Systematic scalability analysis for microservices granularity adaptation design decisions

Hassan, Sara; Bahsoon, Rami; Buyya, Rajkumar

DOI:
10.1002/spe.3069

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Hassan, S, Bahsoon, R & Buyya, R 2022, 'Systematic scalability analysis for microservices granularity adaptation design decisions', *Software: Practice and Experience*, vol. 52, no. 6, pp. 1378-1401. https://doi.org/10.1002/spe.3069

Received: 31 March 2021 | Revised: 29 November 2021 | Accepted: 25 December 2021

DOI: 10.1002/spe.3069

RESEARCH ARTICLE

WILEY

Systematic scalability analysis for microservices granularity adaptation design decisions

Sara Hassan[1] | **Rami Bahsoon**[2] | **Rajkumar Buyya**[3]

[1]School of Computer Science and Digital Technology, Birmingham City University, Birmingham, UK

[2]School of Computer Science, University of Birmingham, Birmingham, UK

[3]Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Parkville, Victoria, Australia

Correspondence
Sara Hassan, School of Computing and Digital Technology, Birmingham City University, Curzon St B4 7XG, Birmingham, UK.
Email: sara.hassan@bcu.ac.uk

Abstract

Microservices have gained wide recognition and acceptance in software industries as an emerging architectural style for autonomous, scalable and more reliable computing. A critical problem related to microservices is reasoning about the suitable granularity level of a microservice (i.e., when and how to merge or decompose microservices). Although scalability is pronounced as one of the major factors for adoption of microservices, there is a general gap of approaches that systematically analyse the dimensions and metrics, which are important for scalability-aware granularity adaptation decisions. To the best of our knowledge, the state-of-art in reasoning about microservice granularity adaptation is neither: (1) driven by microservice-specific scalability dimensions and metrics nor (2) follow systematic scalability analysis to make scalability-aware adaptation decisions. In this article, we address the aforementioned problems using a two-fold contribution. Firstly, we contribute to a working catalogue of microservice-specific scalability dimensions and metrics. Secondly, we describe a novel application of scalability goal-obstacle analysis for the context of reasoning about microservice granularity adaptation. We analyse both contributions by comparing their usage on a hypothetical microservice architecture against ad-hoc scalability assessment for the same architecture. This analysis shows how both contributions can aid making scalability-aware granularity adaptation decisions.

KEYWORDS

goal-oriented analysis, guidance, microservices, scalability, systematic analysis

1 | INTRODUCTION

Microservices have recently made their way to various important industries such as transport,[1] entertainment[2–4] and retail.[5–7] At a very high level, microservices can be regarded as autonomous, replaceable and deployable artefacts that encapsulate distinct functionalities.[8] Microservices typically interact through API gateways which usually hide internal implementation logic.

In Reference 8, we term the transition to microservices as microservitisation. In Reference 9, we define microservitisation as a paradigm shift where services/components are transformed into microservices—a more fine-grained and

autonomous form of services to introduce added value to the architecture. The added value can take several forms such as added replaceability of the service(s), better service traceability and better quality-of-service (QoS) provision under different workloads. This shift involves dramatically changing how a plethora of technical activities (including but not limited to architectural design, development and deployment) are carried out.[9]

Microservitisation takes into consideration the alignment of technical decisions in these activities with a microservice adopter's business objectives. For example, microservitisation involves designing microservices so that they isolate distinct functionalities. This isolation can introduce added value along a number of dimension such as better replaceability or better traceability. This value would be a result of the flexibility of the microservice architecture and thereby its ability to cope with operation, maintenance and evolution uncertainties. Ultimately, this can also relate to reduced maintenance costs and cost-effective QoS provision.

Among the critical microservitisation design decisions is reasoning about the suitable granularity level of a microservice. A granularity level determines "the service size and the scope of functionality a service exposes.[10,p. 426]" Granularity adaptation entails merging or decomposing microservices at any stage of the microservice lifecycle be it in the design or runtime phases. Granularity adaptation involves changing the encapsulation of functionalities across microservices, thereby moving to a finer or coarser grained granularity level. We do not restrict the mode of this change in our assumed definition of granularity adaptation. In other words, merging/decomposing can happen at the source code, deployment and/or physical levels.

Granularity adaptation is critical because "splitting (microservices) too soon can make (other design and operational decisions) very difficult to reason about.[5]" For example, microservice granularity can affect designing the communication links between microservices. Granularity can also affect the choice of containers to host these microservices. Different container choices carry different operational costs.

It will likely happen that you (the software architect) will learn in the process of designing the microservice, hence calling for adapting its granularity both during the design time and runtime phases.

Among the main adopters of microservices are highly competitive industries (e.g., Netflix,[3] Amazon[7] and BBC[2]) which are characterised by scale. Therefore, scalability of a microservice architecture is among the architecturally important requirements that is sought after by microservice adopters. Analysis of microservice-specific literature in Reference 9 concurs with this statement. One of the research objectives of the microservice-specific systematic mapping study reported on in Reference 9 is the following: "which quality attributes are considered when reasoning about microservice granularity and how are they captured?" Scalability was reported in Reference 9 as the most commonly considered quality attribute in the field of microservices when reasoning about microservice granularity. Henceforth, it is essential that microservice granularity adaptation decisions shall be aware of the dimensions and metrics that are important for the scalability of a microservices design. In other words, it is essential that microservice adopters make scalability-aware granularity adaptation decisions. Examples of dimensions that are important for the scalability of a microservice architecture include: workload, the number of logical and physical dependencies across microservices, domain-specific factors (e.g., copyright costs for media content streaming) and the volume of data accessed and/or shared across microservices. This article aims to address the following gap in analysing the scalability of a microservice design: *Inad-granularity adaptation decisions*: The current state-of-art shows inadequacies of knowledge and awareness *equacy of guidance regarding the dimensions and metrics which are important for making scalability-aware* among microservice adopters, regarding the scalability dimensions and metrics that should drive the microservice design in general and micorservice granularity adaptation in particular. "The problem is that scalability is so dependent on the application domain and the system's goals that accommodating all dimensions in pre-defined categories is very challenging, if not impossible.[11,p. 18]" This challenge is critical in microservice architectures; a large number of simultaneous inevitable scalability dimensions and metrics can affect the scalability of a microservice architecture.

Consider a fictional running microservice-based application with a functionality and scale similar to Netflix—called NetWatch. The dimensions affecting NetWatch's scalability include logical interdependencies across microservices and the volume of data shared across them. If granularity adaptation decisions for Netwatch microservices consider scalability with respect to logical interdependencies, then the decisions suggested can be less scalability-aware and might lead to less added value for NetWatch. A relatively more scalability-aware decision would consider scalability with respect to both the logical dependencies and shared data volumes.

To address the above gap, the novel contribution of this article is two-fold:

- A working catalogue of microservice-specific scalability dimensions and metrics. The catalogue builds on a previous systematic mapping study by the authors for the state-of-the-art in reasoning about microservice granularity

adaptation.[9] This catalogue assists in identifying the microservice-specific dimensions and metrics which are important for the scalability of a given microservice architecture and thereby essential for making scalability-aware granularity adaptation decisions. Given this catalogue, it is up to microservice adopters to tailor monitoring tools they utilise in order to track the dimensions and measure metrics they consider relevant from this catalogue. The scope of our catalogue is to provide guidance on *which* dimensions and metrics are relevant rather than *how* to track and measure them.

- An application of scalability goal-obstacle analysis[11,12] in a new context that relates to reasoning about scalability-aware microservice granularity adaptation. The objective is to assist software architects in systematically identifying and analysing the goals that are important to the scalability of a microsevice architectures and the obstacles to these goals. Obstacles are indications of risks that can obstruct satisfiability of scalability goals. Henceforth, the systematic analysis can justify the importance of considering a given scalability dimension when making a granularity adaptation decision. The goal-obstacle analysis for microservice scalability is informed by dimensions from the catalogue. Scalability goal-obstacle analysis was first attempted in References 11,13; it is inspired by keep-all-objectives-satisfied (KAOS) goal-oriented modelling.[14] The input to this analysis is a refined KAOS goal-oriented model of a system. Consequently, scalability goal-obstacle analysis aims to systematically identify, assess, and resolve potential obstacles which can obstruct the system from satisfying its goals if it was scaled along relevant dimensions. Scalability goal-obstacle analysis "attempts to establish a uniform notion of scalability that can be applied in a wide variety of application domains and can support analysis of scalability with respect to a wide variety of system qualities.[13,p. 383]" The generality of scalability goal-obstacle analysis makes it more applicable to our context than other domain-specific approaches that assess scalability (see, e.g., References 15–17). Systems in those domains are fundamentally different from those targeted in our context so scalability assessment approaches developed for these domains are not transferable to our context. Furthermore, scalability goal-obstacle analysis by definition has distinct steps to identify, assess and resolve scalability-obstacles of a system.

