# Formal Modelling and Analysis of the O-RAN O2 Interface in Alloy: Implications for NTN Deployment*

Sean McLaren[1 [0009−0001−9017−3292]], Tsutomu Kobayashi[2 [0000−0002−8795−3183]], Leon Wong[3], and Paul Harvey[1 [0000−0003−1243−938X]]

[1] University of Glasgow, UK
2893505m@student.gla.ac.uk, paul.harvey@glasgow.ac.uk
[2] Japan Aerospace Exploration Agency, Japan
kobayashi.tsutomu@jaxa.jp
[3] Rakuten Mobile, Inc, Japan
leon.wong@rakuten.com

**Abstract.** Open Radio Access Networks (O-RAN) aims to transition telecommunication networks from vendor-specific hardware to open, virtualised control architectures. As interest grows in deploying O-RAN in non-terrestrial networks (NTNs), understanding the robustness of its protocol specifications becomes critical.
This paper reports on the formal modelling of the stable O2 O-RAN interface specification using the Alloy modelling language. We encode ten representative operational scenarios from the O2 specification and formalise safety and feasibility properties relevant to deployment constraints. Bounded model checking reveals several classes of specification weaknesses, including underconstrained pre/post conditions, ambiguous sequencing of protocol steps, and conflicting simultaneous triggers that permit inconsistent system states or violate intended progress conditions.

**Keywords:** Open Radio Access Networks, Alloy, Non-Terrestrial Networks

## 1 Introduction

Correct operation of telecommunication networks is both operationally critical and, in many jurisdictions, a legal requirement [8]. To increase flexibility and reduce costs associated with proprietary hardware and vendor lock-in, operators are transitioning from vertically integrated network appliances to open, software-defined ecosystems in which standardised components interact via well-defined interfaces. The Open Radio Access Network (O-RAN) specification [25] defines a modular architecture of components and interfaces intended to enable interoperable, multi-vendor RAN deployments. Developed by the O-RAN Alliance, a consortium of international operators and vendors, the specification has already underpinned several compliant terrestrial deployments [25].

* This work was supported by EPSRC grant EP/X5257161/1

Building on this terrestrial success, O-RAN has been proposed as a candidate architecture for non-terrestrial network (NTN) deployments. Existing NTN systems are typically customised and proprietary, limiting interoperability and third-party participation [16], both of which have been central to innovation in terrestrial networks. However, NTN environments impose significantly stronger resilience requirements due to limited physical accessibility, high latency, and the cost of intervention. In such contexts, reliable software-based operation is essential. At the same time, recent studies have reported issues in O-RAN implementations [26,31], motivating a rigorous assessment of the specification itself rather than its implementations alone.

In this paper, we formally model aspects of the O-RAN specification in Alloy to assess its robustness for NTN deployment. All Alloy models are available online[4]. Aligned to the proposed regenerative NTN architecture [23], we focus on the O-RAN's O2 interface, a stable and foundational interface responsible for cloud-based infrastructure provisioning, and model ten representative operational scenarios. While prior work has applied formal methods to O-RAN applications (see Section 2), to the best of our knowledge, this is the first study to formally verify the normative behavioural specification of an O-RAN interface itself. Our analysis reveals specification ambiguities, incomplete treatment of error conditions, violated feasibility properties, and potential race conditions, challenging the robustness of O-RAN for deployment in NTN environments.

## 2 Background

### 2.1 O-RAN

Open Radio Access Network [22] is designed to address technical and non-technical challenges in traditional telecommunication systems. At its core, O-RAN is built on two fundamental principles: *vendor-agnostic interoperability* and *open architecture*. The vendor-agnostic nature of O-RAN is achieved through the definition of open interfaces for different tasks, such as RAN control via the E2 or A1 interfaces, or infrastructure provisioning via the O1, O2 interfaces, see Figure 1. These open interfaces enable communication between components from different vendors [22]. The intention is that network operators are no longer limited by proprietary ecosystems, and are able to mix and match hardware and software components to suit their needs [21].

