for rigorously extracting tactics, accompanied by expert opinion.

4.4.4. Patterns

According to [2], patterns are a set of design decisions that solve a recurring problem. Therefore, it seems natural that the pattern specifications contain tactics and hence can be used to identify tactics. S10 used security patterns to extract security tactics. The procedure described by the authors is based on security pattern analysis to identify potential architectural tactics.

4.4.5. Design decisions

S39 used design decisions as a source to recognize tactics. The authors identified design decisions through interviews conducted with project teams. From the design decisions, the authors could then identify tactics. For example, consider the following design decision: *Project A built an integrated test framework to allow the team to simulate the performance of the system under varying conditions. They used the framework to batch transactions and monitor the performance to see if it falls below an established threshold. The integrated test framework supports testing of distributed message communication (e.g., message queues and backend processes).* This integrated test framework is seen as an instance or a variation of the Testability tactic: Specialized access routines/interfaces.

Both S2 and S38 also use design decision principles as a source for recognizing tactics in Safety and Energy Efficiency.

4.4.6. Web repository

Web coding repositories communities (such as Stack Overflow and Github) are essential sources for extracting tactics. The interaction between practitioners regarding issues of code, projects, application frameworks, and technologies allows for a broad collection of knowledge. Since part of this knowledge can be represented by tactics, S85, S87 and S91 use Stack Overflow as a source of information to explore posts about tactics.

[3] https://www.61508.org/knowledge/what-is-iec-61508.php

4.4.7. Documentation

S72 and S76 used application framework documentation to extract tactics. Application frameworks are reusable software elements that provide generic functionality focused on solving recurrent issues [38]. Since frameworks are often based on architectural patterns and tactics, the authors of these studies used open-source framework documentation to recognize tactics.

S19, S45, S48 and S50 use other types of documentation to recognize tactics. S45 and S19 use research literature to recognize tactics. S48 uses architectural security tactics descriptions and information security theory to update architectural security tactics. Finally, S50 uses project documentation as a source for recognizing and characterizing tactics.

Another source for recognizing tactics are agile software development structures. In S32, the authors make a complete study on the importance of architecture in agile projects. The authors study alignments among agile software development structures to derive tactics. The structures are as follows:

- The *Architecture* of the system under design, development, or refinement, what we have called the traditional system or software architecture.
- The *Structure* of the organization: teams, partners, subcontractors, and others.
- The *Production Infrastructure* used to develop and deploy the system, the last activity being especially important in contexts where the development and operations are combined and the system is deployed continuously

4.4.8. Source code

Source code is the most common data source for identifying tactics. Twelve primary studies (S22, S24, S26, S28, S31, S52, S58, S62, S63, S66, S83 and S90) use project source files, logs, configuration files, packages, instances, and other artifacts to apply different analytical techniques to recognize tactics.

Key findings of RQ3

- In general, studies do not bother to detail which data sources they use to recognize tactics.
- Project documentation and source code are the popular data sources for recognizing tactics.
- Recent studies are positioning repositories and web communities (such as Github and Stack Overflow) as favorites for exploring tactics.

4.5. RQ4: Mechanisms to describe tactics

4.5.1. Overview

We recognized 4 mechanisms to describe tactics (see Table 8). Almost half of studies do not identify a mechanism to describe tactics; indeed, they mention and use tactics for other purposes. We discuss the mechanisms in detail in the following sections.

Table 8: Mechanism to describe tactics identified in primary studies

Mechanism	Description	# of studies
Formal language	Languages with formal syntaxis and semantics.	1
Description	Unstructured narrative.	9
Specific template	Usually some fields from the general scenarios described in [2], complemented with some additional fields.	12
Model	Several kinds of model representation (such as UML, feature models, and others).	15
Not described	There is insufficient information provided by the primary study describing the data source used to describe tactics.	54

4.5.2. Formal language

S12 used a Z specification [39] to characterize tactics. This notation's objective is to formally describing the main characteristics of computer systems. The notation uses mathematical data types to model the data of a system.

4.5.3. Description

9 primary studies (S7, S19, S45, S46, S48, S50, S72, S75, and S79) characterized tactics through narrative descriptions. Some studies focus their descriptions on design decision properties; for example, S79 describes scalability tactics based on scalability properties such as scale-out, scale-in, and resource virtualization.

4.5.4. Specific template

12 primary studies (S1, S2, S23, S25, S30, S38, S39, S59, S76, S85, S86, and S91) are inspired by the descriptions of general scenarios [2] (see Table 1, already described in Section 2) to propose their own templates to characterize architecture. Unlike S76, which uses a template inspired by the general scenario description to define, characterize and describe architectural availability tactics, the rest of the studies propose different types of templates that are extended to characterize tactics. The main objective of these templates is not to describe tactics explicitly, but rather to describe scenarios to contextualize another type of architectural analysis.

4.5.5. Model

Another way of representing tactics is through models. These models have several objectives, but all aim to represent architectural design decisions.

S5, S6, S35, S40, S43 and S51 characterize tactics through feature models. These types of models are used to define software product lines and system families. Features are used to identify and organize the commonalities and variabilities within a domain and model functional and quality properties [40].

S27, S33 and S42 use the UML standard to represent tactics; these studies characterize tactics using Class Diagrams and Sequence Diagrams.

S24 characterizes tactics through models that allow visualization of the traceability between quality concerns, tactics and code. S31 operationalizes the S24 proposal and represents the quality characteristics in design and code with a tool called Archie.

Finally, S8, S9, S16 and S37 characterize tactics through conceptual models.

Key findings of RQ4

- A significant group of primary studies prefers abstract representation such as models to describe tactics.
- The general scenario template for describing quality attribute requirements can be adapted to describe tactics.
- There is interest for studies to propose their own definitions for tactics.

4.6. RQ5: Tactics taxonomies

We have identified 10 taxonomies of tactics that update the taxonomies initially proposed in [1] or propose new taxonomies for the discipline (see Appendix A). The quality attributes addressed by these taxonomies are security, safety, fault-tolerance, scalability, deployability and modifiability. Regarding security and modifiability, the primary studies have proposed modifications to the taxonomies described in [1] [2]. In general, the main reason for presenting updates is to extend the tactics to other areas of security and modifiability that the initial taxonomies did not address. On the other hand, in relation to safety, fault-tolerance, scalability and deployability, the proposed taxonomies correspond to new contributions to the discipline.

Regarding security, S20, S25, S44, S48 have proposed different kinds of refinement of the security tactics taxonomy to complement or address other security aspects such as intrusion identification, security information management, and steganography, among others. Security tactics research has inspired some researchers to propose a new look at the security tactics taxonomy originally proposed in [1]. These researchers proposed a new security taxonomy because the original taxonomy was created through informal means; they employed techniques such as Wideband Delphi to create a more methodological approach to the identification of tactics. Other researchers argue that the original taxonomy of security tactics can be supplemented by other aspects of security, such as security principles and policies. Therefore, Figure 11 describes the proposed security taxonomies proposed by S20 and S48, respectively.

Regarding safety, the taxonomies proposed by S2, S7, and S30 coincide in the categories but differ somewhat in the tactics proposed. The difference lies mainly in which safety objective they aim at. More precisely, the tactics described in Figure 12 are oriented towards the design of safety architectures, represent aspects of the IEC 61508 standard and describe decisions focused on process control devices.

Fault tolerance has also sparked interest in proposing tactics. In this respect, the main motivation of primary studies addressing fault tolerance is to propose tactics to design an architecture that ensures better response times to failures. Fault-tolerant architectural design allows the software to be proactively monitored by preventing critical systems from failing or mitigating a critical component's risk. S4 introduced a proposal for fault-tolerance tactics, a subset of the availability tactics proposed in [1] (see Figure 16).

In order to address design decisions related to the ability to handle increasing resources and data loads, scalability tactics taxonomies have been proposed whose main purpose is to provide tactics for designing systems that satisfy a certain degree of scalability, both vertically and horizontally. In this regard, both S19 and S79 proposed taxonomies of scalability tactics, which are described in Figure 13.

The deployability tactics taxonomy proposed in S39 aims to support an architect to evaluate structuring services to be deployed and how to deploy services. Figure 14 depicts tactics that help address deployability design concerns.

The study described in S69 reviewed the literature regarding modifiability and proposed a refined taxonomy of modifiability tactics (see Figure 15). These tactics are used to conduct a study to strengthen the understanding of qualities evolution in service- and microservices-based systems.

Key findings of RQ5

- Safety leads as the quality attribute with the most proposed taxonomies.
- Security is the only quality attribute for which only updates to the original taxonomy have been proposed.

5. Discussion

This section takes the results obtained in each research question and discusses those aspects that we find relevant to the tactics research community. The papers selected in our study reveal different perspectives on how researchers use tactics. The results depicted in Table 9 corroborates this.

5.1. Lack of rigor in characterizing and defining architectural tactics

The original categorizations of tactics largely arose from interviews with architects and practitioners. The intention of capturing this knowledge was to create a more systematic methodology to make design decisions and to evaluate quality requirements.

Table 9: Overview of primary studies for each RQ (the number inside each box is the amount of studies).

		2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	Total
RQ1	Security							2	2		2		2	4	1	1	1	3			18
	Fault tolerance						1	1	1					1						1	5
	Safety		1					1				1						1			4
	Availability						1	1					1					1			4
	Performance						1	1		1			1								4
	Scalability										1							1	1		3
	Deployability												2								2
	Adaptability													1							1
	Dependability									1											1
	Reliability												1								1
	Modifiability																	1			1
	Interoperability																			1	1
RQ2	Code analysis										2	2	1	1	1	3					10
	Manual mapping										1	1	2	3				2		1	10
	Text analysis																		1	3	4
	Multifacetic		1						1		1										3
RQ3	Source code										2	2	1	1	1	3			1	1	12
	Documentation										1			3				2		1	7
	Web repository																			3	3
	Design decisions		1										2								3
	Patterns								1												1
	Experts										1										1
	Standards											1									1
RQ4	Model						1	2	1	1	1	1	5	3							15
	Specific template	1	1								2	1	2			1		1		3	12
	Description							1			1			4				2	1		9
	Formal language								1												1
RQ5	Safety		1					1				1									3
	Security										1			1							2
	Scalability										1								1		2
	Fault tolerance						1														1
	Deployability												1								1
	Modifiability																	1			1

Based on results obtained in RQ2, RQ3 and RQ4, we realized that we could not identify a widespread tactics characterization and definition process. Some papers that proposed techniques to obtain tactics are based on capturing knowledge from source code, using several techniques (such as support vector machines, decision trees, Bayesian logistic regression, AdaBoost, etc). However, tactics are not only represented in code; tactics are also important in decision-making and architectural reasoning. Therefore, only a part of the tactics characterization process is addressed by tactics recovery from source code. Indeed, we believe that tactics and tactics discovery go well beyond source code; this is only one of the possible dimensions of tactics research. The ability to make design decisions and trade-offs, and their impact on quality attributes, are critical dimensions that must be addressed.

