Systematic Scalability Analysis for Microservices Granularity Adaptation Design Decisions

Sara Hassan[a,*], Rami Bahsoon[a], Rajkumar Buyya[b]

[a] *University of Birmingham, Birmingham, UK*
[b] *The University of Melbourne, Melbourne, Australia*

Abstract

Microservices have gained wide recognition and acceptance in software industries as an emerging architectural style for autonomic, 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 decision supports systems for microservice granularity adaptation is neither: 1) driven by microservice-specific scalability dimensions and metrics nor, 2) follow systematic scalability analysis to render scalability-aware adaptation decisions. In this paper, 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 rendering scalability-aware granularity adaptation decisions.

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

1. Introduction

Microservices have recently made their way to various important industries such as transport [1], entertainment [2, 3, 4], and retail [5, 6, 7]. At a very high level, microservices can be regarded as "autonomic, replaceable and deployable artefacts of (the transition to microservices) that encapsulate fine-grained business functionalities presented to system (end) users through standardised interfaces [8, p.3]."

In [8] we term the transition to microservices as microservitization. In [9], we define microservitization as a paradigm shift services/components are transformed into microservices — a more fine-grained and autonomic form of services to introduce added value to the architecture. This shift involves dramatically

*Corresponding author, Present affiliation: Birmingham City University, Birmingham, UK
Email addresses: ssh195@cs.bham.ac.uk (Sara Hassan), r.bahsoon@cs.bham.ac.uk (Rami Bahsoon), rbuyya@unimelb.edu.au (Rajkumar Buyya)

changing how a plethora of technical activities (including but not limited to architectural design, development and deployment) are carried out [9].

Microservitization aims to align technical decisions in these activities with a microservice adopter's business objectives. For example, microservitization involves designing microservices so that they isolate business functionalities and allow them to interact through standardised interfaces. The isolation aims at optimising the autonomy and replaceability of the service(s). It can also promote autonomous management (i.e., decentralised governance) of the service(s), better traceability, accountability and auditing for the service(s) and their provision in the event of failure. These are examples of value added to the software architecture through isolating business functionalities into microservices. The value added can be introduced as a result of the flexibility of the microservice architecture's ability to cope with operation, maintenance and evolution uncertainties. Ultimately, this can also relate to reduced maintenance costs and cost-effective quality of service (QoS) provision to both the service provider and end users.

Among the critical microservitization 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, thereby moving to a finer or more coarse grained granularity level. This problem is critical because "splitting (microservices) too soon can make things very difficult to reason about. It will likely happen that you (the software architect) will learn in the process [5]." This problem is of significance to both brownfield and greenfield developments [5] since it affects how an abstract architecture can be refined to a concrete configuration leveraging microservices.

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 significant requirements that is sought after by microservice adopters. Henceforth, it is essential that any decision support system for microservice granularity adaptation shall be aware of the dimensions and metrics that are important for the scalability of a microservices design. Subsequently, the decision support system should consider these dimensions and metrics to render a scalability-aware granularity adaptation decisions. The inclusion/exclusion of these dimensions and/or metrics as input to a decision support system can render different granularity adaptation strategies. Examples of dimensions that are important for the scalability of a microservice architecture include: end user base size, 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 paper aims to address the following gap in analysing the scalability of a microservice design: *Inadequacy of guidance regarding the dimensions and metrics which are important for the scalability of a microservices architecture and shall be considered to render scalability-aware granularity adaptation decisions:* The current state-of-art shows inadequacies of knowledge and conscience among microservice adopters, regarding the scalability dimensions and metrics that should drive the microservice design. As scalability of a microservice architecture is one of the architecturally significant requirements of microservice adopters, it is

essential that a decision support system shall be aware of the factors that are critical for the scalability of the microservices architecture itself. "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 huge 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. Further consider that microservice granularity adaptation decisions in NetWatch are supported by a fictional system — MicroAdapt; the input to MicroAdapt is determined by NetWatch architects. If the input to MicroAdapt only dictates it should only consider scalability with respect to logical interdependencies, then the granularity adaptation decisions suggested by MicroAdapt can be less scalability-aware and might lead to less added value for NetWatch. A relatively more scalability-aware decision would have been suggested if the input dictated considering scalability with respect to both the logical dependencies and shared data volumes. Informing the input to MicroAdapt requires guidance to identify and systematically analyse the dimensions and metrics important for the scalability of NetWatch.

To address the above gaps, the novel contribution of this paper is in two-folds:

- A working catalogue of the 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 decision support systems for 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 rendering scalability-aware granularity adaptation decisions.

- An application of scalability goal-obstacle analysis [12, 11] in new context that relates to reasoning about the scalability of 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 the scalability goals. Henceforth, the systematic analysis can justify the importance of considering a given scalability dimension, which is important to render a scalability-aware granularity adaptation decisions. The goal-obstacle analysis for microservice scalability is informed by dimensions from the catalogue.