Central to O-RAN's architecture are the RAN Intelligent Controllers (RIC). There are two variants: the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) and the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) [17]. The Near-RT RIC is intended to support control operations with a timescale of ten milliseconds to one second, enabling time-sensitive control through modular applications called *xApps* [22]. The Non-RT RIC focuses on long-term network optimisation and analytics, leveraging *rApps* for policy and decision making [18].

[4] https://github.com/SMcL248/FVTS_Alloy

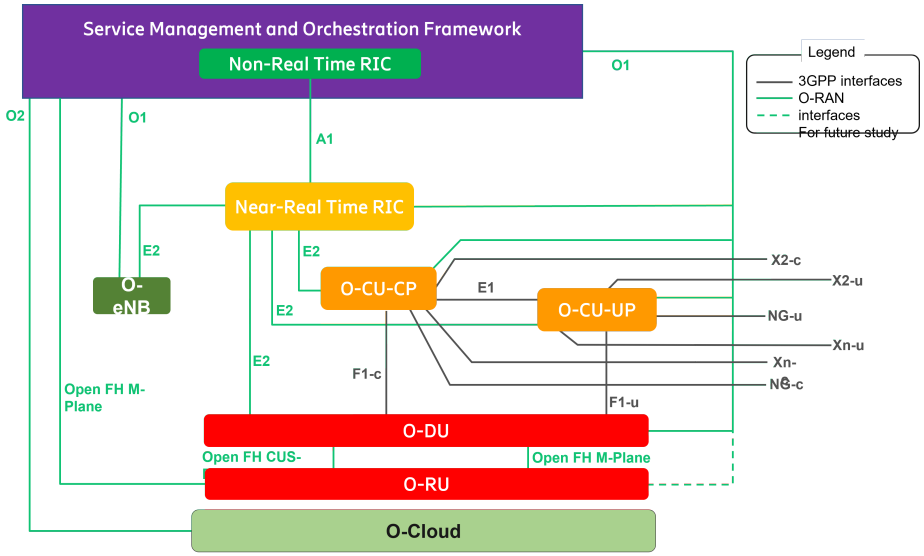


**Fig. 1.** O-RAN Architecture [19]

The Service Management and Orchestration (SMO) provides centralised management, orchestration, and automation functions. The SMO is responsible for lifecycle management, policy control, performance monitoring, and coordination of network functions (NFs) deployed on distributed infrastructure. It interfaces with the O-Cloud, which supplies the virtualised compute, storage, and networking resources required to host O-RAN components. O-Cloud abstracts the underlying physical infrastructure and exposes managed resources through standardised interfaces, including the O2 interface examined in this work.

### 2.2 Formal Modelling and Analysis in Alloy

Alloy [9] is a lightweight formal modelling language based on first-order relational logic, supported by the Alloy Analyzer [29] for automated bounded analysis. Models are expressed declaratively using *signatures* (types), *relations*, and *logical constraints*, enabling concise specification of structural properties and temporal properties. The Analyzer translates models into propositional logic and uses SAT solvers to generate instances of the model and perform bounded model checking. Alloy 6 has enhanced support for modelling and analysis of dynamic behaviour and state transitions within this relational framework.

Alloy has been used to formalise abstractions derived from informal or natural-language specifications, particularly in distributed systems and protocol design [32]. Its bounded model checking approach supports early detection of inconsistencies, underspecification, and unintended interactions, consistent with the "small scope hypothesis," which suggests that many specification errors manifest

in small instances [10]. Overall, Alloy provides a rigorous yet accessible environment for analysing coordination logic and communication protocols, balancing expressiveness with automated verification support. An example of how Alloy was used to model the O2 interface state transition can be seen in Listing 1.2.

### 2.3 Formal Verification In Telecommunications

Formal analysis of communication protocols has a long history, including model checking and state exploration techniques [2,12]. Model checking has been successfully applied to detect deadlocks, race conditions, and feasibility violations in distributed and networked systems [7,11].

In telecommunications, formal methods have been applied to analyse signalling protocols and distributed coordination logic derived from natural-language standards [13,4]. Such efforts often uncover ambiguities and conflicting behaviours not evident during manual review. Our work aligns with this tradition by treating the O-RAN O2 specification as a behavioural artefact subject to formal interpretation and validation. Recent works have also begun to explore how AI tools (e.g. large language models) can support use of formal methods by non-experts in telecommunication networks [30].