Our catalogue acts as a pre-requisite for scalability goal-obstacle analysis. The catalogue can guide the microservice architects to identify the important scalability dimensions and metrics thereby scoping down the goal-obstacle analysis of a microservice architecture by providing more principled input to the analysis. Consequently, scalability goal-obstacle analysis identifies and analyses obstacles along the dimensions from our catalogue. Figure 1 illustrates input,

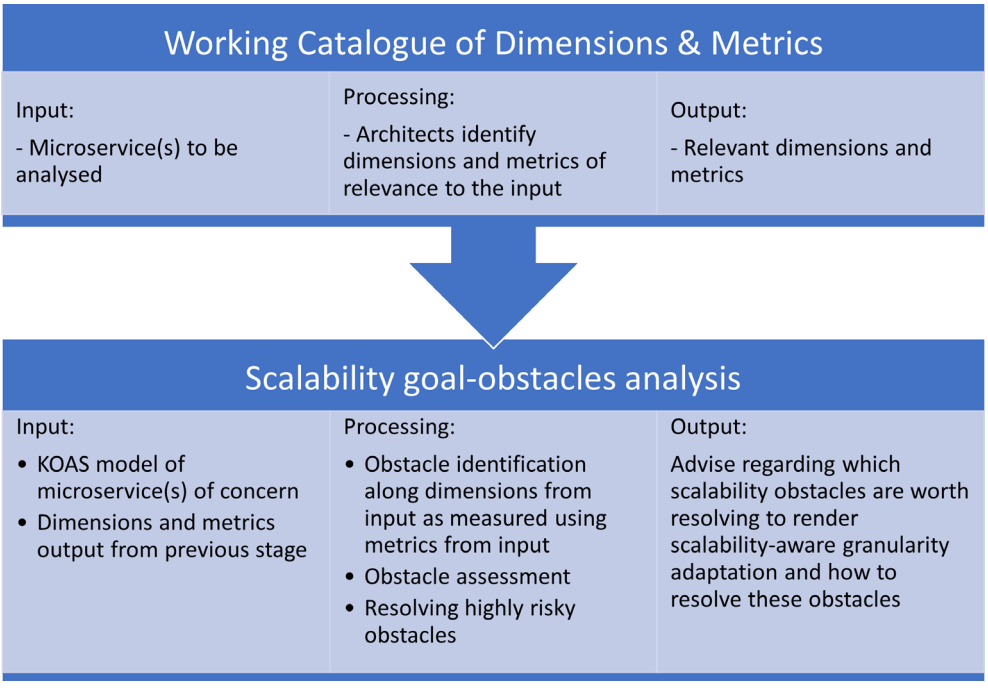

FIGURE 1 Outline of the two-fold contribution of this article

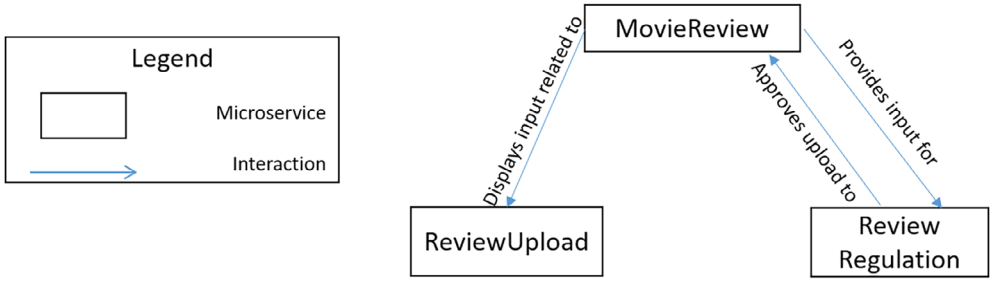

FIGURE 2 Filmflix architecture

processing and output for each of our contributions and shows the link between them. The rest of this article is organised as follows: In Section 2, we use a hypothetical microservice architecture—Filmflix—as a motivating example. In Section 3, we reflect on our analysis of the state-of-the-art and how it was refined into our catalogue. In Section 4, we provide an overview of KOAS goal-oriented modelling and scalability goal obstacle analysis. In Section 5, we analyse and discuss our contributions by comparing their application to the Filmflix architecture from Section 2 against its ad-hoc scalability assessment. In Section 6, we discuss the threats to validity related to our contributions. In Section 7, we compare and contrast our work against relevant existing literature. In Section 8, we conclude by reflecting on the significance of our contributions then we propose some short-term and long-term future research directions that can build upon our contributions.

2 | MOTIVATING SCENARIO

We use a hypothetical online microservice application—Filmflix—as a case study to motivate the problem and highlight its significance. In Filmflix, microservices are used to build both the internal application logic and the user-facing frontend. The drivers for transitioning to microservices, the utilities to be enhanced and the architects' rationale in Filmflix are inspired by the Netflix's experience in adopting microservices.[3,18,19]

Filmflix allows users to upload written movie reviews after they pass a regulation system. The regulation involves looking for a set of a predefined blacklist of "foul" terms in this review. The architecture of Filmflix contains three microservices: ReviewRegulation—implementing the regulation of movie reviews, ReviewUpload—managing the user input requirements when submitting a review and MovieReview—capturing input from the user through an interface and uploading a review that passes the regulation. Figure 2 illustrates the Filmflix architecture including the interaction between its three microservices.

We consider Filmflix to be operating on a similar scale of Netflix. The scale at which Filmflix operates in reality means it is inevitable that multiple scalability dimensions need to be considered when suggesting a granularityadaptation strategy that can result in better QoS provision to the Filmflix architecture. For example, the volume of data shared across Filmflix microservices and the number of geographical locations served by Filmflix are among the dimensions which can affect the scalability of Filmflix. Consequently, both dimensions need to be considering when making scalability-aware granularity adaptation decisions.

If the volume of shared data would be the only provided input and thereby would be considered when reasoning about granularity, the decision might converge to merge the source code of these microservices or even just host them in the same cloud cluster without source-code merging. If the geographical distribution of the end users would be only considered, the decision might then converge to decompose microservices so that functionalities related to each geographical area are encapsulated within the same microservice. Therefore, reasoning about granularity adaptation decisions needs to consider both dimensions simultaneously to suggest a scalability-aware granularity adaptation strategy. Guidance is therefore essential to identify the important scalability dimensions and metrics for Filmflix and then analyse potential obstacles for the satisfiability of Filmflix's goals if it is scaled along these dimensions.

Given the above scenario, we call for: (1) identifying the microservice-specific dimensions and metrics which are important for the scalability of a microservice architecture and thereby essential for making scalability-aware granularity

adaptation decisions and (2) systematically analysing the obstacles along each identified dimension. These obstacles are indications of risks that can obstruct satisfying goals of interest to a microservice architecture. Henceforth, this systematic analysis justifies the importance of considering a given scalability dimension to make a scalability-aware granularity adaptation decision.

We argue that a catalogue of microservice-specific scalability dimensions and metrics would be needed to inform microservices adoption with scalability in mind. We derive this catalogue from microservice-specific literature which we compiled in Reference 9 to make specialised guidance fit for addressing the problem in Section 1. The generality of scalability goal-obstacle analysis[13] makes it flexible enough to be applicable in our context as opposed to other domain-specific scalability assessment approaches. For example, state space size is a scalability indicator of techniques used to tackle state explosion in model checking.[15] Effectiveness calculated as a function of throughput and QoS is an indicator of scalability in parallel computing.[16] Cost-effectiveness of scalability calculated as a function of the system's power and cost of this power is a scalability indicator in distributed systems.[17] Despite their power in the domains they target, these domains are fundamentally different from microservice architectures. Moreover, scalability goal-obstacle analysis by definition has distinct steps (explained in Section 4) making it fitting for addressing the target problem of this article. Both contributions are complementary and aimed at analysing the ability to make scalability-aware granularity adaptation decisions.

3 | WORKING CATALOGUE OF MICROSERVICE-SPECIFIC SCALABILITY DIMENSIONS AND METRICS

In this section, we report on our microservice-specific catalogue of scalability dimensions and metrics (Section 3.2) that compiles its input from a systematic mapping study conducted by the authors of this article.[9] According to this study, scalability is the most common quality considered when reasoning about microservice granularity. This is reasonable given the dynamic, large-scale environment in which microservices operate. This highlights the significance of our contributions; it is essential to make scalability-aware granularity adaptation decisions.

3.1 | State-of-the-art scalability dimensions for microservices

In Reference 9, we study microservice literature which presents processes that help in reasoning about microservice granularity adaptation. In this section, we use this study as a basis for deriving the dimensions and metrics of our catalogue.

Table 1 summarises whether and how scalability is considered when making granularity adaptation decisions. The publications summarised in Table 1 are representative examples from our systematic mapping study.[9] In essence, these processes help reason about microservice granularity adaptation so it is fitting to use them as a basis for deriving the dimensions and metrics of our catalogue.

A scalability dimension can be regarded as any characteristic of the system design and/or its running environment which can exhibit a wide variation in values during the system's lifetime.[11] Scalability metrics can be regarded as any measurable or computable variables that can measure the ability of a system to meet its goals even when it is stressed along one or more scalability dimensions.[16] In essence, a scalability dimension is an independent variable that an architect intentionally changes to scale up/down the system. A scalability metric is a dependent variable which is observed in response to changes in scalability dimensions. Therefore, both scalability dimensions and metrics are measurable variables, but they controlled differently. For example, when testing a system a software architect may want to manually vary the number of concurrent requests given to the system and observe the response time of the system with each change in the number of requests. In that scenario, the scalability dimension is the number of concurrent requests while the scalability metric is the response time of the system.

Reflecting on Table 1, we can make two observations: (1) scalability has not been considered for several examined publications and (2) where scalability has been considered, there is little consensus regarding the dimensions and/or metrics across the examined publications. These observations motivate our first contribution—a working catalogue of microservice-specific scalability dimensions and metrics for microservice adopters. Consequently, these dimensions are essential to consider when making a scalability-aware granularity adaptation decision regarding a microservice architecture.