  Scalability goal-obstacle analysis was first attempted in [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 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 were scaled along relevant dimensions. Scalability goal-obstacle analysis "attempts to establish a uni-

form 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. (e.g., [15], [16], and [17]). Systems in those domains are fundamentally different from those targeted in our context so scalability assessment approaches developed for these domains are non-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 identify and analyse obstacles along the dimensions from our catalogue. Ultimately, the dimensions and metrics which are deemed to be critical to the analysis should be provided as input to decision support systems to render scalability-aware granularity adaptation decisions.

The rest of the paper 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 that has informed our microservice-specific catalogue of scalability dimensions and metrics. 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 compare and contrast our work against relevant existing literature. In Section 7 we conclude by reflecting on the significance of our contributions and 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. 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 1 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 granularity adaptation strategy that can indeed introduce added value to the Filmflix architecture. For

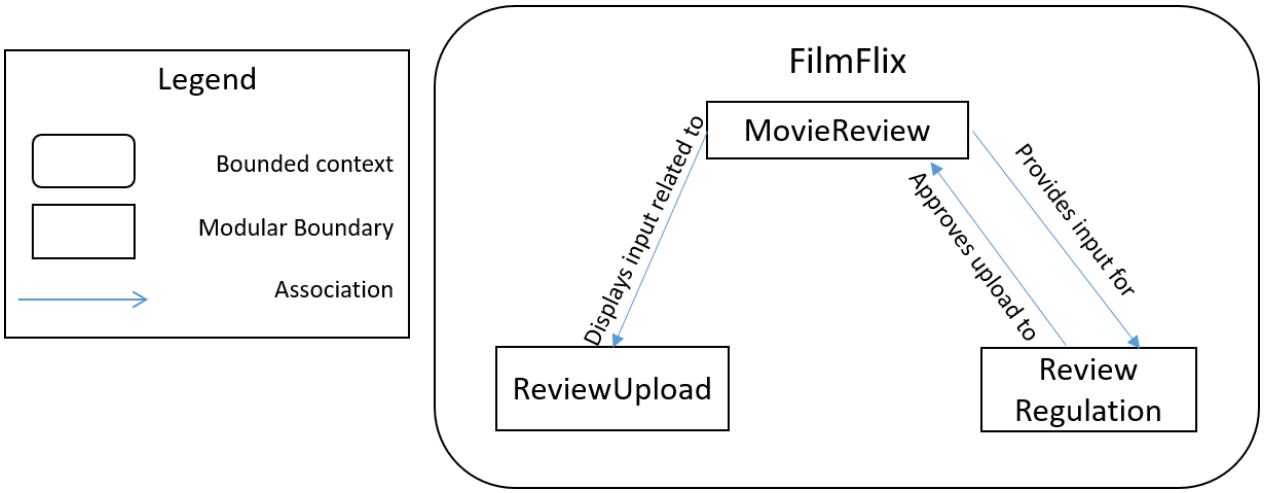

Figure 1: Filmflix architecture; a bounded context is an independent area in the firm's domain, a modular boundary is considered a single microservice and an association is the interaction between microservices

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 rendering 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 the decision of merging the microservices to reduce the data exchange calls across them and enhance the overall Filmflix added value. 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 in order to suggest a scalability-aware granularity adaptation strategy. Guidance is therefore essential to identify the important scalability dimensions and metrics for Filmflix then analyse potential obstacles for the satisfiability of Filmflix's goals if it is scaled along these dimensions. Other cases may require other dimensions as input and as the application domain and/or drivers to microservices entails.

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 rendering 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 render 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 [9] to render 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 and decision support systems for microservice granularity adaptation. Moreover, scalability goal-obstacle analysis by definition has distinct steps (explained in Section 4) making it fitting for addressing the target problem of this paper. Both contributions are complementary and aimed at analysing the ability of a decision support system to render 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 paper [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 render scalability-aware granularity adaptation decisions.

3.1. State-of-the-art Scalability Dimensions for Microservices

Tables 2 and 1 summarise whether and how scalability is considered when rendering granularity adaptation decisions. The publications summarised in Table 2 and 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. We further categorise the representative examples according to their density; Table 1 summarises publications which present a set (rather than a single) of microservice architectural design patterns that can support granularity adaptation decisions. Along with each pattern we summarise the scalability dimensions and/or metrics which can affect adopting them. When compiling Table 1, we made our best effort to not duplicate patterns that have been mentioned in multiple publications. On the other hand, Table 2 summarises publications where a single "process" that supports reasoning about granularity adaptation decisions is presented.

Where the "process" (i.e. decision support system) strives for scalability-aware decisions, the dimensions and/or metrics considered are presented in last two columns of Tables 1 and 2. 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 on the other hand 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].

