Self-adaptation Approaches for Energy Efficiency: A Systematic Literature Review

Fahimeh Alizadeh Moghaddam
S2 & SNE research groups
Vrije Universiteit Amsterdam &
University of Amsterdam
f.alizadehmoghaddam@vu.nl

Patricia Lago
Software and Services (S2) group
Vrije Universiteit Amsterdam
The Netherlands
p.lago@vu.nl

Iulia Cristina Ban
Vrije Universiteit Amsterdam
The Netherlands
iuliacristina.ban@gmail.com

ABSTRACT

The increasing energy demands of software systems have set an essential software quality requirement: energy efficiency. At the same time, the many contextual changes faced by software systems during execution can hamper their functionality and overall quality. To address both problems, self-adaptation approaches can empower software systems, at both design-time and runtime, to adapt to dynamic conditions. In this way, software systems can be more resilient to failure, hence more trustful to satisfy the demands of modern digital society. In this paper, we perform a systematic literature review to study the state-of-the-art on existing self-adaptation approaches for energy efficiency. We analyze the identified approaches from three different perspectives, namely publication trends, application domains, and types of software systems. Our findings can help solution providers to make guided decisions to enable self-adaptability in designing and engineering software systems.

CCS CONCEPTS

• **Software and its engineering → Designing software**; **Software design engineering**;

KEYWORDS

Energy Efficiency, Self-adaptive Software, Systematic Literature Review.

ACM Reference format:
Fahimeh Alizadeh Moghaddam, Patricia Lago, and Iulia Cristina Ban. 2018. Self-adaptation Approaches for Energy Efficiency: A Systematic Literature Review. In *Proceedings of IEEE/ACM 6th International Workshop on Green And Sustainable Software , Gothenburg, Sweden, May 27–26, 2018 (GREENS'18),* 8 pages.
https://doi.org/10.1145/3194078.3194084

1 INTRODUCTION

Energy efficiency is attracting an increasing attention. For instance, the ever-increasing demand for more energy has urged the European Commission to identify energy efficiency as the primary quality requirement for achieving sustainability objectives, which is pointed out in its energy efficiency plan 2011 [6]. Information and communication technology (ICT) contributes to the total energy consumption significantly. Only in 2012 ICT consumed approximately 920 TWh, which is around 4.7% of the total produced electricity [7]. Software and hardware technologies as the main ingredients of ICT play an essential role in defining the level of energy efficiency of ICT. To improve energy efficiency both software and hardware technologies can be targeted. However, software technologies determine the way hardware is exploited [20], and as such can have a more dominant influence on reaching efficiency goals compared to hardware technologies.

Modern software systems cope with many contextual changes during execution, such as changes in environmental conditions and user requirements. Self-adaptability must be enabled for such systems to guarantee the achievement of functional and non-functional requirements. With self-adaptability, software systems can detect unexpected runtime changes and recover from those with available adaptation configurations. Runtime self-adaptation has been described in the MAPE model in the form of four main functionalities [4]: *Monitor*, *Analysis*, *Plan*, and *Execute*. These functionalities must be realized in iteration to enable an autonomic adaptation to a runtime change. There are a number of studies exploring self-adaptation approaches for software systems [10, 11, 23]. Self-adaptation techniques can be beneficial to the energy consumption figures of software systems. Self-adaptive software systems are enabled to react dynamically to the changes of energy efficiency. In this work, we review self-adaptation approaches that specially aim to improve energy efficiency. We study existing approaches to uncover the link between self-adaptability and energy efficiency in software systems. We perform a systematic literature review (SLR). The resulting primary studies are analyzed, and the findings are presented as a guideline including categorizations along various perspectives. As a result, we believe that a wide range of solution providers (from software architects to system engineers) can benefit from this guideline by adopting the best-fitting approaches depending on their own specific requirements. We gather our findings around the notion of *approaches*, which are reusable generic sets of operations. Approaches, in turn, can be adopted and specialized into *solutions*, which are technology-oriented and system-specific. Our results reveal the current and the possible future focus of software solution providers on different approach types.

The rest of the paper is organized as follows. Section 2 describes the research method we follow. In Section 3, we present the analysis of the identified primary studies. This relies on a meta-model illustrating the relevant information extracted from the primary

GREENS'18, May 27–26, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5732-6/18/05…$15.00
https://doi.org/10.1145/3194078.3194084

studies. Section 4 introduces the findings and their categorization. We discuss our main observations in Section 5, and the threats to validity are explained in Section 6. Related work on self-adaptation approaches and energy efficiency improvements is presented in Section 7, while Section 8 concludes the paper.