### 2.4 Formal Modelling in O-RAN

Several recent works have explored applying formal methods to O-RAN, however, they have focused on the third-party applications (xApp, rApp) that *use* interfaces, as opposed to the normative behaviour of the O-RAN interfaces themselves. Exemplar O-RAN applications include PRISM-based analysis of resource allocation [15], threat analysis [28], or energy-efficient service availability [14].

To the best of our knowledge, our work is the first to formally model and verify the normative behavioural semantics of an O-RAN interface, see Section 4.

## 3 O-RAN Operational Scenarios to Model

In the O-RAN specification, the seven key interfaces (A1, O1, O2, E2, Open Fronthaul, Xn, and F1), describe over 300 potential operational scenarios to model across all interfaces. Given space limitations, we focus on the O2 interface. We now describe the O2 interface and the modelled operational scenarios.

### 3.1 The O-RAN O2 Interface

The O2 interface connects the Service Management and Orchestration (SMO) framework to the underlying cloud infrastructure and is responsible for infrastructure resource management, virtual network function lifecycle management, and orchestration of O-Cloud resources. Considering NTN environments,

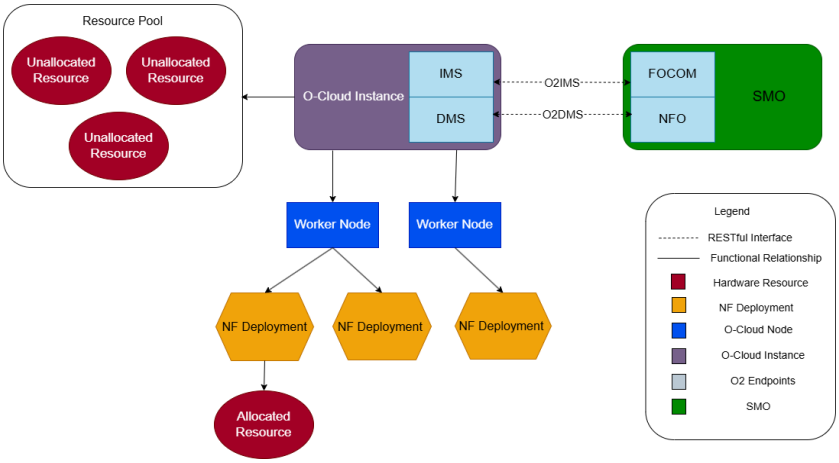


**Fig. 2.** O-Cloud overview showing the Infrastructure Management Services (IMS) and Deployment Management Services (DMS) which communicate with the SMO via the O2 interface. O-Cloud uses Worker Nodes to host Network Function (NF) Deployments. Resources can be scaled in and out of resource pool to increase functional capacity of NF Deployments.

compute platforms may be satellite-borne, intermittently connected, resource-constrained, or geographically distributed. This makes the correctness of infrastructure orchestration mission-critical: Failures in lifecycle operations, state synchronisation, or resource allocation can have amplified consequences due to long propagation delays, limited physical access, and constrained recovery options. By contrast, interfaces such as E2 or A1 primarily govern RAN control and optimisation logic, which, while important, presuppose a stable and correctly managed infrastructure substrate. Verifying O2 procedures therefore targets the foundational control plane governing deployment, scaling, healing, and configuration consistency across heterogeneous and latency-sensitive environments.

The O2 interface was one of the first O-RAN interfaces to be documented, giving it stability, and has several mappings to other standardised telecommunication architectures and frameworks (ETSI NFV MANO [5], ETSI ZSM [6], and 3GPP [1]), and open source projects (ONAP [24], Kubernetes [3]).

### 3.2 Operational Scenarios of the O2 Interface

The O-RAN specification describes 55 operational scenarios of the O2 interface. Given space limitations, we focus on those most commonly found or used. The following summarises the selected operational scenarios and their associated clause in the O-RAN specification [20], with Figure 2 giving an overview of the components, roles, and relations:

**Hardware Infrastructure Scaling Post-Deployment (clause 3.1.5)** describes the addition of a new worker compute node to a previously installed

O-Cloud and related to clause 3.1.5 [20]. It involves a worker Node interacting with the IMS of a given O-Cloud to request a *software image*, which allows the Node to *boot*. Once booted, the O-Cloud Node has been effectively added to the O-Cloud. Subsequentially, the SMO should trigger an inventory update to retrieve an up-to-date list of available resources.

