Self-Awareness in Software Engineering: A Systematic Literature Review

ABDESSALAM ELHABBASH, Lancaster University
MARIA SALAMA, University of Birmingham
RAMI BAHSOON, University of Birmingham
PETER TINO, University of Birmingham

Background: Self-awareness has been recently receiving attention in computing systems for enriching autonomous software systems operating in dynamic environments.

Objective: We aim to investigate the adoption of computational self-awareness concepts in autonomic software systems, and motivate future research directions on self-awareness and related problems.

Method: We conducted a systemic literature review to compile the studies related to the adoption of self-awareness in software engineering and explore how self-awareness is engineered and incorporated in software systems. From 865 studies, 74 studies have been selected as primary studies. We have analysed the studies from multiple perspectives, such as motivation, inspiration, and engineering approaches, among others.

Results: Results have shown that self-awareness has been used to enable self-adaptation in systems that exhibit uncertain and dynamic behaviour. Though the recent attempts to define and engineer self-awareness in software engineering, there is no consensus on the definition of self-awareness. Also, the distinction between self-aware and self-adaptive systems has not been systematically treated.

Conclusions: Our survey reveals that self-awareness for software systems is still a formative field and that there is growing attention to incorporate self-awareness for better reasoning about the adaptation decision in autonomic systems. Many pending issues and open problems, outlining possible research directions.

CCS Concepts: **•General and reference → Surveys and overviews; General literature; •Social and professional topics → Software selection and adaptation; •Software and its engineering → Software configuration management and version control systems;**

General Terms: Management, Performance, Reliability, Design

Additional Key Words and Phrases: Adaptation processes, self-properties, self-adaptive software, self-aware software, software architecture, survey, systematic literature review, research challenges

ACM Reference format:
Abdessalam Elhabbash, Maria Salama, Rami Bahsoon, and Peter Tino. 0. Self-Awareness in Software Engineering: A Systematic Literature Review. *ACM Trans. Autonom. Adapt. Syst.* 0, 0, Article 0 (0), 41 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Software systems have been exhibiting increased complexities due to the dynamism of the operating environments (e.g. the Cloud) and the uncertainties associated with service delivery. The ability

to manage and adapt such systems has gone beyond the capabilities of humans, e.g. administrators and/or designers. Self-adaptation emerged as a methodology to enable software systems to autonomously respond to the changing environment in order to maintain the required Quality of Service [33] [7] [9]. However, most self-adaptive systems lack the awareness about the implicit effects of the adaptation decisions taken at runtime [10], resulting in limited capabilities in facing the continuous changes and meeting the users' and system's quality requirements.

Recently, self-awareness has received more attention in computing systems. Over the past ten years, researchers have been proposing approaches for adopting self-awareness as an enabler for self-adaptation in dynamic environments. The purpose is to enrich the self-adaptation capabilities by gaining in-depth knowledge about the system and the environment's current and future states. The concept of self-awareness in computing was generally inspired by natural ecosystems, biological and human awareness. Yet, the the aspects of the concept widely varied when introduced in computing systems.

The concept of self-awareness in computing systems has been investigated, e.g. under the EU Proactive Initiative Self-Awareness in Autonomic Systems [13] and road-mapping agenda of the Dagstuhl seminar [27]. Such efforts aim to develop a fundamental understanding of what self-awareness in computing means and how it can be best approached. The field of self-awareness has received considerable attention, evidenced by the volume of publications, books, funding calls, seminars and workshops, etc. Despite the increasing attention to the field, the methodological aspects supporting the systematic engineering for this category of software systems necessitate a systematic effort to understand recent trends, approaches, advances and challenges that contributes to the development and evolution of the field, where software aspects and its engineering is fundamental.

Although the research on self-awareness in computing has been ongoing for a decade, there is still a lack of common agreement on the definition of computational self-awareness. A consensus on the definition of self-awareness in computing and clear characterisation of self-aware software systems have not been reached yet. Furthermore, to the best of our knowledge, despite the acceptance of self-awareness as an emerging paradigm in computing systems, no systematic studies have been performed to summarise and characterise approaches to self-awareness in computing and their contribution to the systematic engineering of self-adaptive systems, as well as to show in a unified framework how self-aware software systems are engineered and evaluated. Such gaps motivate this work.

In this paper, we conduct a systematic literature review to compile the studies related to the adoption of self-awareness in software engineering. A systematic literature review is a methodical process to build a body of knowledge on a certain subject or topic [26]. We focus on the studies that have an explicit claim of using self-awareness in software engineering. The review explores and investigates (i) how the studies have defined and characterised self-awareness, (ii) what inspired researchers when engineering self-awareness, (iii) what motivates researchers to use self-awareness, (iv) what software engineering practices and software paradigms have employed self-awareness, (v) how computational self-awareness is engineered to encode self-awareness properties within software systems, (vi) how the proposed self-aware systems are evaluated to quantify the accompanying benefits and overheads, and (vii) the real-world applications that are using self-awareness.

The remainder of this paper is organised as follows. The next section presents and compares related reviews. 3 briefly presents a description of the systematic review protocol adopted in this paper and the research questions. Section 4 presents an overview of the selected studies and the answers to the research questions. Section 5 summarises the main findings, as well as discusses the

impact and limitations of this review. In section 6, we outline different research challenges and open problems. Section 7 concludes the paper and outlines possible future works.

2 RELATED REVIEWS

In this section, we briefly present reviews on self-awareness in the literature. Table 1 shows a comparative summary of surveys on self-awareness in the literature. Abbreviations used in this table are: **Adh**: adhoc, **SLR**: systematic literature review, **SAd**: self-adaptation, **SAw**: self-awareness, **CAw**: context-awareness, **CSAw**: computational self-awareness, **N/A**: not applicable.

An early review has been conducted about self-awareness and its application in computing systems [30]. This work discussed previous efforts that incorporated self-awareness in different computing systems, such as pervasive computing. We have considered this study among the primary studies of this review and discussed its contributions within related research questions. Another early paper surveyed the articles published in the online Awareness Magazine [34]. In this paper, authors analysed the work on self-awareness using three aspects that are the level of awareness being addressed, the field of computer science in which the system has been developed, and the research themes being addressed. The awareness levels have been categorised into meta-self-awareness, self-awareness, consciousness and unconsciousness, where these levels are not strictly dependent.

Regarding the basic concepts of self-awareness, [29] discussed motivations and inspirations of self-awareness and presented an overview of some recent works that approached computational self-awareness. Focusing on specific software domains, authors in [22] focused on reviewing self-awareness approaches used in cloud computing. With respect to context-awareness, the survey conducted by Baldauf et al. [1] presented common principles and elements of context-aware software architectures. This paper then surveyed existing context-aware systems and identified the approaches and aspects of context-aware computing.

From the well-established field of self-adaptive software systems, examples of reviews include [33], [6] and [28]. Salehie et al. [33] presented a classical taxonomy for adaptation using the *when, what, how, where* questions. This survey discussed the definitions and properties of self-adaptation, as well as the techniques developed for self-adaptive systems. The book [6] has widely covered models and middleware for reasoning about self-adaptation, as well as engineering techniques for self-adaptive high-integrity and cloud software. In more details, the chapter [35] covered model-based reasoning of self-adaptive systems and its practices. Focusing on the cloud computing domain, the chapter [16] covered the assurances and evaluation of self-adaptation and employed cost-, time- and resources-awareness, while [5] focused on the emerging techniques for high-integrity (i.e. safety-critical) self-adaptive software. Authors in [28] surveyed engineering approaches for self-adaptive systems.

Giese et al. reviewed self-aware computing systems, focusing on reference architectures and architectural frameworks for building software systems that have similarities with computing systems [18]. Mahdavi-Hezavehi et al. [31] performed an SLR on methods for architecture-based self-adaptive systems. They focused on methods that handle multiple quality attributes. Their results indicate the need for awareness to achieve the level of improvement that the adaptation actions can guarantee. Other surveys focused on one of the self-* properties, such as self-protecting [39] and self-healing [15].

To the best of our knowledge, this is the first systematic literature review on the emerging field of computational self-awareness for software engineering.

Table 1. Comparative Summary of Self-Awareness Reviews in the Literature

Reference	Pub. Year	Methodology	No. of studies		Focus	Analysis Framework	Comparison of Self-Awareness						
			considered	analysed			definitions	motivations	inspirations	practises	approaches	evaluation	applications
[33]	[2009]	Adh	N/A	150	SAd	✓	✓	✓	×	×	×	×	✓
[30]	[2011]	Adh	N/A	28	SAw	×	✓	✓	✓	×	×	×	✓
[34]	[2012]	Adh	N/A	15	CAw	✓	×	×	×	✓	×	×	✓
[35]	[2013]	Adh	N/A	59	SAd	×	×	×	×	×	✓	×	✓
[16]	[2013]	Adh	N/A	32	SAd	×	×	×	×	✓	✓	×	✓
[5]	[2013]	Adh	N/A	59	SAd	✓	×	×	×	✓	×	✓	✓
[39]	[2014]	SLR	1037	107	SAd	✓	×	×	×	✓	×	×	×
[28]	[2015]	Adh	N/A	248	SAd	✓	✓	✓	×	✓	×	×	×
[22]	[2017]	Adh	N/A	79	CSAw	✓	×	✓	×	×	✓	×	✓
[29]	[2017]	Adh	N/A	64	CSAw	✓	✓	✓	×	×	×	×	✓
[18]	[2017c]	Adh	N/A	81	CSAw	×	×	×	×	✓	✓	×	×
[31]	[2017]	SLR	unknown	54	SAd	×	×	×	×	✓	✓	✓	×
This Paper	2019	SLR	865	74	CSAw	✓	✓(RQ1)	✓(RQ2)	✓(RQ3)	✓(RQ4)	✓(RQ5)	✓(RQ6)	✓(RQ7)

3 REVIEW PROTOCOL

In this section, we briefly describe the research method used to conduct this systematic review. The procedure of this study generally followed the guidelines for conducting systematic literature reviews [26]. The overall research objective of the review is to give an overview of the current state-of-the-art related to self-awareness in software engineering in research and practice. The research questions addressed by this study are:

- **RQ1.** How to define and characterise self-awareness?
- **RQ2.** What motivated the application of self-awareness in software engineering?
- **RQ3.** What are the sources of inspiration for its engineering?
- **RQ4.** In which software engineering practices and software paradigms is self-awareness employed?
- **RQ5.** What are the approaches for engineering self-awareness?
- **RQ6.** How are self-aware software systems are evaluated?
- **RQ7.** What are the working real-world applications adopting computational self-awareness?

RQ1 is motivated by the need for defining and consequently characterising self-awareness in software systems; i.e. how to consider that a system is self-aware. RQ2 aims to identify the motivations that stimulated the adoption of computational self-awareness in software engineering. The sources of inspiration for engineering self-awareness are identified in RQ3, in order to investigate how these sources helped to advance self-aware software systems. The goal of RQ4 is to find the software paradigms that employed self-awareness and to explore the characteristics of the environments that can benefit from computational self-awareness. RQ5 identifies the approaches of engineering self-aware software systems and investigate how computational self-awareness has been realised. The aim of RQ6 is to explore the significance of adopting self-awareness in terms of performance evaluation. RQ7 aims to identify the main real-world applications in which self-awareness is adopted.

In brief, the review process includes three main phases: (i) planning, (ii) conducting, and (iii) reporting the review. The first two phases are reported in details in Appendix A. In the remaining of this paper, we report the review results.

4 REPORTING THE REVIEW

In this section, we report the analysis results of the primary studies data (section 4.1), as well as the findings and answers to our research questions (section 4.2 - 4.7).

4.1 Analysis of the Primary Studies

This section describes the primary studies with respect to their publication types and year. Research groups that are active in the field of self-awareness are also presented.

4.1.1 Results of the Quality Assessment. All the primary studies were published in journals, conferences or seminal books that belong to well-established data sources in the software engineering community, as defined in the search strategy in section A.1.2. Most of the studies fulfil the criteria for quality assessment above average. These represent the degree of high quality and potential impact of the selected studies, and provide confidence in the overall quality of the systematic review. Details of the quality assessment results appear in section A.2.4

4.1.2 Publication Types. As shown in Figure 1, a significant number of studies were published in conference proceedings (38), followed by a smaller number of publications as book chapters (22) and journal articles (9). A limited number of technical reports (5) were published. As an observation, ideas and solutions are still being proposed in conferences, and some of them have matured and

reported through journals and books. This indicates that research in this area is still considered maturing. The presence of a number of technical reports reflects the transition between research and practice, as well as the technical work done in this area.

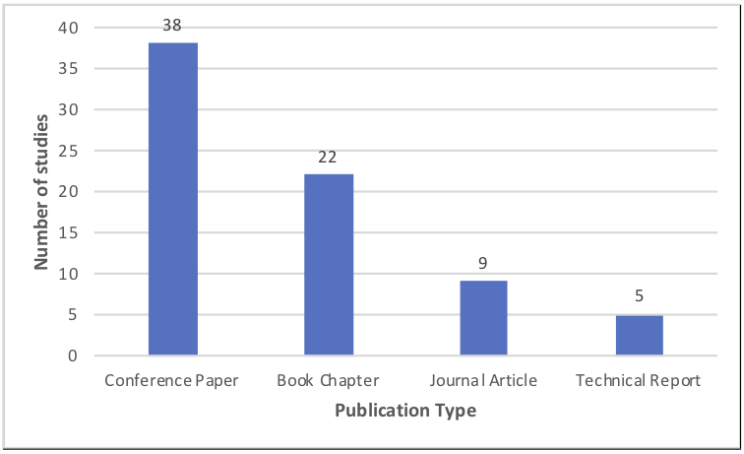

Fig. 1. Distribution of Primary Studies over Publication Types

4.1.3 Publication Years. Checking the distribution of publications over years as shown in Figure 2, it is noticed that the interest on self-awareness has started in 2005, with the exception of very few studies scattered over the years starting 1997. As defined in the search strategy in section A.1.2, we did not set filters on the publication year, yet the time frame of the studies reflects the time frame of interest and advancements in self-awareness. Following the year of 2005, the number of publications is increasing, though it is not a constant increase over the years. Note that the search process has covered only publications for the first quarter of 2018 (this can justify the low number of papers in 2018). The increase in the number of publications indicates that self-awareness has taken its place as the next property among self-* properties, as software systems are their self-adaptation capabilities are becoming more complex.

4.1.4 Active Research groups. To identify the active research groups within the area of self-awareness, we look at the author affiliations that appeared in the publications. Table 2 summarises the active research groups (with at least five publications in self-awareness) along with the number of publications. Publications are mostly dominated by University of Birmingham UK, Aston University UK, Vienna University of Technology Austria, University of Modena and Reggio Emilia Italy and Telécom Paris Tech France (note that we follow the author affiliations as appeared in the time of publication, and some studies appear multiple times under different affiliations).

Analysing the demographic distribution of the researchers by their affiliation countries, Figure 3 illustrates the distribution of this analysis. This shows that self-awareness research is receiving the highest attention in Germany, USA, UK and Italy.