2 RESEARCH METHOD

We conducted an SLR to objectively select the primary studies to base our analysis on. According to [9], an SLR can be organized in four main steps:

(1) *Research Question Identification*: The main goal of this SLR is to review the existing self-adaptation approaches aimed to improve energy efficiency of software systems. We achieve this goal by formulating the following research questions:
   - **RQ1**. What are the types of approaches emerging over time? By answering this research question we aim at characterizing the intensity of scientific works targeting self-adaptation approaches over the years. This may help evaluate possible trends in the coming years.
   - **RQ2**. How can we categorize the application domains of the approaches? The answer to this question will help identify which types of approaches can be more suitable for which domains.
   - **RQ3**. How can we categorize the types of software systems targeted by past research studies? The answer to this question maps the approaches to the types of software systems, namely data-intensive or computation-intensive. This will help selecting better-fitting approaches for a given type of software system and with specific quality requirements.

   The research questions should be answered by extracting the relevant information from the primary studies. Identifying the research questions helps us with the next steps, to select and assess the primary studies.

(2) *Search Strategy Planning*: Thanks to its accessibility, we have selected Google Scholar as the source of relevant studies in our paper. To run the search, we have constructed a search query with specific syntax suitable for Google Scholar:
   ("(energy OR power) (efficiency OR efficient OR saving)") AND ("self-adaptive (software OR system OR framework)" OR "(software OR system OR framework) self-adaptation" OR "self-adaptive application")
   We started with 687 studies as the result of running the search query above[1]. The relevant studies are retrieved if they contain the keywords set up to cover the research question. To ensure the effectiveness of our query, we checked the results against a list of pilot studies that we identified based on our knowledge of the field. Pilot studies are in fact primary studies that should appear among the results of the query execution.

(3) *Study Selection*: We assessed the retrieved studies according to our inclusion and exclusion criteria, which are summarized in Table 1. Basically, we include studies that specialize self-adaptive approaches for energy efficiency and explicitly evaluate their effectiveness on energy consumption. Generic self-adaptive approaches that can be applicable for energy efficiency requirements are not in the scope of this study.
   For efficiency reasons, we have performed the assessment in three rounds: title-based, abstract-based, and body-based. In each round, a number of irrelevant studies were excluded, and the remaining studies were used as input for the next round. The last round resulted in 95 primary studies, which we have used for our analysis. The list of the primary studies is available online[2].

(4) *Primary Studies Management*: During the study selection phase, we used Google Scholar, in which we could assign customized labels to each retrieved study. During the analysis phase, we used an Excel sheet to store relevant information extracted from each primary study.

3 DATA ANALYSIS

We used the coding method [22] to systematically analyze the primary studies. Accordingly, one assigns codes that emerge from the text in each primary study. The codes help compare and discover the similarities between the primary studies. We categorize the extracted codes further to identify more generic concepts. This process resulted in the metamodel (see Figure 1) that illustrates the uncovered concepts and their relations. Among them we identify five main classes:

- *Objective*: This is the goal that each study aims to achieve, whether it is linked to self-adaptability, energy efficiency, or both. *Energy-related objectives* and *self-adaptability objectives* indicate the two types of objectives that have been addressed. Objectives must be specified before initiating the process of forming approaches. At a higher level, objectives can arise from a financial issue. For instance, an organization wants to lower the energy bills. Thus, its objective can be minimizing the energy consumption of its infrastructure, which is an energy-related objective. In another example, for resource-scarce environments, the efficient utilization of computing resources is a necessity. In these cases, software systems have the self-adaptability objectives. It should be noted that as a by-product the energy efficiency of such systems improves.
- *Constraint*: This is a limitation to customize approaches due to environmental conditions and user requirements. For example, financial limitations can narrow down the range of possible approaches to be adopted. Constraints can be zoomed in to more details and restrict a selected approach. For instance, if the users of an application have strict requirements on the output performance, then the energy-related solutions can only hinder performance up to a certain level.
- *Approach*: This is a viewpoint for solving the stated problem in the primary studies and achieving the objectives. An approach comprises an ordered set of (usually high-level) operations designed to deliver the proposed solution. It is a process of working through the details of a problem to realize the proposed solution. With an example we can describe an approach and its relation to a corresponding solution. Assume there is a data center framework that aims to optimize allocation of resources for input workload, which can have

[1]The search query was executed in Google Scholar on January 29, 2018.

