High performance rules engine for 5G and LTE control planes

Flexibility and scale at near line speed with negligible latency

White paper

Nokia’s Agile Rules Technology (A.R.T.) combines the benefits of general-purpose and hard-coded rules engines, without any of their drawbacks. It models the call flow and data requirements of a fundamental set of telecom use cases that represent key business models. Backed by over 150 patents, A.R.T. provides excellent run-time efficiency by anticipating and caching key data requirements of each call flow.

In this paper, we explain the types of rules engines, and then take an in-depth look at A.R.T. We then present the results of a performance study on the engine, and show how this technology, embedded into several Nokia products, helps communication service providers satisfy their subscribers and meet their operational goals.

Contents

Introduction

In a rapidly changing industry, service providers face many challenges that force them to react quickly to competitive threats, to implement new business models and to further reduce operational complexity and cost. To address those challenges, service providers rely on telecoms software solutions. And along with databases and analytics engines, rules engines are a key element of many IT systems.

A rules engine is a software element that executes one or more rules in a run-time production environment. Rules engine software is commonly provided as a component of a larger system, which, among other functions, provides the ability to:

- Register, define, classify and manage all the rules
- Verify consistency of rules definitions
- Define the relationships between different rules
- Relate some of these rules to other applications that are affected or need to enforce one or more of the rules.

In the context of telecommunications, rules could be used to define that a certain set of services or features is offered only to a select group of customers: for example, a rule could define that a music streaming package is offered free of charge only to subscribers of a premium shared data plan who use less than a defined amount of bandwidth every month.

To meet the aforementioned challenges, several multi-dimensional requirements exist for the use of rules engine-based products in telecommunications:

- Ease of use: Rules need to be quickly created and easily modified without long lead times and consulting/implementation costs. A rules engine must therefore be easy to use in house so that service providers can use it in support of their evolving business needs
- Flexibility: The breadth of use cases in this new era of unparalleled broadband usage and the Internet of Things (IoT) is extensive. The rules engine needs to be flexible enough to be able to model all of these cases while maintaining its ease of use and performance characteristics
- High performance and scalability: High performance is required to meet the needs of an ever-increasing number of users and devices. The measure of performance is not only about raw transactions per second (TPS), but the platform must be able to handle the burden of complex use cases, scaled across many sessions that require complex rules and data processing
- Maintainability: An essential aspect of operational efficiency is being able to maintain the product with efficient installation, upgrades, updates, version control, redundancy and recovery. Related to ease of use, maintainability is essential for service providers to integrate the product within its current Operations Support Systems and Business Support Systems (OSS/BSS) environment.

Rules engine types

For telecoms software, typically two rules engine types are leveraged: general purpose rules engines or hard-coded rules engines.

General purpose rules engines

A general purpose or business rules engine is based on an open source or a commercial software system that executes one or more business rules in a run-time production environment. These engines have been used in a variety of industries and have not been built specifically for the telecommunications environment, but rather have been adapted to operate in this context. In addition, the engines have their own programming language that generally requires programming expertise, severely impacting ease of use and restricting many non-technical employees from being able to operate them.

Following a linear step-by-step processing flow approach, a general-purpose rules engine first processes each message and evaluates it as a candidate to trigger a policy. If the engine determines that a policy is to be triggered, it invokes the rules that define that policy. At this stage, the engine collects the data needed to evaluate the policy (this information is often not available in memory). The engine then evaluates the policy conditions and initiates the associated actions.

Although a general-purpose rules engine offers flexibility, its lack of a telecommunications-specific call-flow modeling hampers its performance and maintainability under scale, especially in complex use cases. In addition, ease of use is reduced because adding and modifying business rules often requires a consulting contract with the actual policy vendor.

Hard-coded rules engine

A hard-coded rules engine is specific to a few basic use cases. Within the small scope of the use cases it was written for, it generally performs well but suffers severely in terms of flexibility, ease of use and maintainability. A new software load is often needed to support additional use cases outside the scope of its basic functions, resulting in increased operational costs and lead times.