4.2 Defining and Characterising Self-Awareness in Software Engineering (RQ1)

The concept of computational self-awareness is rapidly developing and many attempts to define this concept have been made. Although significant progress has been made in [S49] to define the concept, our observation is that there is still no general agreement on the definition. In the literature, some authors provided an explicit definition for self-awareness based on how they view this concept

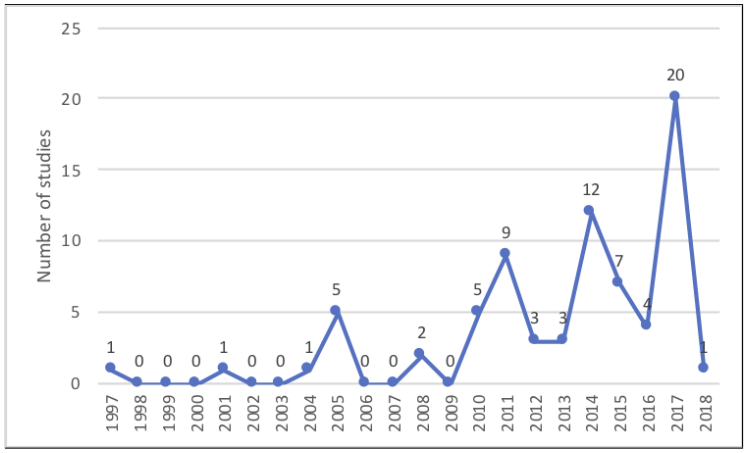

Fig. 2. Number of Publications per Year

Table 2. Active Research Groups in Self-Awareness

Affiliation	Number of studies
University of Birmingham, UK	11
Aston University, UK	8
Vienna University of Technology, Austria	7
Universit di Modena e Reggio Emilia, Italy	7
Telécom Paris Tech, France	7
University of Wrzburg, Germany	6
Karlsruhe Institute of Technology, Germany	5
University of Potsdam, Germany	5
Politecnico di Milano, Italy	5
Lund University, Sweden	5
TU Dresden, Germany	5

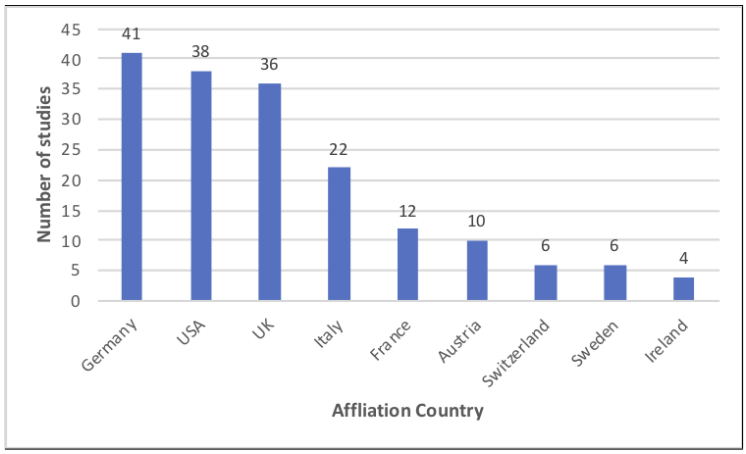

Fig. 3. Distribution of Publications by Affiliation Country

in software engineering or computation in general. Others used the term interchangeably with the term "self-adaptive", i.e. according to their view, a self-aware system is a self-adaptive system and vice versa. Table 3 lists the explicit definitions of self-awareness found in the primary studies.

Table 3. Definitions of Self-Awareness

Study	Definition
[S13]	Self-awareness is "*the ability of an element to autonomously detect deviations in its behaviour that are meaningful.*"
[S5]	Systems are self-aware systems "*if they have an information subsystem which generates an adaptive self-model of the system providing reference for identity check communications. In other words, self awareness implies the evolution of an information sub-system in the first place, and evolution of particular properties of this information subsystem.*"
[S55]	Self-awareness is "*information contained in a system about its global state that feeds back to adaptively control the system's low-level components.*"
[S1]	"*By self-awareness I am referring to an awareness of onefis own thought processes along with the insight that those thought processes can be captured, conceptualized, and named fi and when applied to software, externalized as code*"
[S62]	"*Self-aware computer systems will be able to configure, heal, optimize and protect themselves without the need for human intervention.*"
[S12]	A self-aware Cloud market is "*a market has the ability to change, adapt or even redesign its anatomy and/or the under pinning infrastructure during runtime in order to improve its performance.* "
[S57]	Self-Awareness within software systems as the following: "*To be Self-Aware a node (component of a software system) must contain total information about its internal state along with enough knowledge of its environment to determine the current state of the system as a whole. It may either be focused on its own state or the environments state at any time, but not both at once.*"
[S25]	"*By self-awareness, we mean the ability of each node in the Cloud infrastructure to monitor the level of compliance to SLAs associated with the tasks under its control.*"
[S73]	A component or an ensemble of components is self-aware if it is "*able to recognize the situations of their current operational context requiring self-adaptive actions.*"
[S68]	"*Awareness is a product of knowledge and monitoring.*"
[S3]	"*The SOTA model identifies an n-dimensional virtual-state space in which the execution of a system situates. In the SOTA space, a system is self-aware if it can autonomously recognize its current position and direction of movement in the space, and self-adaptation means that the system is able to dynamically direct its trajectory.*"
[S26]	A self-aware computational node is defined "*as one that possesses information about its internal state and has sufficient knowledge of its environment to determine how it is perceived by other parts of the system.*"
[S69]	A self-aware system is defined as "*a system has detailed knowledge about its own entities,current states, capacity and capabilities, physical connections and ownership relations with other (similar) systems in its environment.*"
[S39]	"*A system that can be called aware should be able to sense or store at least some information about its environment or itself.*"
[S27]	"*Self-awareness describes the ability of a system, to be aware of itself, i.e., to be able to monitor its resources, state, and behavior.*"

Besides the definitions listed in Table 3, some research works ([S53], [S58], [S46], and [S49]) intended to characterise a self-aware software system using a set of sub-properties:

According to [S53], "*To be self-aware a node must:*

- *Possess information about its internal state (private self-awareness).*

- *Possess sufficient knowledge of its environment to determine how it is perceived by other parts of the system (public self-awareness).*

Optionally, it might also:

- *Possess knowledge of its role or importance within the wider system.*
- *Possess knowledge about the likely effect of potential future actions / decisions.*
- *Possess historical knowledge.*
- *Select what is relevant knowledge and what is not.*"

The author of [S58] introduced various dimensions that need to be considered for developing a 'self-aware computer'. These dimensions include the senses that the self-aware computer may have, the ability of a self-aware computer to recognise itself as a computer, and the ability to perform some aspect of performing some sort of self-assessment and self-healing.

The authors of [S40] provide a black-box definition of a self-aware system which is based on relativity to other systems assumed to be identified as self-aware systems. Their approach is to consider a system S as self-aware if it is at least as good as the performance of a self-aware system S_i based on some measure. The authors do not discuss how the system S_i can be classified as a self-aware system, which is a requirement for identifying other systems as self-aware ones.

According to [S46], self-awareness is considered by "*the combination of three properties that a system should possess:*

- *Self-reflective: Aware of its software architecture, execution environment, and hardware infrastructure on which it is running as well as of its operational goals (e.g., QoS requirements, cost- and energy-efficiency targets),*
- *Self-predictive: Able to predict the effect of dynamic changes (e.g., changing service workloads) as well as predict the effect of possible adaptation actions (e.g., changing system configuration, adding/removing resources),*
- *Self-adaptive: Proactively adapting as the environment evolves in order to ensure that its operational goals are continuously met.*"

This definition has been modified in [S49] as follow:
"*Self-aware computing systems are computing systems that:*

(1) *learn models capturing knowledge about themselves and their environment (such as their structure, design, state, possible actions, and run-time behavior) on an ongoing basis and*

(2) *reason using the models (for example predict, analyze, consider, plan) enabling them to act based on their knowledge and reasoning (for example explore, explain, report, suggest, self-adapt, or impact their environment) in accordance with higher-level goals, which may also be subject to change.*"

We extracted the different aspects and characteristics of self-awareness from the definitions cited above. We, then, analysed these definitions to show how each of the definitions found in the primary studies characterises self-awareness and how comprehensive they are. The characteristics are defined as follows:

- *Domain-specific:* determines whether the definition is restricted to the problem domain or is general to cross-cut different domains.
- *Behaviour:* determines whether the definition considers the behaviour of the system implicitly or explicitly. This is to capture the extent to which the definition is consistent with the expectation of self-aware systems to adapt their behaviour at runtime according to their context.
- *Knowledge:* determines the aspects of the definition regarding the treatment of the knowledge, i.e. at a coarse- or fine-grained level. Fine-grained knowledge treatment means that the

knowledge is structured into levels which allow for different level of adaptation. On the contrary, coarse-grained knowledge treatment does not consider such structure. This is to capture the extent to which the self-aware system is expected to acquire detailed knowledge to reason about the different adaptation decisions.
- *Internal State:* determines how the definition considers modelling the internal state of the system; implicitly or explicitly.
- *Environment:* determines how the definition considers modelling the environment state of the system; implicitly or explicitly.
- *Adaptation time:* determines what type of adaptation is supported by the self-aware system (according to the definition) in terms of the time to perform the adaptation; reactive (after the incident is detected) or proactive (before the incident occurs).

Table 4 shows the definitions analysis. It is worth noting here that the absence (declared using '-') of any of the characteristics in this table does not mean that the corresponding work does not support that characteristic. It means that the definition or characterisation of self-awareness in that work does not explicitly mention that characteristic, i.e. in such cases, there may exist inconsistency between the work and the definition.

Table 4. Analysis of Self-Awareness Definitions

Study	Aspects of Self-Awareness					
	Domain	Behaviour	Knowledge	Internal State	Environment	Adaptation time
[S13]	General	Explicit	Coarse	Explicit	-	-
[S5]	General	Implicit	Coarse	Implicit	-	-
[S55]	General	Implicit	Coarse	Explicit	-	-
[S1]	General	Implicit	Coarse	Implicit	-	-
[S62]	General	Implicit	Coarse	Implicit	-	-
[S12]	Cloud	Implicit	Coarse	Explicit	Explicit	-
[S57]	General	Implicit	Coarse	Explicit	Explicit	-
[S25]	Cloud	Implicit	Coarse	Implicit	-	Reactive
[S73]	General	Implicit	Coarse	Implicit	-	-
[S68]	General	Implicit	Coarse	Implicit	Implicit	-
[S3]	General	Implicit	Coarse	Implicit	-	-
[S26]	General	Implicit	Coarse	Explicit	Explicit	-
[S69]	General	Implicit	Coarse	Explicit	Explicit	-
[S39]	General	Implicit	Coarse	Explicit	Explicit	-
[S53]	General	Implicit	Fine	Implicit	Explicit	-
[S27]	General	Explicit	Coarse	Explicit	-	-
[S58]	General	Implicit	Coarse	Explicit	-	-
[S46]	General	Explicit	Coarse	Explicit	Explicit	Proactive
[S49]	General	Explicit	Coarse	Explicit	Explicit	Proactive

It is notable from the surveyed literature that there is no consensus on a definition that covers all aspects of self-awareness. It is also worth to note that there are no clear borderlines between self-aware and self-adaptive systems. Based on the studies considered in this review, most of the researchers use the two terms interchangeably. Among the studies considered in this review, the works of [S53] and [S46] are the only works that have clearly differentiated between the two terms. Both of them view self-awareness an enabler for self-adaptive systems. The former considers a

self-adaptive system to be self-aware if the system defines multi-levels of knowledge modelling and representation and correspondingly supports different levels of self-adaptation. The latter considers the self-adaptive system to be self-aware if the system supports proactive adaptation.

4.3 Motivations for Employing Self-Awareness (RQ2)

This question looks at the motivation that derived the studies in employing self-awareness. The majority of studies have clearly identified the motivation behind having self-awareness as a capability in software systems. Extracted from all studies, we have found that the general motivation that has directed researchers towards self-awareness is the complexity, heterogeneity, large size of modern software systems, evolving functionality and quality requirements during run-time, emergent behaviours, and unpredictable changes of the highly dynamic operating environment [S7] [S20] [S26] [S17] [S39].

More specifically, the motivation of employing self-awareness in software systems varied between a general one related to realising better autonomy for software systems, and others that are more specific. With respect to the former, researchers considered self-awareness for: (i) reasoning and engineering better adaptations with guaranteed functionalities and quality of service during runtime [S5] [S63] [S36] [S53] [S3] [S7] [S2] [S71] [S19] [S33] [S20] [S39] [S59] [S31], (ii) managing complex systems without human intervention [S21] [S38] [S15], (iii) dealing with real-world situations, operational contexts and dynamic environments of modern software systems to respond to such fluctuating environment and associated uncertainty [S70] [S7] [S73] [S10] [S40] [S17] [S39] [S60] [S28] [S42] [S50], (iv) managing complex trade-offs arising from adaptation due to conflicting goals [S37] and the heterogeneity of the system [S26] [S66], and (v) realising intelligent software systems with sophisticated abilities [S55] [S53] [S67] [S68] [S58] [S52] [S51].

Specific motivations varied between domain-specific according to the software paradigm (e.g. ubiquitous applications, pervasive services, cloud-based services, mobile computing) and others driven by software engineering practices (e.g. formal specification, performance management, data access, security). Table 5 summarises these motivations.

4.4 Sources of Inspiration for Self-Awareness (RQ3)

This question looks at what inspired the self-awareness engineering process. Unlike the case of motivation, few of the studies have clearly identified their source of inspiration in engineering self-awareness. Generally, nature and sciences inspired by nature are the main sources of inspiration in all studies.

The critical features of the systems for the presence of self-awareness for each source of inspiration as the following. Biological systems posses (i)memories, which store knowledge, (ii)information subsystem (which processes memories),(iii) and adaptive self-model (which is used for referencing memory communications). Human beings inspiration source is similar where humans have memories (written or recorded contents) and communication systems through which those contents are disseminated; making the relevant community aware of the memories. Natural ecosystems have sensing capabilities, communication means, and decision-making mechanisms. For example in the ants' colony case, ants interact by depositing and locally smelling (sensing) pheromones. These pheromones form a structure that is used by the ants to react (make decisions) based on the shape of such structures. Control theory is based on equipping systems with feedback loops that acquire knowledge about the system behaviour, analyse that behaviour, plan and react to changes. In psychology, self-awareness is classified as private or public. Private is related to knowledge concerning the individual itself whereas public self-awareness is related to knowledge about the environment of the individual. Cognitive science defines different levels of self-awareness including

Table 5. Specific Motivations of using Self-Awareness

Study	Motivation
Driven by Software Paradigm	
[S41]	Autonomous adaptations of hardware/software functionalities in ubiquitous computing applications to meet the dynamic requirements of various environmental situations and provide better QoS
[S12]	Creating cloud markets platforms with self-* properties harmoniously working together in order to be capable to adapt effectively to dynamic changes in user requirements, services, and variability in resources.
[S74]	Modelling integrated pervasive services and their execution environments, in a way that diverse issues of context-awareness, dependability, openness, flexible and robust evolution, can be addressed
[S16]	The need for runtime self-adaptive interactions between pervasive computing services
[S64]	Achieving parallelism within a reasonable cost and time range for data streaming applications operating in distributed environments
[S4]	Acceleration and efficiency of bio-collections' information extraction, while keeping the quality of the results similar to what capable humans can provide
[S66]	The limitations of the security measures on mobile devices, and the lack of cooperation between different security solutions running on the same device
Driven by Engineering Practices	
[S34]	The motivation of including the notion of self within object-oriented formal specification languages is to facilitate reasoning about object interaction.
[S13]	The detection anomalies in the functioning of internet-based services and fault localisation (i.e., locating the responsible sub-services) is easier if service elements are aware of their own health status, determined by whether the current observed behaviour is consistent with expectations.
[S9]	The need to access distributed and dynamic high-dimensional data about resources heterogeneity in a timely fashion in large, decentralised, resource-sharing environments
[S1]	The invention of new abstractions as conceptualisation necessary to determine the behaviour of software needed by users and the implementation details.
[S56]	Enabling change at run-time for evolution purposes
[S47], [S45], [S46]	The need to predict the performance of running services at run-time and related resources management
[S62]	Balancing resources usage in order to improve performance, utilisation, reliability and programmability
[S17]	Solving problems caused by QoS interference in shared resources environment to achieve auto-scaling for cloud-based services
[S28], [S50]	Dynamic context management
[S48]	The complexity of managing end-to-end application performance
[S42]	Performance prediction that is necessary for efficient resource management
[S51]	The need to re-arrange own knowledge structures for compactness and efficiency to survive for long periods in a demanding environment
[S66]	The limitations of the security measures on mobile devices, and the lack of cooperation between different security solutions running on the same device

awareness about basic events, minimal awareness of interaction with the environment, and time of interaction. It also includes advanced self-awareness which includes the previous in addition to

private thoughts and feelings and the ability to construct complex behaviours based on the other levels [30].