[2]The list of the primary studies: https://tinyurl.com/yd5x6cb8

Table 1: Inclusion and exclusion criteria for study selection

#	*Criterion*	*Description*
		Inclusion Criteria
I1	The study is written in English.	For readability purposes, we include only studies that are in English.
I2	The study is peer-reviewed.	We ensure the quality of the primary studies with selecting only from peer-reviewed studies.
I3	Self-adaptability is investigated in the study.	Self-adaptation approaches are the focus of our study. We investigate approaches enabling self-adaptability in software systems at different levels (from software design to system engineering)
I4	The main focus of the study is energy efficiency.	The objective of the studies is to improve on energy efficiency. We also take into account other energy-related concepts, such as power efficiency and energy consumption.
		Exclusion Criteria
E1	The body of the study is available.	The body text of the studies must be available. We exclude the studies that can be retrieved publicly. It should be mentioned that with our university license, we can access top journal publications and conference proceedings.
E2	The study does not provide a solution.	The studies must contribute with a solution to the field. For instance, we exclude discussion studies that do not provide self-adaptation solutions for energy efficiency.
E3	The study does not contribute to the link between energy efficiency and self-adaptability	We exclude the studies that do not investigate the link between energy efficiency and self-adaptability.

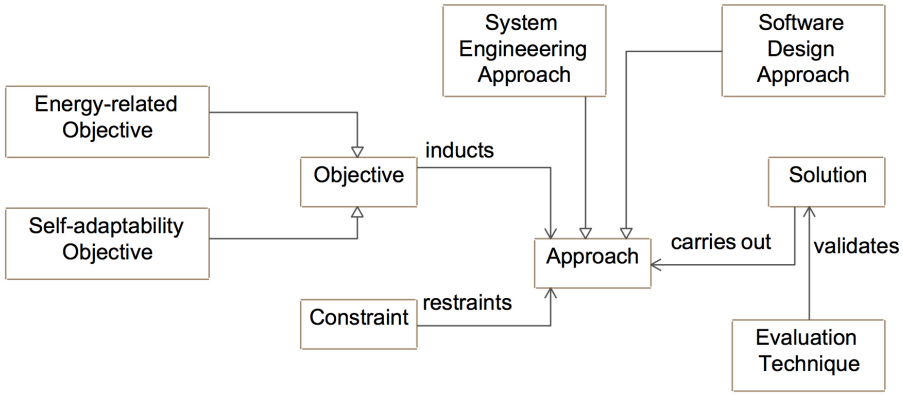

Figure 1: The metamodel presenting the concepts and their relations extracted from the primary studies

a varying pattern during execution. A possible approach to enable self-adaptability for such framework is to implement pattern recognition models that can identify the pattern of each workload, and dynamically select the optimum allocation plan. A solution for this example could be the specific recognition algorithms implemented in the framework.
We identify two types of approaches in the primary studies: *software design approaches* are concerned with the design of software systems throughout their entire life-cycle; *system engineering approaches*, instead, focus on runtime system-level adaptation configurations.

- *Solution*: This is a specific low-level method to solve a problem. Solutions realize approaches, which are generic and applicable for target software systems. For instance, in cyber-foraging an implemented code partitioning algorithm is a self-adaptation solution, while the approach is "to offload the computation".
- *Evaluation Technique*: This is a technique adopted to evaluate the outcome of the proposed solution in each primary study. The evaluation technique determines the degree of improvement on energy efficiency when the self-adaptation solution is applied. A wide range of techniques has been used in our primary studies, such as qualitative discussions, mathematical formulations, empirical experiments, and simulation experiments.

4 RESULTS

The analysis of the primary studies help answer our research questions. The answer to the research questions will help software architects and software engineers to choose the best fitting approaches for their software systems.

RQ1. What are the types of approaches emerging over time?
As mentioned before, we have identified two types of approaches, namely software design approaches (SDA) and system engineering approaches (SEA). Some primary studies propose solutions that realize both types of approaches (SEA+SDA). Figure 2 shows the distribution of each type of approaches in the list of our primary studies over the years. As shown in the figure, there is a growing trend for the both SEA and SDA approach types until 2014, when we witness a peak in adoption of the SEA and SDA types. The second peak is in 2017, in which all the types of approaches have been investigated. Mostly, the approach type SEA is adopted more than other types in each year. We see that only in 2012 and 2015, the primary studies turn their attention more towards the type SDA. This means that a lot of effort has been devoted to system engineering approaches in order to realize runtime adaptation. Many scheduling algorithms and infrastructure configuration mechanisms are implemented at this stage. We expect to see an increase in growth for all types of approaches in the coming years. An example for type SEA is introduced in [16]. They provide a self-adaptive framework that adjusts cluster configurations during runtime, to cover heterogeneity and cluster status variations. The framework has built-in learning algorithms to compare current and previous configurations, and predict cluster performance. All the MAPE functionalities are realized with including essential components such as a monitoring network, a dynamic predictive model, and a runtime scheduler. Differently, Perez-Palacin et al. [18] present an approach of type SDA. They introduce a reference architecture, or an adaptation framework, based on a three-layer software architecture for self-managed systems. The reference architecture uses model-driven techniques to transform design patterns into different Stochastic Petri Nets subnets.