Moreover, there is not enough analysis about the proper way to characterize and define tactics. The primary studies do not dispute the nature of tactics; they only use them. But there is no widely agreed-upon method for defining a tactic. Some primary studies (like S20, S45, S46, and S76) attempt to propose a structure based on the original definition of [1] to refine or propose new tactics; they describe a general scenario in order to help understand tactics. The primary studies use the general scenarios in order to use robust methodological support to justify the characterization of tactics. In this regard, the general scenarios give the possibility to describe specific scenarios for quality attributes. Given that the structure of the general scenario (source of stimulus, stimulus, environment, artifact, response and response measure) describes significant information about quality attributes, this information can be used to define and characterize tactics. However, despite the potential of general scenarios, most studies do not specify a methodology for refining or proposing new tactics.

Research in tactics has the potential to generate a theory: the characterization and definition of tactics could be formalized based on analytic results. This does not mean that the other forms of inquiry identified in this paper can not generate new knowledge. The studies that use case studies, examples and experimental studies can generate significant findings in tactics research. But there is little critical analysis of, for example, How useful was the description of a tactic in a given context? What kinds of results are expected using the definition of a tactic? How have individual tactics evolved and how has the field of study evolved? Hence, we think that a methodological process to extract and characterize tactics in all their dimensions will not only refine the body of knowledge of tactics but also be able to discover other types of tactics.

5.2. Source code and application frameworks

Much of the primary research has studied tactics from two perspectives: source code and application framework. On the source code side, studies such as S22, S24, S26, S28, S28, S31, S52, S58, S62, S63 and S64 argue that design decisions and rationale found in documentation can rarely be traced back to source code. This implies that documented design decisions provide only limited support for keeping developers informed about the underlying decisions during code maintenance. Therefore, these studies propose using source code analysis techniques to identify tactics-related code. These techniques range from manual source code analysis to techniques using machine learning, latent topics and predictor models. This approach of investigating tactics not only helps to detect tactics in source code and support traceability with quality attribute requirements but also supports knowing which tactics are most used by architects to design software architectures. In this respect, we believe that an exploratory study on identification of tactics in source code applied to a significant set of systems could give insights into which tactics are most used by architects to address design concerns.

On the other hand, three studies (S73, S80 and S83) have addressed tactics as a way to evaluate and analyze application frameworks. S73 uses the specification of security tactics to evaluate the functionality of application frameworks based on functional coverage. S80 uses imperfect information to determine the relationship between tactics, patterns and frameworks in the context of microservice architecture. Imperfect information represents a more realistic scenario with respect to the decisions the architect must make in selecting frameworks. Finally, S83 also uses the specification of tactics and patterns to recommend application frameworks.

According to these studies, tactics can be used for different purposes that are not only simply to design decisions. Although it makes sense to use tactics to map design decisions in source code or recommend application frameworks, we believe that these approaches should also be complemented with the original definition of tactics. We think it is interesting to know the architect's intention in using these approaches: Translating a design decision into source code enables the implementation of the architecture. Still, architects rarely describe what stimulus and response they want to address when selecting a tactic. Using tactics to select application frameworks helps the architect evaluate a more restricted set of solutions. We believe that the architect's reasoning as to what responses the system should execute in the face of an external stimulus allows us to contextualize the use of tactics better to recommend application frameworks.

5.3. Little industrial evidence on the use of architectural tactics

The analysis of the primary studies reveals that there is little industrial evidence regarding the use of tactics. Table 10 describes the primary studies that have used industrial systems to validate their proposals.

Table 10: Industrial evidence summary

Study	Description
S9	The authors carried out studies in several companies in different sectors.
S15	The authors used two systems: a business reporting system (BRS) and an industrial control system (ICS) from ABB.
S16	The authors evaluate the efficacy of our approach based on a case study of a Lunar Robot.
S18	e-Bay is used in this study.
S57	A complex private social network system based on the Microsoft Azure cloud is used in this study.
S59	A healthcare system is used in this study to evaluate tactics.
S62	Apache Hive and Hadoop are used in this study.
S63	Chronium, PHP and Thunderbird are used in this study.
S67	An intranet communication system is used in this study.
S70	Industrial Control Systems (developed by 277 vendors) are used in this study.
S71	In this case study, three systems are used: Tactical Cloudlets system, GigaSight system, and AgroTempus system.
S74	The OmniSEC system is used in this study.
S78	Chronium, PHP and Thunderbird are used in this study.

One of the advantages that an architectural approach provides is the early choice of and description of design decisions. Those early decisions have a major impact on the rest of the project and are difficult to change as the project evolves. However, evaluating and managing these decisions based on quality attributes alone can be difficult. Stal [41] mentions that this occurs because quality attributes often affect many parts of a system. For example, practitioners cannot limit security or performance attributes to one part of the system. Such attempts are impossible in most contexts because the concerns are cross-cutting and often even invasive, i.e., they require practitioners to make design and code decisions in existing components.

Kassab et al. [42] mention there is limited evidence of architecture patterns in practice, and their study shows that quality attribute requirements are not a factor in selecting architecture patterns: practitioners select them mainly based on functionality and technological constraints. Although this makes sense in a world where time and resources are limited, this study shows that systematic analysis of software architecture design is (still) not a priority for the industry. Thus there is insufficient evidence of tactics studies in the industry. We know that quality attributes are systemic and need global and strategic treatment.

On the other hand, we also know that architecture patterns are selected through constraints that have weak relation to software design and quality attributes. Nevertheless, the appropriate way to address these issues is the systematization of quality attribute analysis and decisions [43]. For example, Stal [41] mentions that scenario-based approaches (e.g., ATAM and utility trees) are an excellent way to model, document and express quality attributes in a more specific and systemic way. This allows for the introduction of more strategic approaches such as tactics. Indeed, each high priority scenario addressed by an ATAM may result in the deployment of tactics (or "strategies" as they are often called in industry).

6. Threats to validity

Several threats to the study validity have been identified and mitigated. Following Wholin et al. [44] and the suggestions described by Ampatzoglou et al. [45], we address threats to the conclusions, internal, external, construct and external validity.

Conclusions validity refers to the relationship between the extracted and synthesized data and the study findings. To mitigate potential threats to the conclusion, we used a search string to automatically find primary studies, which contained the main concepts addressed by the study. We also replicated the results obtained with the search string with other researchers from our research team. In addition, we defined a template to systematize the data extracted from the primary studies. The three authors participated in the creation of the template as well as in the data refinement. Finally, we used guidelines for systematic mapping studies in order to reduce biases in the review process and in the extraction of information from the primary studies.

Internal validity is related to the level of control of the study on the variables that may influence the study itself. To mitigate internal validity threats, we followed the systematic mapping studies guidelines proposed by Petersen et al. [18]. For data analysis, we used descriptive statistics to interpret the data; and for qualitative data analysis, we used models and representations from the literature to contextualize the primary studies' contribution.

External validity is related to the generalizability of the study findings. To mitigate threats to external validity, we discussed the systematic mapping results with collaborators, and held working sessions with colleagues of our research team to discuss the contribution of each primary study to the research on tactics. To address the potential threat of lacking a set of primary studies representative of the research objective, we used inclusion and exclusion criteria to filter studies; these criteria allowed to identify more precisely those studies that belonged (or not) to the study scope.

Construct validity refers to the validity of extraction and tuning of research question data. To mitigate its threats, we tested the search string in pilot studies in order to refine and improve it, using a trusted source (ACM Digital Library) to evaluate the studies it yield. Once the final search string was accepted, we used it on several prestigious electronic databases to find primary studies. Additionally, we used snowballing to cover and review a larger number of primary studies. Once all the primary studies were collected, we reviewed each of them rigorously and critically. We also defined a thorough mapping process to obtain a set of primary studies suitable for answering the research questions.

7. Related work

Previous studies have revised the tactics literature with specific quality attributes or technologies in mind.

Lewis et al. [9] conducted a systematic literature mapping of tactics for cyber-foraging, arguing that mobile devices have become the dominant way to interact with the Internet, businesses, and social networks. The study describes quality attributes that are relevant for cyber-foraging systems, and found research gaps regarding system-level concerns for operations in cyber-foraging systems and large-scale evaluations. As a further result, they codified design decisions and proposed tactics for cyber-foraging.

Ullah et al. [10] conducted a systematic study of tactics for Big Data Cyber-security Analytics (BDCA) systems. They described critical quality attributes for BDCA systems (namely, performance, accuracy, and scalability), and found a lack of architectural support for some of them. They also remarked the lack of empirical research about the coding of tactics and quality trade-offs among tactics.

Li et al. [46] reported a systematic literature review for microservice architectures. They identified six key quality attributes for microservices architecture (namely, scalability, performance, availability, monitorability, security, and testability), and proposed nineteen tactics to address them. In the same line, Osses et al. [47] also explored tactics and architectural patterns for microservice architectures, and concluded that there is actual but scant evidence of tactics in microservice architectures.

Paradis et al. [48] explored tactics for energy efficiency. They proposed a new taxonomy for energy efficiency to address the identified research gaps; discussed evidence from industry regarding the use of tactics for software energy efficiency; and argued for the need of experimental studies to validate these tactics.

While these studies agree on the relevance and importance of architectural tactics to address quality attributes, they all focus on specific domains or types of systems. This study has explored tactics more broadly rather than for specific domains, and has done so by mapping how the literature has addressed the various aspects of tactics.

8. Conclusions

This paper has reported the results of a systematic mapping study of the existing literature on tactics, from their introduction in 2003 until now. We reviewed and analyzed 91 primary studies in order to answer 5 research questions. We identified 12 quality attributes that have been addressed by the primary studies, with security being the attribute that has attracted the most interest in the community. Also, we realized that 70% of studies do not explicitly describe their method to identifying tactics; however, for those studies that do describe, we have identified 4 methods that allow for the identification of tactics. In the same regard, 69% of the primary studies do not describe which data sources they use to recognize tactics; nevertheless, we have identified 7 data sources used to recognize tactics on those studies that do describe the data source. We also identified 4 mechanisms for characterizing tactics, with models and specific templates predominating as the preferred. And we identified 10 taxonomies that have proposed and/or refined taxonomies of tactics.

The painstaking review, analysis and summarization of tactics proposals led us to the unexpected, but also unavoidable, conclusion that most tactics proposed in the literature do not conform to the description of the original definition, which posited them as design decisions to preserve quality attributes in presence of stimuli.

The evidence gathered shows several research opportunities:

1. More rigorous methods for identifying and characterizing tactics are needed, since many primary studies use (implicit) definitions of tactics quite at variance from the initial definition and spirit.
2. Tactics have become relevant to map architecture decisions onto source code, and automation opportunities beckon within reach.
3. Tactics are a key conceptual tool to reduce solution spaces that architects must evaluate, and should be related to application frameworks whenever possible; and
4. Industry adoption of tactics will need more empirical studies that show how tactics benefit architects' decision making.