TABLE 1 Summarising whether and how scalability is considered by the "processes" (i.e., decision support systems) in Reference 9

Granularity reasoning approach	Scalabilit analysis?	Scalability assessment approach	Pattern affecting granularity adaptation	Scalability dimensions	Scalability metrics
Based on industrial surveys and interviews, a brief catalogue of microservice-specific bad smells "which negatively affect software quality attributes such as understandability, testability, extensibility, reusability, and maintainability of the system under development.[20,p. 59]" The solutions adopted to fix these bad smells are in essence drivers for microservice granularity adaptation.	✓	Discussion	Semantic versioning of APIs	number of codebase versions	
			API gateways	number of cyclical dependencies across microservices, number of interdependent microservices	
			Lightweight communication mechanism adoption	volume of enterprise service buses connecting microservices	
			Microservice discovery adoption	number of available IP addresses	
			Merging microservices accessing same databases	volume of data shared across microservices	
			Polyglot persistence	volume of data shared across microservices	
			Shared library extraction	number of libraries shared across microservices	
			Systematically limiting the number of used development tools	number of development technologies used	
			"Clear analysis of business processes and the need for resources[20,p. 60]"	number of cross-cutting business capabilities	
Microservice architecture design patterns some of which affect the granularity of microservices (e.g., bulkheads); the process of adopting each pattern and the considerations related to it are discussed[21]	✓	Discussion	Service registry/Self-registration/Third-party registration	Number of interdependent microservices	
			API Keys and two-factor authentication	security costs	
			Centralised and decentralised logging	logging costs	
			Command Query Responsibility Segregation (CQRS)	volume of data shared across microservices	
			Multiple instances per host/single server instance per host/single server per virtual machine/single instance per container	containerisation/ virtualisation costs	
			Message broker	caching costs	

(Continues)

TABLE 1 (Continued)

Granularity reasoning approach	Scalabilit analysis?	Scalability assessment approach	Pattern affecting granularity adaptation	Scalability dimensions	Scalability metrics
A guide book of the best practices for architecting microservice-based systems, including how to modernise legacy applications into microservice architectures; this transition drives granularity adaptation decisions[22]	✓	Discussion	Message routing	Network latency across microservices	time of convergence to an adaptation decision, ease of independent development after pursuing adaptation, system throughput/response time
			DevOps and NoOps	operational and infrastructure costs	
			Immutable server/installation scripts	deployment pipeline costs, number of interdependent deployment configuration setting	
			Sidecar	number of development technologies used, number of interdependent teams working on the architecture, number of codebase versions	
			Reference environments/stubs/consumer-driven contract tests Internal and external interfaces	number of integration test cases spanning multiple microservices, number of consumer-driven contracts to be met number of user-facing interfaces	
			Containerisation of service registries	number of microservice registries	
			Transferring/Extracting shared functionalities	number of shared libraries across microservices	
			Client-based load balancing/service discovery/Load balancer per microservice	number of load balancers per microservice	
			Team-based documentation/Microservice-based documentation/Documentation versioning/Microservice templates	volume of documentation	
			Edge-side Includes/Serverside Includes	number of front-end servers	
			Content enricher	number of light-weight communication mechanisms used	
			Event sourcing	number of event buses	
			Circuit breakers	monitoring costs	
			Database replication	number of independent databases, volume of queries received by the data store	

(Continues)

TABLE 1 (Continued)

Granularity reasoning approach	Scalabilit analysis?	Scalability assessment approach	Pattern affecting granularity adaptation	Scalability dimensions	Scalability metrics
A guide book for how to design applications for maximum uptime, performance, and return on investment; these are common drivers of microservitisation. Therefore this book can be regarded as a process supporting granularity adaptation decisions.[23]	✓	Discussion	In-process method calls/Interprocess communication/Remote procedure calls/message-oriented middleware/tuple spaces	Number of interdependent middleware	
			Bulkheads	volume of multi-threading	
			Database clustering	number of available virtual IP addresses	
A presentation of the current solutions for modelling, integrating, testing, deploying, and monitoring microservices; all these dimensions affect or are affected by granularity adaptation decisions hence the presentation in this book can act as procedural guidance for granularity adaptation decision-making[24]					
A series of microservice decomposition strategies mostly based on isolating independent bounded contexts[25]	✓	Discussion	Sagas for managing data transactions	Volume of data transactions	
			Request/Asynchronous One-way (i.e., notifications)	volume of intermicroservice communications	
			Publish/Subscribe and Publish/Asynchronous responses	volume of published events	
			Decomposition by domain, sub-domain or scenarios	number of bounded contexts	
			Request/response messages	volume of requests	
A presentation of best practices that can ensure QoS provision through the microservitisation process; some patterns can drive granularity adaptation decisions (e.g., clustering services according to the client zone they serve)[26]	✓	Discussion	Service routing into zones	geographical spread of end user base	
A case study describing an industrial experience for extracting microservices (i.e., making granularity adaptation decisions)[27]					
An industrial experience reporting on data-driven granularity adaptation decisions[28]					

(Continues)

TABLE 1 (Continued)

Granularity reasoning approach	Scalabilit analysis?	Scalability assessment approach	Pattern affecting granularity adaptation	Scalability dimensions	Scalability metrics
An industrial experience reporting on a monolithic subsystem migration to microservices[29]	✓	Case study application		End user base size, number of countries the application is serving, audit compliance considerations, data transport and synchronisation costs	downtime rate after pursuing adaptation
An industrial experience advocating for microservitisation through "Everything-as-a-Service"[30]	✓	Discussion		technology migration costs, number of interdependent teams	
A case study describing procedural extraction of microservices from a monolithic architecture[31]	✓	Discussion		Number of interdependent database tables, number of RESTful APIs	
Describing a pattern for extracting microservices from monoliths based on incrementally building new functionalities surrounding existing ones[32]					
Describing a pattern for extracting microservices driven by the event flow throughout the architecture[33]					
Proposal of a technique which identifies candidates for microservice decomposition depending on whether they are client-, server- or data-related[34]	✓	Case study application		Code base size (measured in lines of code), number of shared database tables, number of cross-cutting functionalities, number of microservices per crosscutting functionality	
Issues related to componentisation, organisation, endpoints and messaging mechanisms in the microservice architecture are discussed; these issues affect granularity adaptation hence this discussion can support the decision-making process[35]					
An experience report on a migration to decompose an existing application into microservices and on how to decompose an ongoing legacy modernisation project[36]	✓	Case study application		Technology migration costs, number of bounded contexts, number of access points to back-end microservices, number of non-user related functionalities, volume of read/write database operations, infrastructure platform migration costs	transactional consistency after pursuing adaptation

(Continues)

TABLE 1 (Continued)

Granularity reasoning approach	Scalabilit analysis?	Scalability assessment approach	Pattern affecting granularity adaptation	Scalability dimensions	Scalability metrics
A microservice decomposition approach based on extending the usage of web mining techniques and clustering algorithms to characterise the workloads received by a microservice application[37]	✓	Case study application		Volume of requests received by the application	
An experience reporting on a migration process to decompose an existing application into microservices and hte lessons learnt from this transition[38]	✓	Discussion		end user base size, complexity of end user requirements	

Note: A check mark in the second column means scalability is considered when the respective system suggests granularity adaptation decisions; the third column shows dimensions which the respective system deems important to consider for making scalability-aware decisions.

3.2 | Contribution 1: Working catalogue of microservice-specific scalability dimensions and metrics

Based on Table 1, we present our catalogue of scalability dimensions and metrics in Tables 2 and 3, respectively; dimensions and metrics are categorised according to their nature (e.g., organisational, data-related and developmental). Some dimensions and metrics fit under more than one category so they are mentioned more than once in Tables 2 and 3.

It is worth noting that our usage of the term "microservice-specific" is not an indication that the dimensions and metrics below are exclusive to analysing the scalability of microservice architectures. Our usage of this term is an indication that the catalogue below includes the dimensions and metrics which are particularly relevant to the microservice architecture based on the systematically studying microservice-specific literature in Reference 9. Table 1 summarises a portion of this study's results which is relevant to our contribution. This table acts a means to cross-reference our catalogue presented below against the examined literature.

Given these catalogues, microservice adopters can manually elicit the relevant scalability dimensions and metrics. The exact tools used to track the dimensions and measure the metrics elicited from this catalogue depends on the specific

TABLE 2 Working catalogue of microservice-specific scalability dimensions compiled from Reference 9

Nature	Scalability dimensions
Architectural	Number of interdependent microservices, number of user-facing interfaces, number of microservice registries, number of load balancers per microservice, number of event buses, number of interdependent middleware, number of RESTful API gateways, number of bounded contexts
Deployment	Containerisation/virtualisation costs, deployment pipeline costs, number of front-end servers, number of interdependent deployment configuration settings, number of configuration files, computation resource costs
Security	Security costs, number of access points to back-end microservices
Data	Volume of shared data across microservices, number of independent databases, volume of queries received by the data store, data transport and synchronisation costs, volume of data transactions, volume of read/write database operations, data translation costs, database maintenance costs
Testing	Number of integration test cases spanning multiple microservices
Logging	Logging costs, caching costs
Communication	Network latency across microservices, number of light-weight communication mechanisms used, number of available virtual IP addresses, volume of published events in publish-subscribe communication mechanisms, volume of synchronous requests
Operational	Operational and infrastructure costs, monitoring costs
Developmental	Number of development technologies used, number of codebase versions, number of shared libraries across microservices, volume of documentation, volume of multi-threading, technology migration costs
Organisational	Number of interdependent teams working on the architecture, volume of shared knowledge across teams
Monitoring	Monitoring costs
QoS provision	Number of consumer-driven contracts to be met, end user base size, complexity of end user requirements
Geographical	Number of countries the application is serving
Legal	Audit compliance considerations

Note: It is potentially essential to consider these dimensions to make scalability-aware granularity adaptation decisions.

TABLE 3 Working catalogue of microservice-specific scalability metrics compiled from Reference 9

Nature	Scalability metrics
Developmental	Ease of feature introduction after adaptation, ease of independent development after pursuing adaptation
Architectural	Standardisation across interfaces after adaptation, stability of the architecture after adaptation
Data	Data consistency after adaptation, transactional consistency after pursuing adaptation
QoS provision	System performance (in throughput/response time), failure rate of adapted architecture

Note: Relevant scalability metrics can measure the ability of a microservice architecture to scale along the relevant dimensions from Table 2.

microservice application being analysed. Therefore, the exact tools used to track the dimensions and measure the metrics elicited from the catalogue is outside the scope of this article. It is worth noting that not all scalability dimensions are relevant for every given microservice application. In practice, the relevance will vary depending on the microservitisation utilities of concern. Moreover, not all the scalability dimensions and metrics relevant to a microservice application will be critical to address within that application. In some cases, the variance in the values of these dimensions and metrics will be small, imposing very little impact on scalability. Nevertheless, this catalogue can be useful when the relevant dimensions and metrics from the catalogue are systematically linked (through our second contribution—scalability goal-obstacle analysis) to the system's goals, and consequently to the likelihood and criticality of scalability obstacles. In other words, the dimensions from our catalogue can scope the scalability-goal obstacle analysis. The analysis justifies the criticality of dimensions picked from the catalogue given a microservice architecture.