A hard-coded rules engine offers a model that is streamlined to a specific set of use cases or business policies. The engine processes messages individually and determines if they should trigger one of the policies. If one of the policies is to be triggered, specific data is available to help in the real-time processing of the rules leading to evaluation of the conditions and the execution of actions. The available data may be limited by the scope of the specific policy, and the policies themselves may not offer the breadth of control that is required in many environments. This model offers good performance and scale but limited flexibility and maintainability.

Agile Rules Technology

Nokia's Agile Rules Technology (A.R.T.) was originally designed by Alcatel-Lucent to combine the benefits of general-purpose and hard-coded rules engines, without any of their drawbacks. A.R.T. models the call flow and data requirements of a fundamental set of use cases that represent key business models.

Backed by over 150 patents, A.R.T. provides excellent run-time efficiency by anticipating and caching key data requirements of each call flow. The technology then triggers the rules engine only under specific conditions. This specific modeling effectively reduces the complexity of all use cases before they reach the rules engine, resulting in excellent performance that does not deteriorate under scale and complexity.

A.R.T. leverages intimate knowledge of today's broadband environment, coupled with a solid understanding of the key business use cases that service providers need to implement. The A.R.T. model first filters — rather than processes — all incoming messages to find those messages that represent a specific event or rule set that represents a trigger for a specific rule. This important differentiator reduces the load on the rules engine because it processes only those messages that meet the filter requirement.

A.R.T. assigns each rule group one or more triggers. If an incoming message represents a trigger that is assigned to a specific rule group, A.R.T. evaluates this rule group. A rule group is an A.R.T. object that represents a specific business policy and consists of several rules. Each rule has a condition and an action, and an action is executed if the condition is evaluated to be true.

The A.R.T. data modeling capability caches all required data needed to evaluate these conditions in real time. This modular approach provides a powerful rules engine that offers performance and scale across multiple complex use cases while offering excellent ease of use and maintainability.

Comparing rules engine types

Figure 1 depicts the different approaches described above.