We have identified a few studies of type SEA+SDA. For instance, Cioara et al. [5] propose a methodology for energy-aware management in data centers. They have defined an ontological model for representing the key indicators in data centers, namely energy and performance. The model helps calculate the level of energy efficiency in data centers at runtime and apply reconfigurations when needed.

RQ2. How can we categorize the application domains of the approaches?
The self-adaptation approaches are generic guidelines that can be applied to different domains. From the primary studies we identify several application domains the approaches are customized for. Figure 3 shows the popularity of each type of self-adaptation approache for different application domains. The cloud computing domain has benefited the most from the all types of approaches. The second popular domains are computer systems and wireless sensor networks (WSN), in which many approaches focus on runtime resource optimizations. In particular, the limited battery life of sensors in WSN has motivated a lot of effort on runtime adaptations. In general, we can see from the figure that the approaches of the type SEA are usually proposed for specific domains. The reason is that these approaches are mostly technology-driven, and hence domain-dependent.

Another observation is that many of the proposed approaches are not proposed for a specific domain. We label these approaches as *Domain-independent* that enable self-adaptability from a general-purpose point of view. For instance, a primary study that introduces an online learning algorithm to make dynamic adaptation decisions falls under this category. The domain-independent category itself can be sub-categorized into more detailed categorizations such as requirements engineering, software design, and optimization algorithms. It should be noted that general-purpose approaches are mostly from type SDA focusing on self-adaptation software architecture.

Type SEA+SDA has been specified for the cloud computing domain. This domain has attracted the highest number of approaches, which is an indicator of the the maturity of the cloud computing domain in general. Table 2 illustrates the primary studies focusing on each application domain.

RQ3. How can we categorize the types of software systems targeted by the primary studies?
The ratio between the size of computed data and the size of computing power can show the types of software systems, which are namely, data-intensive and computation-intensive. Most primary studies do not specify a system type, classified as *Type-independent*. Figure 4 shows the classification of the primary studies based on the types of software systems. From an overall perspective we see that computation-intensive tasks are mostly the target for self-adaptation approaches. We see this pattern for the types SEA and SEA+SDA. However, the same pattern does not apply to the type SDA, in which approaches are mostly focused on data-intensive tasks rather than computational-intensive ones. Table 3 shows the primary studies that are proposed for the two system types.

5 DISCUSSION

By zooming into the self-adaptation approaches, we can add more details to each approach type as listed in Table 4. In particular, *software design approaches* (SDA) target the software design at different stages ranging from software development life-cycle to execution time. We have identified three categories of SDA approaches on *requirements engineering*, *architecture modeling*, and *architecture reconfiguration*. For instance, Bencomo *et al.* present a requirement engineering approach that systematically and incrementally builds software architectures from system goals [3]. They use goal-based requirements specification for self-adaptive software systems. *Architecture modeling* approaches introduce reference architectures that guide software architects in building self-adaptive software systems. The design patterns proposed by Said *et al.* for self-adaptive real-time embedded systems fall under this category of approaches [21]. Our analysis show that the proposed reference architectures mostly adopt a layered architectural style. The second most common architectural style is the service-oriented style that with an automated composition of software services can seamlessly adapt to runtime changes. Another interesting category of the type SDA is *architecture reconfiguration*, which suggests modifications to the