Examples of nature's inspiration include: biological systems [S5] [S55] [S53], natural ecosystems [S62] [S74] [S16] and human beings [S5] [S1] [S67]. Sciences inspiring self-awareness are control theory [S36], psychology [S57] [S7] [S26] [S18] [S28] [S50], and cognitive science [S53]. Table 6 summarises inspirations cited in primary studies.

These inspirations had led to engineering approaches that mimic models and architectures for computational self-awareness. For example, the inspiration from the control theory inspired designing architectures for collecting data about the system performance and using that data to build models that represent the system state thus achieving self-awareness. Another example is the approaches inspired by cognitive science attempt to build different models representing the different levels of self-awareness. This leads to having various approaches of engineering self-awareness, each with different focus and capabilities and thus each having partial self-awareness (if we consider that all the different focuses together constitute computational self-awareness). Therefore, a holistic self-aware system may need to consolidate those various focuses - an open challenge that is worthy of future research.

Table 6. Source of inspiration for engineering Self-Awareness

Study	Inspiration
Inspiration from Nature	
[S5]	biological cell and the system of a human organisation (e.g., a company or government department)
[S55]	biological systems: the immune system and ant colonies
[S1]	human beings
[S62]	biological organic nature
[S67]	human wisdom
[S74], [S16]	natural ecosystems
Inspiration from Sciences	
[S36]	Control Theory
[S53], [S52]	Biology and cognitive science
[S57], [S26], [S18]	Psychology
[S7]	Psychology, philosophy and medicine
[S28], [S50]	Psychology and philosophy

Within the studies mentioning their source of inspiration, we have also found that the majority of studies named only their source of inspiration. More details, albeit in an abstract form, on about how self-awareness approaches are inspired by nature or sciences are found in few numbers of studies; such as [S5] [S55] [S67] [S57] [S7] [S16]. The exception that could be found is [S53], where the authors have explicitly mentioned how self-awareness have been inspired by biology and cognitive science. The mapping between the source of inspiration and the research work conducted in the study is expected to be clearly communicated. Further, studies investigating how self-awareness could be inspired by nature and other sciences can help to advance self-aware software systems.

4.5 Software Paradigms and Engineering Practices Employing Self-Awareness (RQ4)

This research question explores the software paradigms and the software engineering practices that employed self-awareness.

Table 7 lists the software paradigms found in the primary studies and related studies. Figure 4 shows the distribution of studies by software paradigms (note that some studies appear multiple

times under different categories, which interprets the total number of studies appearing in the figure is greater than the number of primary studies). The majority of studies considered self-awareness for autonomic computing, i.e. engineering self-adaptive software systems as a general software paradigm, not explicitly designed for a particular paradigm or application type. Service-oriented systems and cloud-based services also received attention in a good number of studies, and less attention to ubiquitous and pervasive computing and distributed systems. Within distributed systems, some studies considered a certain type of applications operating in decentralised environments, such as artificial intelligence systems [S55], distributed smart cameras [S7] [S18]. Single works focused on software-intensive systems [S68], stream programming [S64], mobile computing [S66] and Internet of Things [S65].

Table 7. Software Paradigms employing Self-Awareness

Software Paradigms	**Studies**
Self-adaptive Systems	[S22], [S5], [S63], [S1], [S56], [S36], [S62], [S37], [S67], [S3], [S7], [S2], [S38], [S19], [S33], [S20], [S39], [S69], [S27], [S48], [S6], [S8], [S11], [S24], [S32], [S30], [S35], [S42], [S44], [S43], [S49], [S51], [S54], [S72]
Service-oriented Systems	[S13], [S21], [S47], [S73], [S10],[S14] [S23], [S46], [S59], [S15], [S50]
Cloud-based Services	[S45], [S12], [S25], [S7], [S26], [S18], [S17], [S60], [S61], [S65]
Distributed Systems	[S9], [S55], [S7], [S71], [S18]
Ubiquitous and Pervasive Computing	[S70], [S41], [S53], [S52], [S74], [S16], [S29]
Software-Intensive Systems	[S68]
Stream Programming	[S64]
Mobile Computing	[S66]
Internet of Things	[S65]

The observation that the majority of the proposed work tends to be generic and not explicitly designed for a particular paradigm or application type implies that generality can come with advantages and disadvantages. Generality can imply application and evaluation of the proposed work under different contexts and applications, reflection on their strengths and weaknesses in dealing with the said paradigm. This can consequently provide inputs for further improvements and extensions. On the other hand, employing self-awareness can take simplistic assumptions, or tend to be limited when addressing the requirements of some paradigms, where speciality and customisation are desirable for more effective adaptations. Self-awareness that considers characteristics of particular software paradigms will result in advancing these paradigms. Yet, the validity of these observations can be subject to further empirical studies.

With respect to the software engineering practices that employed self-awareness, Figure 5 shows the number of studies by engineering practices (also note that some studies appear multiple times under different categories, which explains why the total number of studies appearing in the figure is greater than the number of primary studies). Table 8 lists the practices and related primary studies. We considered the studies that reported on reviews and roadmaps for self-awareness separately (10 studies). We have found that architecture design (18 studies) is the practice that most employed self-awareness, as well as system design (14 studies) and engineering adaptations (8 studies). A number of studies also employed self-awareness for QoS resources management (with some explicitly focusing on performance), system specification (including formal methods), as well as knowledge representation and reasoning. Operation management during runtime and service composition also received some attention. Single research efforts considered various practices, such

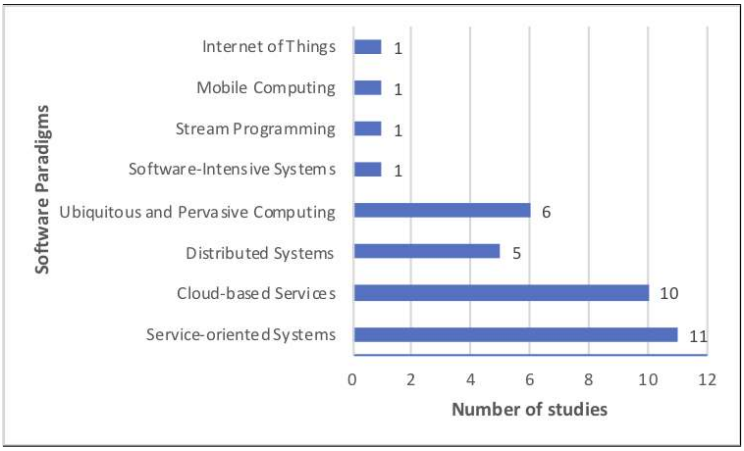

Fig. 4. Distribution of studies by software paradigms

as system development for stream programming [S64] and language semantics for Object-Z [S34], security design and information extraction. These studies are domain-specific, which interprets their minimal number.

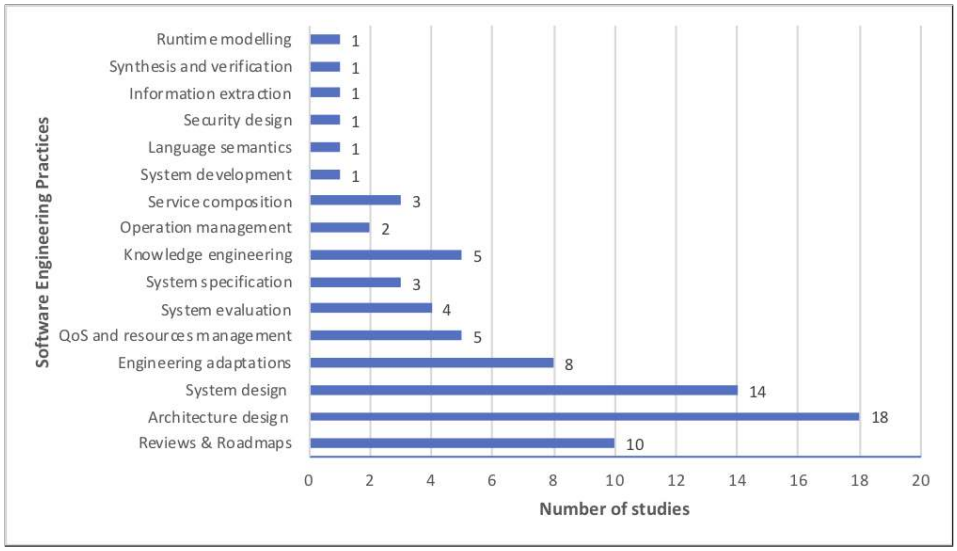

Fig. 5. Distribution of studies by engineering practices

Meanwhile, some studies investigated the concept of self-awareness in software engineering. For instance, the works of [S63] [S57] [S53] [S73] reviewed the concept of self-awareness and its applications in computing systems. Other studies presented roadmaps for realising self-awareness in software systems [S5], developing sustainable systems [S21] and software-intensive systems [S68], as well as realising service composition [S73] and enabling change and evolution in development platforms and tools that are used for incremental and iterative development [S56]. The work of [S62] discussed related technologies for enabling self-awareness. Though these studies did not

Table 8. Engineering Practices employing Self-Awareness

Engineering Practices	Studies
Reviews & Roadmaps	[S5], [S63], [S56], [S21], [S62], [S53], [S68], [S11], [S31], [S43]
Architecture design	[S13], [S22], [S41], [S36], [S3], [S2], [S10], [S26], [S18], [S60], [S28], [S61], [S32], [S30], [S42], [S6], [S72], [S65]
System design	[S70], [S55], [S33], [S19], [S46], [S48], [S28], [S27], [S29], [S42], [S49], [S50], [S54], [S72]
Engineering adaptations	[S56], [S37], [S7], [S71], [S52], [S16], [S39], [S43]
QoS and resources management	[S47], [S25], [S45], [S38], [S17]
System evaluation	[S24], [S35], [S44], [S51]
System specification	[S1], [S20], [S59]
Knowledge engineering	[S9], [S67], [S69], [S58], [S40]
Operation management	[S12], [S74]
Service composition	[S73], [S23], [S14]
System development	[S64]
Language semantics	[S34]
Security design	[S66]
Information extraction	[S4]
Synthesis and verification	[S15]
Runtime modelling	[S8]

explicitly considered certain engineering practices, yet they are contributing to formalising the concept of self-awareness and guiding the research community.

4.6 Approaches for Engineering Self-Awareness (RQ5)

Engineering self-awareness aims for encoding self-aware properties within the software systems in an attempt to provide systematic treatment for managing the software system state, knowledge and execution environment. This section provides details about how self-awareness has been engineered in self-aware software systems and categorises the engineering approaches.

In literature, different approaches for engineering self-awareness in software engineering are found. On one hand, we have observed that 23 out of the 74 primary studies did not provide any engineering approaches for self-awareness in software engineering. These works have presented visions, outlined challenges, and raised questions. On the other hand, the remaining 51 studies claimed to provide engineering approaches for self-awareness. We have categorised these approaches into model-driven, architecture-centric, programming-driven, knowledge-centric, and development lifecycle-based approaches. Table 9 lists the engineering approaches categories and their related studies.

Figure 6 shows the distribution of studies with respect to the classification of engineering approaches. Architecture-centric and model-driven approaches are found the most dominant approaches in the current literature. Other categories of approaches have taken less attention in the research community.

4.6.1 Model-driven approaches. The model-driven approaches attempt to create abstract models that represent the software system and its execution environment [14]. Environments are characterised by dynamic behaviours and a demanding need for self-adaptation, e.g. the self-aware

Table 9. Engineering Approaches and Related Studies

Engineering Approach	Studies
Model-driven	[S47], [S12], [S45], [S3], [S46], [S39], [S59], [S28], [S27], [S48], [S4], [S8], [S15], [S29], [S50], [S51], [S42]
Architecture-centric	[S13], [S22], [S70], [S36], [S41], [S62], [S25], [S37], [S73], [S74], [S7], [S2], [S38], [S10], [S18], [S19], [S23], [S26], [S33], [S16], [S17], [S60], [S61], [S6], [S42], [S32], [S30], [S66], [S65]
Programming-driven	[S34], [S64]
Knowledge-centric	[S9], [S67], [S68], [S69], [S20]
Development lifecycle-based	[S71]

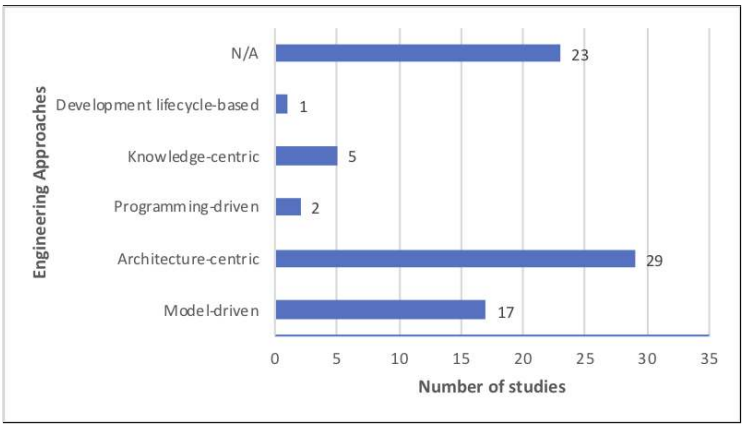

Fig. 6. Distribution of studies by engineering approaches

systems environments. This calls for runtime model-driven approaches [2] which capture the runtime system state, in order to help the system to decide when and how to adapt to accommodate changes.

In the literature of self-aware software systems, few numbers of model-driven approaches have been proposed. In [S12], the authors have proposed a model-driven monitoring methodology to enable self-awareness in cloud platforms. The methodology presents a model for mapping the low-level metrics to the cloud market goals, in order to evaluate the performance of the goals. The work of [S59] has presented a model for expressing self-adaptive behaviour of service-oriented applications using the SCA-ASM modelling language [32]. This extension of the SCA-ASM offers mechanisms to monitor the environment and the system itself and to perform adaptation actions. In [S39], authors have introduced a novel graphical language, namely, "Extended Behavior Trees (XBTs)", for modelling adaptive and self-aware agents. The approach introduces a new combined Q-learning strategy that allows the interleaving of reasoning, learning and actions.