Decision Support System	Scalability Considered	Scalability Assessment Approach	Pattern Supporting Granularity Adaptation Decisions	Scalability Dimensions Considered	Scalability Metrics Considered
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	
			Systematic standardisation of 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	
A guide book of the best practices for architecting microservice-based systems, including how to modernize 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	

Decision Support System	Scalability Considered	Scalability Assessment Approach	Pattern Supporting Granularity Adaptation Decisions	Scalability Dimensions Considered	Scalability Metrics Considered
			Reference environments/ stubs/ consumer-driven contract tests	number of integration test cases spanning multiple microservices, number of consumer-driven contracts to be met	
			Internal and external interfaces	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/Server-side 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	
A guide book for how to design applications for maximum uptime, performance, and return on investment; these are common drivers of microservitization. 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	

Decision Support System	Scalability Considered	Scalability Assessment Approach	Pattern Supporting Granularity Adaptation Decisions	Scalability Dimensions Considered	Scalability Metrics Considered
			Request/Asynchronous One-way (i.e. notifications)	volume of inter-microservice 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 microservitization 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	

Table 1: Summarising whether and how scalability is considered when representative examples from [9] (i.e. decision support systems) suggest granularity adaptation decisions; the examples in this table present a set of patterns and/or best practices that can entail granularity adaptation; for each pattern the scalability dimensions which are deemed relevant to it (as discussed in the respective publication) are presented

Decision Support System	Scalability Considered	Scalability Assessment Approach	Scalability Dimensions Considered	Scalability Metrics Considered
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]				
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 microservitization 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]				

Decision Support System	Scalability Considered	Scalability Assessment Approach	Scalability Dimensions Considered	Scalability Metrics Considered
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 business functionalities, number of microservices per cross-cutting business 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 modernization 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 business functionalities, volume of read/write database operations, infrastructure platform migration costs	transactional consistency after pursuing adaptation
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	

Table 2: Summarising whether and how scalability is considered by the "processes" (i.e. decision support systems) in [9]; 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 rendering scalability-aware decisions

Looking at Table 2, 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 pave the way for our first contribution — the working catalogue of microservice-specific scalability dimensions and metrics aiming to provide specialised guidance to identify the dimensions and metrics which are important for the scalability of a microservice architecture. Consequently, these dimensions are essential to consider in order to render a

scalability-aware granularity adaptation decision regarding a specific microservice architecture.

3.2. Catalogue of Microservice-Specific Scalability Dimensions and Metrics

Compiling Table 2, we present our catalogue of scalability dimensions and metrics in Tables 3 and 4 respectively; dimensions and metrics are categorised according to their nature (e.g., organisational, data-related and developmental).

Category	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 bounded contexts
Deployment	containerisation/virtualisation costs, deployment pipeline costs, number of front-end servers, number of interdependent deployment configuration settings, number of RESTful API gateways, 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 requests in synchronous communication mechanisms
Operational	Operational and infrastructure 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

Table 3: Working catalogue of microservice-specific scalability dimensions compiled from Tables 1 and 2; it is potentially essential to consider these dimensions to render scalability-aware granularity adaptation decisions

Category	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

Table 4: Working catalogue of microservice-specific scalability metrics compiled from Table 2; relevant scalability metrics can measure the ability of a microservice architecture to scale along the relevant dimensions from Table 3

Given these catalogues, microservice adopters can manually elicit the relevant scalability dimensions and metrics. It is worth noting that not all scalability dimensions are critical for the development of microservices and its scalability. In practice, some of them will vary within small ranges, imposing very little impact on scalability. The same can be said about scalability metrics; the relevance of a scalability metric highly depends on its criticality to the microservice architecture in question. 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. On the other hand, the analysis justifies the criticality of dimensions picked from the catalogue given a microservice architecture.

4. Systematic Scalability Analysis for Microservice Granularity

4.1. KAOS Goal-Oriented Modelling

The Keep All Objectives Satisfied (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 business 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].

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 characterize 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*. Goal loads are also referred to as scaling dimensions — application domain and system design properties that may exhibit a wide variation in values during the lifetime of the system.

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 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 [41, 12] so we utilise the same technique for consistency. We envision however that objective techniques such as those used in [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 [44, 11]: 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. The state-of-the-art and -practice in the microservice industry has been to subjectively assess scalability. On the

other hand, scalability goal-obstacle analysis has not been applied in the context of microservices. Hence, our application of scalability goal obstacle analysis to microservices in this paper is a two-fold novelty.

5. Analysis

In this section we use our Filmflix architecture from Section 2 as a case study for showing how our contributions address the problems of concern in Section 1. We argue that our contributions are flexible enough such that the same illustration applies regardless the size of the microservice architecture. Therefore, we envision that our analysis results are applicable for microservices that are larger than Filmflix. Nevertheless, we acknowledge that application on industrial-scale case studies is necessary in the future to verify this.

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.

Reflecting on utilising our catalogue for Filmflix, we discuss the potential comprehensiveness of Tables 3 and 4. 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 input to the system providing decision support 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 2 using the notation explained in Figure 3. In this subsection, we describe the role of each agent to justify their responsibilities for different goals.

