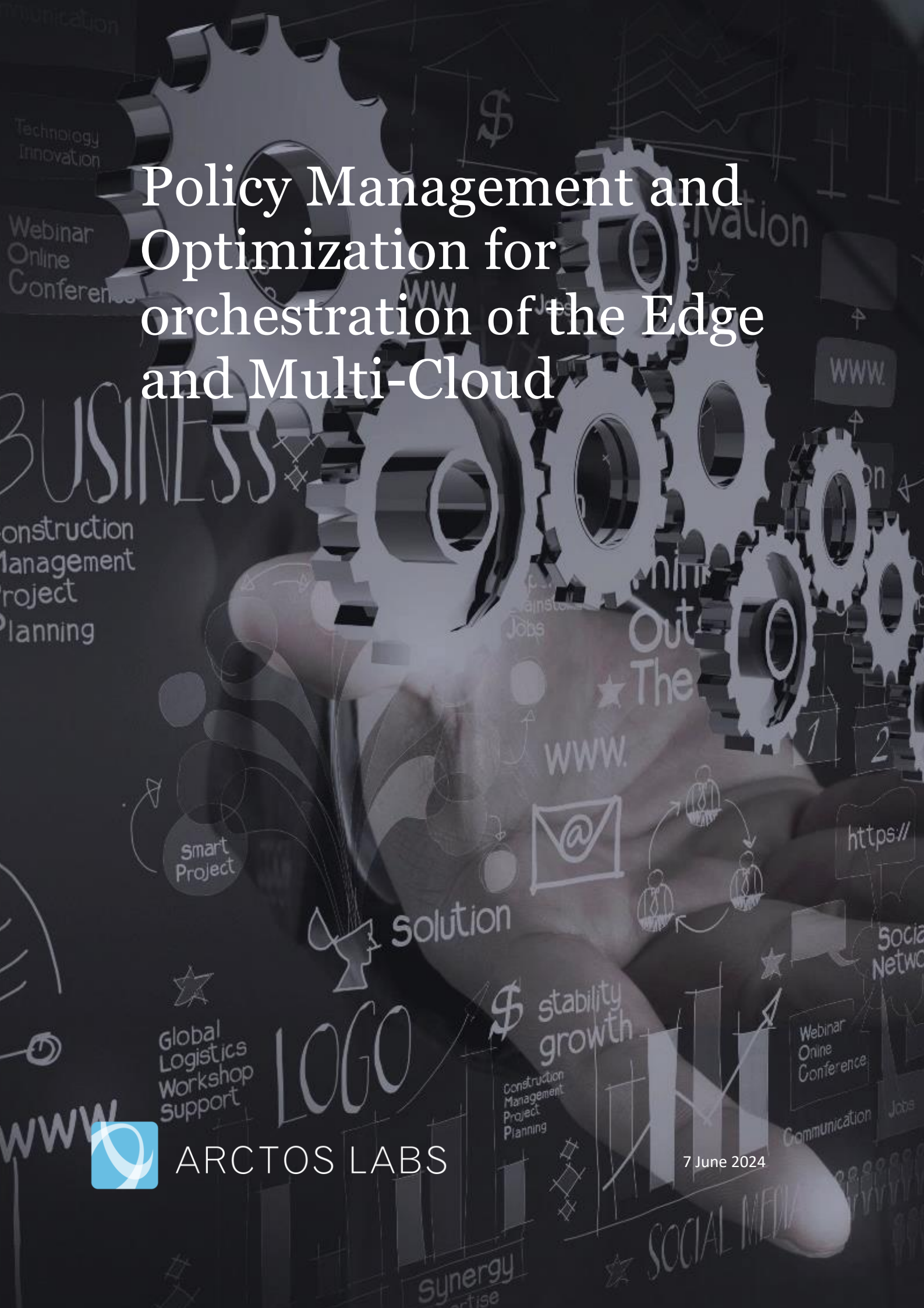


Policy Management and Optimization for orchestration of the Edge and Multi-Cloud

ARCTOS LABS

7 June 2024

There are very few services that work in a “one-size-fits-all” fashion. Most services [1] benefit from tailoring at various stages of their life cycle.

This paper addresses how policy management could be utilized to achieve such tailoring during the service orchestration process.

With the policy management paradigm service administrators can manage different aspects of how services are being orchestrated, thereby impacting their final behavior and achieving intended tailoring.