The works of Kounev et al. [S47], [S45], [S48], and [S42] have taken the model-driven self-aware systems a step further by introducing the dynamic performance models as a 'mind' that controls a self-aware system. Such models enable predicting changes in the system workload and the execution environments leading to proactively adapting the system in order to avoid the violation of the requirements. In [S46], the authors designed the Descartes Modelling Language (DML) as a tool for modelling QoS and resource management aspects of self-aware systems.

The works of [S29] [S28] [S28] [S50] developed a service-based solution built upon the Models@Runtime notion to enable self-awareness for context management. The solution reduces self-awareness to a set of components (services) that collects various contextual information.

[S15] introduces formal verification techniques to enable self-awareness. The techniques use Markov chains and Markov decision processes to detect the violation of and establish QoS properties.

In the context of self-awareness for self-modelling, Landauer [S51] proposed a conceptual architecture defines a set of activities for self-aware experiments in self-modelling systems. In [S8] the same authors define some methods to model aspects of the system's internal behaviour and operational environment.

In [S4] self-awareness is adopted for information extraction from digitized bio-collections. The idea is to develop self-aware methods that digitise bio-collection and assess the quality of their output based on an acceptance method. The process can be considered as a direct application of the MAPE-K framework for information extraction.

4.6.2 Architecture-centric approaches. The architecture-centric approaches introduce reference architectures for representing the system's design decisions and constraints [17]. In general, such approaches have similar design trends, basically the design of agent control loops, (e.g. the MAPE-K [24]). They consist of the following five components (which are [21]: (i) Monitoring, Observing or Sensing, which collects the knowledge about the system state and the environment state, (ii) Knowledge, which accumulates and represents the information gathered by the monitoring components, (iii) Analysis or Evaluation, which processes the acquired knowledge to assess the system goals and to report the need for adaptation when required, (iv) Plan, which processes the adaptation reports, inspects possible adaptation actions, and selects the optimal one, and (v) Execute, Act, or Self-express, which applies the adaptation actions.

The architectural framework proposed by [S18] brings forward the MAPE-K by extending the knowledge modelling component, to enrich the self-adaptation capabilities by adopting the computational self-awareness principles. Inspired from Psychology and based on self-awareness types introduced in [S52], the architecture introduces five levels of self-awareness:

- *Stimulus-awareness*: This level is related to the knowledge about the basic events affecting the system. It does not support any ability of learning or prediction. Hence, it provides knowledge for basic levels of adaptation, e.g. replacing failed services in SOA-based application.
- *Goal-awareness*: This level models the knowledge about the system's goals and objectives and the extent to which the goals are being achieved.
- *Interaction-awareness*: This level is able to model the knowledge about the interactions among the different systems components and the interactions of the system with the environment. This enables to anticipate of how adaptations decisions can affect the interacted components and the environment.
- *Time-awareness*: This level enables modelling the knowledge about the past performance of the different system components, e.g. the historical performance of the services in an SOA-based application; which provides useful input for the services selection and composition.
- *Meta-self-awareness*: This level acts as a cognitive system that reasons about the adoption of any of the other levels of awareness, based on the benefits and overhead of each of them. At runtime and using real data, this level analyses and predicts the benefits and the overhead of each awareness levels and decides which of them to be adopted for the adaptation decision making.

Also, different architecture-centric approaches have been proposed to monitor the system and environment states to reason about the autonomous adaptation decisions at different levels; the

infrastructure level, the architecture level, or the application level. The work of [S17] is an example of self-awareness in cloud computing at the infrastructure level, where the self-awareness is used in the process of auto-scaling the hardware resources in the cloud-based on the changes in the workload. Introducing a set of quality-driven architectural patterns [S61], one of the patterns, namely the Meta-self-awareness pattern is an example of using self-awareness to adapt at the architectural level, where an architecture adaptation manager manages the trade-offs between different QoS requirements to switch between different architectural patterns. The work of [S37] is an example of a self-aware architecture-centric approach for adaptation at the application level, where the approach presents an architectural framework that enables automatic scheduling of adaptation actions to react to the changes and fluctuations in the available resources.

The works in [S32] [S32] proposed general architectural notations to describe the self-aware system architecture. The notation can be used to conceptually describe the system components, the links between them, and the data flow. The work of [S61] proposed a markovian-based analytical model for dynamically assessing the impact of architectural strategies on the stability of the quality attributes of self-Aware Cloud architectures during runtime. In [S6] a methodology and high-level suggestions are presented to analyse and increase the level of self-awareness in various computing architectures. In [S66] a conceptual architecture for self-aware for mobile devices security is proposed. The architecture is basically to monitor mobile device operations, detect malicious behaviour, and perform an action to adapt. All of the above works provide contributions at the conceptual level which needs further research to be realised.

4.6.3 Programming-driven approaches. Self-awareness has been rarely incorporated in an explicit way to propose self-aware programming paradigms.

The work of [S64] proposed the inclusion of self-awareness in stream programming model in which stream data arrive continuously and change dynamically in rate or content due to the changes in computing resources or communication infrastructure. The proposed model, called StreamAware, enables dynamic and automatic task rescheduling, as well as data parallelism in response to the changes of the stream data.

The work of [S34] has proposed the inclusion of the notion of 'self' in object-oriented formal specification languages, in order to express the awareness by an object of its own identity. This results in "self-aware" objects which support the reasoning about object interaction in object-oriented programming paradigm.

4.6.4 Knowledge-centric approaches. Knowledge representation is a key activity towards achieving self-aware systems. It enables modelling the acquired knowledge (whether it is related to the internal system state or the system environment), which is required to reason about the adaptation decision making.

Few approaches have been proposed for the knowledge representation in self-aware software systems. [S9] proposed a multi-dimensional access structure, called "Heterogeneity-Aware Distributed Access Structure (HADAS)", that can be used in self-aware systems to make the system's nodes self-aware by storing reflective information about its own state, such as processing power, storage, etc. In [S67], the authors introduced an abstract approach for knowledge representation to show that knowledge can be represented by rule-based models, frames, semantic networks, concept maps, ontologies and logic. Following this work, an approach for implementing self-awareness based on the KnowLang framework [36] was later proposed in [S68] and [S69]. The framework provides a knowledge base that abstracts some context and a reasoner that allows for knowledge access in that context. In [S20], the authors have introduced the SCEL (Software Component Ensemble Language), that is an approach for providing linguistic abstractions for describing the

behaviour and knowledge of self-aware systems taking into consideration the evolution of the system ensembles and interactions among them. These works address knowledge representation at a coarse-grain level. They provide less focus on the potential structure of the collected knowledge and do not explicitly deal with dynamic knowledge management in self-adaptive systems, which have the effect of limiting the effectiveness of the adaptation.

4.6.5 Development lifecycle-based approaches. [S71] proposed a general software development life-cycle to engineer self-adaptive systems. The approach is based on the decomposition of a complex system into service components. The local awareness of a component informs local adaptive behaviour. Then, collective awareness is achieved by grouping the inter-related elementary components into ensembles to enable communication and knowledge exchange.

4.7 Evaluation of Self-Aware Software Systems (RQ6)

This research question investigates the approaches that have been used to evaluate the proposed self-aware approaches. The question also looks for the evaluation criteria, reported performance and overhead of the approaches.

4.7.1 Evaluation Approaches. We observed that 28 papers out of the 39 (that proposed engineering approaches) have provided some kind of evaluation for their approaches. We categorise these approaches into the following categories: analysis-, illustrative example-, illustrative application-, and simulation-based evaluation. Table 10 lists the evaluation approaches categories and their related studies.

Table 10. Evaluation Approaches and Related Studies

Evaluation Approach	Studies
Analysis	[S34]
Illustrative example	[S22], [S47], [S62], [S45], [S73], [S3], [S2], [S20], [S46], [S26], [S59], [S39], [S44]
Illustrative application	[S13], [S41], [S36], [S37], [S7], [S38], [S10], [S18], [S33], [S19], [S16], [S64], [S48], [S29], [S42], [S50], [S4], [S8], [S15], [S65]
Simulation	[S12], [S23], [S60], [S61]

Analysis-based evaluation. The approach presented in [S34] has provided an analysis-based evaluation approach to show how the concept of self-awareness supports the reasoning about object interaction in object-oriented programming paradigm.

Illustrative example. The studies listed under this category have presented case studies to explore their approaches and validate the applicability of the approaches. But, they do not provide any measurements related to the performance of the proposed approaches. For instance, [S46] provides examples to show how their modelling languages can be used to model the self-adaptive systems. [S20] provides examples to show how the proposed knowledge representation approach can be used to represent the captured knowledge to reason about the adaptation.

Illustrative application. The approaches listed under this category have provided real implementation as an illustrative application for their approaches, in order to demonstrate the applicability of the approach in real life. However, the experiments are performed on small scale cases due to the complexity of performing large-scale experiments in a real setting. For example, [S18] evaluates the proposed self-aware approach using a cloud-based application that has the ability to select the adaptation strategy according to the demand of the cloud-based services at

runtime. The illustrative application demonstrates the adaptation capabilities of the approach. However, the experiments have been performed using two physical machines with one or two virtual machines hosted on each of them; a case in which the scale is too small compared to the large scale of cloud systems.

Simulation-based evaluation. Studies adopting simulation-based evaluation have provided an experimental evaluation based on simulations featuring large scale experiments. Such simulations provide the possibility to perform scalable and repeated experiments in a relatively fast and inexpensive controlled environment. However, these approaches still need to demonstrate the applicability of the approaches in real environments.

Figure 7 illustrates the distribution of studies by evaluation approach categories. The majority of studies have evaluated their work using either an illustrative example or illustrative application. Simulation-based evaluation, featuring scalability, is significantly less used.

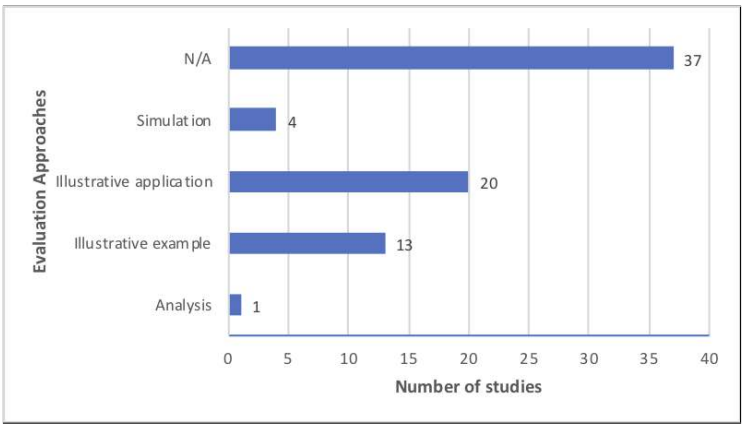

Fig. 7. Distribution of studies by evaluation approaches

4.7.2 Evaluation Criteria. Below, we present the evaluation criteria that have been used in the mentioned studies, and then we present how each of the approaches addressed them. Table 11 lists the evaluation criteria and the corresponding studies.

4.7.2.1 Performance. The studies listed under the *illustrative application* and *simulation* categories have reported on the performance of the proposed approaches. Through the motivation behind adopting self-awareness in the proposed approaches is to enrich the adaptation capabilities and to manage the trade-offs that exist among the different evaluation criteria, we observed that most of the evaluation approaches do not demonstrate how the improvements in one or more of the considered evaluation criteria affected the performance in terms of one or more of the "conflicting" criteria. Also, we observed that some approaches claim the benefits of their self-aware systems without comparing their performance with non-self-aware, or other self-aware approaches.

The approaches presented in [S36], [S37] and [S38] have been evaluated using the *accuracy* and *efficiency* criteria. *Accuracy* measures the extent to which the actual performance of the system meets the performance goals. On the other hand, *efficiency* reflects the ability to minimise power consumption while meeting the performance goal. The results show that the proposed approach has achieved higher accuracy, but with higher power consumption compared with a static approach. Similarly, [S13] has evaluated the work using the *accuracy* criteria. They view *accuracy* as the

Table 11. Evaluation Criteria and Related Studies

Study	Evaluation Criteria	Trade-offs
[S13]	Accuracy	-
[S36], [S37], [S38]	Accuracy, Efficiency	Accuracy, Efficiency
[S41]	Processing time	-
[S12]	Number of bids, asks, allocations, average price, Market revenue	-
[S7]	Reduction in communication	-
[S23]	Number of violations	-
[S33]	Power efficiency, Execution time	Power efficiency, Execution time
[S10]	Power consumption	-
[S19]	Lookahead, Latency, Number of achieved goals	Lookahead, Latency, Number of achieved goals
[S18]	Accuracy, Adaptation quality, Overhead, Reliability	Accuracy, Overhead
[S16]	Local resources consumption, Time performance	-
[S64]	Performance per Watt	-
[S48]	Number of violations	-
[S42]	Prediction accuracy of response time	-
[S50]	Lines of code, Complexity, Technical debt	-
[S4]	Number of required humans, Cost	-

probability of detecting email anomaly based on some adaptive measures, such as the mean and standard deviation of the captured data, which are application-specific evaluation criteria.

The evaluation of [S33] has considered both *power efficiency* and *execution time*. The results highlight that a self-aware solution can achieve low execution time with minimal power consumption. The work of [S10] has also considered power consumption in the smartphone case study. The results show that applying the self-aware strategies to activate system components on-demand has reduced the power consumption compared to a naive non-adaptive method.

The work of [S19] has used three evaluation criteria, that are: *lookahead* that specifies the planning window in the future, *latency* that is the time required to finish planning, and the number of achieved goals. The results show that the larger the *lookahead* the higher the *latency* and *the number of achieved goals*. However, the above works do not demonstrate how their self-aware approaches are compared to non-self-aware (or other self-aware) approaches.

In [S41], the authors evaluate the performance of their approach in terms of processing time. The results show that the proposed approach exhibit better performance compared to a "conventional" approach. The work of [S18] has used the weighted sum of *accuracy, adaptation quality, overhead, and reliability*, (assuming that the corresponding thresholds are specified in the Service Level Agreement) to evaluate the proposed approach. The results show that the weighted sum (called global benefit) in the self-aware case is higher than the weighted sum in the non-self-aware case. [S64] has evaluated the self-aware approach using the normalised performance (in terms

of computation time) per Watt in the presence of fluctuating input data streams and compares the self-aware approach with a set of static (non-self-aware) approaches. The reported results demonstrate that the approach's ability to adapt to the data stream fluctuations while keeping the performance per Watt close to the best static approach. [S16] evaluates the approach using the *local resource consumption* and the *time performance* criteria. The paper claims acceptable time performance and resources consumption with an increasing workload. However, the evaluation of these approaches does not demonstrate how the self-aware approach compare to non-self-aware (or other self-aware) approaches and does not address other quality attributes that may be adversely affected.

Simple metrics are found in the works of [S23] and [S48] that have evaluated their approaches using the number of violations. The results show that the number of violation is reduced leading to a more stable state. The work presented in [S7] has considered a very abstract quality criterion, namely, *communication*. The self-aware scenario results in a reduction of communication between the system objects compared to the non-self-aware scenario. The evaluation scenario of [S12] has considered a number of market-based metrics (number of bids, asks, allocations, average price, and market revenue) to show that the proposed cloud-market monitoring model is able to detect sudden changes in the demand for resources. The case study in [S42] shows 20% of *prediction error* for the response times. The evaluation of the illustrative application in [S50] showed a reduction in the *number of lines of code* compared to a reference application. The experiment of [S4] illustrated a reduction of 32% in the *number of required humans* required to obtain acceptable accuracy for the information extraction in digitized bio-collections. However, all the evaluation approaches mentioned above do not address other quality attributes that may be adversely affected.