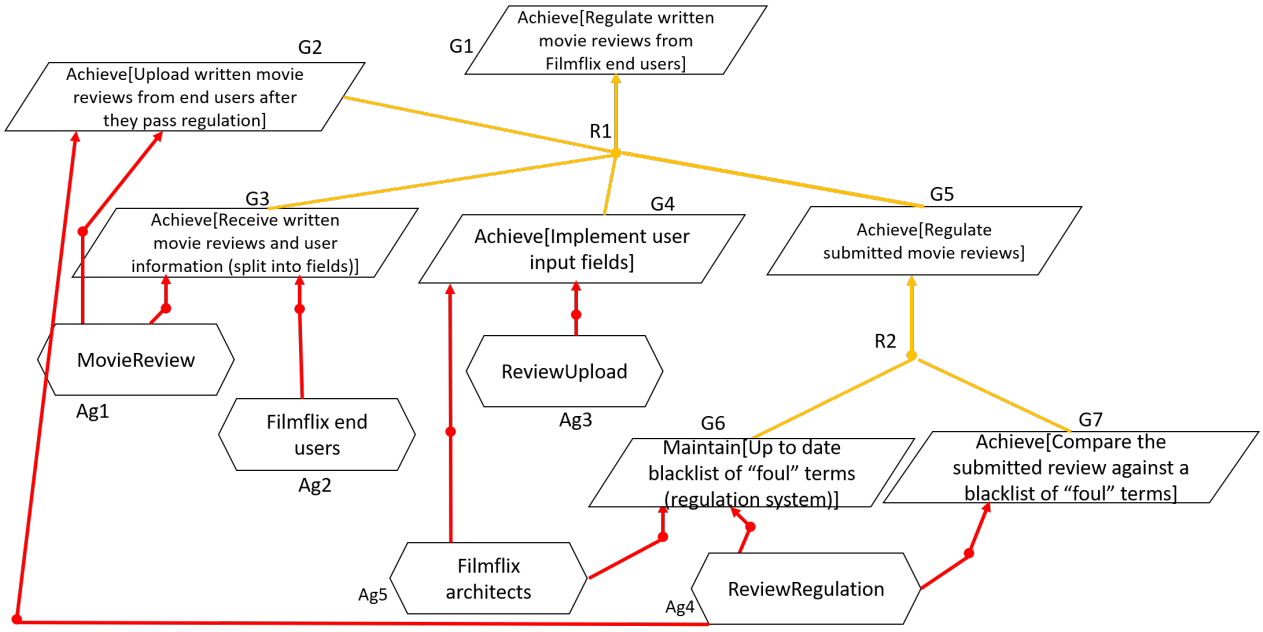

Figure 2: Refinement of the *Achieve[Regulate written movie reviews from Filmflix end users]* goal in Filmflix's architecture

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 2). Receiving input is represented by G3 —

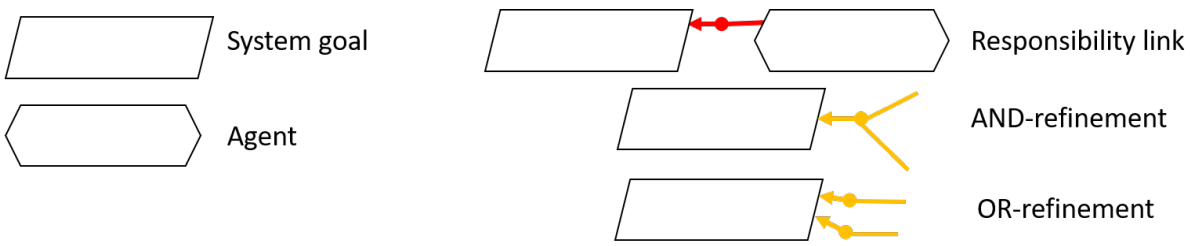

Figure 3: Legend for the KAOS modelling notation

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 2 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: 1) compares it to a pre-defined blacklist (G7 in Figure 2) and, 2) 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 2) 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 2.

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 5 and 6 summarise the obstacle identification and assessment results.

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	

Table 5: Identifying relevant scalability dimensions and metrics (guided by Tables 3 and 4) for the modelled goals of the Filmflix architecture; this table is used to identify the scalability obstacles in Table 6

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 2. 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.

Scalability Obstacle	Criticality	Likelihood	Rationale
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 2).
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 2 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 are obstruct achieving G2, G6 and G7 with acceptable performance.