Policies are technology-independent expressions that complement and enhance service orchestration implementations, thereby adding a certain degree of programmability without the need to modify the service orchestration implementations. As outlined by this document, the policies are either rule-based, which means that decisions are taken depending on the fulfillment of certain criteria, or the policy decision is subject to an optimization evaluation.

The service orchestration implementation can thereby call for policy evaluation during the process of service orchestration. Orchestration here refers to both the provisioning, assurance, and retirement of services. Rule policies and optimization policies can enable intelligent and optimal workload placement, guide the use of resources, configure services, or reduce energy consumption just to name a few examples.

Policies may use information from a large range of sources, and may for example be related to users, usage situations, infrastructure resources and their usage, networking conditions, environmental issues, acceptable costs, configuration settings, and regulations. This implies that the policies are specific to the service type, and how it is supposed to be used (use case).

1 What is Policy Management

From Wikipedia [2] the definition of Policy Management is:
Policy-based management is a technology that can simplify the complex task of managing networks and distributed systems. Under this paradigm, an administrator can manage different aspects of a network or distributed system in a flexible and simplified manner by deploying a set of policies that govern its behaviour. Policies are technology independent rules aiming to enhance the hard-coded functionality of managed devices by introducing interpreted logic that can be dynamically changed without modifying the underlying implementation. This allows for a certain degree of programmability without the need to interrupt the operation of either the managed system or of the management system itself. Policy-

[1] Service in this document is used to capture various forms of functionality delivered by software being deployed at data centers, edge compute locations, networks, or clouds.

[2] https://en.wikipedia.org/wiki/Policy-based_management

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

based management can increase significantly the self-managing aspects of any distributed system or network, leading to more autonomic behaviour demonstrated by Autonomic computing systems.

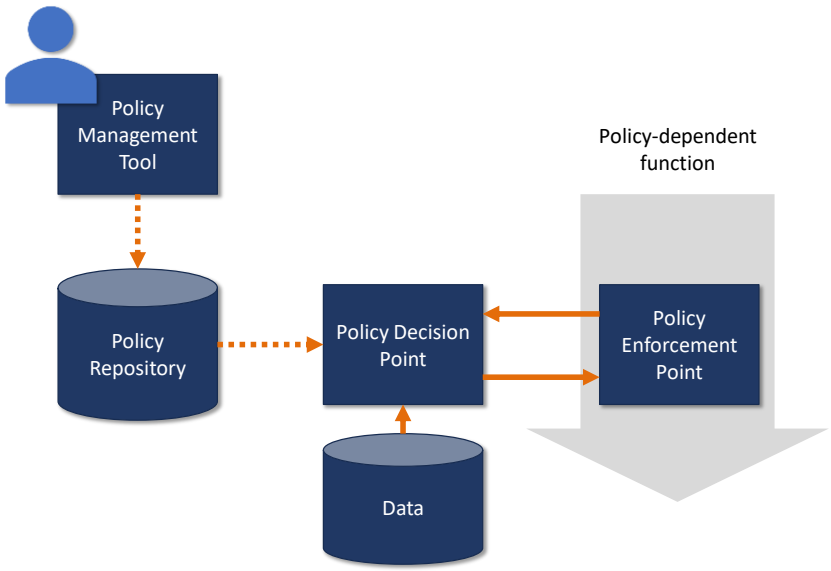
Moving from a centralized and homogeneous nature of the Cloud towards the heterogeneity of Multi-Cloud and Edge enhances the need for a powerful and flexible policy management to give enterprises and service providers proper influence on how services are orchestrated.

In the scope of this document, policies are being applied to

a) tailor generic services to better fit specific use cases or customers

b) dynamically adapt service orchestration to changing situations

IETF and DTMF have defined a generic high-level architecture for policy-based management. Here Policy Management is built from four elements: the Policy Management Tool, Policy Repository, Policy Decision Point (PDP), and Policy Enforcement Point (PEP).



As outlined so far, policies are used to add and tailor service orchestration. This means that the artifacts containing the policies have a life cycle of their own being separate from the policy-dependent function (in the context of this document being the service orchestration implementation[3]) that implements the policy enforcement.

Such separation opens for programmability where e.g. system administrators are given tools whereby they can tailor the final service orchestration behavior.

[3] The Service Orchestration implementation refers to functionality that implements the orchestration process for a specific service type. I.e. the functionality that runs inside an orchestrator.