Ongoing research is exploring barriers to adoption of tactics by architects in industry, and specifically their perceptions on whether and how their key design decisions can (and should) be usefully expressed using tactics. Future research will focus on proposing a systematic method to define and characterize tactics, to better discover new ones, describe them adequately, and drive their use.

9. Acknowledgments

This work was supported in part by the Centro Basal CCTVal (PIA/APOYO AFB180002) from ANID (Chile), and by awards CCF-1823214 and CCF-1817267 from the National Science Foundation (USA). We also thanks to the department of Electronics and Informatics from the Universidad Técnica Federico Santa María.

Appendix A Taxonomies identified in the SMS

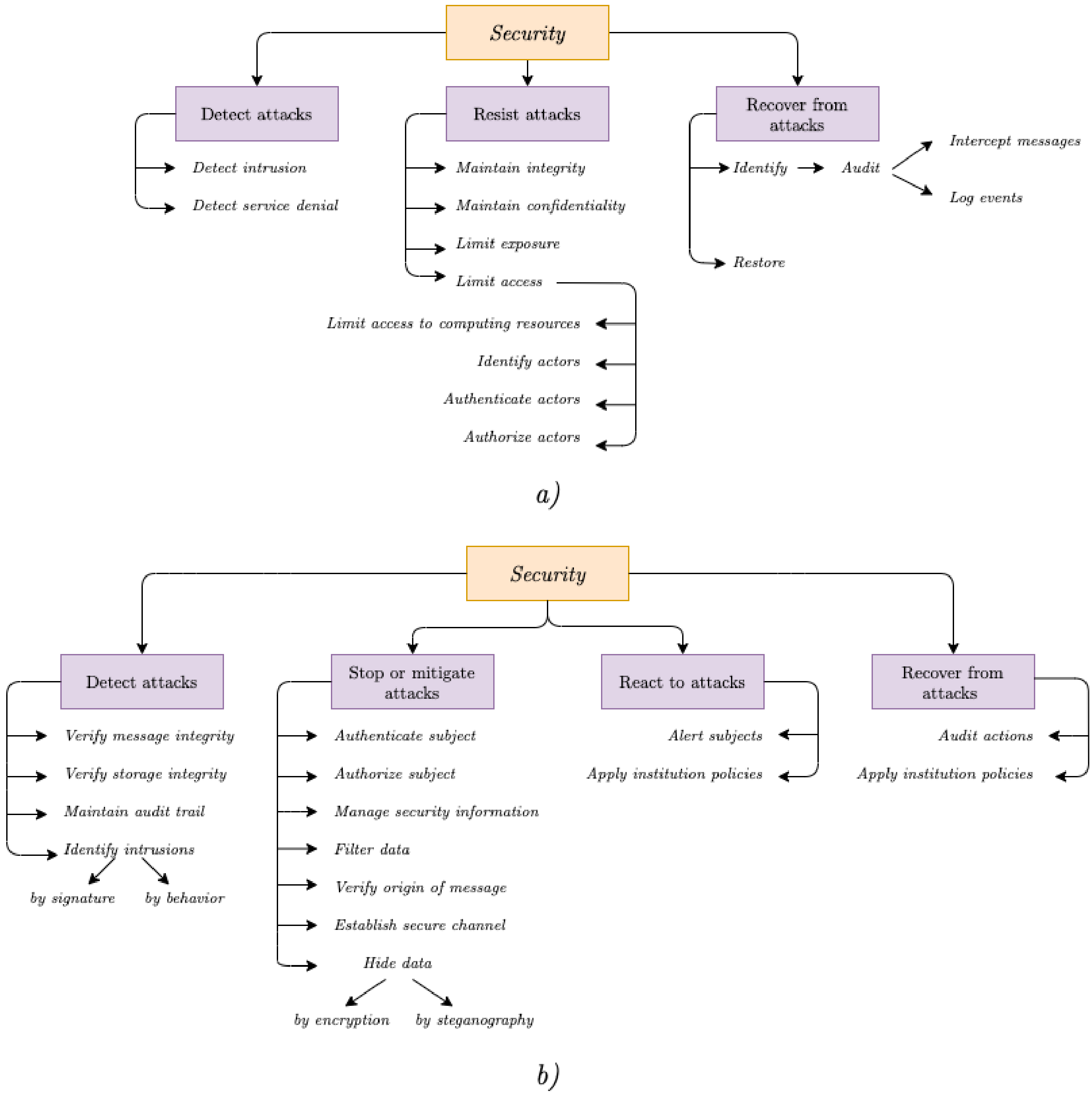

Figure 11: Security tactics taxonomy proposed by S20 (a) and S48 (b).

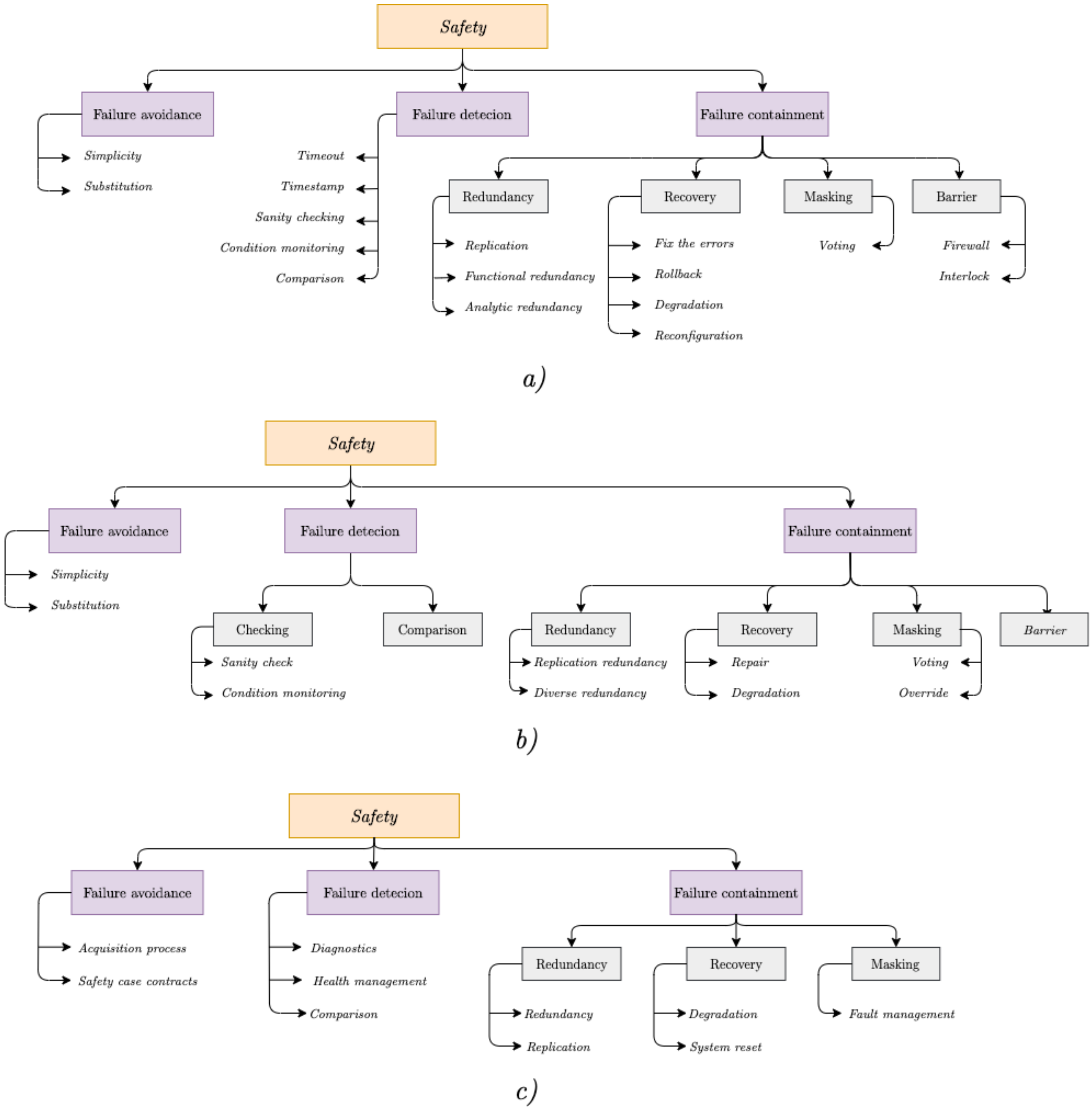

Figure 12: Safety tactics taxonomy proposed by S2 (a), S30 (b) and S7 (c)