4.7.2.2 Overhead. In this section, we investigate the overhead resulting from adopting self-awareness in software systems. Only 8 of the studies have reported the overhead of adopting self-awareness. All of them considered overhead in terms of computation time.

[S62] reported that the proposed approach is low-overhead without presenting experimentation results to demonstrate this claim. In [S36] [S37] and [S38], the authors reported that the overhead of the proposed approach is very low and that the system can take adaptation decisions in 20.09 nanoseconds. However, other overheads related to adopting self-awareness, e.g. the overhead of monitoring, registering events and taking an action, have not been taken into account. [S33] reported on the overhead related to the monitoring component of the approach. The reported runtime overhead is within 1-2%, which the authors consider it to be negligible compared to the normal system's execution time. [S16] reported the overhead of propagating the monitoring information across a network and stated that the overhead is "acceptable" and limited. These approaches consider only the overhead of the monitoring activity. [S50] reported a slight increase in the execution time due to the time needed to build contextual models and considered that increase as negligible.

The study of [S48] has provided a more profound analysis of the overhead. The authors reported on the overhead of analysing the captured information and forecasting, as well as the overhead of the adaptation process. They reported that both overheads depend on the data, configuration settings, the techniques used for performance forecasting and the application specifications.

4.8 Applications Adopting Self-Awareness(RQ7)

This section explores major application domains adopting computational self-awareness. Although not an exhaustive list, we identify the following domains:

- **Smart-*.** The functionalities of the devices or components of a smart-* application (e.g. smart-home, smart-city, .etc) are exposed as services that need to compose to satisfy the users' needs.

Self-awareness principles enable dynamic context management in order to reason about the application needs. Examples of studies consider this domain are [S29] [S50] [S65] [S59].

- **Online Resource Management.** Self-awareness resource management techniques can detect or predict changes at run-time and re-actively or proactively adapt the system. Auto-scaling cloud services is an example application, where cloud applications are deployed on cloud services that exhibit uncertainty in performance related to dynamics of the environment. Self-awareness can be used to improve cloud services auto-scaling methods by enabling online reasoning to enable run-time auto-scaling and SLA management to replace the offline analysis adopted for dealing with such a dynamic environment. Examples of studies consider this domain include [S18] [S17] [S26] [S25] [S47] [S45] [S46] [S48] [S37] [S38] [S44].
- **Dynamic Software Architectures.** The capabilities of knowledge acquisition and representation provided by self-awareness allow for dynamically adapting software architectures by selecting or adapting the software components based on the context. Self-awareness can provide built-in run-time management of QoS, aiming to achieve stability of QoS provision. Examples of studies consider this domain include[S60] [S61] [S26]
- **Health Monitoring.** Sensing devices collect data that relate to patients measurements such as blood pressure and heart rate. A health monitoring application can be equipped with a self-awareness module to increase the robustness of the data analysis by better informing the data collection methods. This includes managing the trade-offs related to the data collection including aspects of energy efficiency (as devices are usually battery operated), latency, and location-awareness, among others. Examples of studies consider this domain include [S65].
- **Emergency System.** A application for distributed management of personnel to deal with emergency situations. Self-awareness enables dealing with cases of emergency information centres failure or fluctuation of performance. Examples of studies consider this domain include [S16] [S15].

5 CHARACTERISATION AND DISCUSSION

In this section, we summarise the main findings, discuss the implications of the review on the research community, as well as report on the limitations and threats to validity of the review.

5.1 Characterisation of Self-Awareness in Software Engineering

We constructed a thematic map about self-awareness in software engineering. The thematic map is depicted in Figure 8. This covers data extracted from the primary studies, i.e. reflects the current state of the art. The thematic map represents a taxonomy for characterising self-awareness in software engineering. Along with the analysis of the primary studies, it could help to reflect on the areas that need further development in the literature, as discussed in the next sections.

5.2 General observations

We conducted this review with the vision of answering the five research questions. The main findings of this systematic review are as follows:

RQ1. There is growing attention to adopting self-awareness in modern software systems. However, there is no common agreement on the definition of self-awareness. Many researchers use the terms ‘self-aware’ and ‘self-adaptive’ interchangeably. Recent attempts to define self-aware systems include that a self-aware system should have multi-levels of knowledge representation and/or support of proactive adaptation. The definition introduced in [S49] can be considered as the most recent and the widest agreed-on definition. Nevertheless, as it has been acknowledged by several authors and research programmes (e.g. Self-aware related EU projects, USA), the

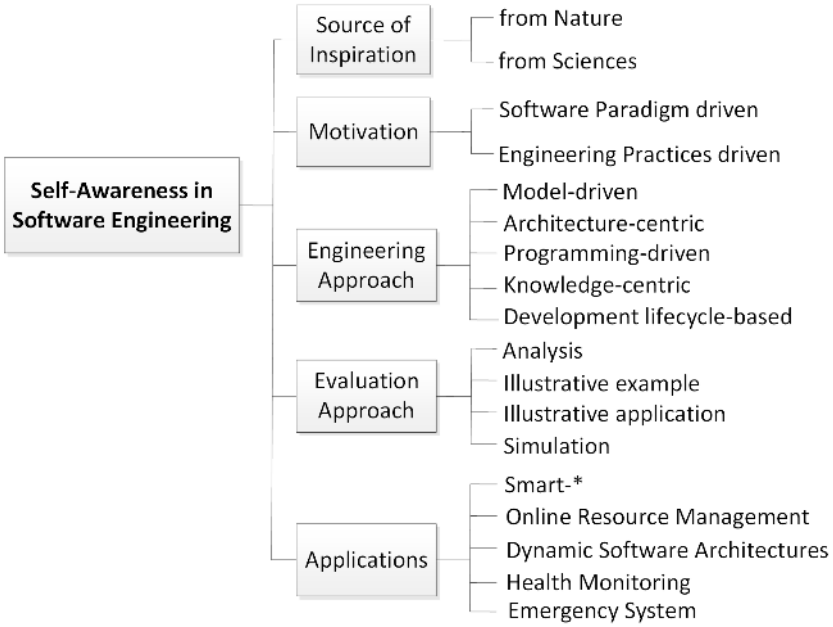

Fig. 8. Thematic Analysis of Self-Awareness in Software Engineering

definition of self-awareness is still an open research question and had been subject to research debate. Nevertheless, reaching a definition is necessary to provide meaningful guidance for developments in the field. This is particularly important to inform the methodological incubation of steps including, but not limited to, the engineering of the requirements that explicate this category of the systems, architectures and patterns that meet these requirements and dynamic trade-offs, testing, evaluation and/or verification procedures for the awareness, its satisfaction/optimisation, added value, etc.

RQ2. Motivations for employing self-awareness were found clearly identified in the studies. Motivations varied between the general purpose of realising better autonomy for software systems and domain-specific purposes. Better understanding for the motivation of leveraging self-awareness from one side and the motivation behind choosing one method over its alternatives can provide engineers and practitioners for this category of systems with the necessary guidance on the drivers, consequences, recommended-/anti- use, lessons learnt etc. It can also help practitioners to develop customised evaluation, testing and verification frameworks that are informed by these drivers.

RQ3. The sources of inspiration were mainly nature, psychology, control theory, but the mapping between the self-awareness in software engineering and the source of inspiration is not well detailed in the majority of studies. Furthermore, these inspirations had led to contributions which could be criticised to have partial focus, rather than holistic focus, in realising self-awareness.

RQ4. Self-awareness was considered for self-adaptive software systems as a general software paradigm, with few studies focusing on a particular software paradigm or application type. Architecture design was found the most contributing software engineering practice in realising self-awareness, as well as system design and engineering adaptations.

RQ5. The approaches for engineering self-aware software systems can be categorised as model-driven, architecture-centric, programming-driven, knowledge-centric and development lifecycle-based approaches. We have found that self-aware systems vary from centrally managed to fully distributed. However, most of the approaches tend to integrate self-awareness in decentralised settings.

RQ6. Some of the studies have provided an experimental evaluation of the proposed approaches. However, the evaluations do not demonstrate the value added by self-awareness to the adaptation capabilities compared to the self-adaptive systems. The evaluations also need to report on the overhead accompanied by adopting self-awareness. As discussed in the shortcomings that relate to both the definition and motivation, future research needs to align the evaluation with the drivers for self-awareness and its scope. Research needs to discuss metrics, evaluation criteria that relate to the evaluation of the presence and/or absence of self-awareness on the system and what does it mean for the system to be self-aware. Research may also need to evaluate settings and patterns for realising self-awareness - ranging from centralised to semi/decentralised etc.

RQ7. The application domains of computational self-awareness are various and numerous. However, there is no real-world application, i.e. a product, that realises computational self-awareness. The applications identified from the reviewed studies are, in fact, case studies utilised as proof-of-concepts.

5.3 Critical Reflections on the State of the Art

In the following, we discuss a number of limitations that are observed from the reviewing and analysis of the literature.

- **Proactive adaptation.** The vast majority of literature utilises captured knowledge for reactive adaptation. Little has been done to tackle proactive self-adaptation in order to self-adapt the system before it becomes necessary. Proactive adaptation can make autonomous systems more dependable as it reduces interruptions by avoiding unwanted cases of failure or requirements violations [28].
- **Human-in-the-loop.** It is obvious from the reviewed studies that they have been engineered as fully autonomic systems. The capturing of the knowledge about the system internal state and the environment does not explicitly consider the capturing of human knowledge that may also evolve over the lifetime of the application. Although this issue has been observed in the literature of self-adaptive systems [19], it is still lacking consideration in the computational self-awareness literature though the latter is, to a big extent, motivated by the potential to improve self-adaptivity. Frameworks for involving the potential evolving human knowledge are required so that the can contribute to making systems more self-aware.
- **Goals mapping.** The majority of work express user goals in terms of metrics that are assumed to be straightway monitored. This can be valid in some cases. For example, goals like Processing time and Processing Power can be considered also as low-level metrics that can be monitored. However, goals like Accuracy and Efficiency are high-level goals that need to be mapped into 'monitorable' low-level metrics. Also, it might be clear that it is not realistic to assume that users can specify goals using low-level metrics (e.g. number of violations and Lines of code). Users need to specify high-level goals that should be mapped to low-level ones. Our observation is that little has been done with respect to such mapping in the reviewed papers.
- **Privacy-awareness.** Obviously, the realisation of computational self-awareness is based only on data collection. This raises concerns about the privacy of the owners of the collected data. There is here a trade-off between self-awareness and privacy. On one hand, in order for a

system to increase its self-awareness, it needs to collect more data. On the other hand, the more data is collected the less privacy is achieved. Therefore, a balance is required to optimise for both self-awareness and privacy. None of the reviewed studies addressed this concern although it is discussed in [S40] as a challenge.

- **Realistic evaluation.** Evaluation is mostly performed using simulations or, in the best case, illustrative case studies. It can be suggested that this is due to the costs and complexity of developing and experimenting with real self-aware applications on a large scale. Although that suggestion might be valid, this observation leads us to doubt of the generability of the outcomes as most evaluation approaches are merely proof of concept ideas that are of little practical use to other researchers. In other words, the literature still requires a real-world self-aware application that is evaluated and that are usable by others.

5.4 Implications for research

The aim of this systematic review on self-awareness in software engineering is to investigate how current research has adopted computational self-awareness to enrich the self-adaptation capabilities of autonomous software systems. This paper provides the first comprehensive review that summarises the relevant literature and reports on possible gaps. Overall, the review provides a quite representative state of the relevant literature. The findings can support researchers interested in future research for advancing self-aware systems. The reported challenges can guide the researches to direct their future research accordingly to look for solutions for filling the mentioned gaps.

5.5 Limitations and threats to validity

The main limitations and validity threats of this review are related to the studies selection bias, inaccuracy in data extraction and analysis of collected studies.

- **Missing relevant studies.** The search was based on meta-data (abstract, title, and keywords) only and might have missed some studies that have considered self-awareness in software engineering as part of their proposed work, and have not mentioned this explicitly in the title, abstract and keywords. Though the meta-data are specified by the authors of the papers, we reasonably rely on how well the digital databases classify and index papers. Studies have been collected from data sources that are basically academic indexing services. We have not considered other sources, e.g. companies websites, that might have addressed self-awareness in their industry-focused research and might have interesting findings.
- **Studies selection bias.** With respect to the selection of the initial studies, we adopted a set-up to guide the selection process, thus avoiding selection bias. For example, if the number of search results is more than 200 results, we selected the first 200 sorted by relevance. Furthermore, every study is screened by the first two authors to increase the reliability of the selection. If the two authors do not agree on the decision of selecting a paper as a primary study after discussions, the other authors are involved for further discussion and agreement. Such set-up directed the selection based on both the search results and the widest possible consensus of the selection.
- **Inaccuracy in data extraction.** Inaccuracy can be introduced in the data extraction process due to different reasons, such as the background of the researcher, subjectivity of the researcher, and the way the authors used to present their approaches and findings. Aiming at minimising the inaccuracy in data extraction, we adopted a strategy for the data extraction process, such that data extracted from certain studies by one reviewer are checked by the other reviewer. Also, we had thorough discussions to eliminate any confusion which lead us to believe that the effect of this error is minimal.

6 RESEARCH CHALLENGES

While self-awareness is getting popularity as an enabler for self-adaptation in software systems, there are several issues and challenges which need considerations. In this section, we present different challenges related to the adoption of self-awareness in software engineering.