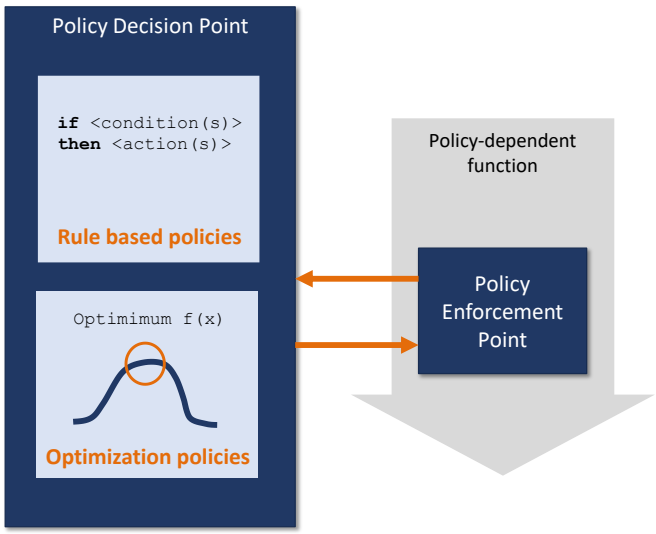
Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

2 Various types of policies

Policies are used to provide “decisions” to a policy-dependent function. It means that such policy-dependent functions have enforcement points where the decision outcome is unknown at the time of design of the policy-dependent functionality. The scheme is that the policy-dependent function invokes a policy decision point for a decision when a policy enforcement point has been reached.

The policy decision point executes a function that determines the response using data according to what is designed into the policy. Such data may come from the system that contains the policy-dependent function or it may be external.

We have two types of policies. They are either rule-based, which means that decisions are taken depending on the fulfillment of certain criteria, or the policy decision is subject to an optimization evaluation.



The policies executed by the policy decision point are artifacts (policy implementations). Those implementations are therefore subject to design and administration before they can be invoked and properly executed.

To support this, there is a policy management tool along with a policy repository. The policy repository is an entity that stores policies so that specific policies can be retrieved for policy evaluation.

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

3 Policy Management by Arctos Labs ECO

Arctos Labs offers two Policy management technologies; Rule-based built on Open Policy Agent (OPA)/Rego [4] and optimization policies built on MiniZinc [5]. Both are open-source technologies.

They serve two distinct purposes. OPA makes it easy to express policies related to, for example, scaling, regulatory rules, geography, quality, and topology, on both services as well as resource usage. OPA is a well-established technology with a large ecosystem and a wide range of application examples and serves as a suitable solution to manage policies related to service orchestration.

MiniZinc on the other hand is suitable for optimization problems, e.g., when there is a need to optimize the trade-off between certain parameters (e.g., quality vs cost). This may apply when there are several solutions possible and where a MiniZinc optimization policy then recommends the most optimal. It may also apply when there is resource competition where a MiniZinc optimization policy can select how resources are distributed between several competing services. Examples here are assurance and failover scenarios to keep as many services up and running as possible.

In our Edge Cloud Optimization portfolio, Arctos Labs has extended both technologies to make them more purposeful in an edge-to-cloud continuum, as described in the following sections.

3.1 Extending Open Policy Agent and MiniZinc with Policy dispatching

Open Policy Agent (OPA) is a powerful choice for a policy management solution and has an increasing uptake for a variety of use cases. It is easy to use, has a growing ecosystem of companies engaged, and scales well.

In the type of use cases Arctos Labs are addressing, we however identified the need for improvement. Those improvements have been added to our OPA-based technology for rule-based policy management.

Polices are implemented in OPA's native query language Rego and are executed by the Policy Decision Point at the time of the policy query. One of the key advantages of splitting the policy-dependent function from the policy implementations is to create a loose binding so that policies can be used to tailor service orchestration behavior.

It implies that a policy designer is a role separated from the design of the policy-dependent function. In larger organizations, there may be a range of persons that are tailoring the same service orchestration (policy-dependent function) required for a range of reasons.

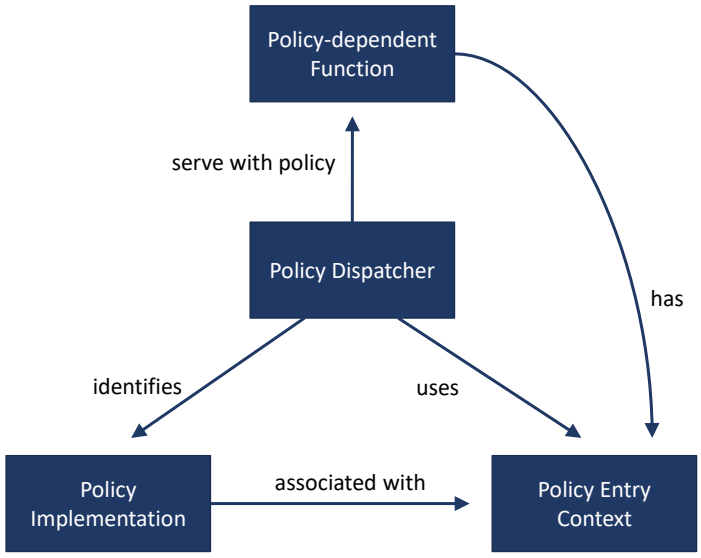
This may apply for example to tailoring for service quality or functionality specific to a country or market, or preferred use of resources depending on certain customer types