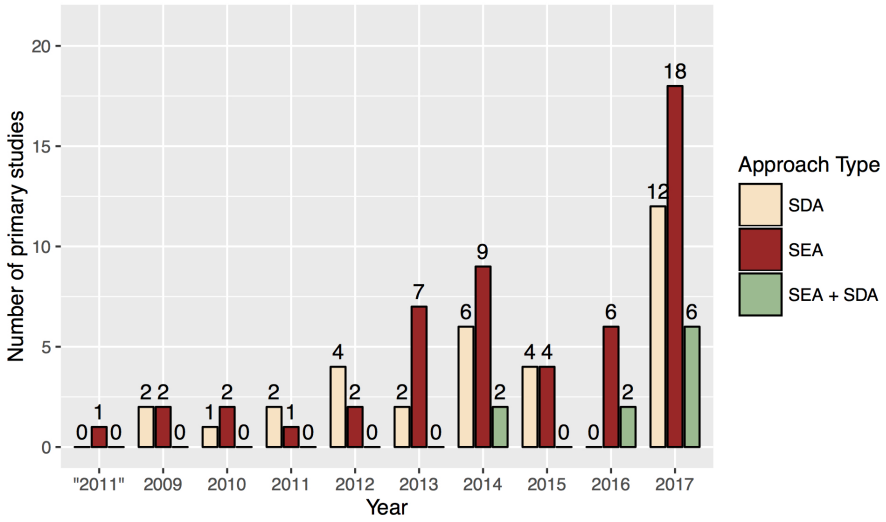

Figure 2: The emerging trend of the adopted approaches throughout the years. Different colors indicate different types of approaches, namely software design approaches (SDA), system engineering approaches (SEA), and both (SEA+SDA).

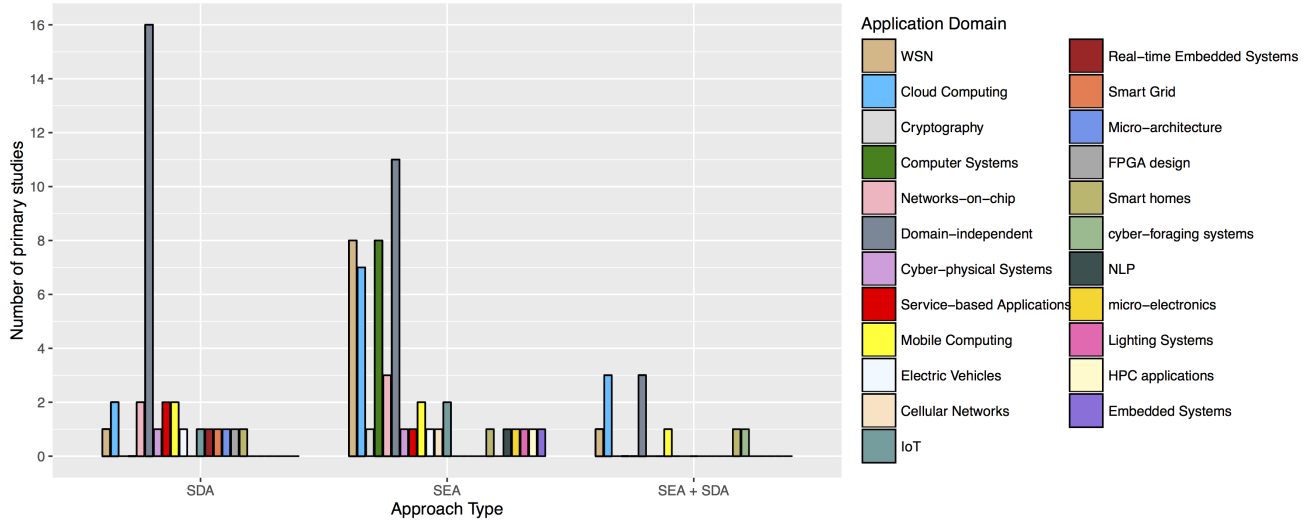

Figure 3: The distribution of the types of the self-adaptation approaches in different application domains.

software architecture at runtime. A nice example for this category is the work by Mizouni *et al.* that presents a battery-aware cyber-foraging mobile application [14]. In their work, architecture reconfigurations are introduced in the form of features availability in the application, at both the mobile side and the surrogate side.

System engineering approaches (SEA) target the MAPE model functionalities at the system level. The type SEA includes the specific solutions to realize each of the MAPE functionalities in terms of specific tools and methods. We have identified two categories of SEA approaches namely, *MAPE functionalities* and *system architecture*. Self-adaptation approaches that focus on the seamless adaptation at runtime, usually cover all the MAPE functionalities at once. However, we have seen some approaches that investigate on only one functionality. For example, Mishra *et al.* propose a probabilistic graphical model-based learning algorithm to enable energy savings [13]. Another category of the type SEA is *system architecture*, which introduces an architecture for tools and technologies to enable self-adaptability. The work presented by Alzamil *et al.* is an example of this type [2]. They propose a system architecture for profiling and assessing the energy efficiency of cloud infrastructure resources.