- **Holistic self-awareness**. As discussed in section 4.4 a holistic self-aware system that can consolidate various aspects of computational self-awareness based on the various and different inspirations of self-awareness is an open challenge that is worthy of future research. This can equip the self-aware system with comprehensive knowledge about its internal state and the environment and various features and capabilities to model knowledge from different perspectives, leading to a wide range of adaptation choices.
- **Self-awareness vs. Self-adaptation.** The visions of the self-aware initiatives represented by [S53] and [S46] are only a step forward towards distinguishing self-aware systems from self-adaptive systems. However, the difference between self-aware and self-adaptive systems in terms of the definition is not the only aspect to consider. Distinctions between self-adaptive and self-aware systems are required in order to evaluate the enhancement occurring after using self-awareness. Though self-awareness can be viewed as property for enriching self-adaptivity similar to other self-* properties (e.g. self-healing, self-organising, etc.), clear illustration on the complementary use is still lacking. Another observation is that the relation between other self-* properties and self-awareness is not entirely clear. Self-awareness has been explicitly linked to self-expression; however, it can be argued that self-expression can still relate to properties that relate to healing, organising, protecting, securing, configuring, etc. Similarly, the self-* properties can also be engineered as part of the self-aware "engine". Nevertheless, the literature and existing solutions do not explicate this topic.
- **Inspirations for self-awareness.** As observed in the literature that the majority of studies named only their source of inspiration, without a clear description about how self-awareness in software engineering is inspired by nature of sciences. We argue that mapping between the source of inspiration and the research in software engineering needs to be clearly present. Further, studies investigating how self-awareness could be inspired by nature and sciences would contribute to advancing self-aware software systems.
- **Evaluating the presence of self-awareness in a system.** Existing approaches have evaluated self-awareness in relation to the primitives that enable self-awareness and enrich the adaptivity. As an example, [S18] has structured the proposed framework around levels for self-awareness benefiting from stimuli, goal, knowledge, interaction, yet evaluating self-awareness presence is challenging to achieve. This calls for novel metrics, qualitative and quantitative frameworks for evaluating the presence/absence of self-awareness. This can consequently beg the question: what are the distinct characteristics of a self-aware system? to what extent a system is self-aware? how can we claim that a system is more self-aware than its competitor?
- **Testing self-awareness properties.** Given a clear characterisation for self-aware software systems, testing self-awareness properties themselves is not considered in the literature. We argue that a testing framework and tool would be essential for advancing self-awareness in software engineering. The role of such tool is to test the properties of self-awareness themselves. Inspirations could be drawn from Artificial Intelligence (AI) studies that claim to propose more intelligent systems; i.e. more self-aware systems.
- **Proactive adaptation.** Achieving proactive adaptation is challenged by the need for prediction models that provide convenient accuracy of predictions, as inaccurate predictions result in sub-optimal adaptations. The development of such models is also challenged by the lack of knowledge in some cases (in cases of rare events), the volatility of knowledge (the cases

when the environment is very dynamic and knowledge becomes quickly obsolete), and the prediction of user behaviours. Furthermore, proactive adaptation comes with the overhead of predictions. therefore, the feasibility of this type of adaptation needs to be assessed, as unnecessary adaptations would reduce or cancel its desired benefits.

- **Evaluating the quality and overhead of self-awareness.** As found in the primary studies, the evaluation of the quality and overhead of self-awareness is not considered as it should be. The majority of the studies evaluated the enhancement in quality attributes after employing self-awareness capabilities in comparison with non-self-aware ones. Meanwhile, how well the self-awareness performed is not evaluated. Also, the quality and accuracy of adaptations made with the help of self-awareness are not evaluated. A framework for evaluating self-awareness, similar to self-adaptivity [37], can be used by researchers as a base for evaluating how well their proposed self-awareness engineering approaches are performing, as well as for comparing different approaches.
- **Dynamic knowledge management.** Dynamic knowledge management in the studied self-aware software systems architectures is given little consideration; although it is a vital requirement for self-awareness. Most of the self-aware frameworks address knowledge management at a coarse-grained level. We argue that "finer" knowledge representation can better address the users and systems requirements in the environments that exhibit uncertainty and dynamism, as well as can improve the quality and accuracy of adaptation. The knowledge should be treated as moving targets that can change and evolve over time. Self-aware systems should be able to capture the evolution trends and use this information to better inform the adaptation decision.

7 CONCLUSION AND FUTURE WORK

The main contribution of this paper is to systemically review and investigate the adoption of computational self-awareness concepts in autonomic software systems. Our goal is to provide a quite representative state-of-the-art of the current research in the self-aware software engineering topic, in order to identify the accomplishments done so far, the pending issues and open problems. We have considered six main data sources to conduct the search from which we identified and reviewed 865 studies. From these, we have found 74 papers presenting approaches that exhibit acceptable relevance, which we selected as primary studies. The main findings show that there is growing attention to incorporate self-awareness for better reasoning about the adaptation decision making in autonomic systems. However, many pending issues and open problems still need to be addressed, e.g. dynamic knowledge management.

Our future work aims at addressing the challenges outlined in this paper. We will focus on closing the gaps in the relevant literature, such as the dynamic knowledge management. We aim to investigate the evaluation of self-aware systems and compare their performance with self-adaptive systems in order to identify the value-added by self-awareness to self-adaptive systems.

PRIMARY STUDIES

[S1] R. Abbott and C. Sun. 2008. Abstraction Abstracted. In *2nd International Workshop on The Role of Abstraction in Software Engineering (ROA)*. ACM, 23–30.

[S2] D. B. Abeywickrama, N. Hoch, and F. Zambonelli. 2013. SimSOTA: Engineering and Simulating Feedback Loops for Self-adaptive Systems. In *International C* Conference on Computer Science and Software Engineering (C3S2E)*. ACM, 67–76.

[S3] D. B. Abeywickrama, F. Zambonelli, and N. Hoch. 2012. Towards Simulating Architectural Patterns for Self-Aware and Self-Adaptive Systems. In *IEEE 6th International Conference on Self-Adaptive and Self-Organizing Systems Workshops*

(SASOW). 133–138.

[S4] I. Alzuru, A. Matsunaga, M. Tsugawa, and J. A. B. Fortes. 2017. SELFIE: Self-Aware Information Extraction from Digitized Biocollections. In *IEEE 13th International Conference on e-Science (e-Science)*. 69–78.

[S5] P. Andras and B. G. Charlton. 2005. Self-aware software - Will it become a reality? In *Self-Star Properties in Complex Information Systems: Conceptual and Practical Foundations*. Lecture Notes In Computer Science, Vol. 3460. 229–259.

[S6] M. Autili, K. L. Bellman, A. Diaconescu, L. Esterle, M. Tivoli, and A. Zisman. 2017. *Transition Strategies for Increasing Self-awareness in Existing Types of Computing Systems*. Springer International Publishing, 305–336.

[S7] T. Becker, A. Agne, P. R. Lewis, R. Bahsoon, F. Faniyi, L. Esterle, A. Keller, A. Chandra, A. R. Jensenius, and S. C. Stilkerich. 2012. EPiCS: Engineering Proprioception in Computing Systems. In *IEEE 15th International Conference on Computational Science and Engineering (CSE)*. 353–360.

[S8] K. L. Bellman, C. Landauer, P. Nelson, N. Bencomo, S. Gtz, P. R. Lewis, and L. Esterle. 2017. *Self-modeling and Self-awareness*. Springer International Publishing, 279–304.

[S9] A. G. Beltran, P. Milligan, and P. Sage. 2005. Heterogeneity-aware distributed access structure. In *5th IEEE International Conference on Peer-to-Peer Computing (P2P)*. 152–153.

[S10] N. Bicocchi, D. Fontana, and F. Zambonelli. 2014. A self-aware, reconfigurable architecture for context awareness. In *IEEE Symposium on Computers and Communications (ISCC)*. 1–7.

[S11] R. Birke, J. Cámara, L. Y. Chen, L. Esterle, K. Geihs, E. Gelenbe, H. Giese, A. Robertsson, and X. Zhu. 2017. *Self-aware Computing Systems: Open Challenges and Future Research Directions*. Springer International Publishing, 709–722.

[S12] I. Breskovic, C. Haas, S. Caton, and I. Brandic. 2011. Towards Self-Awareness in Cloud Markets: A Monitoring Methodology. In *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*. 81–88.

[S13] A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, and I. Cohen. 2001. Self-aware services: using Bayesian networks for detecting anomalies in Internet-based services. In *IEEE/IFIP International Symposium on Integrated Network Management*. 623–638.

[S14] Giacomo Cabri and Franco Zambonelli. 2014. Towards Self-Aware and Self-Composing Services. In *The Computer After Me*, Jeremy Pitt (Ed.). Imperial College Press, Chapter 2, 21–36. https://doi.org/10.1142/9781783264186_0002

[S15] R. Calinescu, M. Autili, J. Cámara, A. Di Marco, S. Gerasimou, P. Inverardi, A. Perucci, N. Jansen, J. P. Katoen, M. Kwiatkowska, and et. al. 2017. Synthesis and Verification of Self-aware Computing Systems. In *Self-Aware Computing Systems*. Springer, 337–373.

[S16] G. Castelli, M. Mamei, A. Rosi, and F. Zambonelli. 2015. Engineering Pervasive Service Ecosystems: The SAPERE Approach. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 10, 1, Article 1 (2015), 27 pages.

[S17] T. Chen and R. Bahsoon. 2015. Toward a Smarter Cloud: Self-Aware Autoscaling of Cloud Configurations and Resources. *Computer* 48, 9 (2015), 93–96.

[S18] T. Chen, F. Faniyi, R. Bahsoon, P. R. Lewis, X. Yao, L. Minku, and Lu. Esterle. 2014. *The handbook of engineering self-aware and self-expressive systems*. Technical Report. School of Computer Science, University of Birmingham.

[S19] D. Dannenhauer, M. T. Cox, S. Gupta, M. Paisner, and D. Perlis. 2014. Toward Meta-level Control of Autonomous Agents. *Procedia Computer Science* 41 (2014), 226 – 232. 5th Annual International Conference on Biologically Inspired Cognitive Architectures (BICA).

[S20] R. De Nicola, M. Loreti, R. Pugliese, and F. Tiezzi. 2014. A Formal Approach to Autonomic Systems Programming: The SCEL Language. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 9, 2, Article 7 (2014), 29 pages.

[S21] S. Dustdar, C. Dorn, F. Li, L. Baresi, G. Cabri, C. Pautasso, and F. Zambonelli. 2010. A Roadmap Towards Sustainable Self-aware Service Systems. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. ACM, 10–19.

[S22] A. Egyed. 2004. Architecture Differencing for Self Management. In *1st ACM SIGSOFT Workshop on Self-managed Systems (WOSS '04)*. ACM, New York, NY, USA, 44–48.

[S23] A. Elhabbash, R. Bahsoon, and P. Tino. 2014. Towards Self-Aware Service Composition. In *IEEE International Conference on High Performance Computing and Communications, IEEE 6th International Symposium on Cyberspace Safety and Security, IEEE 11th International Conference on Embedded Software and Systems (HPCC,CSS,ICESS)*. 1275–1279.

[S24] L. Esterle, K. L. Bellman, S. Becker, A. Koziolek, C. Landauer, and P. R. Lewis. 2017. *Assessing Self-awareness*. Springer International Publishing, 465–481.

[S25] F. Faniyi and R. Bahsoon. 2011. Engineering Proprioception in SLA Management for Cloud Architectures. In *9th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. 336–340.

[S26] F. Faniyi, P. R. Lewis, R. Bahsoon, and X. Yao. 2014. Architecting Self-Aware Software Systems. In *IEEE/IFIP Conference on Software Architecture (WICSA)*. 91–94.

[S27] E. Gerbert-Gaillard, S. Chollet, and P. Lalanda. 2016. Model-driven approach for self-aware pervasive systems. In *IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. 1–6.

[S28] E. Gerbert-Gaillard and P. Lalanda. 2016. Self-Aware Model-Driven Pervasive Systems. In *IEEE International Conference on Autonomic Computing (ICAC)*. 221–222.

[S29] E. Gerbert-Gaillard, P. Lalanda, S. Chollet, and J. Demarchez. 2017. A self-aware approach to context management in pervasive platforms. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 256–261.

[S30] H. Giese, T. Vogel, A. Diaconescu, S. Götz, and K. L. Bellman. 2017. Generic Architectures for Individual Self-aware Computing Systems. In *Self-Aware Computing Systems*. Springer, 149–189.

[S31] H. Giese, T. Vogel, A. Diaconescu, S. Götz, N. Bencomo, K. Geihs, S. Kounev, and K. L. Bellman. 2017. State of the Art in Architectures for Self-aware Computing Systems. In *Self-Aware Computing Systems*. Springer, 237–275.

[S32] H. Giese, T. Vogel, A. Diaconescu, S. Götz, and S. Kounev. 2017. Architectural Concepts for Self-aware Computing Systems. In *Self-Aware Computing Systems*. Springer, 109–147.

[S33] R. Gioiosa, G. Kestor, D. J. Kerbyson, and A. Hoisie. 2014. Cross-Layer Self-Adaptive/Self-Aware System Software for Exascale Systems. In *IEEE 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. 326–333.

[S34] A. Griffiths. 1997. ldquo;self rdquo;-conscious objects in Object-Z. In *Technology of Object-Oriented Languages and Systems (TOOLS 25)*. 210–224.

[S35] N. Herbst, S. Becker, S. Kounev, H. Koziolek, M. Maggio, A. Milenkoski, and E. Smirni. 2017. *Metrics and Benchmarks for Self-aware Computing Systems*. Springer International Publishing, 437–464.

[S36] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. 2010. SEEC: A framework for self-aware computing. (2010).

[S37] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. 2011. SEEC: A general and extensible framework for self-aware computing. (2011).

[S38] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. 2013. A generalized software framework for accurate and efficient management of performance goals. In *International Conference on Embedded Software (EMSOFT)*. 1–10.

[S39] M. Hölzl and T. Gabor. 2015. *Reasoning and Learning for Awareness and Adaptation*. Springer International Publishing, 249–290.

[S40] M. Hölzl and M. Wirsing. 2014. Issues in Engineering Self-Aware and Self-Expressive Ensembles. In *The Computer After Me*, Jeremy Pitt (Ed.). Imperial College Press, Chapter 3, 37–54. https://doi.org/10.1142/9781783264186_0003

[S41] C. H. Huang, J. S. Shen, and P. A Hsiung. 2010. A Self-Adaptive Hardware/Software System Architecture for Ubiquitous Computing Applications. In *Ubiquitous Intelligence and Computing (Lecture Notes in Computer Science)*, Z. Yu, R. Liscano, G. L. Chen, D. Q. Zhang, and X. S. Zhou (Eds.), Vol. 6406. Nokia; Ind Corp China, 382–396. 7th International Conference on Autonomic and Trusted Computing, NW Polytechn Univ, Xian, PEOPLES R CHINA, OCT 26-29, 2010.

[S42] N. Huber, F. Brosig, S. Spinner, S. Kounev, and M. Baehr. 2017. Model-Based Self-Aware Performance and Resource Management Using the Descartes Modeling Language. *IEEE Transactions on Software Engineering* 43, 5 (2017), 432–452.

[S43] J. O. Kephart, A. Diaconescu, H. Giese, A. Robertsson, T. Abdelzaher, P. R. Lewis, A. Filieri, L. Esterle, and S. Frey. 2017. *Self-adaptation in Collective Self-aware Computing Systems*. Springer International Publishing, 401–435.

[S44] J. O. Kephart, M. Maggio, A. Diaconescu, Ho. Giese, H. Hoffmann, S. Kounev, A. Koziolek, P. R. Lewis, A. Robertsson, and S. Spinner. 2017. *Reference Scenarios for Self-aware Computing*. Springer International Publishing, 87–106.

[S45] S. Kounev, F. Brosig, and N. Huber. 2011. Self-aware QoS Management in Virtualized Infrastructures. In *8th ACM International Conference on Autonomic Computing (ICAC)*. ACM, 175–176.

[S46] S. Kounev, F. Brosig, and N. Huber. 2014. *The descartes modeling language*. Technical Report. Department of Computer Science, University of Wuerzburg.

[S47] S. Kounev, F. Brosig, N. Huber, and R. Reussner. 2010. Towards Self-Aware Performance and Resource Management in Modern Service-Oriented Systems. In *IEEE International Conference on Services Computing (SCC)*. 621–624.

[S48] S. Kounev, N. Huber, F. Brosig, and X. Zhu. 2016. A Model-Based Approach to Designing Self-Aware IT Systems and Infrastructures. *Computer* 49, 7 (2016), 53–61.

[S49] S. Kounev, P. R. Lewis, K. L. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. O. Kephart, and A. Zisman. 2017. *The Notion of Self-aware Computing*. Springer International Publishing, 3–16.