4 | CONTRIBUTION 2:APPLYING SCALABILITY GOAL-OBSTACLE ANALYSIS TO MICROSERVICE GRANULARITY

4.1 | KAOS goal-oriented modelling

The KAOS framework allows modelling a software system as a collection of top-level goals operationalised through a hierarchy of AND/OR refinements to relate top-level goals to lower level sub-goals which ensure them.[39] An AND-refinement relates a goal to a set of sub-goals; this means that satisfying all the sub-goals in is necessary for achieving the parent goal. OR-refinement links relate a goal to a set of alternative sub-goals (which may include further refinements); achieving one of the alternative sub-goals is sufficient for achieving the parent goal. Each goal has is a prescriptive statement including its pattern (e.g., Achieve, Maintain, Avoid), name and natural language definition.[12] Each goal is assessable by satisfaction criteria and/or metrics.

Each goal is connected to an agent(s) through a responsibility link. Agents are active system components, such as humans, hardware devices and software components, that are capable of goals they are responsible for. Goals range from high-level objectives whose satisfaction involves multiple agents (e.g., providing efficient decision-making support), to fine-grained technical properties involving fewer agents (e.g., monitoring runtime evidence variables related QoSs of concern).[12]

Unlike goals that are prescriptive, domain hypotheses and assumptions are descriptive statements about the system or its usage context which are subject to change but their validity is necessary for goal achievement.[12]

Compared to other goal-oriented modelling approaches, KOAS is particularly useful to scalability analysis for the following reasons (among others)[11]:

1. Explicitly capturing the assumptions made about the system's environment. "The scalability of a system is highly dependent not only on the assumptions made about the current system environment, but also on the estimation of this environment in the future.[11,p. 79]"
2. Providing traceability across the hierarchy of goals. This is instrumental to updates in assumptions made about the system's environment which can affect its scalability.
3. Assigning the responsibility of achieving a system goal to an agent. This is particularly helpful in assigning the responsibility of achieving a scalability goal to a particular agent in the system.
4. Objectively capturing quality system goals.

5. Allowing for variations in assumptions made about the system environment. This is particularly relevant to environments that are highly dynamic which is the case for microservice architectures.

4.2 | Scalability goal-obstacle analysis

Given a refined KAOS goal-oriented model of a system, scalability goal-obstacle analysis aims "to take a pessimistic view of the model elaborated so far by systematically considering how the actual system might deviate from the model[12,p. 4]". This entails the following steps[12]:

1. Identifying as many scalability obstacles as possible by systematically considering all leaf goals and assumptions in the goal graph;
2. Assessing the relative importance of the identified obstacles in terms of their likelihood and criticality to top-level goals;
3. Resolving the highly risky obstacles (which are both highly critical and highly likely) using obstacle resolving tactics. These include modifying existing goals, requirements and assumptions, or by introducing new ones so as to prevent, reduce or tolerate the obstacles.

A scalability obstacle is a condition that obstructs the goal from being satisfied when the load imposed by the goal on agents involved in its satisfaction exceeds the capacity of the agents. Each goal is connected to the obstacles obstructing it using an obstruction link. A scalability obstacle uses the concept of goal load and agent capacity to denote measures that characterise the amount of work needed and the amount of resources available to the agent to satisfy the goal, respectively.[12] Therefore, a scalability obstacle takes the form *Goal Load Exceeds Agent Capacity*. A goal load is another term for a scalability dimension.

Assessing the relative importance of a scalability obstacle is a product of its likelihood and criticality inspired by the risk analysis matrix technique.[40] In this matrix, the likelihood of an obstacle is estimated qualitatively on a scale from low to high and a similar scale is used to estimate criticality. This technique has been used in the context of scalability goal-obstacle analysis in References 12,41 so we utilise the same technique for consistency. We envision, however, that objective techniques such as those used in References 42,43 can also be applied to assess scalability obstacles.

Resolving scalability obstacles can be done using a range of tactics that satisfy the following strategies[11,44]: goal substitution, agent substitution, obstacle prevention, goal weakening, obstacle reduction, goal restoration, obstacle mitigation and do-nothing. The obstacle prevention strategy for example can be satisfied using tactics such as introducing either a domain assumption to be satisfied by some agent or a scalability goal. A scalability goal is a quality goal constrained by expected variations on the scaling dimensions.[12]

Scalability goal-obstacle analysis is a systematic rather than ad-hoc approach to assessing scalability. Looking at the third column of Table 1, the state-of-the-art in assessing scalability when reasoning about granularity adaptation is ad-hoc case study application or discussions. Hence, our application of scalability goal-obstacle analysis in the field of microservices is a novel introduction of systematic solution to an ad-hocly solved existing problem.

5 | ANALYSIS AND DISCUSSION OF CONTRIBUTIONS

In this section, we use our Filmflix architecture (presented in Section 2) as a case study for showing how our contributions address the inadequacy of microservice-specific scalability guidance.

In this section, we first ad-hocly discuss the dimensions and metrics which can affect the scalability of Filmflix. Then we apply scalability goal-obstacle analysis to Filmflix; the analysis is scoped to dimensions from our catalogue which we deemed relevant to Filmflix's scalability. As mentioned in Section 3.2, not all the dimensions and metrics would be relevant to a particular case study. Therefore, not all the dimensions and metrics in Tables 2 and 3 are relevant to Filmflix. Also, the relevance of a particular dimension and/or metric from the catalogue to Filmflix is a subjective matter depending on the microservitisation utility of concern. As far as the motivating scenario in Section 2 is concerned, the

microservitisation utility of concern is improved QoS provision of the architecture. We do acknowledge that under different studies of Filmflix with different microservitisation utilities, different dimensions and metrics might be relevant. One of the aims of the discussion in this section is to show the potential comprehensiveness of our catalogue. Therefore, we are not concerned in this discussion about the underlying tools used to track the dimensions and measure the metrics deemed relevant for Filmflix. We do appreciate applicability concerns of this matter in Section 8.

Reflecting on utilising our catalogue for Filmflix, we discuss the potential comprehensiveness of Tables 2 and 3. Nevertheless, we acknowledge that further investigation is needed to extend and/or refine our catalogue. Comparing both scalability assessment approaches of Filmflix, we show how scalability goal-obstacle analysis leads to much more informed results than ad-hoc scalability assessment (Section 5.3).

5.1 | Filmflix Ad-hoc scalability assessment

Our scalability assessment in this section is inspired by our observations of FilmFlix's architecture from Section 2.

We regard the size of "foul" terms blacklist, the number of user input fields when uploading a review, and the number of reviews submitted simultaneously to Filmflix as the dimensions that can potentially affect Filmflix's scalability. This is grounded on the intuition that only the inputs are directly relevant to the functionality of microservice in Filmflix can affect its scalability. ReviewRegulation compares each submitted review against each term in the blacklist, so its size affects the ability of Filmflix to perform acceptably. Depending on the number of input fields required by Filmflix architects, the performance of ReviewUpload can be affected. MovieReview is the user-facing interface through which movie reviws are submitted; the number of submitted reviews impacts Filmflix's ability to perform acceptably. Furthermore, we regard the response time of Filmflix as the only critical scalability metric. Thereafter, our ad-hoc assessment identifies three potential scalability obstacles:

- As the number of foul terms in the blacklist increases, the response time of the Filmflix architecture can deteriorate to an unacceptable extent.
- As the number of input fields required by Filmflix architects increases, the response time of the Filmflix architecture can deteriorate to an unacceptable extent.
- As the number of simultaneous reviews submitted to Filmflix increases, the response time of the Filmflix architecture can deteriorate to an unacceptable extent.

5.2 | Filmflix systematic scalability analysis

In this section, we apply scalability goal-obstacle analysis to the Filmflix architecture. We then discuss how the analysis results can inform suggesting scalability-aware granularity adaptation decisions in Filmflix. This section presents a usage scenario of our contributions and serves the following purposes:

- Illustrating how goal-obstacle analysis can aid in systematically highlighting and justifying dimensions which need to be considered when suggesting scalability-aware granularity adaptation decisions (one dimension of our contributions' benefits presented in Section 1).
- Serving as an example of how microservice architects can get more informed granularity adaptation decisions if they provide more well-rounded dimensions to reason about before suggesting those decisions.
- Serving as an example for microservice architects to replicate our guidance on other microservice architectures.

5.2.1 | KOAS goal-oriented modelling of Filmflix

Filmflix has seven goals assigned to five agents; the goal model is presented and refined in Figure 3 using the notation explained in Figure 4. In this subsection, we describe the role of each agent to justify their responsibilities for different goals.