Table 4 lists the primary studies relevant for each category of approaches. As shown in the table, some primary studies fall under more than one categories. Most of the primary studies from

Table 2: The application domains identified in the primary studies

Application Domain	*Approach Types*	*Primary Studies*	*#*
Domain-independent	SEA / SDA / SEA+SDA	P11, P16, P21, P28, P31-P32, P36-P37, P40-P44, P49, P51, P55-P61, P64, P66, P73, P75, P79, P87-P88, P95	30
Cloud Computing	SEA / SDA / SEA+SDA	P3, P6, P7, P15, P23, P25, P27, P33, P38, P45, P82, P91	11
WSN	SEA / SDA	P1, P2, P5, P8, P25, P46, P65, P85, P93	9
Computer Systems	SEA	P9, P19-P20, P22, P47, P52, P76, P81	8
Networks-on-chip	SDA / SEA	P10, P30, P78, P63, P67, P68	6
Mobile Computing	SEA / SDA	P14, P29, P50, P54, P62	5
Service-based Applications	SEA / SDA	P13, P18, P53	3
Smart Homes	SDA / SEA / SEA+SDA	P71, P83-P84	3
IoT	SEA / SDA	P26, P34, P80	3
Cyber-physical Systems	SDA / SEA	P12, P90	2
Electric Vehicles	SDA / SEA	P17, P86	2
Real-time Embedded Systems	SDA	P39	1
Smart Grid	SDA	P48	1
Micro-architecture	SDA	P69	1
FPGA	SDA	P70	1
Cyber-foraging Systems	SEA+SDA	P72	1
NLP	SEA	P74	1
Micro-electronics	SEA	P77	1
Cryptography	SEA	P4	1
Cellular Networks	SEA	P24	1
Lighting Systems	SEA	P89	1
HPC	SEA	P92	1
Embedded Systems	SEA	P94	1

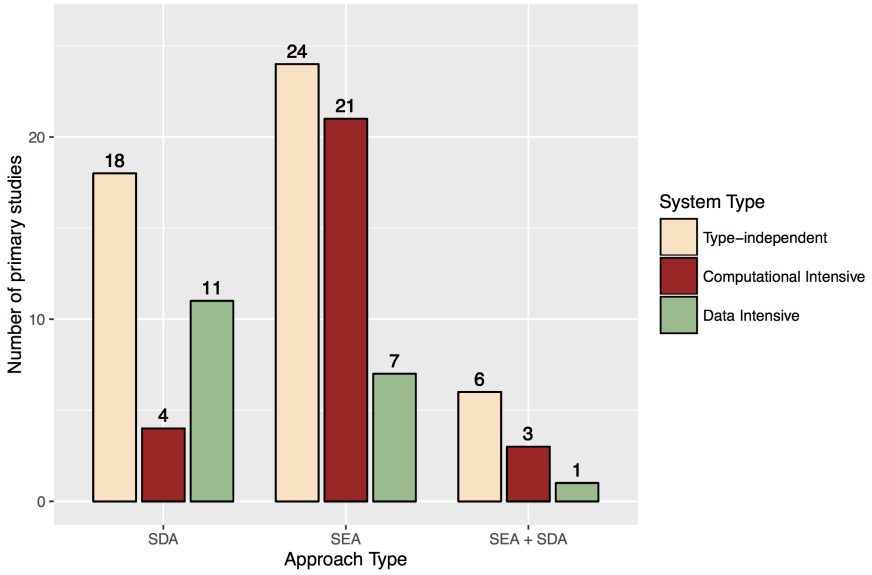

Figure 4: The distribution of types of software systems targeted by the primary studies.

the type SEA focus on the *MAPE functionalities* rather than their *system architectures*. Differently, the primary studies from the type SDA focus more on the *architecture modeling* rather than other categories. The few number of primary studies on the *architecture reconfiguration* indicates that it is a young field, and more advances on this category will emerge in the following years.

Table 3: The types of system tasks identified in the primary studies

Software System Type	*Approach Types*	*Primary Studies*	*#*
Computation-intensive	SEA / SDA / SEA+SDA	P2-P3, P5-P15, P23-P25, P27-P28, P33, P41, P45, P47, P52, P54, P56-P57, P59, P76	28
Data-intensive	SEA / SDA / SEA+SDA	P18, P26, P30, P32, P34-P35, P37-P39, P43-P44, P46, P49-P51, P53, P58, P60-P61	19

Table 4: The categorization of self-adaptation approaches identified in the primary studies