Figure 1. Rules engine models

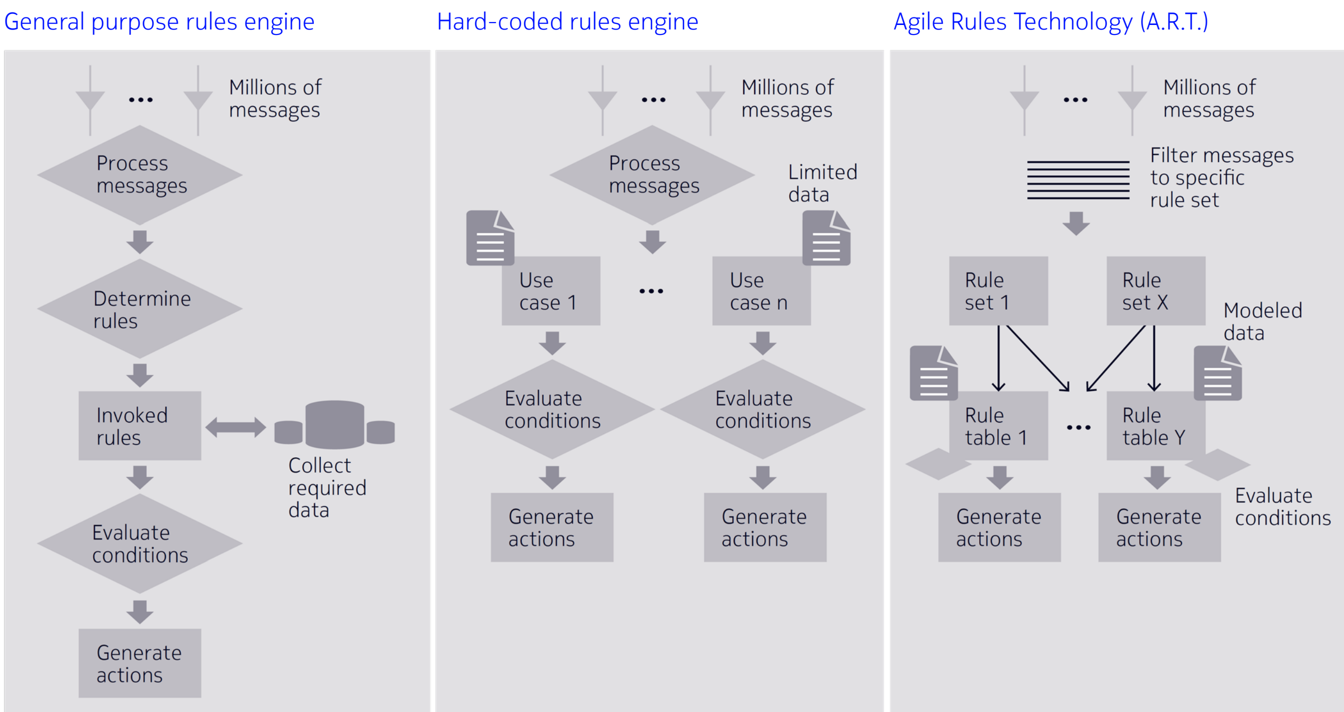


Table 1 compares advantages and disadvantages of the different approaches.

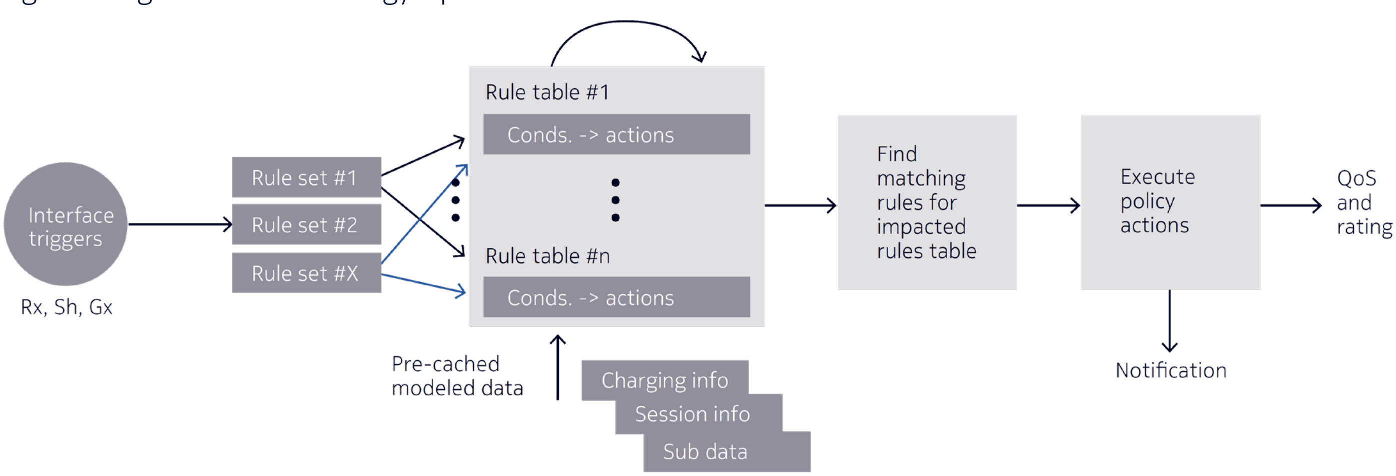
Table 1. Characteristics of A.R.T. compared to general purpose and hard-coded rules engines

Specifications	General purpose	Hard-coded	Agile Rules Technology
Ease of use	Detailed and complex configuration, many with own programming language	Configuration straightforward but rigid	Quick and easy configuration/ modification
Flexibility	Flexibility provides for wide range of use cases	Rigid approach provides for only a few basic use cases	Flexibility of general purpose with performance of hard-coded
Performance	Performance suffers with weak use case modeling	Performance is good for the few supported use cases	Performance consistent across many service provider-specific use cases
Scalability	Inefficient processing leads to limited scale	Scale is good for a limited number of use cases	Efficient, modular processing leads to large scale
Maintainability	Often requires specialized professional services to modify policy rules Rules may need reconfiguration on software upgrade	Often requires a new release to offer specific use cases	Easy, in-service software upgrades with no rule reconfiguration

An in-depth look at A.R.T.