[S50] P. Lalanda, E. Gerber-Gaillard, and S. Chollet. 2017. Self-Aware Context in Smart Home Pervasive Platforms. In *IEEE International Conference on Autonomic Computing (ICAC)*. 119–124.

[S51] C. Landauer and K. L. Bellman. 2017. An Architecture for Self-Awareness Experiments. In *IEEE International Conference on Autonomic Computing (ICAC)*. 255–262.

[S52] P. R. Lewis. 2014. Computational Self-Awareness and Learning Machines. In *The Computer After Me*, Jeremy Pitt (Ed.). Chapter 18, 267–280. https://doi.org/10.1142/9781783264186_0018

[S53] P. R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and X. Yao. 2011. A Survey of Self-Awareness and Its Application in Computing Systems. In *5th IEEE Conference on Self-Adaptive and Self-Organizing*

Systems Workshops (SASOW). 102–107.
[S54] S. Linkola, A. Kantosalo, T. Männistö, and H. Toivonen. 2017. Aspects of Self-awareness: An Anatomy of Metacreative Systems. In *8th International Conference on Computational Creativity (ICCC)*.
[S55] M. Mitchell. 2005. Self-awareness and control in decentralized systems.. In *AAAI Spring Symposium: Metacognition in Computation*. 80–85.
[S56] O. Nierstrasz, M. Denker, T. Gîrba, A. Lienhard, and D. Röthlisberger. 2008. *Change-Enabled Software Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 64–79.
[S57] S. Parsons, R. Bahsoon, P. R. Lewis, and X. Yao. 2011. *Towards a Better Understanding of Self-Awareness and Self-Expression within Software Systems*. Technical Report CSR-11-03. University of Birmingham, School of Computer Science.
[S58] Jeremy Pitt. 2014. Introduction: The Computer After Me. In *The Computer After Me*. Chapter 1, 1–18. https://doi.org/10.1142/9781783264186_0001
[S59] E. Riccobene and P. Scandurra. 2015. Formal Modeling Self-adaptive Service-oriented Applications. In *30th Annual ACM Symposium on Applied Computing (SAC)*. ACM, 1704–1710.
[S60] M. Salama and R. Bahsoon. 2015. Quality-Driven Architectural Patterns for Self-Aware Cloud-Based Software. In *IEEE 8th International Conference on Cloud Computing*. 844–851.
[S61] M. Salama, A. Shawish, and R. Bahsoon. 2016. Dynamic Modelling of Tactics Impact on the Stability of Self-Aware Cloud Architectures. In *IEEE 9th International Conference on Cloud Computing (CLOUD)*. 871–875.
[S62] M. D. Santambrogio, H. Hoffmann, J. Eastep, and A. Agarwal. 2010. Enabling technologies for self-aware adaptive systems. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 149–156.
[S63] R. Sterritt and M. Hinchey. 2005. Why computer-based systems should be autonomic. In *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS)*. 406–412.
[S64] Y. Su, F. Shi, S. Talpur, Y. Wang, S. Hu, and J. Wei. 2015. Achieving self-aware parallelism in stream programs. *Cluster Computing - The Journal of Networks Software Tools and Applications* 18, 2, SI (2015), 949–962.
[S65] K. Tammemäe, A. Jantsch, A. Kuusik, J. Preden, and E. Õunapuu. 2018. *Self-Aware Fog Computing in Private and Secure Spheres*. Springer International Publishing, 71–99.
[S66] N. K. Thanigaivelan, E. Nigussie, S. Virtanen, and J. Isoaho. 2017. Towards Self-aware Approach for Mobile Devices Security. In *Computer Network Security*, J. Rak, J. Bay, I. Kotenko, L. Popyack, V. Skormin, and K. Szczypiorski (Eds.). Springer International Publishing, 171–182.
[S67] E. Vassev and M. Hinchey. 2011. Knowledge Representation and Reasoning for Intelligent Software Systems. *Computer* 44, 8 (2011), 96–99.
[S68] E. Vassev and M. Hinchey. 2012. Awareness in Software-Intensive Systems. *Computer* 45, 12 (2012), 84–87.
[S69] E. Vassev and M. Hinchey. 2015. *Knowledge Representation for Adaptive and Self-aware Systems*. Springer International Publishing, 221–247.
[S70] E. E. Veas, K. Kiyokawa, and H. Takemura. 2005. Self-aware Framework for Adaptive Augmented Reality. In *International Conference on Augmented Tele-existence (ICAT)*. ACM, 70–77.
[S71] Nikola Šerbedžija, Tomáš Bureš, and Jaroslav Keznikl. 2013. Engineering Autonomous Systems. In *17th Panhellenic Conference on Informatics (PCI)*. ACM, 128–135.
[S72] J. Walter, A. Di Marco, S. Spinner, P. Inverardi, and S. Kounev. 2017. *Online Learning of Run-Time Models for Performance and Resource Management in Data Centers*. Springer International Publishing, 507–528.
[S73] F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, and M. Puviani. 2011. On Self-Adaptation, Self-Expression, and Self-Awareness in Autonomic Service Component Ensembles. In *5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*. 108–113.
[S74] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo, M. Risoldi, A. E. Tchao, S. Dobson, G. Stevenson, J. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, and B. Wally. 2011. Self-aware Pervasive Service Ecosystems. *Procedia Computer Science* 7 (2011), 197 – 199.

REFERENCES

[1] M. Baldauf, S. Dustdar, and F. Rosenberg. 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* 2, 4 (2007), 263–277.
[2] G. Blair, N. Bencomo, and R. B. France. 2009. Models@ run.time. *Computer* 42, 10 (2009), 22–27.
[3] P. Bozzelli, Q. Gu, and P. Lago. 2013. *A systematic literature review on green software metrics*. Report Technical Report. VU University Amsterdam, Department of Computer Science, The Netherlands. http://www.sis.uta.fi/~pt/TIEA5_Thesis_Course/Session_10_2013_02_18/SLR_GreenMetrics.pdf
[4] O. P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* 80, 4 (2007),

571–583.

[5] R. Calinescu. 2013. *Emerging Techniques for the Engineering of Self-Adaptive High-Integrity Software.* Springer Berlin Heidelberg, Berlin, Heidelberg, 297–310.

[6] J. Camara, R. de Lemos, C. Ghezzi, and A. Lopes. 2013. *Assurances for self-adaptive systems : principles, models, and techniques.* Springer, Berlin New York.

[7] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. 2009. *Software Engineering for Self-Adaptive Systems: A Research Roadmap.* Springer-Verlag, 1–26.

[8] D. S. Cruzes and T. Dyba. 2011. Recommended Steps for Thematic Synthesis in Software Engineering. In *Proceedings of International Symposium on Empirical Software Engineering and Measurement.* 275–284.

[9] R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. Goschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezze, C. Prehofer, W. Schafer, R. Schlichting, D. Smith, P. J. Sousa, L. Tahvildari, K. Wong, and J. Wuttke. 2013. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap.* Lecture Notes in Computer Science, Vol. 7475. Springer-Verlag, 1–32.

[10] S. Dustdar, C. Dorn, F. Li, L. Baresi, G. Cabri, C. Pautasso, and F. Zambonelli. 2010. A Roadmap Towards Sustainable Self-aware Service Systems. In *Proceedings of ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS).* ACM, New York, NY, USA, 10–19.

[11] T. Dybå and T. Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9-10 (2008), 833–859.

[12] T. Dybå, T. Dingsøyr, and G. Hanssen. 2007. Applying Systematic Reviews to Diverse Study Types: An Experience Report. In *1st International Symposium on Empirical Software Engineering and Measurement (ESEM).* 225–234.

[13] European Commission (FP7). 2010. FP7: FET Proactive Initiative: Self-Awareness in Autonomic Systems (AWARENESS). http://cordis.europa.eu/fp7/ict/fet-proactive/aware_en.html

[14] R. B. France and B. Rumpe. 2007. Model-driven Development of Complex Software: A Research Roadmap. In *Future of Software Engineering (FOSE).* 37–54.

[15] R. Frei and G. D. M. Serugendo. 2014. Self-Healing Software. In *The Computer After Me*, Jeremy Pitt (Ed.). Chapter 18, 71–82. https://doi.org/10.1142/9781783264186_0005

[16] A. Gambi, G. Toffetti, and M. Pezz. 2013. *Assurance of Self-adaptive Controllers for the Cloud.* Springer Berlin Heidelberg, Berlin, Heidelberg, 311–339.

[17] D. Garlan and M. Shaw. 1993. *An introduction to software architecture.* Series on Software Engineering and Knowledge Engineering, Vol. 2. World Scientific.

[18] H. Giese, T. Vogel, A. Diaconescu, S. Götz, N. Bencomo, K. Geihs, S. Kounev, and K. L. Bellman. 2017. *State of the Art in Architectures for Self-aware Computing Systems.* Springer International Publishing, 237–275.

[19] M. Gil, V. Pelechano, J. Fons, and M. Albert. 2016. Designing the Human in the Loop of Self-Adaptive Systems. In *Ubiquitous Computing and Ambient Intelligence*, C. R. García, P. Caballero-Gil, M. Burmester, and A. Quesada-Arencibia (Eds.). Springer International Publishing, 437–449.

[20] T. Greenhalgh and R. Peacock. 2005. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ* 331, 7524 (2005), 1064–1065.

[21] M. C. Huebscher and J. A. McCann. 2008. A Survey of Autonomic Computing - Degrees, Models, and Applications. *Comput. Surveys* 40, 3, Article 7 (2008), 7:1–7:28 pages.

[22] A. Iosup, X. Zhu, A. Merchant, E. Kalyvianaki, M. Maggio, S. Spinner, T. Abdelzaher, O. Mengshoel, and S. Bouchenak. 2017. *Self-awareness of Cloud Applications.* Springer International Publishing, 575–610.

[23] JabRef Development Team. accessed: 2018. *JabRef.* http://www.jabref.org

[24] J. O. Kephart and D. M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50.

[25] B. A. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. 2009. Systematic Literature Reviews in Software Engineering - A Systematic Literature Review. *Journal of Information and Software Technology* 51, 1 (2009), 7–15.

[26] B. A. Kitchenham and S. Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Report Technical Report. Keele University.

[27] S. Kounev, X. Zhu, J. O. Kephart, and M. Kwiatkowska. 2015. Model-driven Algorithms and Architectures for Self-Aware Computing Systems (Dagstuhl Seminar 15041). *Dagstuhl Reports* 5, 1 (2015), 164–196.

[28] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17, Part B (2015), 184–206.

[29] P. R. Lewis. 2017. Self-aware computing systems: From psychology to engineering. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 1044–1049.
[30] P. R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and X. Yao. 2011. A Survey of Self-Awareness and Its Application in Computing Systems. In *5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*. 102–107.
[31] S. Mahdavi-Hezavehi, V. H.S. Durelli, D. Weyns, and P. Avgeriou. 2017. A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems. *Information and Software Technology* 90 (2017), 1 – 26.
[32] E. Riccobene, P. Scandurra, and F. Albani. 2011. A Modeling and Executable Language for Designing and Prototyping Service-Oriented Applications. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications*. 4–11.
[33] M. Salehie and L. Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4, 2 (2009), 1–42.
[34] J. Schaumeier, J. Pitt, and G. Cabri. 2012. A Tripartite Analytic Framework for Characterising Awareness and Self-Awareness in Autonomic Systems Research. In *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*. 157–162.
[35] G. Steinbauer and F. Wotawa. 2013. *Model-Based Reasoning for Self-Adaptive Systems fi Theory and Practice*. Springer Berlin Heidelberg, Berlin, Heidelberg, 187–213.
[36] E. Vassev, M. Hinchey, and B. Gaudin. 2012. Knowledge Representation for Self-adaptive Behavior. In *5th International C* Conference on Computer Science and Software Engineering (C3S2E)*. ACM, 113–117.
[37] N. M. Villegas, H. A. Müller, G. Tamura, L. Duchien, and R. Casallas. 2011. A framework for evaluating quality-driven self-adaptive software systems. In *Proceedings of 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 80–89.
[38] D. Weyns, M. U. Iftikhar, S. Malek, and J. Andersson. 2012. Claims and supporting evidence for self-adaptive systems: a literature study. In *P7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE Press, 89–98.
[39] E. Yuan, N. Esfahani, and S. Malek. 2014. A Systematic Survey of Self-Protecting Software Systems. *ACM Transactions on Autonomous and Adaptive Systems* 8, 4 (2014), 1–41.

A REVIEW PROTOCOL

In this appendix, we present the research method and systematic process we followed in conducting the review.

The procedure of this study generally followed the guidelines for conducting systematic literature reviews [26]. The process has also been informed by other reviews relevant to software engineering [25] [3]. We first define the research questions that derive our research, and then we describe the planned protocol to be followed for conducting the review. The research methodology is depicted in Figure A.1.

A.1 Planning the Review

This section presents the details of the first phase.

A.1.1 Research Questions. The overall research objective of the review is to give an overview of the current state-of-the-art related to self-awareness in software engineering in research and practice. The research questions addressed by this study are:

- **RQ1.** How to define and characterise self-awareness?
- **RQ2.** What motivated the application of self-awareness in software engineering?
- **RQ3.** What are the sources of inspiration for its engineering?
- **RQ4.** In which software engineering practices and software paradigms is self-awareness employed?
- **RQ5.** What are the approaches for engineering self-awareness?
- **RQ6.** How are self-aware software systems are evaluated?
- **RQ7.** What are the working real-world applications adopting computational self-awareness?

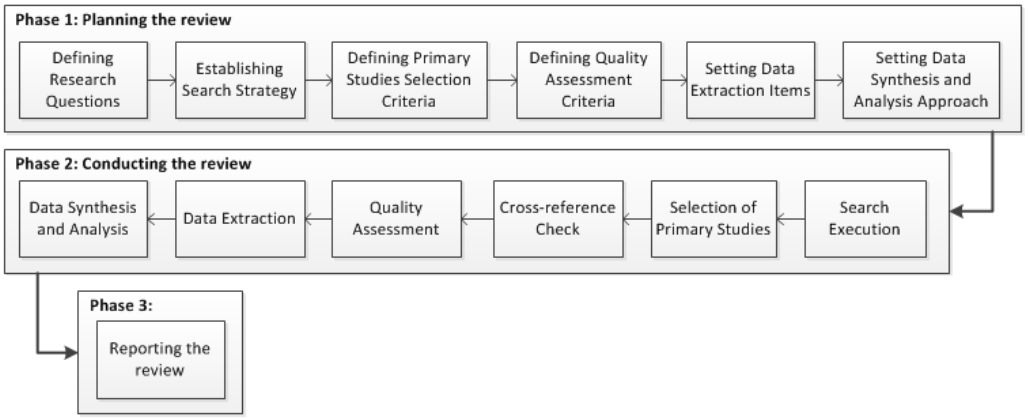

Fig. A.1. Research methodology

RQ1 is motivated by the need for defining and consequently characterising self-awareness in software systems; i.e. how to consider that a system is self-aware. RQ2 aims to identify the motivations that stimulated the adoption of computational self-awareness in software engineering. The sources of inspiration for engineering self-awareness are identified in RQ3, in order to investigate how these sources helped to advance self-aware software systems. The goal of RQ4 is to find the software paradigms that employed self-awareness and to explore the characteristics of the environments that can benefit from computational self-awareness. RQ5 identifies the approaches of engineering self-aware software systems and investigate how computational self-awareness has been realised. The aim of RQ6 is to explore the significance of adopting self-awareness in terms of performance evaluation. RQ7 aims to identify the main real-world applications in which self-awareness is adopted.