Approach Type	*Approach Category*	*Primary Studies*	*#*
Software Design (SDA)	Architecture Modeling	P13-P14, P16-P17, P27-P28, P30, P33-P35, P38-P40, P43, P48, P57-P58, P62, P64, P67-P70, P75, P82, P84, P87, P90, P95	29
	Requirement Engineering	P7, P21, P37, P42, P53, P55, P66, P79, P83	9
	Architecture Reconfiguration	P31, P50, P60, P61, P72	5
System Engineering (SEA)	MAPE Functionalities	P1-P12, P15, P18-P20, P22-P29, P32, P36, P41, P43-P44, P46-P47, P49, P51-P52, P54, P56, P59, P62-P63, P65, P67, P71, P73-P78, P80, P82-P83, P85, P88-P89, P91-P93	57
	System Architecture	P3, P8, P19, P36, P45, P47, P54, P65, P81, P86, P91, P94	12

6 THREATS TO VALIDITY

In the following, we identify a number of threats to the internal and external validity of our work, and explain our mitigation strategy.

The threats to **internal validity** target the design and execution of the review. To prevent possible design errors, we have followed the steps described in the systematic literature review protocol summarized in Section 2. In addition, to mitigate a possible subjective execution of our review, three researchers independently followed the inclusion/exclusion criteria used to select the primary studies.

The threats to **construct validity** target the alignment between the research question and the measurements of the review. We evaluate the effectiveness of our search query with a list of pilot studies that has proven to answer the research query based on our knowledge in the field.

The threats to **external validity** target the generalizability of our results. To mitigate this type of threats, we have used generic keywords to construct a search query that can capture as many relevant studies as possible. Furthermore, we have validated the list of obtained primary studies against a list of pilot studies as an objective common ground in the field.

7 RELATED WORK

The related works can be considered from two different perspectives, namely self-adaptability and energy efficiency. Each perspective has received a lot of attention from researchers in the last few years but we could not find any research paper specifically focusing on the link between the two perspectives.

From the perspective of **self-adaptability**, Macias-Escriva *et al.* review various methods and techniques employed in the design of self-adaptive systems [11]. They evaluate current progress on self-adaptability from the viewpoint of computer sciences and cybernetics, based on the analysis of state-of-the-art strategies reported in the literature. Krupitzer *et al.* present a comprehensive taxonomy for self-adaptation. They further provide a structured overview on approaches for engineering self-adaptive software systems [10]. Yang *et al.* conduct a systematic literature review to explore the modeling methods and the activities regarding requirement engineering for self-adaptive systems [23]. They also specify the application domains and the quality attributes that the primary studies focus on. Mahdavi-Hezavehi *et al.* review existing architecture-based methods to achieve multiple quality attributes in self-adaptive software systems [12]. Muccini *et al.* explore the existing self-adaptive approaches for cyber-physical systems [15]. They identify the self-adaptation concerns and application domains covered by the primary studies. Energy efficiency has been observed as the most dominant application domain for cyber-physical systems.

In turn, from the perspective of **energy efficiency**, Hammadi and Mhamdi [8] conduct a survey on energy efficiency solutions in data centers such as virtualization, energy-aware routing, dynamic voltage/frequency scaling, dynamic power management, renewable energy supply, and energy-aware cooling systems. Orgerie *et al.* survey the solutions to improve the energy efficiency of computing and communication resources in distributed systems [17]. Alizadeh Moghaddam *et al.* carried out a systematic literature review of the energy-efficient networking solutions in cloud-based environments [1]. Pinto and Castor provide an overview on the contributions of software engineering community to improve energy-related qualities [19]. They mostly focus on the two main issues in this domain, namely: *lack of knowledge* and *lack of tools*.

Differently from all other surveys we found in the literature, the work presented in this paper focuses on the link between self-adaptability and energy efficiency. In our study, we distinguish between software design approaches and system engineering approaches that can make our findings beneficial to both software architects and system engineers.

8 CONCLUSION

The energy consumption of software systems has been the focus of many research studies. Improvements in energy efficiency can result from applying self-adaptation to software systems. We believe the link between self-adaptability and energy efficiency is a promising target for building energy-aware software systems. Therefore, we carry out a systematic literature review to identify pre-existing approaches that focus on this link. We, in particular, look for *self-adaptation approaches for energy efficiency in software systems*. Our findings from the SLR indicate two types of self-adaptation approaches that have been implemented by the primary studies, namely software design approaches (SDA) and system engineering approaches (SEA). The publication trends of the primary studies show an increasing growth in the adoption of the self-adaptation approaches over the years. In particular, more attention has been paid to the type SEA that focuses on runtime realization of the MAPE (monitor-analyze-plan-execute) model functionalities.