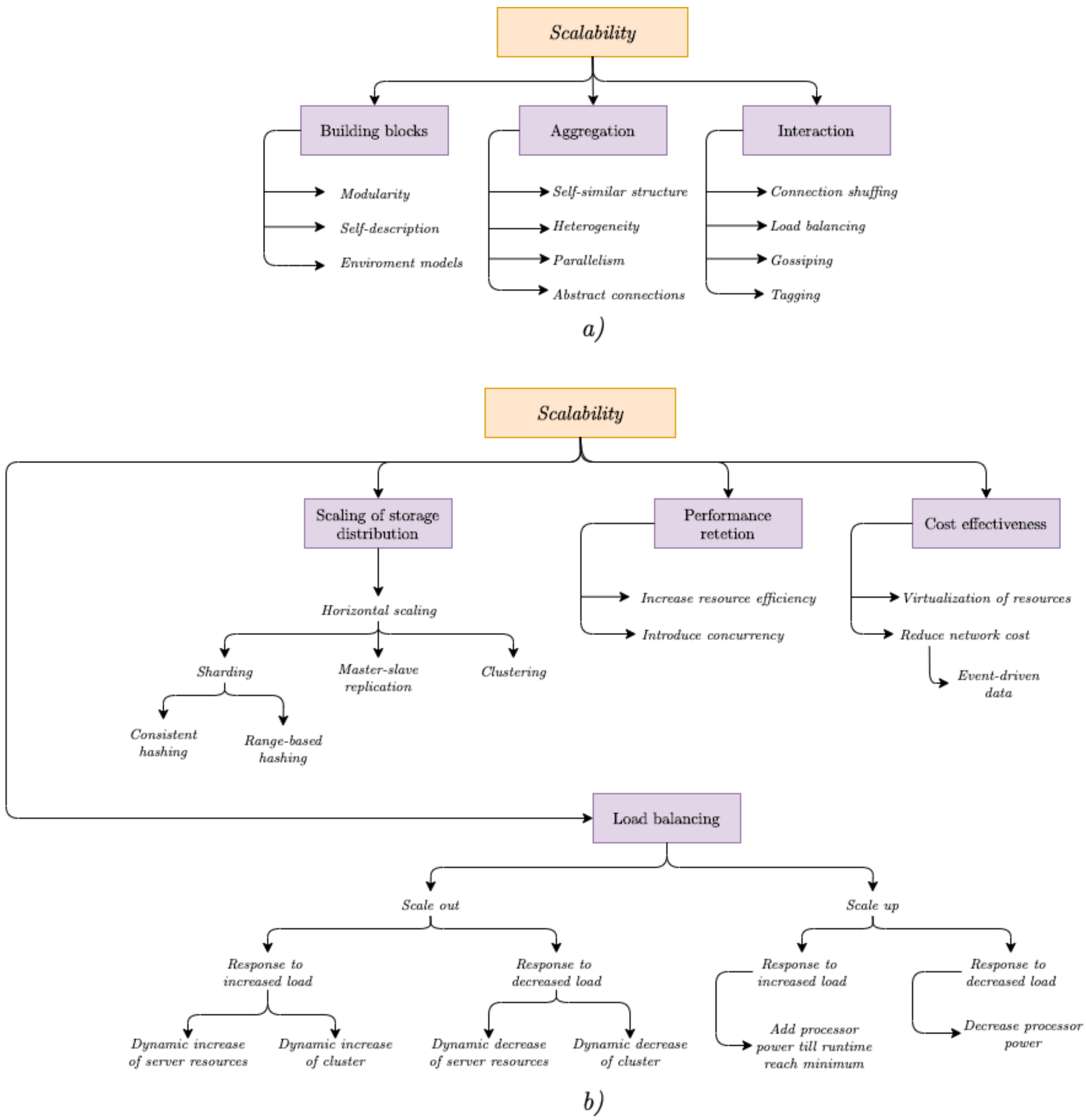

Figure 13: Scalability tactics taxonomy proposed by S19 (a) and S79 (b)

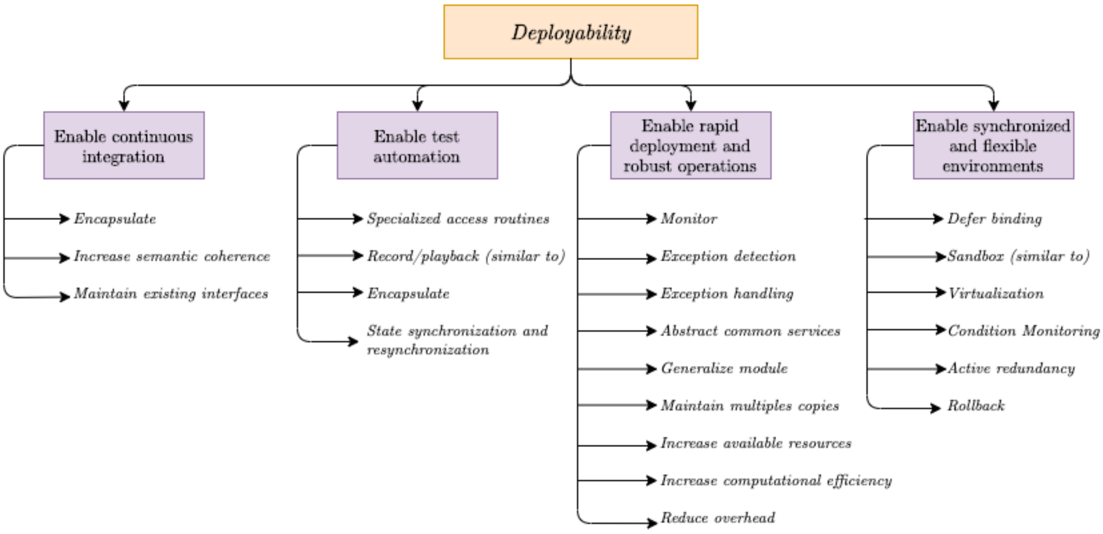

Figure 14: Deployability tactics taxonomy proposed by S39

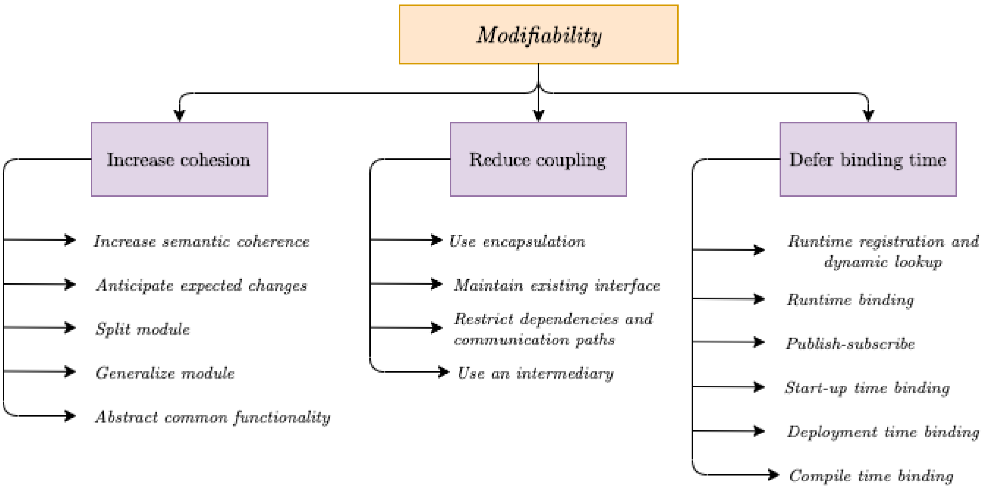

Figure 15: Modifiability tactics taxonomy proposed by S69

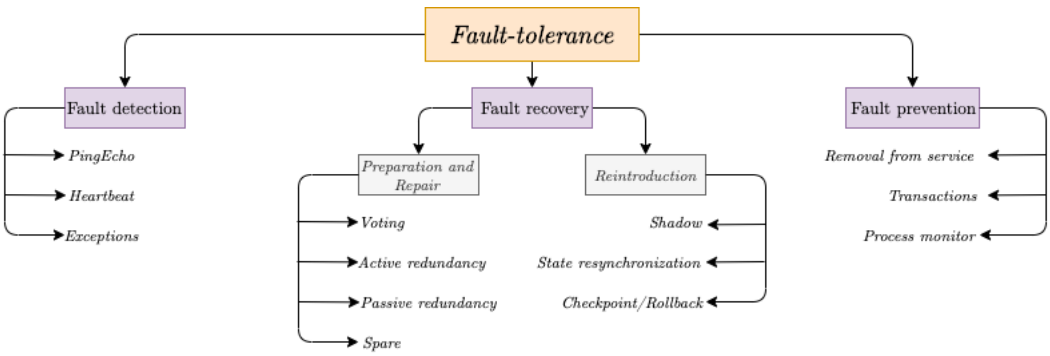

Figure 16: Fault-tolerance tactics taxonomy proposed by S4

Appendix B Primary studies

Table 11: Primary studies (part I)

ID	Authors	Title	Venue	Cite
S1	F. Bachmann, L. Bass, and M. Klein	Moving from quality attribute requirements to architectural decisions	International Software Requirements to Architectures Workshop	[5]
S2	W. Wu and T. Kelly	Safety tactics for software architecture design	Annual International Computer Software and Applications Conference	[49]
S3	H. Reza, D. Jurgens, J. White, J. Anderson, and J. Peterson	An architectural design selection tool based on design tactics scenarios and nonfunctional requirements	IEEE International Conference on Electro Information Technology	[50]
S4	N. B. Harrison and P. Avgeriou	Incorporating fault tolerance tactics in software architecture patterns	Joint International Workshop on Software Engineering for Resilient Systems	[51]
S5	S. Kim, D. K. Kim, N. Lu, and S. Y. Park	A tactic-based approach to embodying non-functional requirements into software architectures	International IEEE Enterprise Distributed Object Computing Conference	[52]
S6	S. Kim, D. K. Kim, L. Lu, and S. Park	Quality-driven architecture development using architectural tactics	Journal of Systems and Software	[53]
S7	A. E. Hill and M. Nicholson	Safety tactics for reconfigurable process control devices	IET International Conference on Systems Safety	[54]
S8	T. Marew, J. S. Lee, and D. H. Bae	Tactics based approach for integrating non-functional requirements in object-oriented analysis and design	Journal of Systems and Software	[55]
S9	N. B. Harrison and P. Avgeriou	How do architecture patterns and tactics interact? A model and annotation	Journal of Systems and Software	[6]
S10	J. Ryoo, P. Laplante, and R. Kazman	A methodology for mining security tactics from security patterns	Annual Hawaii International Conference on System Sciences	[56]
S11	N. B. Harrison, P. Avgeriou, and U. Zdun	On the impact of fault tolerance tactics on architecture patterns	International Workshop on Software Engineering for Resilient Systems	[57]
S12	A. Wyeth and C. Zhang	Formal specification of Software Architecture Security Tactics	International Conference on Software Engineering and Knowledge Engineering	[58]
S13	S. Kim, D. K. Kim, and S. Park	Tool support for quality-driven development of software architectures	IEEE/ACM International Conference on Automated Software Engineering	[59]
S14	S. H. Al-Daajeh, R. E. Al-Qutaish, and F. Al-Qirem	Engineering dependability to embedded systems software via tactics	International Journal of Software Engineering and its Applications	[60]
S15	A. Koziolek, H. Koziolek, and R. Reussner	PerOpteryx: Automated Application of Tactics in Multi-Objective Software Architecture Optimization	ACM SIGSOFT conference	[61]
S16	M. Mirakhorli and J. Cleland-Huang	Using tactic traceability information models to reduce the risk of architectural degradation during system maintenance	IEEE International Conference on Software Maintenance	[62]
S17	A. Sanchez, A. Aguiar, L. S. Barbosa, and D. Riesco	Analysing tactics in architectural patterns	IEEE Software Engineering Workshop	[63]
S18	J. M. Cañete-Valdeón	Annotating problem diagrams with architectural tactics for reasoning on quality requirements	Information Processing Letters	[64]
S19	R. Kazman and P. Kruchten	Design approaches for taming complexity	IEEE International Systems Conference	[65]

Table 12: Primary studies (part II)