Table 6: Assessing scalability obstacles of the Filmflix architecture

Reflecting on Table 6, there are three high risk obstacles that need to be resolved in order 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 [12, 11], 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 4. On 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 4). If Film-

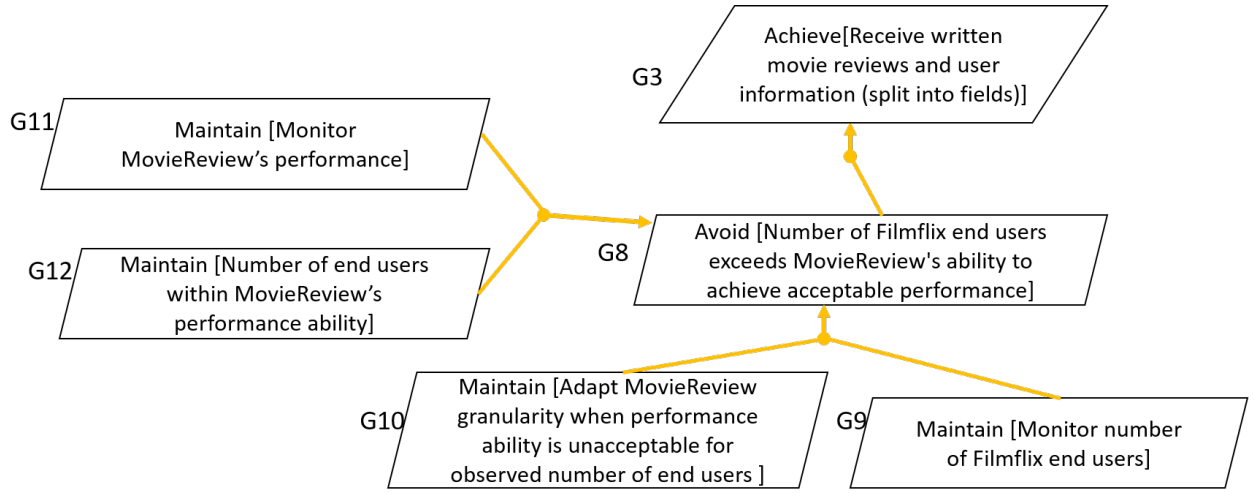

Figure 4: Resolving the *Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix end users exceeds MovieReview's ability to achieve acceptable performance]* scalability obstacle prevention goal

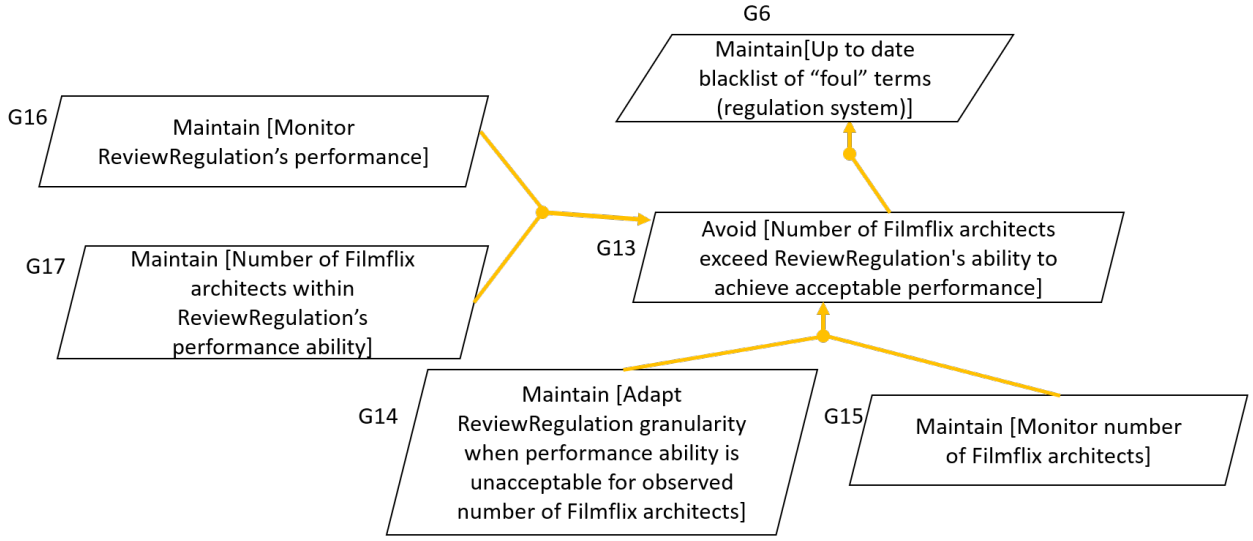

Figure 5: Resolving the *Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix architects exceed ReviewRegulation's ability to achieve acceptable performance]* scalability obstacle prevention goal

flix 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 4 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 4). This ability is derived from monitoring MovieReview's performance (G11 in Figure 4). This route of achieving G8 does not involve adapting MovieReview's granularity and therefore it is does not require using microservice granularity adaptation decision support.