A.1.2 Search Strategy. Our search strategy includes determining the data sources and the search string, as they appear in the sub-sections below.

Data sources. The search process for this study is based on automated search in the following digital libraries and indexing systems that are considered as the largest and most complete scientific databases for conducting literature reviews in computer science [4] [12]:

- IEEE Xplore (http://ieeexplore.ieee.org/)
- ACM Digital Library (http://dl.acm.org/)
- ScienceDirect (http://www.sciencedirect.com/)
- Web of Science (http://www.webofknowledge.com/)
- SpringerLink (http://link.springer.com/)
- Wiley InterScience (http://onlinelibrary.wiley.com/)
- World Scientific (https://www.worldscientific.com/)

As technical reports do not appear in these libraries, we also considered Google Scholar (http://scholar.google.co.uk/) for this type of publications only.

Furthermore, we identified the most relevant conference proceedings and journals in the field of software engineering for manual search, according to our previous experience and the results obtained during trial searches. The full list of sources considered for manual search is presented in Table A.1.

Table A.1. Manual search sources

Type	Data Source	Publisher
Journal	ACM Transactions on Software Engineering and Methodology	ACM
	ACM Transactions on Autonomous and Adaptive Systems	ACM
	IEEE Transactions on Software Engineering	IEEE
	Journal of Software and Systems	Elsevier
	Information and Software Technology	Elsevier
	Software and System Modelling	Springer
Conference	ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)	ACM
	International Conference on Software Engineering (ICSE)	IEEE
	International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)	IEEE
	IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)	IEEE
Magazine	Software	IEEE
	Computer	IEEE
	Awareness Magazine: Self-Awareness in Autonomic Systems	European Commission (FP7)
	Dagstuhl Reports	Schloss Dagstuhl - Leibniz Center for Informatics

Search String. The aim of the search string is to capture all results related to self-awareness in the context of software engineering. Trial searches were performed in each database with the intention of checking the number of returned papers and their relevance. The objective of the trial searches is to check the feasibility of the search string and adjust it accordingly.

The general search string used on all databases is: `(self-aware*) AND (software)`. The first term captures the different ways self-awareness could be used, i.e. self-aware or self-awareness. The second term makes it explicit for software. The keywords `system(s)` and `computing` have returned a huge number of results related to computing systems, hardware, robots and networks. Other combined keywords, such as software engineering and software systems - when tried - had led to a vast wide set of irrelevant results. The simplicity and generality of the search string help in maximising the number of returned relevant papers, as it places as few restrictions as possible on the search string. We used the search string in the automated search engines of the data sources defined earlier, searching by meta-data (i.e. title, abstract and keywords).

Regarding the search in the specialised data sources, we first checked whether the papers published in these venues are retrieved in the databases included in the automated search. We found that manual search is needed only for the Awareness Magazine (http://www.awareness-mag.eu/index.php) and the Dagstuhl Reports (http://drops.dagstuhl.de/opus/institut_dagrep.php?fakultaet=07).

A.1.3 Studies Selection Criteria. After the search is executed, the study selection will be performed on the resulting set of studies. During the screening of search results, the title, abstract, introduction and conclusion for each candidate paper should be examined closely to determine the relevance of the paper. In some cases when these do not provide enough information to decide the relevance of the paper, the whole paper should be read. The selection is to be performed with respect to the inclusion and exclusion criteria defined in Table A.2.

When similar studies are reported in several papers as work-in-progress, the most comprehensive version is to be considered, unless significant details were reported in the earlier version.

Table A.2. Studies Selection Criteria

	Inclusion Criteria
I1.	Papers published in conferences and journals, as full research paper, short and position paper presenting new and emerging ideas, as well as doctoral symposiums
I2.	Literatures published as books, book chapters and technical reports
I3.	Papers employing self-awareness concepts in engineering software systems (e.g. cloud-based, service-oriented)
I4.	Papers implementing or extending self-awareness concepts
I5.	Papers discussing general or particular aspects of self-awareness
	Exclusion Criteria
E1.	Papers not in the form of a full research paper, i.e. in the form of abstract, tutorials, presentation, or essay.
E2.	Papers with abstract not available
E3.	Papers not written in English language
E4.	Papers focusing on awareness or context-, situation-awareness or any other form of awareness (i.e. not self-awareness), or not explicitly addressing self-awareness
E5.	Papers not focusing on self-awareness in software engineering; i.e. other computer science fields, such as networking or robotics or hardware

A.1.4 Cross-references Check. In order not to miss any relevant studies, we designate the cross-referencing technique to find potentially relevant studies, by applying the "snow-balling" search method [20] [3]. This is performed by tracking the references contained in the "References" section in each selected primary study [20] [3].

A.1.5 Quality Assessment Criteria. Primary studies are evaluated according to the quality assessment criteria shown in Table A.3, in order to assess the quality of the studies under consideration. The quality assessment checklist is based on the assessment method for research studies proposed in [11] [38].

Table A.3. Quality Assessment Checklist

	Quality item
QA1.	Problem definition of the study
QA2.	Reporting on background and context
QA3.	Description of the research method
QA4.	Evaluation of the research method
QA5.	Contributions of the study
QA6.	Reporting on the insights derived from the study
QA7.	Reporting on the limitations of the study and threats to validity

The scoring procedure is 1 if the quality item is present, 0.5 if it is partially present, 0 if not present or unknown. Based on that, the quality assessment score (maximum of 7) for a study is calculated by summing up the scores for all the quality items.

A.1.6 Data Extraction Items. For each selected primary study, the whole paper should be read to extract the data items, that will help in answering the research questions. Data items to be extracted and their relevant research questions are listed in Table A.4.

Table A.4. Data items extracted from Primary Studies

Data item	Description	Relevant RQ
BibTeX key	a unique key identifying the study for reference	Documentation
Title	title of the study	Documentation
Year	publication year	Demographics
Authors' affiliations	the affiliations of all authors as appearing in the study	Demographics
Affiliation Countries	the countries of the authors' affiliations	Demographics
Definition	definition of the self-awareness concept	RQ1
Characteristics	characteristics to consider software as a self-aware one	RQ1
Motivation	the motivation for employing self-awareness	RQ2
Source of inspiration	what inspired the self-aware approach	RQ3
Software Engineering Practices	the software engineering practices that employed self-awareness; i.e. requirements engineering, architecture design	RQ4
Software paradigm	which types of software considered self-awareness; i.e. cloud-, mobile-, service-based	RQ4
Engineering Approach	what is the approach used to realise self-awareness; i.e. prediction, machine learning	RQ5
Evaluation tool	how the self-aware system is evaluated; i.e. simulation, experiments	RQ6
Performance	how the self-aware system performed compared with non-self-aware	RQ6
Overhead	what is the overhead caused by self-awareness	RQ6
Applications	real-world applications adopting self-awareness	RQ7

A.1.7 Data Synthesis and Analysis Approach. Data synthesis should involve collating and summarising data extracted from primary studies. In this stage, statistics are also extracted and the results are further analysed. For the data synthesis, the extracted data should be inspected for similarities in order to define how results could be encapsulated. Our approach for synthesizing findings will be based on the synthesis method "thematic analysis/synthesis" [8], with the difference that instead of identifying themes derived from the findings reported in each primary study, we consider the synthesis and analysis targeted to answer the research questions.

A.2 Conducting the Review

This section summarises the execution of the review protocol (the second phase).

A.2.1 Search Execution. The search was executed during March 2018, then updated during June 2019 by the first author according to the search strategy defined in section A.1.2. In practice, particular settings were built for each search engine (details in Table A.5), since each digital library works in a specific manner. This was attempted to minimise duplications and rejections by setting the appropriate options in each search engine. Particularly, filters were applied - when available - for setting the search engine to retrieve only studies published by its own engine or to retrieve documents in English language only. Minimising results by excluding irrelevant disciplines was also used, whenever available. In cases where the search engine does not imply enough filters and a large number of irrelevant results were retrieved, we used the first sets of search results sorted

by relevance. This decision was made after carefully checking the next set and found complete irrelevance.

Table A.5. Search Execution (search strings and settings)

Database	Search string	Search settings
ACM Digital Library	"self-aware*" AND software	N/A
IEEE Xplore	"self-aware*" AND software	refined by Publisher: IEEE
ScienceDirect	"self-aware*" AND software	Publications tiles: Procedia Computer Science Journal of Systems and Software Future Generation Computer Systems Expert Systems with Applications Science of Computer Programming Computer Standards & Interfaces Decision Support Systems Journal of Network and Computer Applications
Web of Science	"self-aware*" AND software	Language: English Categories: COMPUTER SCIENCE THEORY METHODS COMPUTER SCIENCE INFORMATION SYSTEMS COMPUTER SCIENCE SOFTWARE ENGINEERING COMPUTER SCIENCE INTERDISCIPLINARY APPLICATIONS
SpringerLink	"self-aware*" AND software	Discipline: Computer Science SubDiscipline: SWE Language: English
World Scientific	"self-aware*" AND software	N/A
Wiley InterScience	self-aware* AND software	N/A
Google Scholar	"self-aware*" AND software	N/A

During the course of executing the search, we used a spreadsheet to keep track of the search execution process and perform quantitative analysis on the results. This spreadsheet contains:

- Data source - the name of the data source;
- URL - the URL of the data source;
- Search Query - the query string as entered to the search engine;
- Search filters - further filters used to refine the search results (e.g., language, discipline);
- Search results - the total number of search results retrieved;
- Considered results - the total number of search results considered for primary studies selection;
- Search results file - the bibliography file of the search results;

Search results were extracted as a bibliography in BibTeX format, having a final collection of bibliographies for each data source. We have also created a spreadsheet listing the search results with their meta-information. We, then, used JabRef [23] to merge the search results files into one *.bib* file after detecting and removing duplicates.

As a result of the automated search execution, we found 51,414 studies in total, without the Awareness Magazine and Dagstuhl Reports (where we performed a manual search for all articles). In case a large number of results was retrieved, we used the first sets of search results sorted by relevance. More specifically, we considered the first 200 results from each of SpringerLink, World Scientific, and Google Scholar sorted by relevance. Table A.6 shows the total results of automated search execution in each data source and the considered results.

The end results of the automated search execution are 890 studies to be considered with 25 duplicates. Then, we performed primary studies selection on the 865 candidate studies and all the

articles published in the two specialised data sources considered for manual search; i.e. 51 articles in the Awareness Magazine and 6 volumes (with 12 issues each) of the Dagstuhl Reports.

Table A.6. Automated Search Results

Database	Search results	Considered results
ACM Digital Library	55	55
IEEE Xplore	98	98
ScienceDirect	80	80
Web of Science	57	57
SpringerLink	30,430	200
World Scientific	3568	200
Wiley InterScience	0	0
Google Scholar	17,126	200
Total	51,414	890
Total after removing duplicates	-	865

A.2.2 Selection of Primary Studies. Selection of primary studies was performed using the inclusion and exclusion criteria (defined earlier in Table A.2). We used a spreadsheet to collect data related to this stage. This spreadsheet contains:

- Selection - whether the study is selected or not;
- BibTeX key - a unique key identifying the study for reference;
- Title - the title of the study;
- Year - publication year;
- I1 - I5 - whether the study fulfils the inclusion criteria;
- E1 - E5 - whether the study fulfils the exclusion criteria;

This step results in 74 selected primary studies.

A.2.3 Cross-references Check. Cross-references check was performed on the References section of each selected primary study. Bibliography data about every cited reference was collected, similar to the search results. We collected 712 new studies. Then, we performed the same study selection process as described in section A.1.3. This results in 13 more studies after removing duplicates. The final set of primary studies includes 74 studies. The reference list of the primary studies appears in Appendix A.

Study selection and cross-referencing check were completed by the first and second authors by agreement and supervised by the other authors.

A.2.4 Quality Assessment. We performed quality assessment check on the 74 collected studies, according to the criteria defined earlier in section A.1.5. Quality assessment was completed the two principal researchers by agreement, passing through extensive discussions to agree on a final score.

Figure A.2 shows the number of studies with different scores for each quality assessment criterion (described in section A.1.5 and process reported in section 4.1). The results show that researchers explicitly provide descriptions of the problem they tackle (QA1), report on background and context (QA2), as well as a description of the research method (QA3). Evaluation of the research method (QA4) and insights (QA6) are also reported, but not always explicitly. This reflects that the evaluation of self-aware software systems needs to receive more attention (as we will discuss in the next section). Providing explicit evaluation and insights is important in general. With respect to QA5, the majority of the studies did not report significant contributions to self-awareness. This reflects the need for clear vision and roadmap for self-awareness in the community (as reflected in section

6). Most studies ignore reporting limitations of the results and threats to validity (QA7). This could reflect some weaknesses to the studies addressing self-awareness in software systems. However, reporting the limitations deserves attention, as this should be part of any research study.

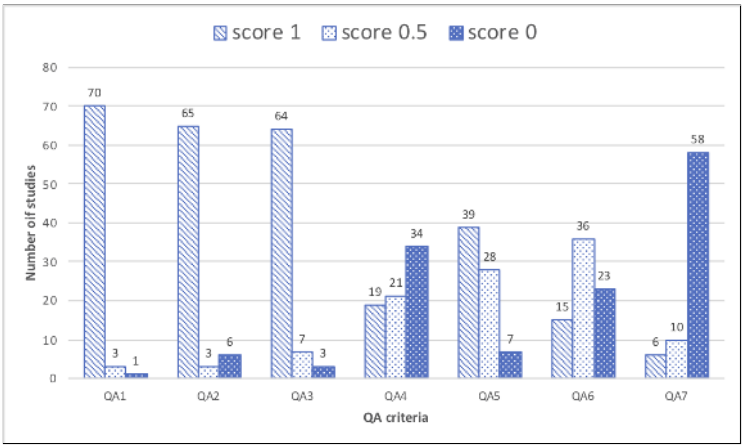

Fig. A.2. Quality Scores for the Primary Studies

The results of the quality assessment show that the average quality score is 4.74. The number of studies with respect to different total score ranges is shown in Table A.7. The quality score for the majority of studies (53 studies, 71.62%) ranges between 4.0 and 6.0 with an average of 4.73. A small percentage of studies (6 studies, 8.11%) are highly scored with an average of 6.58, and another small percentage (15 studies, 20.27%) had a score of 3.5 or lower with an average score of 2.80.

Table A.7. Quality Assessment Total Results

Total score	Number of studies	Percentage	Mean score
≥ 6.5	6	8.11	6.58
4.0 - 6.0	53	71.62	4.73
≤ 3.5	15	20.27	2.8
Average QA score			4.74

Generally, the mean quality score of the majority of studies ranging between 4.0 and 6.0 could be attributed to the quality criteria that were scored low for a large number of studies. These include the lack of strong evaluation (QA4), reporting on future insights (QA5), and reporting on limitations and threats to validity (QA7).

A.2.5 Data Extraction and Synthesis. Finally, we performed data extraction and synthesis on the primary studies. This step was performed by the two principal researchers, and supervised by the other researchers reviewing the whole process, having thorough discussions and looking iteratively at the literature. For each study, data items defined earlier in section A.1.6 were extracted and recorded in a spreadsheet.

Received ; revised ; accepted