ID	Authors	Title	Venue	Cite
S20	J. Ryoo, P. Laplante, and R. Kazman	Revising a security tactics hierarchy through decomposition reclassification and derivation	IEEE International Conference on Software Security and Reliability Companion	[66]
S21	E. B. Fernández and H. Astudillo	Should we use tactics or patterns to build secure systems?	International Symposium on Software Architecture and Patterns	[67]
S22	M. Mirakhorli, P. Mäder, and J. Cleland-Huang	Variability points and design pattern usage in architectural tactics	ACM SIGSOFT International Symposium on the Foundations of Software Engineering	[68]
S23	S. H. Al-daajeh, R. E. Al-qutaish, and F. Al-qirem	A Tactic-Based Framework to Evaluate the Relationships between the Software Product Quality Attributes	International Journal of Software Engineering	[69]
S24	M. Mirakhorli, Y. Shin, J. Cleland-Huang, and M. Cinar	A tactic-centric approach for automating traceability of quality concerns	International Conference on Software Engineering	[70]
S25	C. Preschern	Catalog of Security Tactics linked to Common Criteria Requirements	Conference on Pattern Languages of Programs	[71]
S26	M. Mirakhorli, J. Carvalho, C. H. Jane, and P. Mäder	A domain-centric approach for recommending architectural tactics to satisfy quality concerns	International Workshop on the Twin Peaks of Requirements and Architecture	[72]
S27	X. Qiu and L. Zhang	Providing support for specifying redundancy tactics using aspect-oriented modeling	International Symposium on the Physical and Failure Analysis of Integrated Circuits	[73]
S28	M. Mirakhorli	Preventing erosion of architectural tactics through their strategic implementation, preservation and visualization	IEEE/ACM International Conference on Automated Software Engineering	[74]
S29	M. Kassab and G. El-Boussaidi	Towards quantifying quality tactics and architectural patterns interactions	International Conference on Software Engineering and Knowledge Engineering	[75]
S30	C. Preschern, N. Kajtazovic, and C. Kreiner	Catalog of Safety Tactics in the light of the IEC 61508 Safety Lifecycle	VikingPLoP Conference	[76]
S31	M. Mirakhorli, A. Fakhry, A. Grechko, M. Wieloch, and J. Cleland-Huang	Archie: A tool for detecting monitoring and preserving architecturally significant code	ACM SIGSOFT International Symposium on the Foundations of Software Engineering	[77]
S32	R. L. Nord, I. Ozkaya, and P. Kruchten	Agile in Distress: Architecture to the Rescue	International Conference on Agile Software Development	[78]
S33	X. Qiu and L. Zhang	Test scenario generation for reliability tactics from uml sequence diagram	Asia-Pacific Software Engineering Conference	[79]
S34	S. Tahmasebipour and S. M. Babamir	Ranking of Common Architectural Styles Based on Availability Security and Performance Quality Attributes	Journal of Computing and Security	[80]
S35	J. Chavarriaga, C. Noguera, R. Casallas, and V. Jonckers	Architectural tactics support in cloud computing providers: The jelastic case	International ACM SIGSOFT Conference on Quality of Software Architectures	[81]
S36	G. Pedraza-García, H. Astudillo, and D. Correal	A methodological approach to apply security tactics in software architecture design	IEEE Colombian Conference on Communications and Computing	[82]
S37	X. Qiu and L. Zhang	Specifying redundancy tactics as crosscutting concerns using aspect-oriented modeling	Frontiers of Computer Science	[83]

Table 13: Primary studies (part III)

ID	Authors	Title	Venue	Cite
S38	G. Procaccianti, P. Lago, and G. A. Lewis	A catalogue of green architectural tactics for the cloud	IEEE International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems	[8]
S39	S. Bellomo, N. Ernst, R. Nord, and R. Kazman	Toward design decisions to enable deployability: Empirical study of three projects reaching for the continuous delivery holy grail	Annual IEEE/IFIP International Conference on Dependable Systems and Networks	[84]
S40	S. N. Lee, D. Ko, S. Park, and S. Kim	An approach to building domain architectures using domain component model and architectural tactics	International Journal of Engineering Systems Modelling and Simulation	[85]
S41	R. Noel, G. Pedraza-García, H. Astudillo, and E. B. Fernández	An exploratory comparison of security patterns and tactics to harden systems	Ibero-American Conference Software Engineering	[86]
S42	N. A. M. Alzahrani and D. C. Petriu	Modeling fault tolerance tactics with reusable aspects	International ACM SIGSOFT Conference on Quality of Software Architectures	[87]
S43	S. Kim	A quantitative and knowledge-based approach to choosing security architectural tactics	International Journal Ad Hoc and Ubiquitous Computing	[88]
S44	A. Alebrahim, S. Fassbender, M. Filipczyk, M. Goedicke, and M. Heisel	Towards a reliable mapping between performance and security tactics and architectural patterns	ACM International Conference Proceeding Series	[89]
S45	G. Lewis and P. Lago	A catalog of architectural tactics for cyber-foraging	International ACM SIGSOFT Conference on Quality of Software Architectures	[90]
S46	M. B. Kjaergaard and M. Kuhrmann	On architectural qualities and tactics for mobile sensing	International ACM SIGSOFT Conference on Quality of Software Architectures	[91]
S47	A. E. Sabry	Decision Model for Software Architectural Tactics Selection Based on Quality Attributes Requirements	Procedia Computer Science	[92]
S48	E. B. Fernandez, H. Astudillo, and G. Pedraza-García	Revisiting Architectural Tactics for Security	European Conference on Software Architecture	[93]
S49	M. Mirakhorli and J. Cleland-Huang	Modifications, tweaks and bug fixes in architectural tactics	IEEE International Working Conference on Mining Software Repositories	[94]
S50	H. M. Chen, R. Kazman, S. Haziyev, V. Kropov, and D. Chtchourov	Architectural Support for DevOps in a Neo-Metropolis BDaaS Platform	IEEE Symposium on Reliable Distributed Systems	[95]
S51	J. Chavarriaga, C. Noguera, R. Casallas, and V. Jonckers	Managing trade-offs among architectural tactics using feature models and feature-solution graphs	Colombian Computing Conference	[96]
S52	J. Ryoo, B. Malone, P. A. Laplante, and P. Anand	The Use of Security Tactics in Open Source Software Projects	IEEE Transactions on Reliability	[97]

Table 14: Primary studies (part IV)

ID	Authors	Title	Venue	Cite
S53	S. Adam and A. Abran	The software architecture mapping framework for managing architectural knowledge	International Conference on Software Engineering and Knowledge Engineering	[98]
S54	D. E. Krutz and M. Mirakhorl	Architectural clones: toward tactical code reuse	Annual ACM Symposium on Applied Computing	[99]
S55	G. Pedraza-García, R. Noel, S. Matalonga, H. Astudillo, and E. B. Fernández	Mitigating security threats using tactics and patterns: a controlled experiment	European Conference on Software Architecture Workshops	[100]
S56	M. Kassab and G. Destefanis	Estimating Development Effort for Software Architectural Tactics	International Andrei Ershov Informatics Conference	[101]
S57	D. Gesvindr and B. Buhnova	Architectural tactics for the design of efficient PaaS cloud applications	Working IEEE/IFIP Conference on Software Architecture	[102]
S58	M. Mirakhorli and J. Cleland-Huang	Detecting, Tracing and Monitoring Architectural Tactics in Code	IEEE Transactions on Software Engineering	[103]
S59	G. Márquez and H. Astudillo	Selecting components assemblies from non-functional requirements through tactics and scenarios	International Conference of the Chilean Computer Science Society	[104]
S60	G. Márquez and H. Astudillo	Selection of software components from business objectives scenarios through architectural tactics	IEEE/ACM International Conference on Software Engineering Companion	[105]
S61	A. M. Alashqar, H. M. El-Bakry, and A. A. Elfetouh	A Framework for Selecting Architectural Tactics Using Fuzzy Measures	International Journal of Software Engineering and Knowledge Engineering	[106]
S62	R. Gopalakrishnan, P. Sharma, M. Mirakhorli, and M. Galster	Can Latent Topics in Source Code Predict Missing Architectural Tactics?	IEEE/ACM International Conference on Software Engineering	[107]
S63	J. C. S. Santos, A. Peruma, M. Mirakhorli, M. Galstery, J. V. Vidal, and A. Sejfia	Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium PHP and Thunderbird	IEEE International Conference on Software Architecture	[108]
S64	M. Salama, A. Shawish, and R. Bahsoon	Dynamic modelling of tactics impact on the stability of self-aware cloud architectures	IEEE International Conference on Cloud Computing	[109]
S65	M. A. Al Imran, S. P. Lee, and M. A. M. Ahsan	Quality driven architectural solutions selection approach through measuring impact factors	International Conference on Electrical Engineering and Computer Science	[110]
S66	I. J. Mujhid, J. C. Joanna, R. Gopalakrishnan, and M. Mirakhorli	A search engine for finding and reusing architecturally significant code	Journal of Systems and Software	[111]
S67	F. Osses, G. Márquez, M. M. Villegas, C. Orellana, M. Visconti, and H. Astudillo	Security tactics selection poker (TaSPeR): A card game to select security tactics to satisfy security requirements	European Conference on Software Architecture: Companion Proceedings	[112]
S68	F. Alizadeh Moghaddam, G. Procaccianti, G. A. Lewis, and P. Lago	Empirical validation of cyber-foraging architectural tactics for surrogate provisioning	Journal of Systems and Software	[113]

Table 15: Primary studies (part V)

ID	Authors	Title	Venue	Cite
S69	J. Bogner, S. Wagner, and A. Zimmermann	Using architectural modifiability tactics to examine evolution qualities of Service- and Microservice-Based Systems: An approach based on principles and patterns	Software-Intensive Cyber-Physical Systems	[114]
S70	D. Gonzalez, F. Alhenaki, and M. Mirakhorli	Architectural security weaknesses in industrial control systems (ICS) an empirical study based on disclosed software vulnerabilities	IEEE International Conference on Software Architecture	[115]
S71	G. Lewis, P. Lago, S. Echeverría, and P. Simoens	A tale of three systems: Case studies on the application of architectural tactics for cyber-foraging	Future Generation Computer Systems	[116]
S72	G. Márquez, M. M. Villegas, and H. Astudillo	An Empirical Study of Scalability Frameworks in Open Source Microservices-based Systems	International Conference of the Chilean Computer Science Society	[117]
S73	H. Cervantes, H., Kazman, R., Ryoo, J., Cho, J., Cho, G., Kim, H., and Kang, J.	Data-Driven Selection of Security Application Frameworks During Architectural Design	Annual Hawaii International Conference on System Sciences	[38]
S74	C. Orellana, M. M. Villegas, and H. Astudillo	Mitigating security threats through the use of security tactics to design secure cyber-physical systems (CPS)	European Conference on Software Architecture	[118]
S75	F. Wessling, C. Ehmke, O. Meyer, and V. Gruhn	Towards Blockchain Tactics: Building Hybrid Decentralized Software Architectures	International Conference on Software Architecture Companion	[119]
S76	G. Márquez and H. Astudillo	Identifying availability tactics to support security architectural design of microservice-based systems	European Conference on Software Architecture	[120]
S77	C. Preschern, N. Kajtazovic, and C. Kreiner	Safety architecture pattern system with security aspects	Transactions on Pattern Languages of Programming IV	[121]
S78	J. C. S. Santos, K. Tarrit, A. Sejfia, M. Mirakhorli, and M. Galster	An empirical study of tactical vulnerabilities	Journal of Systems and Software	[122]
S79	S. P. Nanda and H. Reza	Deriving Scalability Tactics for Development of Data-Intensive Systems	International Conference on Information Technology–New Generations	[123]
S80	G. Marquez, Y. Lazo, and H. Astudillo	Evaluating Frameworks Assemblies in Microservices-based Systems Using Imperfect Information	IEEE International Conference on Software Architecture Companion	[124]
S81	M. Alenezi, A. Agrawal, R. Kumar, and R. A. Khan	Evaluating Performance of Web Application Security through a Fuzzy Based Hybrid Multi-Criteria Decision-Making Approach: Design Tactics Perspective	IEEE Access	[125]
S82	Agrawal, A., Seh, A. H., Baz, A., Alhakami, H., Alhakami, W., Baz, M., Rajeev, K. and Khan, R. A.	Software security estimation using the hybrid fuzzy ANP-TOPSIS approach: Design tactics perspective	Symmetry	[126]
S83	Keim, J., Kaplan, A., Koziolek, A., and Mirakhorli, M.	Does BERT Understand Code?–An Exploratory Study on the Detection of Architectural Tactics in Code	European Conference on Software Architecture	[127]