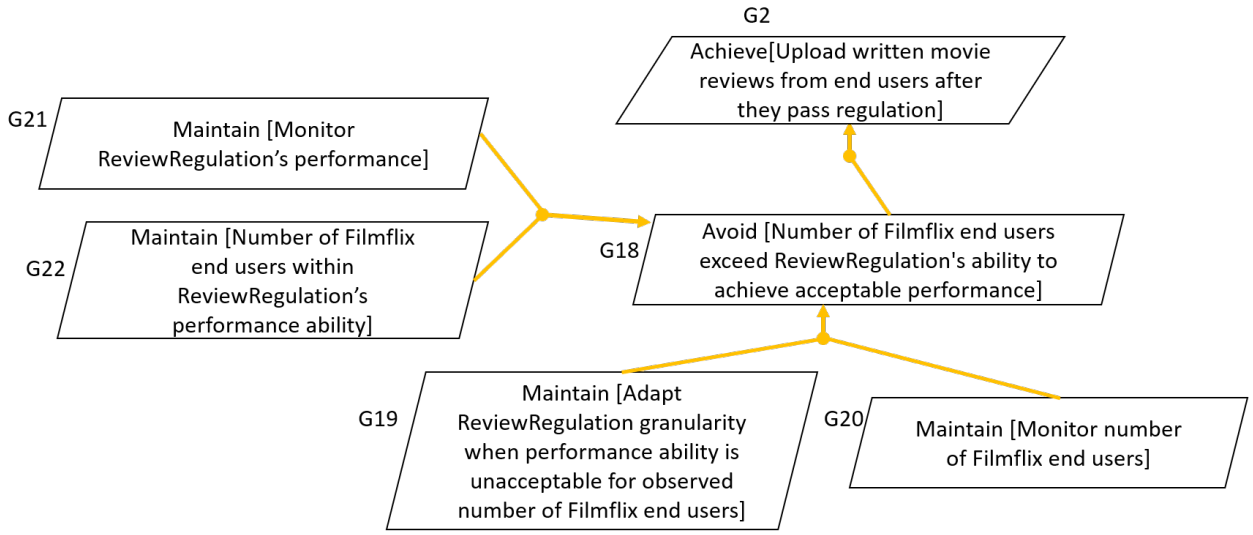

Figure 6: Resolving the *Number of Filmflix end users exceed ReviewRegulation's ability to achieve acceptable performance* obstacle by introducing and refining the *Avoid [Number of Filmflix end users exceed ReviewRegulation's ability to achieve acceptable performance]* scalability obstacle prevention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

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 5. Although G13 is presented as a refinement of G6 in Figure 5, we appreciate that G13 is also relevant to achieving G2 and G7 from Figure 2. 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 5). The 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 5 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 5). This ability is estimated from monitoring ReviewRegulation's performance (G16 in Figure 5). This route of achieving G13 does not involve adapting ReviewRegulation's granularity and therefore it is does not require using microservice granularity adaptation decision support.

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 ReviewRegulation's ability to achieve acceptable performance]*. This goal can be achieved via two routes, illustrated as an OR-refinement of G18 in Figure 6.

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 6). 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 6 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 6). This ability is derived from monitoring ReviewRegulation's performance (G21 in Figure 6). This route of achieving G18 does not involve adapting ReviewRegulation's granularity and therefore it is does not require using microservice granularity adaptation decision support.

5.3. Reflection

5.3.1. Catalogue Comprehensiveness

Reflecting on the scalability dimensions in Table 6, 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 3. The number of end users is in essence the end user base size under the QoS provision category of Table 3. 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 6 potentially maps to the number of countries the application is serving and audit compliance considerations in Table 3. One of the scalability dimensions which have not been deemed critical — volume of received reviews — can be mapped to the end user base size under the architectural category in Table 3. The mapping is based on the relationship between volume of received reviews and number of Filmflix end users captured in Table 6. 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 6. All the scalability metrics considered 5 are related to performance; it is present under the QoS provision category of Table 4.

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 [9]. In particular, we used all the relevant publications from [9] to compile the catalogues due to our confidence in this paper'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 3 has a corresponding category of metrics in Table 4. The categories in Table 4 are a direct representation of the metrics we came across in the examined microservice literature (summarised in Tables 2 and 1). Therefore, our catalogue unveils a research gap in the microservice state-of-the-art and -practice regarding the existence of adequate scalability metrics.

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 input to microservice granularity adaptation decision support; 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.

6. Related Work

In this section, we compare and contrast our work against existing literature that accounts for microservice scalability requirements. Table 7 summarises the existing literature we examined along with the comparison results along three dimensions: acknowledging the significance of scalability for microservices, providing systematic guidance for scalability-aware design of microservices, and whether or not this guidance is specific to a certain application domain.

Work	Acknowledges Scalability Significance	Provides Systematic Scalability Guidance	Provides non-domain specific guidance
S. Hassan, R. Bahsoon, R. Kazman (2019) [9]	✓		
C. Joseph, K. Chandrasekaran (2019) [45]	✓		
M. Ahmadvand, A. Ibrahim (2016) [46]	✓		✓
N. Dragoni et al. (2017) [47]	✓		
N. H. Do et al. (2017) [48]	✓	✓	
S. N. Srirama, M. Adhikari, S. Paul (2020) [49]	✓	✓	
M. Abdullah, W. Iqbal, A. Erradi[50]	✓	✓	
A. Avritzer et al. (2020) [51]	✓		✓
Microservice-specific scalability guidance (this work)	✓	✓	✓

Table 7: Summarising the results of comparing and contrasting our work against existing relevant literature

Our contributions in this paper are inspired by the systematic mapping study conducted and reported in

[9]. However, the objectives of [9] are broader than the target of this paper. In this paper we focus on compiling guidance for rendering scalability-aware granularity adaptation decisions. In [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 [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 [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 renders 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, [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, [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 generic hence making it applicable to applications where other approaches to scaling are taken. Furthermore, our work is focussed on rendering scalability-aware granularity adaptation decisions. In [48], the approach is targeted at scaling microservice architectures regardless granularity of the microservices.