[4] https://www.openpolicyagent.org/
[5] https://www.minizinc.org/

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

or segments, etc. The above examples would see market or customer delivery teams developing their specific policy implementations.

To avoid that all these different tailorings must be captured by one single policy implementation we introduce a mechanism whereby the policy query is dispatched to the correct policy implementation based on context information included in the policy query.



This addition enables a service provider to have various parts of its organization develop their policy implementations and enter them into a common system relying on the policy dispatcher will resolve the policy query to the correct policy implementation.

The policy dispatching capability applies also to MiniZinc optimization policies.

3.2 Auto-generation of MiniZinc optimization policies

Writing MiniZinc policies (programs) can be challenging for example when it comes to representing objects and their relations to accurately reflect a real network with its resources and their properties, and parameterizing the program at execution time.

Each specific optimization policy is represented by a high-level optimization model that captures and expresses the specific optimization problem based on a network abstraction, optimization intents, and service constraints. ECO has the capability to code-generate and automatically execute MiniZinc optimization policies upon a policy query. The code generation is data-driven, so service constraint values, the network at hand, etc are included during the generation of the MiniZinc code.

Those high-level optimization models are onboarded onto our ECO portfolio and become the implementation of a specific solution.

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

3.3 Examples of optimization models

These solutions are covered in more depth by other documentation.

Model title	Main features
Workload placement optimization (Multi-cloud & Hybrid IT)	This model optimally distributes individual SW components/ workloads (of a multi-component service) across a resource infrastructure with multiple compute locations arranged in a topology both horizontally (geographically) and vertically (cloud-to-edge continuum) taking service constraints, such as latency, into account
Energy consumption optimization	This model optimizes a fleet of cloud and edge locations w.r.t energy consumption. The model will strive to reach optimal utilization of every location, also taking data transport into account given a specified service demand.
Cloud and Edge Infrastructure optimization	This model optimizes capacity dimensioning and TCO for a fleet of compute locations given a specified service demand
Network slice optimization	This model optimizes how cell capacity is assigned to network slices

4 Invoking policies

Both OPA policies and MiniZinc optimization policies are used in a similar pattern, which is that they are invoked by a Policy-dependent function (e.g., a service orchestration implementation) that is designed to use policies. For both technologies, the policy-dependent function uses the policy by querying the Policy decision point with a set of parameters. Upon execution of the policy implementation and using the included sources of data, a result is returned.

A policy-dependent function can contain multiple policy enforcement points.

4.1 Combining different types of policies

Arctos Labs ECO portfolio supports policies as a mechanism by which policies on the orchestration process (in the policy-dependent function) are imposed, i.e., reducing the possible “solution space” that the orchestrator can consider. Solution space here means the possible set of configurations, parameter sets, capacity allocations, etc.

It means that the service orchestration process may query several policies in an order decided by the service orchestration process. This order can include any mix of rule-based or optimization policies that fits the needs decided by the designer of the service orchestration process.

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152

Optimization policies can support the service orchestration in selecting the most optimal (e.g., lowest cost, most performing) solution when there are several options available during the orchestration process. Such an approach is powerful when optimal is subject to a multitude of combinatorial aspects.

Optimization policies can also assist the orchestration process in case of resource competition.

This may apply when:

a) provisioning multiple services that sometimes experience a lack of resource capacity.
b) in case of resource failures where the remaining capacity needs to be divided among existing services in a way that is as optimal as possible to keep services operational, albeit not fully preserving their intended QoS.

The specific sequence of policy queries is captured in each service orchestration implementation. Further, the possible outcomes of policies, as described above, need to be specifically implemented in the service orchestration implementation.

Arctos Labs Scandinavia AB, Bangårdsgatan 14, 972 35 Luleå, Sweden. E-mail: info@arctoslabs.com. Web: www.arctoslabs.com. Reg. no: 556743-9152