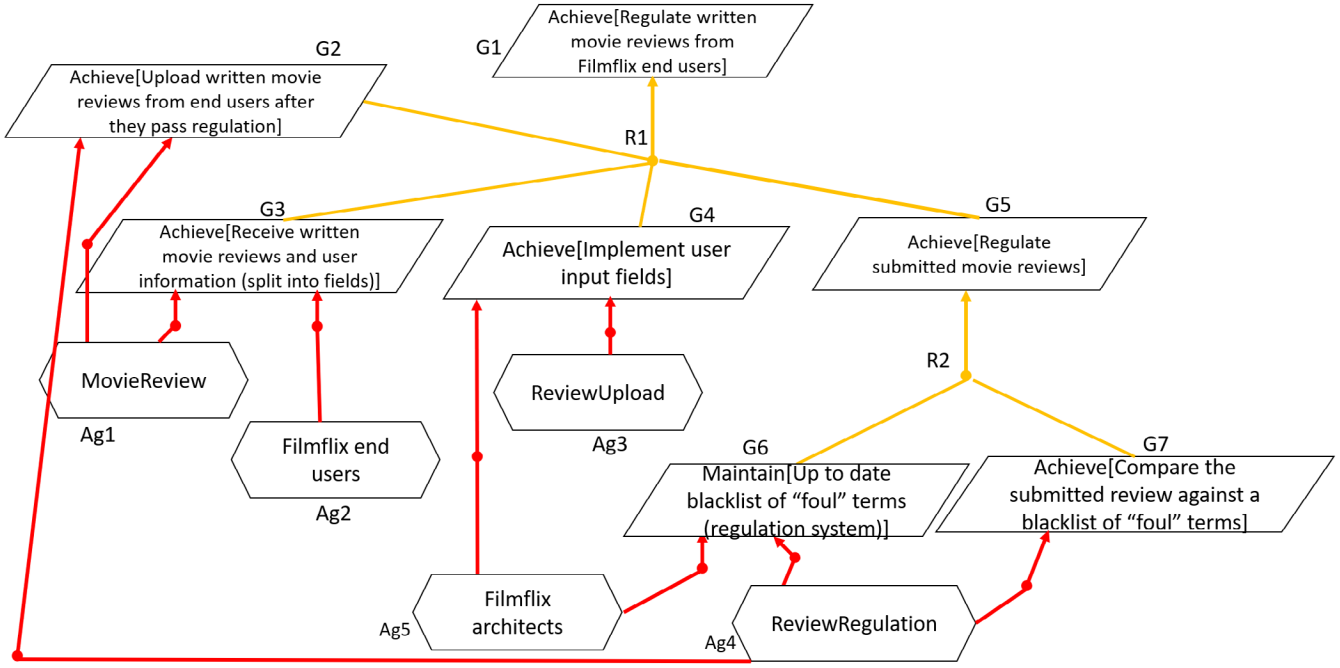

FIGURE 3 Refinement of the *Achieve[Regulate written movie reviews from Filmflix end users]* goal in Filmflix's architecture

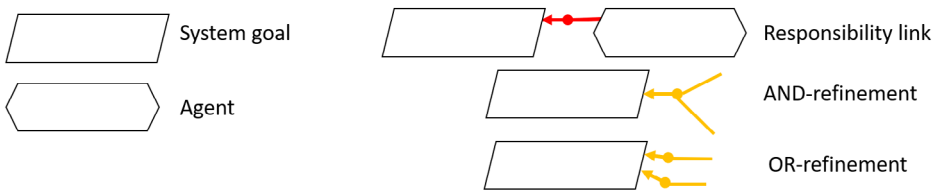

FIGURE 4 Legend for the KAOS modelling notation

MovieReview: This agent is user-facing interface of Filmflix; it is responsible for receiving input and displaying output related to the high-level goal (G1 in Figure 3). Receiving input is represented by G3—*Achieve[Receive written movie reviews and user information (split into fields)]*; displaying output is represented by G2—*Achieve[Upload written movie reviews from end users after they pass regulation]*.

Filmflix end users: This agent refers to any active Filmflix end user which submits input to MovieReview. Therefore, active Filmflix end users share with the responsibility of achieving G3 in Figure 3 with MovieReview.

ReviewUpload: This agent is a Filmflix microservice which is not facing end users but implements Filmflix architects' requirements regarding the required end user input fields (e.g., name, age, ethnicity, email, etc.). Therefore, ReviewUpload contributes to the design of MovieReview by achieving G4—*Achieve[Implement user input fields]*.

ReviewRegulation: This agent is a Filmflix microservice which is core to regulating the output displayed by MovieReview. Given each review submitted by a Filmflix end user, this microservice: (a) compares it to a pre-defined blacklist (G7 in Figure 3) and (b) allows MovieReview to upload reviews which do not contain any term on the blacklist (contributing to G2). In parallel, ReviewRegulation has to maintain this blacklist (G6 in Figure 3) given any compliance requirements dictated by Filmflix architects.

Filmflix architects: This agent refers generically to any source which the architects utilise to determine input information required by Filmflix end users and foul terms that need to be included in the blacklist used by ReviewRegulation. These sources include but are not limited to data privacy laws, historical data and compliance regulations. Therefore, Filmflix architects provide the input to achieve G4 and G6 in Figure 3.

5.2.2 | Scalability goal-obstacle analysis of Filmflix

Conducting scalability obstacle analysis on the goal model revealed five potential scalability obstacles related to four scaling dimensions. In this subsection, we present the process of identifying, assessing and resolving these obstacles. Based on this systematic analysis, we discuss how goal obstacle resolution tactics can inform reasoning about scalability-aware granularity adaptation decisions. Tables 4 and 5 summarise the obstacle identification and assessment results.

Reflecting on Table 5, there are three high risk obstacles that need to be resolved to ensure Filmflix's architecture achieves its goals with acceptable performance. In other words, these obstacles need to be considered when reasoning about adapting the granularity of Filmflix's architecture.

Using resolution tactics from References 11,12, we propose preventing the *Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance* obstacle from occurring by introducing and refining a scalability obstacle prevention goal: *Avoid [Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance]*. This goal can be achieved via two routes, illustrated as an OR-refinement of G8 in Figure 5. On the one hand, it can be achieved by ensuring the MovieReview's granularity level enables it to perform acceptably given observed numbers of Filmflix end users (G9 and G10 in Figure 5). If Filmflix architects were to achieve G9 and G10, architects can reason about granularity while considering the relationship between number of Filmflix end users and MovieReview performance.

Alternatively, G8 in Figure 5 can be achieved if Filmflix architects can ensure that the number of Filmflix end users never stresses MovieReview beyond its performance ability (G12 in Figure 5). This ability is derived from monitoring

TABLE 4 Identifying relevant scalability dimensions and metrics (guided by Tables 2 and 3) for the modelled goals of the Filmflix architecture

Goal	Scalability metric	Scalability dimension	Influenced by
Achieve[Regulate written movie reviews from Filmflix end users]	MovieReview performance	volume of received reviews, number of "foul" terms in blacklist, number of Filmflix end users, number of Filmflix architects, number of user input fields	volume of received reviews <depends on> [number of Filmflix end users], number of "foul" terms in blacklist <depends on> [number of Filmflix architects], number of user input fields <depends on> [number of Filmflix architects]
Achieve[Upload written movie reviews from end users after they pass regulation]	MovieReview performance	volume of reviews that passed the regulation	
Achieve[Receive written movie reviews and user information (split into fields)]	MovieReview performance	volume of received reviews, number of user input fields	
Achieve[Implement user input fields]	ReviewUpload performance	number of Filmflix architects, number of user input fields	number of user input fields <depends on> [number of Filmflix architects]
Achieve[Regulate submitted movie reviews]	ReviewRegulation performance	number of "foul" terms in blacklist, number of Filmflix architects, volume of received reviews, number of Filmflix end users	number of "foul" terms in blacklist <depends on> [number of Filmflix architects], volume of received reviews <depends on> [number of Filmflix end users]
Maintain[Up to date blacklist of "foul" terms (regulation system)]	ReviewUpload performance	number of "foul" terms in blacklist, number of Filmflix architects	number of "foul" terms in blacklist <depends on> [number of Filmflix architects]
Achieve[Compare the submitted review against a blacklist of "foul" terms]	ReviewRegulation performance	number of "foul" terms in blacklist	

Note: This table is used to identify the scalability obstacles in Table 5.

TABLE 5 Assessing scalability obstacles of the Filmflix architecture

Scalability obstacle	Criticality	Likelihood	Rationale
Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance	High	High	Based on the assumption from Section 2 that Filmflix on a scale similar to Netflix, there is a high likelihood of having a large number of active end users submitting reviews. This will affect MovieReviews ability to achieve G3 in Figure 3. Since MovieReview is the user-facing interface, it is critical for its performance to remain acceptable to avoid losing the interest of a large number of end users.
Volume of reviews which passed the regulation exceed MovieReview's ability to achieve acceptable performance	High	Low	Even for the scale at which Filmflix operates, only a fraction of the received reviews will be uploaded by MovieReview, so they likelihood of this obstacle is low. Nevertheless, if this obstacle where to occur it would be critical since it affects achieving a user-facing goal (G2 in Figure 3).
Number of user input fields exceeds MovieReview's ability to achieve acceptable performance	Low	Low	Since received user information is not subject to regulation and it is not uploaded to MovieReview along with an accepted review, the number of fields has little impact on MovieReview's performance.
Number of Filmflix architects exceeds ReviewUpload's capacity to achieve acceptable performance	Low	High	The likelihood of this obstacle depends on possibility of conflicts across international data privacy and compliance rules (both of which are sources for Filmflix architects to determine the input to ReviewUpload). Possibility of such conflicts is high given the potential geographical distribution of Filmflix end users. Even if this obstacle were to occur, its criticality to the overall goal G1 in Figure 3 is low since ReviewUpload does not utilise or regulate the submitted reviews.
Number of Filmflix end users exceeds ReviewRegulation's ability to achieve acceptable performance	High	High	The number of active end users determines the volume of reviews which have to regulated. For the scale of Filmflix, it is highly likely to have a large number of active end users leading to a large volume of reviews to be regulated and high likelihood of this obstacle. If ReviewRegulation does not achieve acceptable performance, then the user-facing G2 is potentially obstructed leading to the risk of losing end users' interest.
Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance	High	High	Given the potential geographical spread of Filmflix end users, it is highly likely a large number of sources is used by Filmflix architects to determine the "foul" terms which ReviewRegulation has to maintain. There are two highly likely implications of this: (1) a large number of terms against which each review needs to be compared and (2) frequent updates to the blacklist during which no regulation can be done. Both implications are critical since they obstruct achieving G2, G6 and G7 with acceptable performance.