Table 16: Primary studies (part VI)

ID	Authors	Title	Venue	Cite
S84	Milhem, H., Weiss, M., and Some, S. S.	Modeling and Selecting Frameworks in Terms of Patterns, Tactics and System Qualities	International Journal of Software Engineering and Knowledge Engineerings	[128]
S85	Malavolta, I., Chinnappan, K., Swanborn, S., Lewis, G. A., and Lago, P.	AMining the ROS ecosystem for green architectural tactics in robotics and an empirical evaluation	International Conference on Mining Software Repositories	[129]
S86	Yánez, W., Bahsoon, R., Zhang, Y., and Kazman, R.	Architecting Internet of Things Systems with Blockchain: A Catalog of Tactics	ACM Transactions on Software Engineering and Methodology	[130]
S87	Bi, T., Liang, P., Tang, A., and Xia, X.	Mining architecture tactics and quality attributes knowledge in Stack Overflow	Journal of Systems and Software	[131]
S88	Valle, P. H. D., Garcés, L., and Nakagawa, E. Y.	Architectural strategies for interoperability of software-intensive systems: practitioners' perspective	Annual ACM Symposium on Applied Computing	[132]
S89	AlDaajeh, S. H., Harous, S., and Alrabaee, S.	Fault-Detection Tactics for Optimized Embedded Systems Efficiency	IEEE Access	[133]
S90	Shokri, A., Santos, J., and Mirakhorli, M.	ArCode: Facilitating the Use of Application Frameworks to Implement Tactics and Patterns	International Conference on Software Architecture	[134]
S91	Chinnappan, K., Malavolta, I., Lewis, G. A., Albonico, M., and Lago, P.	Architectural Tactics for Energy-Aware Robotics Software: A Preliminary Study	European Conference on Software Architecture	[135]

References

[1] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice (2nd Edition), SEI Series in Software Engineering, 2003.

[2] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice (3rd Edition), SEI Series in Software Engineering, 2013.

[3] L. Bass, M. H. Klein, G. A. Moreno, Applicability of general scenarios to the architecture trade-off analysis method, Tech. rep., Software Engineering Institute, Carnegie-Mellon University (2001).

[4] F. Bachmann, L. Bass, M. Klein, Deriving architectural tactics: A step toward methodical architectural design, No. CMU/SEI-2003-TR-004. Carnegie-Mellon University, Software Engineering Institute.

[5] F. Bachmann, L. Bass, M. Klein, Moving from quality attribute requirements to architectural decisions, The Second International Workshop on From Software Requirements to Architectures (STRAW) (2003) 122–129.

[6] N. B. Harrison, P. Avgeriou, How do architecture patterns and tactics interact? a model and annotation, Journal of Systems and Software 83 (10) (2010) 1735–1758, Doi: `https://doi.org/10.1016/j.jss.2010.04.067`.

[7] L. Bass, G. Moreno, et al., Applicability of general scenarios to the architecture tradeoff analysis method, Tech. rep., Carnegie-Mellon University of Pittsburgh PA Software Engineering Institute. (2001).

[8] G. Procaccianti, P. Lago, G. A. Lewis, A catalogue of green architectural tactics for the cloud, 8th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA) (2014) 29–36Doi: `https://doi.org/10.1109/MESOCA.2014.12`.

[9] G. Lewis, P. Lago, Architectural tactics for cyber-foraging: Results of a systematic literature review, Journal of Systems and Software 107 (158-186), Doi: `https://doi.org/10.1016/j.jss.2015.06.005`.

[10] F. Ullah, M. A. Babar, Architectural tactics for big data cybersecurity analytics systems: a review, Journal of Systems and Software 151 (2019) 81–118, Doi: `https://doi.org/10.1016/j.jss.2019.01.051`.

[11] G. Márquez, H. Astudillo, R. Kazman, Architectural Tactics in Software Architecture: A Systematic Mapping Study - Study protocol, Doi: `10.5281/zenodo.7290575` (Oct. 2022).
URL `https://doi.org/10.5281/zenodo.7290575`

[12] F. Bachmann, L. Bass, M. Klein, llluminating the fundamental contributors to software architecture quality, Tech. rep., Software Engineering Institute, Carnegie-Mellon University (2002).

[13] L. Bass, M. Klein, F. Bachmann, Quality attribute design primitives, Tech. rep., Software Engineering Institute, Carnegie-Mellon University (2000).

[14] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice (4th Edition), SEI Series in Software Engineering, 2021.

[15] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) (2008) 1–10Doi: `10.14236/ewic/EASE2008.8`.

[16] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, Proceedings of the 18th international conference on evaluation and assessment in software engineering (2014) 1–10Doi: `https://doi.org/10.1145/2601248.2601268`.

[17] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01.

[18] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and Software Technology 64 (2015) 1–18, Doi: .

[19] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requirements engineering 11 (1) (2006) 102–107, Doi: .

[20] M. Kuhrmann, P. Diebold, J. Münch, Software process improvement: a systematic mapping study on

the state of the art, PeerJ Computer Science 2 (2016) e62, doi:https://doi.org/10.7717/peerj-cs.62.

[21] M. Shaw, Writing good software engineering research papers, International Conference on Software Engineering (2003) 726–736doi:10.1109/ICSE.2003.1201262.

[22] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative research in psychology 3 (2) (2006) 77–101, Doi: 10.1191/1478088706qp063oa.

[23] P. Tarvainen, Adaptability evaluation at software architecture level, The Open Software Engineering Journal 2 (1), Doi: 10.2174/1874107X00802010001.

[24] A. Avizienis, J. C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE transactions on dependable and secure computing 1 (1) (2004) 11–33, Doi: 10.1109/TDSC.2004.2.

[25] ISO 25000 software and data quality, ISO/IEC 25010, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, accessed on 2020-09-19.

[26] L. Bass, "Deployability" in *Software Quality Assurance*, Morgan Kaufmann, 2016, pp. xxiii–xxvii.

[27] R. Kazman, P. Kruchten, Design approaches for taming complexity, IEEE International Systems Conference (2012) 1–6Doi: 10.1109/SysCon.2012.6189488.

[28] Software Safety, https://www.systemsafetyengineering.com/software-safety.html.

[29] The MITRE Corporation, Common vulnerabilities and exposures (CVE), https://cve.mitre.org, accessed on 2020-08-27.

[30] K. Moløkken-Østvold, C. H. Nils, C. B. Hans, Using planning poker for combining expert estimates in software projects, Journal of Systems and Software 81 (12) (2008) 2106–2117, Doi: https://doi.org/10.1016/j.jss.2008.03.058.

[31] L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos, The nfr framework in action, Non-Functional Requirements in software engineering. International Series in Software Engineering 5 (2000) 15–45, Doi: https://doi.org/10.1007/978-1-4615-5269-7_2.

[32] D. K. Kim, The role-based metamodeling language for specifying design patterns, Design Pattern Formalization Techniques (2007) 23Doi: 10.4018/978-1-59904-219-0.ch009.

[33] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805.

[34] J. Cleland-Huang, R. Settimi, X. Zou, P. Solc, The detection and classification of non-functional requirements with application to early aspects, IEEE International Requirements Engineering Conference (2006) 39–48Doi: 10.1109/RE.2006.65.

[35] M. Mirakhorli, C.-H. J. Shin, Y., M. Cinar, A tactic-centric approach for automating traceability of quality concerns, International conference on software engineering (ICSE) (2012) 639–649Doi: 10.1109/ICSE.2012.6227153.

[36] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (Jan) (2003) 993–1022.

[37] E. Alpaydin, Introduction to machine learning, MIT press, 1 Rogers Street in Cambridge, MA 02142, USA, 2020.

[38] H. Cervantes, R. Kazman, J. Ryoo, J. Cho, G. Cho, H. Kim, J. Kang, Data-driven selection of security application frameworks during architectural design, 52nd Hawaii International Conference on System Sciences (2019) 7331–7340.

[39] J. M. Spivey, J. R. Abrial, The Z notation, Hemel Hempstead: Prentice Hall, 1992.

[40] J. White, J. A. Galindo, T. Saxena, B. Dougherty, D. Benavides, D. C. Schmidt, Evolving feature model configurations in software product lines, Journal of Systems and Software 87 (2014) 119–136, Doi: https://doi.org/10.1016/j.jss.2013.10.010.

[41] M. Stal, Faster, Better, Higher – But How?, https://www.infoq.com/articles/add-nfrs/ (2012).

[42] M. Kassab, M. Mazzara, J. Lee, G. Succi, Software architectural patterns in practice: an empirical study, Innovations in Systems and Software Engineering 14 (4) (2018) 263–271, Doi: https://doi.org/10.1007/s11334-018-0319-4.

[43] M. Richards, N. Ford, Fundamentals of Software Architecture: An Engineering Approach, O'Reilly,

2020.

[44] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science and Business Media.

[45] A. Ampatzoglou, S. Bibi, P. Avgeriou, A. Chatzigeorgiou, Guidelines for managing threats to validity of secondary studies in software engineering, Contemporary Empirical Methods in Software Engineering (2020) 415–441Doi: https://doi.org/10.1007/978-3-030-32489-6_15.

[46] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, M. A. Babar, Understanding and addressing quality attributes of microservices architecture: A systematic literature review, Information and Software Technology (2020) 106449Doi: https://doi.org/10.1016/j.infsof.2020.106449.

[47] F. Osses, G. Márquez, H. Astudillo, An exploratory study of academic architectural tactics and patterns in microservices: A systematic literature review, Iberoamerican Conference on Software Engineering (CIbSE) (2018) 71–84.

[48] C. Paradis, R. Kazman, D. A. Tamburri, Architectural tactics for energy efficiency: Review of the literature and research roadmap, 54th Hawaii International Conference on System Sciences (2021) 7197 Url: http://hdl.handle.net/10125/71488.