The heart of A.R.T. is based on detailed and specific modeling of the fundamental business use cases that the service provider is primarily interested in. This modeling is done by leveraging the knowledge of the standards-based architecture and the specific data models required to support a wide range of use cases. Over 150 patents have been filed as part of the design, and many of these patented features are mentioned indirectly in the following discussion.

Figure 2. Agile Rules Technology - process flow



Operational simplicity of A.R.T.

As outlined above, the basic building blocks of A.R.T. involve the rule groups, the triggers and the real-time data modeling. This simple yet sophisticated model not only provides for a scalable, high-performing rules engine, but also is conducive to building many operational functions using a simple IF/THEN rules in a modular GUI-driven model.

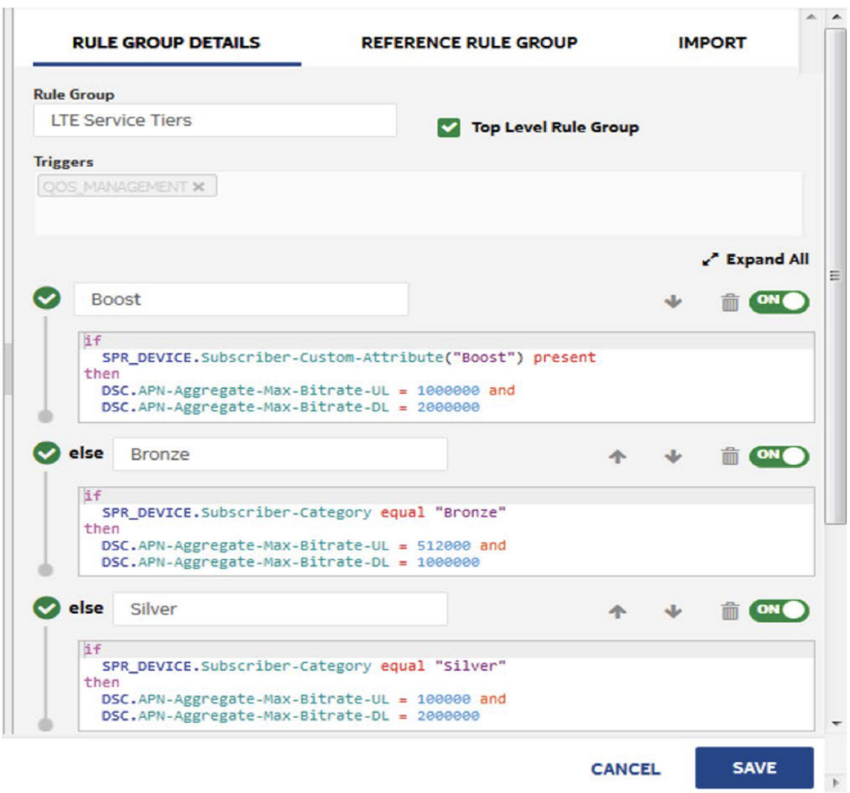
With the aforementioned design, A.R.T. offers an easy-to-follow GUI-driven interface that service providers will use to quickly and easily create new rules and modify existing rules. A set of new rules can be easily created within its own dedicated rule group containing several rules, each having their own condition and action(s) while being assigned specific triggers rule that define when this rule group should be evaluated.

Figure 3 shows how the rules are edited in the GUI as a series of simple IF/THEN blocks which provide a clear summary of the rules logic.

The following example rule group (i.e. policy) defines the settings for four levels of LTE service tiers (speed boost, bronze, silver, and default). Each of these settings is represented by a rule with a condition (IF …) and an action (THEN …).

This example rule group is associated with the QoS Management trigger, meaning that when a QoS Management event takes place, this rule and its associated rules will be evaluated.

Figure 3. Example screen of rule group for setting LTE service tiers



This level of operational ease of use will relinquish the dependency on an external system integrator or software vendor to implement new or modify existing business rules. This saves money, time, and perhaps more importantly empowers innovation.