Another practical auto-scaling policy is proposed in [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 utilizing 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 [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 generic since it does not restrict the drivers of adaptation to improving performance only.

An objective, systematic approach for assessing scalability of microservice architectures is proposed in [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 [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.

7. Conclusions and Future Directions

In this paper, 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 in order to render scalability-aware granularity adaptation decisions for it. We compile our catalogue by reviewing the state-of-the-art in decision support systems for microservice granularity adaptation from [9]. Secondly, we report on a new application of scalability goal-obstacle analysis [12, 11] 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 against 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. We also appreciate that the practicality of transferring goal-obstacle analysis to industrial-scale microservice applications needs to be investigated in the short-term future. Such investigation will pave the way to another important criterion: "does a scalability-aware granularity adaptation decision actually add more value to a microservice architecture compared to the value added if the decision does not consider scalability?"

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

References

[1] E. Reinhold, Lessons learned on uber's journey into microservices, https://www.infoq.com/presentations/uber-darwin (Jul 2016).

[2] S. Godwin, Cloud-based microservices powering bbc iplayer (jun 2016).
URL https://www.infoq.com/presentations/bbc-microservices-aws?utm_campaign=infoq_content&utm_source=infoq&utm_medium=feed&utm_term=Microservices

[3] K. Probst, J. Becker, Engineering trade-offs and the netflix api re-architecture, https://medium.com/netflix-techblog/engineering-trade-offs-and-the-netflix-api-re-architecture-64f122b277dd (aug 2016).

[4] P. Calcado, No free lunch, indeed: Three years of micro-services at soundcloud, http://www.infoq.com/presentations/soundcloud-microservices (Jan).

[5] Z. Dehghani, Zhamak dehghani real world microservices: Lessons from the frontline, https://youtu.be/hsoovFbpAoE (feb 2015).

[6] G. Steinacker, On monoliths and microservices, https://dev.otto.de/2015/09/30/on-monoliths-and-microservices/ (sep 2015).

[7] T. Wagner, Microservices without the servers (Sep 2015).
URL https://aws.amazon.com/blogs/compute/microservices-without-the-servers/

[8] S. Hassan, R. Bahsoon, Microservices and their design trade-offs: A self-adaptive roadmap, in: 13th IEEE International Conference on Services Computing (SCC), San Francisco, USA, 2016.

[9] S. Hassan, R. Bahsoon, R. Kazman, Microservice transition and its granularity problem: A systematic mapping study, CoRR abs/1903.11665. arXiv:1903.11665.
URL http://arxiv.org/abs/1903.11665

[10] N. Kulkarni, V. Dwivedi, The role of service granularity in a successful soa realization a case study, in: 2008 IEEE Congress on Services - Part I, 2008, pp. 423–430.

[11] L. D. de Cerqueira, A framework for the characterization and analysis of software systems scalability, Ph.D. thesis, University College London (University of London) (2010).

[12] L. Duboc, E. Letier, D. S. Rosenblum, Systematic elaboration of scalability requirements through goal-obstacle analysis, IEEE Transactions on Software Engineering 39 (1) (2013) 119–140.

[13] L. Duboc, D. Rosenblum, T. Wicks, 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, ACM, New York, NY, USA, 2007, pp. 375–384.

[14] R. IT, A kaos tutorial (oct 2007).
URL http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

[15] J. Edmund M. C., O. Grumberg, D. Peleg, Model Checking, The MIT Press, 1999.

[16] P. Jogalekar, M. Woodside, Evaluating the scalability of distributed systems, IEEE Transactions on Parallel and Distributed Systems 11 (6) (2000) 589–603.

[17] P. Jogalekar, C. Woodside, A scalability metric for distributed computing applications in telecommunications, in: V. Ramaswami, P. Wirth (Eds.), Teletraffic Contributions for the Information Age, Vol. 2 of Teletraffic Science and Engineering, Elsevier, 1997, pp. 101 – 110.

[18] S. Tonse, Scalable microservices at netflix. challenges and tools of the trade, http://www.infoq.com/presentations/netflix-ipc (mar 2015).

[19] C. Watson, S. Emmons, B. Gregg, A microscope on microservices, https://medium.com/netflix-techblog/a-microscope-on-microservices-923b906103f4 (feb 2015).

[20] D. Taibi, V. Lenarduzzi, On the definition of microservice bad smells, IEEE Software 35 (3) (2018) 56–62.