[49] W. Wu, K. T., Safety tactics for software architecture design, Computer Software and Applications Conference (COMPSAC) (2004) 368–375Doi: 10.1109/CMPSAC.2004.1342860.

[50] H. Reza, D. Jurgens, J. White, J. Anderson, J. Peterson, An architectural design selection tool based on design tactics, scenarios and nonfunctional requirements, IEEE International Conference on Electro Information Technology (2005) 6–ppDoi: 10.1109/EIT.2005.1627052.

[51] N. B. Harrison, P. Avgeriou, Incorporating fault tolerance tactics in software architecture patterns, Proceedings of the 2008 RISE/EFTS Joint International Workshop on Software Engineering for Resilient Systems (2008) 9–18Doi: https://doi.org/10.1145/1479772.1479775.

[52] S. Kim, D. K. Kim, L. Lu, S. Y. Park, A tactic-based approach to embodying non-functional requirements into software architectures, 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC'08) (139-148), Doi: 10.1109/EDOC.2008.18.

[53] S. Kim, D. K. Kim, L. Lu, S. Park, Quality-driven architecture development using architectural tactics, Journal of Systems and Software 82 (8) (2009) 1211–1231, Doi: https://doi.org/10.1016/j.jss.2009.03.102.

[54] A. E. Hill, M. Nicholson, Safety tactics for reconfigurable process control devices, International Conference on System Safety 2009. Incorporating the SaRS Annual ConferenceDoi: 10.1049/cp.2009.1562.

[55] T. Marew, J. S. Lee, D. H. Bae, Tactics based approach for integrating non-functional requirements in object-oriented analysis and design, Journal of Systems and Software 82 (10) (2009) 1642–1656, Doi: https://doi.org/10.1016/j.jss.2009.03.032.

[56] J. Ryoo, P. Laplante, R. Kazman, A methodology for mining security tactics from security patterns, 43rd Hawaii International Conference on System Sciences (HICSS) (2010) 1–5Doi: 10.1109/HICSS.2010.18.

[57] N. B. Harrison, P. Avgeriou, U. Zdun, On the impact of fault tolerance tactics on architecture patterns, Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems (2010) 12–21Doi: https://doi.org/10.1145/2401736.2401738.

[58] A. Wyeth, C. Zhang, Formal specification of software architecture security tactics, International Conference on Software Engineering and Knowledge Engineering (2010) 172–175.

[59] S. Kim, D. K. Kim, S. Park, Tool support for quality-driven development of software architectures, Proceedings of the IEEE/ACM international conference on Automated software engineering (2010) 127–130Doi: 10.1145/1858996.1859018.

[60] S. H. Al-Daajeh, R. E. Al-Qutaish, F. Al-Qirem, Engineering dependability to embedded systems software via tactics.

[61] A. Koziolek, H. Koziolek, R. Reussner, Peropteryx: automated application of tactics in multi-objective software architecture optimization, Proceedings of the joint ACM SIGSOFT conference (QoSA) (2011) 33–42Doi: 10.1145/2000259.2000267.

[62] M. Mirakhorli, J. Cleland-Huang, Using tactic traceability information models to reduce the risk of

architectural degradation during system maintenance, 27th IEEE International Conference on Software Maintenance (ICSM) (2011) 123–132Doi: 10.1109/ICSM.2011.6080779.
[63] A. Sanchez, A. Aguiar, L. S. Barbosa, D. Riesco, Analysing tactics in architectural patterns, Software Engineering Workshop (SEW) (2012) 32–41Doi: 10.1109/SEW.2012.10.
[64] J. M. Cañete-Valdeón, Annotating problem diagrams with architectural tactics for reasoning on quality requirements, Information Processing Letters 112 (16) (2012) 656–661, Doi: https://doi.org/10.1016/j.ipl.2012.06.002.
[65] R. Kazman, P. Kruchten, Design approaches for taming complexity, Systems Conference (SysCon) (2012) 1–6Doi: 10.1109/SysCon.2012.6189488.
[66] J. Ryoo, P. Laplante, R. Kazman, Revising a security tactics hierarchy through decomposition, reclassification, and derivation, IEEE Sixth International Conference on Software Security and Reliability Companion (SERE-C) (2012) 85–91Doi: 10.1109/SERE-C.2012.18.
[67] E. B. Fernandez, H. Astudillo, Should we use tactics or patterns to build secure systems, First International Symposium on Software Architecture and Patterns, in conjunction with the 10th Latin American and Caribbean Conference for Engineering and Technology (2012) 8.
[68] M. Mirakhorli, P. Mäder, J. Cleland-Huang, Variability points and design pattern usage in architectural tactics, Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (2012) 52Doi: 10.1145/2393596.2393657.
[69] S. H. Al-Daajeh, R. E. Al-Qutaish, F. Al-Qirem, A tactic-based framework to evaluate the relationships between the software product quality attributes, International Journal of Software Engineering 5 (1) (2012) 5–26.
[70] M. Mirakhorli, Y. Shin, J. Cleland-Huang, M. Cinar, A tactic-centric approach for automating traceability of quality concerns, 34th International Conference on Software Engineering (ICSE) (2012) 639–649Doi: 10.1109/ICSE.2012.6227153.
[71] C. Preschern, Catalog of security tactics linked to common criteria requirements, Proceedings of the 19th Conference on Pattern Languages of Programs (2012) 7.
[72] M. Mirakhorli, J. Carvalho, J. Cleland-Huang, P. Mäder, A domain-centric approach for recommending architectural tactics to satisfy quality concerns, 3rd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks) (2013) 1–8Doi: https://doi.org/10.1109/TwinPeaks-2.2013.6617352.
[73] X. Qiu, L. Zhang, Providing support for specifying redundancy tactics using aspect-oriented modeling, 13th International Conference on Quality Software (2013) 183–186Doi: 10.1109/QSIC.2013.61.
[74] M. Mirakhorli, Preventing erosion of architectural tactics through their strategic implementation, preservation, and visualization, International Conference on Automated Software Engineering (ASE) (2013) 762–765Doi: 10.1109/ASE.2013.6693152.
[75] M. Kassab, G. El-Boussaidi, owards quantifying quality, tactics and architectural patterns interactions, SEKE 2013-January (2013) 441–446.
[76] C. Preschern, N. Kajtazovic, C. Kreiner, Catalog of safety tactics in the light of the iec 61508 safety lifecycle, Proceedings of VikingPLoP 2013 Conference (2013) 79.
[77] M. Mirakhorli, A. Fakhry, A. Grechko, M. Wieloch, J. Cleland-Huang, Archie: A tool for detecting, monitoring, and preserving architecturally significant code, 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineerin (2014) 739–742Doi: https://doi.org/10.1145/2635868.2661671.
[78] R. L. Nord, I. Ozkaya, P. Kruchten, Agile in distress: Architecture to the rescue, International Conference on Agile Software Development (2014) 43–57Doi: https://doi.org/10.1007/978-3-319-14358-3_5.
[79] X. Qiu, L. Zhang, Test scenario generation for reliability tactics from UML sequence diagram, 21st Asia-Pacific Software Engineering Conference 1 (11-18), Doi: 10.1109/APSEC.2014.11.
[80] S. Tahmasebipour, S. M. Babamir, Ranking of common architectural styles based on availability, security and performance quality attributes, Journal of computing and security 1 (2).
[81] J. Chavarriaga, C. A. Noguera, R. Casallas, V. Jonckers, Architectural tactics support in cloud com-

puting providers: the jelastic case, Proceedings of the 10th international ACM Sigsoft conference on Quality of software architectures (2014) 13–22Doi: 10.1145/2602576.2602580.

[82] G. Pedraza-Garcia, H. Astudillo, D. Correal, A methodological approach to apply security tactics in software architecture design, IEEE Colombian Conference on Communications and Computing (COLCOM) (2014) 1–8Doi: https://doi.org/10.1109/ColComCon.2014.6860432.

[83] X. Qiu, L. Zhang, Specifying redundancy tactics as crosscutting concerns using aspect-oriented modeling, Frontiers of Computer Science 8 (6) (2014) 977–995, Doi: 10.1007/s11704-014-3390-5.

[84] S. Bellomo, N. Ernst, R. Nord, R. Kazman, Toward design decisions to enable deployability: Empirical study of three projects reaching for the continuous delivery holy grail, 4th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2014) 702–707Doi: 10.1109/DSN.2014.104.

[85] S. N. Lee, D. Ko, S. Park, S. Kim, An approach to building domain architectures using domain component model and architectural tactics, International Journal of Engineering Systems Modelling and Simulation 6 (1-2) (2014) 54–61, Doi: https://doi.org/10.1504/IJESMS.2014.058424.

[86] R. Nöel, G. Pedraza-García, H. Astudillo, E. B. Fernández, An exploratory comparison of security patterns and tactics to harden systems, Proceedings of the 17th Ibero-American Conference Software Engineering (2014) 378–391.

[87] N. A. M. Alzahrani, D. C. Petriu, Modeling fault tolerance tactics with reusable aspects, 11th International ACM SIGSOFT conference on quality of software architectures (2015) 43–52Doi: https://doi.org/10.1145/2737182.2737189.

[88] S. Kim, A quantitative and knowledge–based approach to choosing security architectural tactics, International Journal of Ad Hoc and Ubiquitous Computing 18 (1-2) (2015) 45–53, Doi: https://doi.org/10.1504/IJAHUC.2015.067780.

[89] A. Alebrahim, S. Fassbender, M. Filipczyk, M. Goedicke, M. Heisel, Towards a reliable mapping between performance and security tactics, and architectural patterns, Proceedings of the 20th European Conference on Pattern Languages of Programs (2015) 39Doi: 10.1145/2855321.2855361.

[90] G. Lewis, P. Lago, A catalog of architectural tactics for cyber-foraging, 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA) (2015) 53–62Doi: https://doi.org/10.1145/2737182.2737188.

[91] M. B. Kjærgaard, M. Kuhrmann, On architectural qualities and tactics for mobile sensing, Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures (2015) 63–72Doi: 10.1145/2737182.2737196.

[92] A. E. Sabry, Decision model for software architectural tactics selection based on quality attributes requirements, Procedia Computer Science 65 (2015) 422–431, Doi: https://doi.org/10.1016/j.procs.2015.09.111.

[93] E. B. Fernandez, H. Astudillo, G. Pedraza-García, Revisiting architectural tactics for security, European Conference on Software Architecture (2015) 55–69Doi: https://doi.org/10.1007/978-3-319-23727-5_5.

[94] M. Mirakhorli, J. Cleland-Huang, Modifications, tweaks, and bug fixes in architectural tactics, IEEE/ACM 12th Working Conference on Mining Software Repositories (2015) 377–380Doi: 10.1109/MSR.2015.44.