Another operational benefit that A.R.T. delivers is maintainability. Within the rules engine itself, the rules are contained with a rule system. This allows the service provider to save a snapshot of all rule groups and associated rules at any given point in time. This snapshot can then be exported to another system for testing and troubleshooting or another snapshot can be imported into an existing system for backup and disaster recovery. Creating, saving, importing and exporting a rules system is done in a matter of minutes.

Performance study for complex use cases

To validate the claims of this paper, a comprehensive internal performance study was done that involved modeling several realistic and complex use cases across millions of subscribers with millions of combined IP-CAN and AF sessions. Traffic was generated in the form of session establishment and deletion messages through both the Gx interface (for IP-CAN sessions) and the Rx interface (for AF sessions). Traffic was generated at a rate to reach maximum system capacity measured through TPS, where a single transaction represents a request and a response. The other performance metric that was used is latency, measuring the time between request and response in milliseconds.

The Policy control use cases that were modeled in this study included a range of realistic services that service providers may find compelling to implement while offering enough complexity to aggressively tax the system. Due to the simplicity and elegance of A.R.T.'s design, each use case was modeled completely within its own rule group. Each rule group was assigned a specific trigger that ensured that it would be evaluated often throughout the study.

The performance of the rules engine (i.e., A.R.T.) was measured by a control scenario that used no policies (i.e., rule groups) at all. Specifically, IP-CAN and AF session establishment and deletion messages were generated to max out the system processing capacity while no rules were processed. A specific maximum TPS was recorded and an average latency was measured.

After the results of this control scenario were captured, all of the rules were then turned on. The test was run in such a way that each of these use cases was evaluated continuously. Many use cases were evaluated upon each session establishment message, thus really taxing the processing capability of the system. The results indicated that the maximum TPS reached by the system in the control test decreased by only 10 percent and the message latency only decreased by 12 percent. This is a remarkable finding as it indicates that the system performance holds up extremely well across realistic and complex use cases. The reason for this involves the simplicity of the A.R.T. modeling (triggers, rule groups, data modeling) that minimizes the run-time processing requirements.

A.R.T. "inside"

Policy Control: Nokia Policy Controller

Nokia Policy Controller combines A.R.T. with a state-of-the-art cloud-native architecture to provide the Policy and Charging Rules Function (PCRF) in mobile and fixed networks. It enables service providers to map business demands and network constraints to easy-to-manage network policy rules.

The role of policy management has dramatically changed over the last few years, resulting in ever-increasing complexity of rules, creating a clear need for a better rules engine.

A simple policy control related use case is described in the section titled "Operational simplicity of A.R.T." on page 6, above.

Real-time rating and charging: Nokia Converged Charging

Charging use cases often require integration with other applications. The concept of a rule profile has been implemented in support of that requirement. Rule profiles allow the assignment of rules to other objects in the system as part of a workflow. For example, rather than creating a set of rule groups in a trigger to describe the different bundles for charging, one rule profile is assigned to a charging service based on applicability condition and another one for its tariff. This means that when users are creating bundles, they can focus on the charging service they are building without worrying about the underlying rule groups and triggers. Once the user has defined a bundle, it is scanned and translated into rule groups from the rule profiles. The order in which the rule groups are executed is defined by the priority of the charging service, and the list of the specific set of services for a subscriber.

Another benefit to rule profiles is the ability to make provisioning easier for rules common to a feature. In the case of a "ThresholdProfile", it's possible for a user to assign various thresholds for a bucket or counter. For example, a user can provision a 25% recurring threshold to send a usage notification to the subscriber, and a 50% threshold that reduces the quality of service for the remainder of a 2GB bucket. Rather than creating several rules for sending notifications at the 25%, 50%, 75% and 100% thresholds, and an additional rule for changing the QoS once the 50% threshold is reached, the user can simply specify the percentages and recurrence values in the threshold object itself. From these values, the system generates the underlying rule groups, and assigns the appropriate actions. Not only does the provisioning of thresholds become easier, users can also add additional conditions in a standard rules format. For instance, it's possible to add a condition to only send a 25% threshold notification if the subscriber is not roaming. The recurring threshold is still defined as 25% and the relevant rules is generated; however, in this example, it is combined with the additional condition of "not roaming."