[21] L. Krause, Microservices: Patterns and Applications: Designing Fine-Grained Services by Applying Patterns, Lucas Krause, 2015.
URL `https://books.google.co.uk/books?id=dd5-rgEACAAJ`

[22] E. Wolff, Microservices: Flexible Software Architecture, Pearson Education, 2016.
URL `https://books.google.co.uk/books?id=zucwDQAAQBAJ`

[23] M. Nygard, Release It! Design and Deploy Production-Ready Software, Pragmatic Bookshelf, 2007.

[24] S. Newman, Building Microservices, 1st Edition, O'Reilly Media, 2015.

[25] C. Richardson, Microservice Patterns, Manning Publications Company, 2018.
URL `https://books.google.co.uk/books?id=UeK1swEACAAJ`

[26] C. Posta, The hardest part of microservices: Calling your services, http://blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/ (apr 2017).

[27] S. Penchikala, Susanne kaiser on microservices journey from a startup perspective, https://www.infoq.com/news/2017/07/kaiser-microservices-journey (jul 2017).

[28] S. Penchikala, Managing data in microservices, https://www.infoq.com/news/2017/06/managing-data-in-microservices (jun 2017).

[29] S. Vlaovic, R. Pilani, S. Parulekar, S. Handa, Netflix billing migration to aws, https://medium.com/netflix-techblog/netflix-billing-migration-to-aws-451fba085a4 (Jan 2016).

[30] D. Iffland, Q&a with intuit's alex balazs, https://www.infoq.com/articles/intuit-alex-balazs-node-services (Jun 2016).

[31] C. Posta, Low-risk monolith to microservice evolution part i, http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution/ (jun 2015).

[32] M. Fowler, Strangler application, https://www.martinfowler.com/bliki/StranglerApplication.html (Jun 2004).

[33] M. Fowler, Event interception, http://www.martinfowler.com/bliki/EventInterception.html (Jun 2004).

[34] A. Levcovitz, R. Terra, M. T. Valente, Towards a technique for extracting microservices from monolithic enterprise systems, CoRR abs/1605.03175. arXiv:1605.03175.
URL http://arxiv.org/abs/1605.03175

[35] K. Bakshi, Microservices-based software architecture and approaches, in: 2017 IEEE Aerospace Conference, 2017, pp. 1–8.

[36] H. Knoche, W. Hasselbring, Using microservices for legacy software modernization, IEEE Software 35 (3) (2018) 44–49.

[37] O. Mustafa, J. M. Gómez, Sustainable approach for improving microservices based web application, in: Sustainability Dialogue: International Conference on Sustainability and Environmental Management, 2017.

[38] P. Calçado, Layering microservices, http://philcalcado.com/2018/09/24/services_layers.html (Sept 2018).

[39] A. van Lamsweerde, Goal-oriented requirements engineering: a guided tour, in: Proceedings Fifth IEEE International Symposium on Requirements Engineering, 2001, pp. 249–262.

[40] U. S. D. of Defense, Risk Management Guide for DOD Acquisition, Unitest States Department of Defence, 6th Edition (aug 2006).

[41] F. Al-Rebiesh, Adaptively improving performance stability of cloud based application using the modern portfolio theory, Ph.D. thesis, School of Computer Science (2016).

[42] R. Kazman, M. Klein, P. Clements, Atam: Method for architecture evaluation, Tech. Rep. CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2000).

[43] R. Saaty, The analytic hierarchy process—what it is and how it is used, Mathematical Modelling 9 (3) (1987) 161 – 176.

[44] A. van Lamsweerde, E. Letier, Handling obstacles in goal-oriented requirements engineering, IEEE Transactions on Software Engineering 26 (10) (2000) 978–1005.

[45] C. T. Joseph, K. Chandrasekaran, Straddling the crevasse: A review of microservice software architecture foundations and recent advancements, Software: Practice and Experience 49 (10) (2019) 1448–1484.

[46] M. Ahmadvand, A. Ibrahim, Requirements reconciliation for scalable and secure microservice (de)composition, in: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), 2016, pp. 68–73.

[47] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina, Microservices: yesterday, today, and tomorrow, in: Present and ulterior software engineering, Springer, 2017, pp. 195–216.

[48] N. H. Do, T. Van Do, X. Thi Tran, L. Farkas, C. Rotter, A scalable routing mechanism for stateful microservices, in: 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), 2017, pp. 72–78.

[49] S. N. Srirama, M. Adhikari, S. Paul, Application deployment using containers with auto-scaling for microservices in cloud environment, Journal of Network and Computer Applications 160 (2020) 102629.

[50] M. Abdullah, W. Iqbal, A. Erradi, Unsupervised learning approach for web application auto-decomposition into microservices, Journal of Systems and Software 151 (2019) 243 – 257.

[51] A. Avritzer, V. Ferme, A. Janes, B. Russo, A. van Hoorn, H. Schulz, D. Menasché, V. Rufino, Scalability assessment of microservice architecture deployment configurations: A domain-based approach leveraging operational profiles and load tests, Journal of Systems and Software 165 (2020) 110564.