[95] H. M. Chen, R. Kazman, S. Haziyev, V. Kropov, D. Chtchourov, Architectural support for devops in a neo-metropolis BDaaS platform, 34th Symposium on Reliable Distributed Systems Workshop (SRDSW) (2015) 25–30Doi: https://doi.org/10.1109/SRDSW.2015.14.

[96] J. Chavarriaga, C. Noguera, R. Casallas, V. Jonckers, Managing trade-offs among architectural tactics using feature models and feature-solution graphs, 10th Computing Colombian Conference (10CCC) (2015) 124–132Doi: 10.1109/ColumbianCC.2015.7333406.

[97] J. Ryoo, B. Malone, P. A. Laplante, P. Anand, The use of security tactics in open source software projects, IEEE Transactions on Reliability 65 (3) (2016) 1195–1204, Doi: 10.1109/TR.2015.2500367.

[98] S. Adam, A. Abran, The software architecture mapping framework for managing architectural knowledge, SEKE (2016) 357–362Doi: 10.18293/SEKE2016-183.

[99] D. E. Krutz, M. Mirakhorl, Architectural clones: toward tactical code reuse, Proceedings of the 31st Annual ACM Symposium on Applied Computing (2016) 1480–1485Doi: 10.1145/2851613.2851787.
[100] G. Pedraza-García, R. Noël, S. Matalonga, H. Astudillo, E. B. Fernandez, Mitigating security threats using tactics and patterns: a controlled experiment, Proccedings of the 10th European Conference on Software Architecture Workshops (2016) 37Doi: 10.1145/2993412.3007552.
[101] M. Kassab, G. Destefanis, Estimating development effort for software architectural tactics, International Andrei Ershov Memorial Conference on Perspectives of System Informatics (2015) 158–169.
[102] D. Gesvindr, B. Buhnova, Architectural tactics for the design of efficient PaaS cloud applications, 13th Working IEEE/IFIP Conference on Software Architecture (2016) 158–167Doi: 10.1109/WICSA.2016.42.
[103] M. Mirakhorli, J. Cleland-Huang, Detecting, tracing, and monitoring architectural tactics in code, IEEE Transactions on Software Engineering 42 (3) (2015) 205–220, Doi: 10.1109/TSE.2015.2479217.
[104] G. Márquez, H. Astudillo, Selecting components assemblies from non-functional requirements through tactics and scenarios, 35th International Conference of the Chilean Computer Science Society (SCCC) (2016) 1–11Doi: 10.1109/SCCC.2016.7836020.
[105] G. Márquez, Selection of software components from business objectives scenarios through architectural tactics, Proceedings of the 39th International Conference on Software Engineering Companion (2017) 441–444Doi: https://doi.org/10.1109/ICSE-C.2017.35.
[106] A. M. Alashqar, H. M. El-Bakry, A. A. Elfetouh, A framework for selecting architectural tactics using fuzzy measures, International Journal of Software Engineering and Knowledge Engineering 27 (3) (2017) 475–498, Doi: https://doi.org/10.1142/S0218194017500176.
[107] R. Gopalakrishnan, P. Sharma, M. Mirakhorli, M. Galster, Can latent topics in source code predict missing architectural tactics?, Proceedings of the 39th International Conference on Software Engineering (2017) 15–26Doi: 10.1109/ICSE.2017.10.
[108] J. C. Santos, A. Peruma, M. Mirakhorli, M. Galster, J. V. Vidal, A. Sejfia, Understanding software vulnerabilities related to architectural security tactics: An empirical investigation of chromium, php and thunderbird, International Conference on Software Architecture (ICSA) (2017) 69–78Doi: 10.1109/ICSA.2017.39.
[109] M. Salama, A. Shawish, R. Bahsoon, Dynamic modelling of tactics impact on the stability of self-aware cloud architectures, International Conference on Cloud Computing (CLOUD) (2016) 871–875Doi: 10.1109/CLOUD.2016.0126.
[110] M. A. Al Imran, S. P. Lee, M. M. Ahsan, Quality driven architectural solutions selection approach through measuring impact factors, International Conference on Electrical Engineering and Computer Science (ICECOS) (2017) 131–136Doi: 10.1109/ICECOS.2017.8167119.
[111] I. J. Mujhid, J. C. Santos, R. Gopalakrishnan, M. Mirakhorli, A search engine for finding and reusing architecturally significant code, Journal of Systems and Software 130 (2017) 81–93, Doi: https://doi.org/10.1016/j.jss.2016.11.034.
[112] F. Osses, G. Márquez, M. M. Villegas, C. Orellana, M. Visconti, H. Astudillo, Security tactics selection poker (TaSPeR) a card game to select security tactics to satisfy security requirements, 12th European Conference on Software Architecture: Companion Proceedings (2018) 1–7Doi: https://doi.org/10.1145/3241403.3241459.
[113] F. Alizadeh Moghaddam, G. Procaccianti, G. A. Lewis, P. Lago, Empirical validation of cyber-foraging architectural tactics for surrogate provisioning, Journal of Systems and Software 138 (2018) 37–51, Doi: https://doi.org/10.1016/j.jss.2017.11.047.
[114] J. Bogner, S. Wagner, A. Zimmermann, Using architectural modifiability tactics to examine evolution qualities of service-and microservice-based systems, SICS Software-Intensive Cyber-Physical Systems 34 (2) (2019) 141–149, Doi: https://doi.org/10.1007/s00450-019-00402-z.
[115] D. Gonzalez, F. Alhenaki, M. Mirakhorli, Architectural security weaknesses in industrial control systems (ICS) an empirical study based on disclosed software vulnerabilities, IEEE International Conference on Software Architecture (ICSA) (2019) 31–40Doi: 10.1109/ICSA.2019.00012.
[116] G. Lewis, P. Lago, S. Echeverría, P. Simoens, A tale of three systems: Case studies on the application

of architectural tactics for cyber-foraging, Future Generation Computer Systems 96 (2019) 119–147, Doi: https://doi.org/10.1016/j.future.2019.01.052.

[117] G. Márquez, M. M. Villegas, H. Astudillo, An empirical study of scalability frameworks in open source microservices-based systems, 37th International Conference of the Chilean Computer Science Society (SCCC)Doi: 10.1109/SCCC.2018.8705256.

[118] C. Orellana, M. M. Villegas, H. Astudillo, Mitigating security threats through the use of security tactics to design secure cyber-physical systems (CPS), 13th European Conference on Software Architecture 2 (2019) 109–115, Doi: https://doi.org/10.1145/3344948.3344994.

[119] F. Wessling, C. Ehmke, O. Meyer, V. Gruhn, Towards blockchain tactics: Building hybrid decentralized software architectures, IEEE International Conference on Software Architecture Companion (ICSA-C) (2019) 234–237Doi: 10.1109/ICSA-C.2019.00048.

[120] G. Márquez, H. Astudillo, Identifying availability tactics to support security architectural design of microservice-based systems, 13th European Conference on Software Architecture 2, Doi: https://doi.org/10.1145/3344948.3344996.

[121] C. Preschern, N. Kajtazovic, C. Kreiner, Safety architecture pattern system with security aspects, Transactions on Pattern Languages of Programming IV (2019) 22–75Doi: https://doi.org/10.1007/978-3-030-14291-9_2.

[122] J. C. Santos, K. Tarrit, A. Sejfia, M. Mirakhorli, M. Galster, An empirical study of tactical vulnerabilities, Journal of Systems and Software 149 (263-284), Doi: https://doi.org/10.1016/j.jss.2018.10.030.

[123] S. P. Nanda, H. Reza, Deriving scalability tactics for development of data-intensive systems, International Conference on Information Technology–New Generations (2020) 285–290Doi: https://doi.org/10.1007/978-3-030-43020-7_38.

[124] G. Márquez, Y. Lazo, H. Astudillo, Evaluating frameworks assemblies in microservices-based systems using imperfect information, IEEE International Conference on Software Architecture Companion (ICSA-C) (2020) 250–257Doi: 10.1109/ICSA-C50368.2020.00049.

[125] M. Alenezi, A. Agrawal, R. Kumar, R. A. Khan, Evaluating performance of web application security through a fuzzy based hybrid multi-criteria decision-making approach: Design tactics perspective, IEEE Access 8 (2020) 25543–25556, Doi: 25543-25556.

[126] A. Agrawal, A. H. Seh, A. Baz, H. Alhakami, W. Alhakami, M. Baz, R. Kumar, R. A. Khan, Software security estimation using the hybrid fuzzy ANP-TOPSIS approach: design tactics perspective, Symmetry 12 (4) (2020) 598, Doi: https://doi.org/10.3390/sym12040598.

[127] J. Keim, A. Kaplan, A. Koziolek, M. Mirakhorli, Does BERT understand code?–an exploratory study on the detection of architectural tactics in code, European Conference on Software Architecture (2020) 220–228Doi: 10.1007/978-3-030-58923-3_15.

[128] H. Milhem, M. Weiss, S. S. Some, Modeling and selecting frameworks in terms of patterns, tactics and system qualities, International Journal of Software Engineering and Knowledge Engineering 30 (11n12) (2020) 1819–1850, Doi: ttps://doi.org/10.1142/S021819402040032X.

[129] I. Malavolta, K. Chinnappan, S. Swanborn, G. A. Lewis, P. Lago, Mining the ROS ecosystem for green architectural tactics in robotics and an empirical evaluation, International Conference on Mining Software Repositories (MSR) (2021) 300–311)Doi: 10.1109/MSR52588.2021.00042.

[130] W. Yánez, R. Bahsoon, Y. Zhang, R. Kazman, Architecting internet of things systems with blockchain: A catalog of tactics, ACM Transactions on Software Engineering and Methodology (TOSEM) 30 (3) (2021) 1–46, Doi: https://doi.org/10.1145/3442412.

[131] T. Bi, P. Liang, A. Tang, X. Xia, Mining architecture tactics and quality attributes knowledge in stack overflow, Journal of Systems and Software 180 (2021) 111005, Doi: https://doi.org/10.1016/j.jss.2021.111005.

[132] P. H. D. Valle, L. Garcés, E. Y. Nakagawa, Architectural strategies for interoperability of software-intensive systems: practitioners' perspective, Annual ACM Symposium on Applied Computing (2021) 1399–1408Doi: https://doi.org/10.1145/3412841.3442015.

[133] S. H. Aldaajeh, S. Harous, S. Alrabaee, Fault-detection tactics for optimized embedded systems effi-

ciency, IEEE Access 9 (2021) 91328–91340, Doi: 10.1109/ACCESS.2021.3091617.

[134] A. Shokri, J. Santos, M. Mirakhorli, ArCode: Facilitating the use of application frameworks to implement tactics and patterns, arXiv preprint arXiv:2102.08372.

[135] K. Chinnappan, I. Malavolta, G. A. Lewis, M. Albonico, P. Lago, Architectural tactics for energy-aware robotics software: A preliminary study, European Conference on Software Architecture (2021) 164–171.