MovieReview's performance (G11 in Figure 5). This route of achieving G8 does not involve adapting MovieReview's granularity.

To resolve the *Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance* obstacle, we propose introducing and refining a scalability obstacle prevention goal: *Avoid [Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance]*. This goal can be achieved via two routes, illustrated as an OR-refinement of G13 in Figure 6. Although G13 is presented as a refinement of G6 in Figure 6, we appreciate that G13 is also relevant to achieving G2 and G7 from Figure 3. G13 can be achieved by ensuring the ReviewRegulation's granularity level enables it to perform acceptably given observed numbers of Filmflix architects (G14 and G15 in Figure 6). The

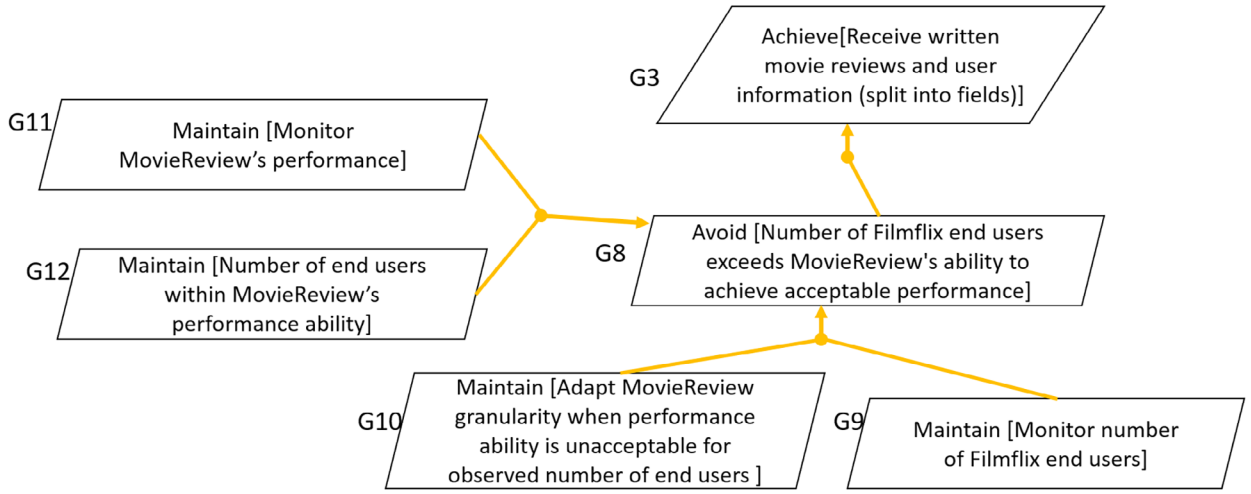

FIGURE 5 Resolving the *Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance]* scalability obstacle prevention goal

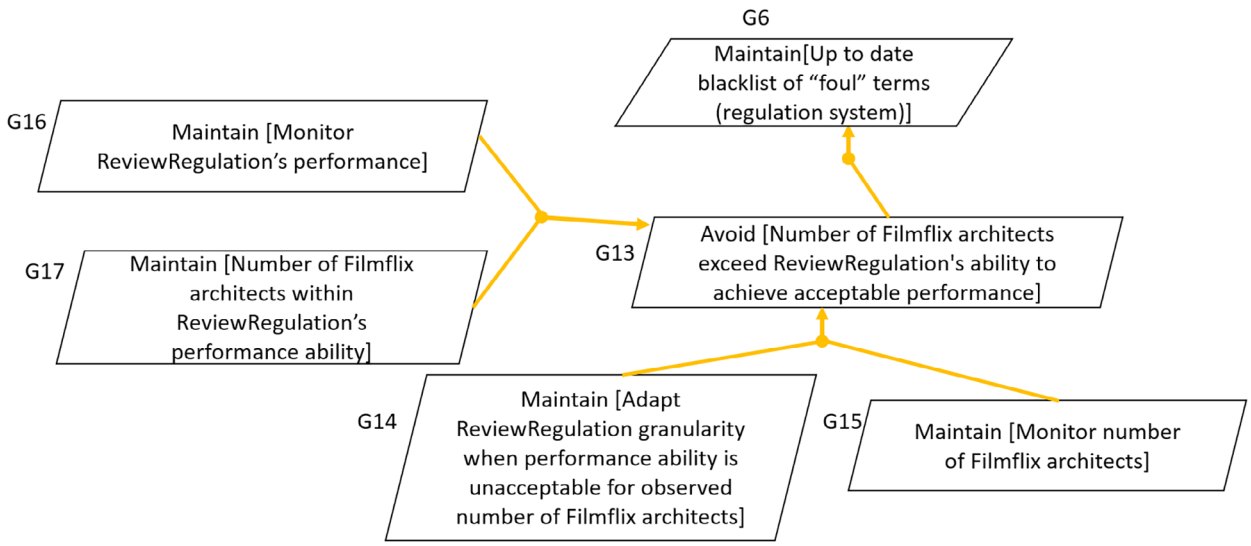

FIGURE 6 Resolving the *Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance]* scalability obstacle prevention goal

number of Filmflix architects is a generic term referring to the number of sources which are consulted to build the blacklist used by ReviewRegulation.

If Filmflix architects were to achieve G14 and G15, architects can justify reasoning about granularity adaptation decisions while considering the relationship between number of Filmflix architects and ReviewRegulation performance.

G13 in Figure 6 can also be achieved if Filmflix architects can ensure that the number of sources used to compile the blacklist in ReviewRegulation never stresses that microservice beyond its performance ability (G17 in Figure 6). This ability is estimated from monitoring ReviewRegulation's performance (G16 in Figure 6). This route of achieving G13 does not involve adapting ReviewRegulation's granularity.

To resolve the *Number of Filmflix end users exceeds ReviewRegulation's ability to achieve acceptable performance* obstacle, we propose introducing and refining a scalability obstacle prevention goal: *Avoid [Number of Filmflix end users exceed*

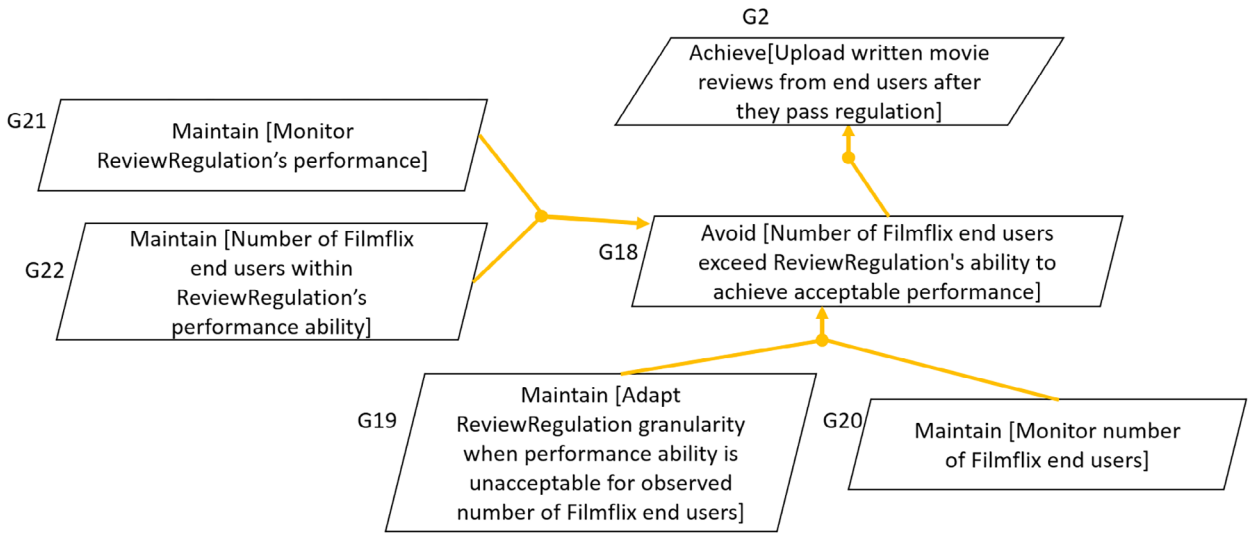

FIGURE 7 Resolving the *Number of Filmflix end users exceed ReviewRegulation's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix end users exceed ReviewRegulation's ability to achieve acceptable performance]* scalability obstacle prevention goal

ReviewRegulation's ability to achieve acceptable performance]. This goal can be achieved via two routes, illustrated as an OR-refinement of G18 in Figure 7.

G18 can be achieved by ensuring the ReviewRegulation's granularity level enables it to perform acceptably given observed numbers of Filmflix end users (G19 and G20 in Figure 7). If Filmflix architects were to achieve G19 and G20, architects can use them to consider the relationship between number of Filmflix end users and ReviewRegaulation performance when reasoning about granularity adaptation.

G18 in Figure 7 can alternatively be achieved if Filmflix architects can ensure that the number of sources used to compile the blacklist in ReviewRegulation never stresses that microservice beyond its performance ability (G22 in Figure 7). This ability is derived from monitoring ReviewRegulation's performance (G21 in Figure 7). This route of achieving G18 does not involve adapting ReviewRegulation's granularity.

5.3 | Discussion

5.3.1 | Catalogue comprehensiveness

Reflecting on the scalability dimensions in Table 5, we observe that the most critical scalability dimensions for Filmflix (i.e., number of Filmflix end users and number of Filmflix architects) are present in Table 2. The number of end users is in essence the end user base size in Table 2. According to Section 5.2, Filmflix architects is a generic term referring sources which include but are not limited to data privacy laws, historical data, and compliance regulations. Therefore, the number of Filmflix architects in Table 5 potentially maps to the number of countries the application is serving and audit compliance considerations in Table 2. One of the scalability dimensions which have not been deemed critical—volume of received reviews—can be mapped to the end user base size in Table 2. The mapping is based on the relationship between volume of received reviews and number of Filmflix end users captured in Table 5. The same can be said about number of "foul" terms in blacklist and number of user input fields since they both depend on number of Filmflix architects which is a dimension implicitly present in Table 5. All the scalability metrics considered 4 are related to performance; it is present under the QoS provision in Table 3.