Our results show that most approaches are application domain-independent, i.e. can be applied to any domain. Cloud computing is the most common application domain as the target of the approaches. Also, our findings show that computation-intensive software systems have attracted the highest attention of research works so far.

Our analysis show that four primary studies address runtime *software architecture reconfiguration*, which we identify as one of the categories of the type SDA. In the coming years, we expect to see more attention dedicated to this category of approaches as a response to the increasing maturity of this field. As our future work, we aim to examine the effectiveness of software architecture reconfiguration combined with runtime adaptation mechanisms.

ACKNOWLEDGEMENT

We thank Sarah Haddou for her assistance in collecting and selecting the primary studies.

REFERENCES

[1] Fahimeh Alizadeh Moghaddam, Patricia Lago, and Paola Grosso. 2015. Energy-efficient networking solutions in cloud-based environments: A systematic literature review. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 64.

[2] Ibrahim Alzamil, Karim Djemame, Django Armstrong, and Richard Kavanagh. 2015. Energy-aware profiling for cloud computing environments. *Electronic Notes in Theoretical Computer Science* 318 (2015), 91–108.

[3] Nelly Bencomo, Paul Grace, and Peter Sawyer. 2009. Revisiting the relationship between software architecture and requirements: the case of dynamically adaptive systems. In *Self-Organizing Architectures SOAR 2009 Workshop at Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), WICSA/ECSA.*

[4] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. 2009. Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*. Springer, 48–70.

[5] Tudor Cioara, Ionut Anghel, and Ioan Salomie. 2017. Methodology for energy aware adaptive management of virtualized data centers. *Energy Efficiency* 10, 2 (2017), 475–498.

[6] EC (European Commission). 2011. Energy Efficiency Plan 2011. (2011). Retrieved August 21, 2017 from https://ec.europa.eu/clima/sites/clima/files/strategies/2050/docs/efficiency_plan_en.pdf

[7] EINS Consortium et al. 2013. Overview of ICT energy consumption (D8. 1). *Report FP7-2888021. European Network of Excellence in Internet Science* (2013).

[8] Ali Hammadi and Lotfi Mhamdi. 2014. A survey on architectures and energy efficiency in data center networks. *Computer Communications* 40 (2014), 1–21.

[9] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology* 51, 1 (2009), 7–15.

[10] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.

[11] Frank D Macías-Escrivá, Rodolfo Haber, Raul Del Toro, and Vicente Hernandez. 2013. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications* 40, 18 (2013), 7267–7279.

[12] Sara Mahdavi-Hezavehi, Vinicius HS Durelli, Danny Weyns, and Paris 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.

[13] Nikita Mishra, Huazhe Zhang, John D Lafferty, and Henry Hoffmann. 2015. A probabilistic graphical model-based approach for minimizing energy under performance constraints. In *ACM SIGPLAN Notices*, Vol. 50. ACM, 267–281.

[14] Rabeb Mizouni, M Adel Serhani, Abdelghani Benharref, and Oubai Al-Abassi. 2012. Towards battery-aware self-adaptive mobile applications. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*. IEEE, 439–445.

[15] Henry Muccini, Mohammad Sharaf, and Danny Weyns. 2016. Self-adaptation for cyber-physical systems: a systematic literature review. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 75–81.

[16] Xinyu Niu, Kuen Hung Tsoi, and Wayne Luk. 2012. Self-Adaptive Heterogeneous Cluster with Wireless Network. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 306–311.

[17] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. 2014. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 47.

[18] Diego Perez-Palacin, Raffaela Mirandola, and José Merseguer. 2012. QoS and energy management with Petri nets: A self-adaptive framework. *Journal of Systems and Software* 85, 12 (2012), 2796–2811.

[19] Gustavo Pinto and Fernando Castor. 2017. Energy efficiency: a new concern for application software developers. *Commun. ACM* 60, 12 (2017), 68–75.

[20] Giuseppe Procaccianti, Patricia Lago, Antonio Vetro, Daniel Méndez Fernández, and Roel Wieringa. 2015. The green lab: Experimentation in software energy efficiency. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. 941–942.

[21] Mouna Ben Said, Yessine Hadj Kacem, Mickaël Kerboeuf, Nader Ben Amor, and Mohamed Abid. 2014. Design patterns for self-adaptive RTE systems specification. *International Journal of Reconfigurable Computing* 2014 (2014), 8.

[22] Anselm L Strauss. 1987. *Qualitative analysis for social scientists*. Cambridge University Press.

[23] Zhuoqun Yang, Zhi Li, Zhi Jin, and Yunchuan Chen. 2014. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 55–71.