4G and 5G signaling control: Nokia Cloud Signalling Director (CSD)

4G and 5G Signaling Control is another area where the advantages of A.R.T. truly shine. While not required for every signaling routing use case, A.R.T. proves especially beneficial in the context of a Diameter Routing Agent (DRA), a 5G Service Communication Proxy (SCP) or 5G Security Edge Protection Proxy (SEPP) for use cases such as:

- Message Mediation: Mediation is altering the message structure for some specific use cases or for inter-operability. Nokia Cloud Signaling Director (CSD) supports AVP addition, removal, move, as well as modification
- Server off-load: Server off load is to respond on behalf of a server, such as OCS offload. The aim of an OCS offload feature is to filter out messages that can be safely disregarded by the OCS, for example usage reports that contain information about zero usage. Alternatively, in situations where the DRA has detected that the OCS is in an overloaded state, some update messages could be filtered out by simply discarding the message or by responding to the PDN Gateway (PGW) with a "Diameter Busy" message
- Roaming steering: In this use case, CSD routes messages between different service providers for roaming based on a defined set of rules for Diameter (Diameter Edge Agent), or SEPP use cases
- Session binding: In this use case, routing decisions are based on pre-existing sessions, which requires an implementation that keeps track of previous actions
- Load balancing: The CSD is used to load-balance between different instances of the same function, for example PGW or PCRF instances
- Redirect: The CSD is used by the Subscriber Location Function (SLF) in the IP Multimedia Subsystem (IMS) to identify the right Home Subscriber Server (HSS), or in the 5GC to provide subscriber specific selection of AUSF, AMF, etc.
- Topology hiding: For example, the CSD could be used to hide the actual servers in 4G and 5G cores

Conclusion

With the explosion of broadband services and the rise of the IoT, service providers have a tremendous burden: offering new, innovative services with scale and performance and continuing to satisfy their subscribers while generating new streams of revenue. Performance, scale and flexibility are essential to meeting these needs. With over 150 patents filed, A.R.T. leverages the benefits of general purpose and hard-coded rules engines without adopting any of their weaknesses. Nokia products based on A.R.T. offer a range of benefits:

Intuitive rules visualization GUI: Service providers are empowered to quickly and easily create and modify rules in house, without the need to understand detailed programming. This eliminates dependency on vendors or system integrators, reducing costs and time

Excellent flexibility: Offering a wide range of service provider-specific use cases, A.R.T. provides the flexibility benefits of a general-purpose rules engine without the performance issues. Users can configure, test and deploy a new set of rules in minutes

High performance: A.R.T. enables high performance across massive subscriber and session scale for a new breed of complex use cases

Powerful operation and maintainability: Specifically tailored for service providers, A.R.T. reduces time and costs to operate, install, upgrade, troubleshoot and debug the system and associated rules.

To find out more about Nokia's 4G and 5G signaling solutions and A.R.T., please visit our website at https://www.nokia.com/networks/products/cloud-signaling-director/. For more information about Nokia's revenue management solutions please visit https://www.nokia.com/networks/portfolio/monetization/.

Abbreviations

AF	Application Function
A.R.T.	Agile Rules Technology
BSS	Business Support System
CSD	Cloud Signaling Director
DRA	Diameter Routing Agent
DSC	Dynamic Services Controller
HSS	Home Subscriber Server
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IP-CAN	IP connectivity access network
LTE	Long Term Evolution
MME	Mobility Management Entity
OCS	Online Charging System
OSS	Operations Support System
PGW	PDN gateway
PCRF	Policy and Charging Rules Function
QoS	Quality of Service
SLF	Subscriber Location Function
TPS	Transactions per second

Nokia OYJ
Karakaari 7
02610 Espoo
Finland
Tel. +358 (0) 10 44 88 000

Document code: CID195805 (April)