It is worth noting, however, that while the application to Filmflix can act as evidence for the comprehensiveness of our catalogues, they can only be as good as the model of the analysed case study. Moreover, the comprehensiveness of our catalogue relies heavily on the completeness we strived for in Reference 9. In particular, we used all the relevant publications from Reference 9 to compile the catalogues due to our confidence in this article's coverage of the relevant literature.

It is also worth noting that although some of dimensions and metrics in our catalogue have not been deemed to be potential obstacles in Section 5, this does not mean those metrics/dimensions need not be considered for other microservice architectures. The aim of our catalogue is to provide comprehensive guidance for microservice architects about the possible scalability dimensions and metrics. Therefore, it is through systematic scalability goal-obstacle analysis of a particular microservice architecture that the dimension/metric significance to it can be justified.

Another point is worth noting regarding the catalogue of metrics in particular. Not every category of dimensions in Table 2 has a corresponding category of metrics in Table 3. The categories in Table 3 are a direct representation of the metrics we came across in the examined microservice literature in Reference 9. Therefore, our catalogue unveils a research gap in the microservice state-of-the-art and -practice regarding the existence of adequate scalability metrics.

We subjectively assigned the dimensions and metrics to their respective categories based on their usage in the examined publications. Nevertheless, we acknowledge that objective categorisation of the metrics and dimensions can be a future refinement of these catalogues.

5.3.2 | Scalability goal-obstacle analysis significance

Comparing Filmflix scalability assessment results in Sections 5.1 and 5.2, we observe that goal-obstacle analysis delved to the actual dimensions which impact the ones identified by ad-hoc scalability assessment. Moreover, ad-hoc scalability assessment failed to systematically identify how obstacle resolution tactics can inform microservice granularity adaptation; this is possible through goal-obstacle analysis. Although we apply scalability goal-obstacle analysis to one microservice architecture, the same experience can be copied to other microservice architectures with different scalability metrics and perhaps conflicting system goals. We acknowledge that the practicality of applying scalability goal-obstacle analysis needs to be investigated further in cases where there are: (a) conflicting system goals and (b) conflicting obstacle resolution tactics. As for scalability metrics, we envision that assessing system goals using metrics other than performance would not impact the practicality of the approach. After all, the exercise of monitoring architectural metrics is passive rather than active. Therefore, monitoring failure rate for example alongside performance would not render our approach impractical.

6 | THREATS TO VALIDITY

We reflect on threats to validity regarding our contributions' maturity. We reflect on internal and external threats to validity.

Internal: We acknowledge that for some microservice architectures making scalability-aware granularity adaptation decisions might not be critical for a microservice architecture to satisfy its goals. Our systematic analysis approach addresses this issue by relating the scalability obstacle resolution tactics (which include granularity adaptation) to goal satisfaction.

External: We acknowledge that our discussion in Section 5.3 is derived from the Filmflix case study. In other words, it might not be the case that our discussion generalises to all microservice case studies of different sizes and/or application domains. In fact, we view this as a positive point for the potential development of our working catalogue. As mentioned in Section 5.3.1, using our catalogue for more case studies can refine both the depth and breadth of our catalogue. As for the scalability goal-obstacle analysis, our target in Section 5 was to show the significance of scalability goal-obstacle analysis compared to ad-hoc assessment which the state-of-the-art in the microservice field. Exhaustively proving such significance on all possible scopes, application domains and complexities would be infeasible. Nevertheless, we do acknowledge the need for objective experimentation with a wider variety of microservice case studies (particularly from industry) to strengthen the argument for the generality of our work.

7 | RELATED WORK

In this section, we compare and contrast our work against existing literature that accounts for microservice scalability requirements when reasoning about granularity adaptation. Table 6 summarises the existing literature we examined along

TABLE 6 Summarising the results of comparing and contrasting our work against existing relevant literature

Work	Acknowledges scalability significance	Provides systematic scalability guidance	Provides non-domain specific guidance	Provides scalability-aware design decision advice
9	✓			
45	✓		✓	
46	✓		✓	✓
47	✓			
48	✓	✓		✓
49	✓	✓		✓
50	✓	✓		✓
51	✓		✓	
52	✓		✓	
53	✓		✓	
45	✓			
54	✓		✓	
55	✓			
Microservice-specific scalability guidance (this work)	✓	✓	✓	✓

with the comparison results along three dimensions: acknowledging the significance of scalability for microservices, providing systematic guidance for scalability-aware design of microservices, whether or not this guidance is specific to a certain application domain, and whether or not recommendations for scalability-aware design decisions are provided.

Our contributions in this article are inspired by the systematic mapping study conducted and reported in Reference 9. However, the objectives of Reference 9 are broader than the target of this paper. In this article, we focus on compiling guidance for making scalability-aware granularity adaptation decisions. In Reference 9, the objectives entail more aspects of granularity adaptation (e.g., how microservices are modelled and what quality attributes are considered when reasoning about granularity adaptation).

Another literature survey that explicitly acknowledges scalability in relation to microservices is Reference 45. This surveys broadly studies and categorises microservice literature into a round taxonomy. Among the categories is microservice load balancing, which is one of the main techniques of achieving microservice scalability. By including this category, this work overlaps with our objective of acknowledging the significance of designing for microservice scalability. Nevertheless, our work takes a further step by providing systematic guidance for achieving scalability-aware granularity adaptation decisions.

In Reference 46, the authors propose a "conceptual methodology using which security and scalability requirements are incorporated in decomposing system into microservices." We appreciate that this work provides a systematic methodology for scaling a microservice architecture. We also acknowledge that their work makes security-aware design decisions. However, our work is unique in studying the resolution tactics given potential scalability obstacles in a microservice architecture.

Similar to our work, Reference 47 discusses the importance of scalability for microservice architectures. In that sense, both this paper and ours overlap regarding the significance of scalability when designing microservice architecture. However, our work is unique in providing guidance that actually manifests this significance in granularity adaptation decisions.

On a more practical front, Reference 48 proposes a scalable routing mechanism for applications designed according to the microservice architecture. We appreciate that this is a systematic, efficient approach for addressing scalability requirements of a microservice architecture. Nevertheless, our scalability goal obstacle analysis approach is more general hence making it applicable to applications where other approaches to scaling are taken. Furthermore, our work is focussed on making scalability-aware granularity adaptation decisions. In Reference 48, the approach is targeted at scaling microservice architectures regardless granularity of the microservices. Another practical auto-scaling policy is proposed in Reference 49 integrated with a container-aware application scheduling strategy. The contribution in this paper is aimed at efficiently deploying microservices with minimum processing time and cost, while utilising the computing resources efficiently on the cloud. We appreciate that the contribution of this paper can help in designing scalable microservice architectures. Nevertheless, this work does not explicitly focus on granularity adaptation decisions as we do in our contributions. Similar focus on auto-scaling has been presented in Reference 50 where a complete automated system to decompose, deploy and auto-scaling microservices to maintain the desired response time has been proposed. This work definitely puts granularity adaptation at its forefront which aligns with our objectives. Nevertheless, we argue that our work is more general since it does not link scalability only to improved performance.

An objective, systematic approach for assessing scalability of microservice architectures is proposed in Reference 51. It uses operational profiles to generate load tests to automatically assess scalability pass/fail criteria of microservice configuration alternatives. Our work takes this assessment a step further by proposing ways to resolve scalability obstacles that can be uncovered by such assessment. Therefore, we envision that the contribution in Reference 51 can complement our scalability goal-obstacle analysis.

Overall, our work overlaps with existing literature in acknowledging the significance of considering scalability when designing microservice architecture. However, our work is unique in linking scalability to granularity adaptation decisions in particular and providing systematic non-domain specific guidance for this link.

8 | CONCLUSIONS AND FUTURE WORK

In this article, we contribute to a working catalogue of microservice-specific scalability dimensions and metrics. Our catalogue helps identify dimensions and metrics which are important for the scalability of a given microservice architecture; they need to be considered to make scalability-aware granularity adaptation decisions for it. We compile our catalogue by reviewing the state-of-the-art in microservice granularity adaptation from Reference 9. Secondly, we report on a new application of scalability goal-obstacle analysis[11,12] in the context of reasoning about microservice granularity adaptation. Applying scalability goal-obstacle analysis to a microservice architecture helps identify obstacles along each dimension of importance from our catalogue.

We analyse and discuss our contributions by comparing their usage to both Filmflix architecture in Section 2 and ad-hoc scalability assessment. Comparing both assessment approaches, we show how our contributions lead to more informed results than ad-hoc scalability assessment. Finally, we discuss how scalability goal-obstacle analysis can be applied to other microservice architectures.

Our contributions pave the way to future research directions. In the short-term, we appreciate that further investigation is required to assess the comprehensiveness of our catalogue and ensure that no dimensions and/or metrics in the literature have been wrongly skipped or made redundant. Investigation here can be via interviewing microservice adopters and/or applying our contributions to an industrial case study. We also intend to experimentally and quantitatively evaluate the applicability of our contributions in industrial-scale microservice applications. We intend to experiment with a set of open-source microservice case studies of different scales larger than Filmflix and with a myriad microservitisation concerns. Such experimentation would involve quantitatively measuring relevant scalability metrics before and after resolving critical scalability obstacles. Such experimentation will pave the way to evaluate the following criteria: (a) does a scalability-aware granularity adaptation decision actually result in better scalability metric values compared to the values measured if the decision does not consider scalability and (b) what is the scope of applicability of our guidance and what sizes of case studies can be tackled by the proposed guidance? Regarding the second criterion in particular, we envision that the hierarchical nature of KAOS can help to iteratively scope the model of the microservice system of concern. In other words, our approach can be applied initially on a high-level model of the large-scale system which abstracts away from the lower level goals. Subsequently, a more fine-grained KAOS model of a sub-system can be the input to the next iteration of applying our approach.

In the long-term, we envision that scalability goal-obstacle analysis can itself be developed into a semiautomated tool to assess the impact of scalability on microservice granularity. Another interesting research direction is to develop guidance for making granularity adaptation decisions that are aware of other dimensions (e.g., availability-aware, maintainability-aware, and/or reliability-aware).

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID

Sara Hassan https://orcid.org/0000-0001-7481-0434
Rami Bahsoon https://orcid.org/0000-0002-1139-5795
Rajkumar Buyya https://orcid.org/0000-0001-9754-6496

REFERENCES

1. Reinhold E. Lessons learned on uber's journey into microservices; July 2016. https://www.infoq.com/presentations/uber-darwin
2. Godwin S. Cloud-based microservices powering bbc iplayer; June 2016. https://www.infoq.com/presentations/bbc-microservices-aws?utm_campaign=infoq_content-utm_source=infoq-utm_medium=feed-utm_term=Microservices
3. Probst K, Becker J. Engineering trade-offs and the netflix api re-architecture; August 2016. https://medium.com/netflix-techblog/engineering-trade-offs-and-the-netflix-api-re-architecture-64f122b277dd
4. Calcado P. No free lunch, indeed: Three years of micro-services at soundcloud; January. http://www.infoq.com/presentations/soundcloud-microservices
5. Dehghani Z. Zhamak dehghani real world microservices: Lessons from the frontline; February 2015. https://youtu.be/hsoovFbpAoE
6. Steinacker G. On monoliths and microservices; September 2015. https://dev.otto.de/2015/09/30/on-monoliths-and-microservices/
7. Wagner T. Microservices without the servers; September 2015. https://aws.amazon.com/blogs/compute/microservices-without-the-servers/
8. Hassan S, Bahsoon R. Microservices and their design trade-offs: a self-adaptive roadmap. Proceedings of the 13th IEEE International Conference on Services Computing (SCC); 2016; San Francisco.
9. Hassan S, Bahsoon R, Kazman R. Microservice transition and its granularity problem: a systematic mapping study. *Softw Pract Exper*. 2020;50:1651-1681.
10. Kulkarni N, Dwivedi V. The role of service granularity in a successful soa realization a case study. Proceedings of the 2008 IEEE Congress on Services-Part I; 2008:423-430.
11. de Cerqueira LD. *A Framework for the Characterization and Analysis of Software Systems Scalability*. Ph.D. thesis. University College London; 2010.
12. Duboc L, Letier E, Rosenblum DS. Systematic elaboration of scalability requirements through goal-obstacle analysis. *IEEE Trans Softw Eng*. 2013;39(1):119-140.
13. Duboc L, Rosenblum D, Wicks T. A framework for characterization and analysis of software system scalability, in: Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC-FSE '07; 2007:375-384; ACM, New York, NY.
14. R. IT, A kaos tutorial; October 2007. http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf
15. Edmund J, Grumberg MCO, Peleg D. *Model Checking*. The MIT Press; 1999.
16. Jogalekar P, Woodside M. Evaluating the scalability of distributed systems. *IEEE Trans Parallel Distrib Syst*. 2000;11(6):589-603.
17. Jogalekar P, Woodside C. A scalability metric for distributed computing applications in telecommuni-cations. In: Ramaswami V, Wirth P, eds. *Teletraffic Contributions for the Information Age, Vol. 2 of Teletraffic Science and Engineering*. Elsevier; 1997:101-110.
18. Tonse S. Scalable microservices at netflix. challenges and tools of the trade; March 2015. http://www.infoq.com/presentations/netflix-ipc
19. Watson C, Emmons S, Gregg B. A microscope on microservices; February 2015. https://medium.com/netflix-techblog/a-microscope-on-microservices-923b906103f4
20. Taibi D, Lenarduzzi V. On the definition of microservice bad smells. *IEEE Softw*. 2018;35(3):56-62.
21. L. Krause, *Microservices: Patterns and Applications: Designing Fine-Grained Services by Applying Patterns*. Lucas Krause, 2015. https://books.google.co.uk/books?id=dd5-rgEACAAJ
22. E. Wolff, *Microservices: Flexible Software Architecture*. Pearson Education, 2016. https://books.google.co.uk/books?id=zucwDQAAQBAJ
23. Nygard M. *Release It! Design and Deploy Production-Ready Software*. Pragmatic Bookshelf; 2007.
24. Newman S. *Building Microservices*. 1st ed. O'Reilly Media; 2015.
25. Richardson C. Microservice patterns, manning publications company; 2018. https://books.google.co.uk/books?id=UeK1swEACAAJ
26. Posta C. The hardest part of microservices: calling your services; April 2017. http://blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/
27. Penchikala S. Susanne kaiser on microservices journey from a startup perspective. https://www.infoq.com/news/2017/07/kaiser-microservices-journey (jul 2017).
28. Penchikala S. Managing data in microservices; June 2017. https://www.infoq.com/news/2017/06/managing-data-in-microservices

29. Vlaovic S, Pilani R, Parulekar S, Handa S. Netflix billing migration to aws; January 2016. https://medium.com/netflix-techblog/netflix-billing-migration-to-aws-451fba085a4
30. Iffland D. Q&a with intuit's alex balazs; June 2016, https://www.infoq.com/articles/intuit-alex-balazs-node-services
31. Posta C. Low-risk monolith to microservice evolution part i; June 2015. http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution/
32. Fowler M. Strangler application; June 2004. http://www.martinfowler.com/bliki/StranglerApplication.html
33. Fowler M. Event interception; June 2004. http://www.martinfowler.com/bliki/EventInterception.html
34. Levcovitz A, Terra R, Valente MT. Towards a technique for extracting microservices from monolithicenterprise systems.
35. Bakshi K. Microservices-based software architecture and approaches. Proceedings of the 2017 IEEE Aerospace Conference; 2017:1-8.
36. Knoche H, Hasselbring W. Using microservices for legacy software modernization. *IEEE Softw*. 2018;35(3):44-49.
37. Mustafa O, G'omez JM. Sustainable approach for improving microservices based web application. Proceedings of the Sustainability Dialogue: International Conference on Sustainability and Environmental Management; 2017.
38. Cal ̧cado P. Layering microservices; September 2018. http://philcalcado.com/2018/09/24/services layers.html
39. van Lamsweerde A. Goal-oriented requirements engineering: a guided tour, in: Proceedings of the 5th IEEE International Symposium on Requirements Engineering; 2001:249-262.
40. U. S. D. of Defense. *Risk Management Guide for DOD Acquisition*. 6th ed. Unitest States Department of Defence; 2006.
41. Al-Rebiesh F. *Adaptively Improving Performance Stability of Cloud Based Application Using the Modern Portfolio Theory*. Ph.D. thesis. School of Computer Science, University of Birmingham, UK; 2016.
42. Kazman R, Klein M, Clements P. Atam: method for architecture evaluation. Technical report CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA; 2000.
43. Saaty R. The analytic hierarchy process—what it is and how it is used. *Math Model*. 1987;9(3):161-176.
44. van Lamsweerde A, Letier E. Handling obstacles in goal-oriented requirements engineering. *IEEE Trans Softw Eng*. 2000;26(10):978-1005.
45. Joseph CT, Chandrasekaran K. Straddling the crevasse: a review of microservice software architecture foundations and recent advancements. *Softw Pract Exper*. 2019;49(10):1448-1484.
46. Ahmadvand M, Ibrahim A. Requirements reconciliation for scalable and secure microservice (de)composition. Proceedings of the 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW); 2016:68-73.
47. Dragoni N, Giallorenzo S, Lafuente AL, et al. Microser-vices: yesterday, today, and tomorrow. *Present and Ulterior Software Engineering*. Springer; 2017:195-216.
48. Do NH, Van Do T, Thi Tran X, Farkas L, Rotter C. A scalable routing mechanism for stateful microservices. Proceedings of the 2017 20th Conference on Innovations in clouds, Internet and Networks (ICIN); 2017:72-78.
49. Srirama SN, Adhikari M, Paul S. Application deployment using containers with auto-scaling for microservices in cloud environment. *J Netw Comput Appl*. 2020;160:102629.
50. Abdullah M, Iqbal W, Erradi A. Unsupervised learning approach for web application auto-decomposition into microservices. *J Syst Softw*. 2019;151:243-257.
51. Avritzer A, Ferme V, Janes A, et al. Scalability assessment of microservice architecture deployment configurations: a domain-based approach leveraging operational profiles and load tests. *J Syst Softw*. 2020;165:110564.
52. Freire AFAA, Sampaio AF, Carvalho LHL, Medeiros O, NC M ̧c. Migrating pro-duction monolithic systems to microservices using aspect oriented programming. *Softw Pract Exper*. 2021;51:1280–1307.
53. Chen F, Zhang L, Lian X. A systematic gray literature review: the technologies and concerns of microservice application programming interfaces. *Softw Pract Exper*. 2021;51:1483-1508.
54. Balalaie A, Heydarnoori A, Jamshidi P, Tamburri DA, Lynn T. Microservices migration patterns. *Softw Pract Exper*. 2018;48(11):2019-2042.
55. Taneja M, Jalodia N, Byabazaire J, Davy A, Olariu C. Smartherd management: a microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming. *Softw Pract Exper*. 2019;49(7):1055-1078.

How to cite this article: Hassan S, Bahsoon R, Buyya R. Systematic scalability analysis for microservices granularity adaptation design decisions. *Softw: Pract Exper*. 2022;1-24. doi: 10.1002/spe.3069