**Network Function Lifecycle (clause 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.5)** The lifecycles of Network Functions (NFs) are defined through 5 operational scenarios. NF Deployments can be instantiated (clause 3.2.1) or terminated in the O-Cloud (clause 3.2.5). The O-Cloud can alter the processing power of NF via scaling. NF Deployments can have further resources allocated to it via scale out (clause 3.2.2), and have resources deallocated through scale in (clause 3.2.3). A software modification (clause 3.2.4) procedure is also detailed. This operation follows a *replace-and-build* approach. This consists of a NF with the new software version being instantiated, and depending on the status of this NF, the new or existing NF is terminated. Each of these use cases are initiated by request via the O2 interface, and generate SMO responses.

**Network Slices (clause 3.10.1, 3.10.2).** Network slices may be created (clause 3.10.1) and destroyed (clause 3.10.2) over multiple O-Cloud instances. Such slices consist of NF Deployments which may or may not need to be instantiated as part of the slice creation. For each O-Cloud in the slice, a unique ID (VLAN-ID) is used to mark appropriate NFs. Such VLAN-IDs are requested from the given O-Cloud, and must be locally unique. The deletion of a network slice will include the termiantion of specified NFs and the deallocation of VLAN-IDs in each O-Cloud included in the network slice.

**O-Cloud Node Clusters (clause 3.11.2.3, 3.11.3.2).** The creation and deletion of O-Cloud Node Clusters are described in the specification. Clusters are groupings of O-Cloud resources that are used to deploy workloads. To create a Cluster, a request is sent the O-Cloud. Upon receiving a request, an O-Cloud Node Cluster ID shall be generated and used to book required resources. When all resources are successfully booked, they are allocated and the Cluster is created. This then generates an SMO response. Upon failure, it is stated that all actions should be rolled back. In deletion, all booking IDs are removed and resources are deallocated. Similarly to creation, an SMO response is generated and the system is told to roll-back upon error.

## 4 Formalising the O2 Interface in Alloy

We now describe the process of formally modelling the described O2 operational scenarios in Alloy, as well as verification of safety and feasibility.

### 4.1 Modelling the O2 Interface in Alloy

In the O-RAN specification, each operational scenario has a defined goal, participating actors, preconditions, starting conditions, prescribed steps and actions, end conditions, exception flows, and post-conditions [20].

**Listing 1.1.** Model structure of NF Deployment Instantiation

```
sig Node {var deployment: set NF}
sig NF {}
sig Request {
    target: one oCloud,
    payload: one NF,
    var underlying_error: lone Response,
    var processed: lone Response
}
sig oCloud { dms: one DMS, var nodes: set Node}
sig DMS {cloud: one oCloud, var requests: set Request}
one sig SMO {var responses: set Request->Response}
enum Response{SUCCESS, ERROR}
```

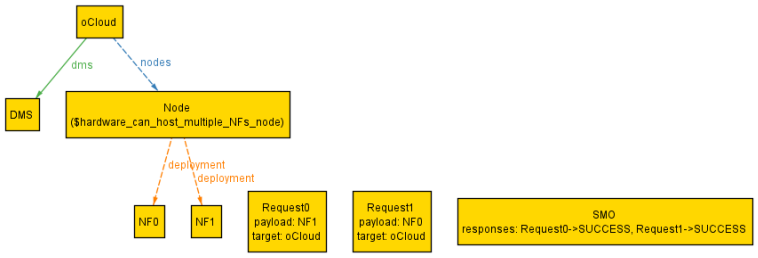


**Fig. 3.** Alloy Visualisation of Listing 1.1

We represent O2 components and interactions in Alloy using signatures and predicates, respectively. Each Alloy signature contains a list of fields that allow for mappings between O2 components. Listing 1.1 shows the structure of the NF Deployment Instantiation protocol, with Figure 3 showing this visually. From the specification, the SMO sends requests to the DMS. O-Cloud Worker Nodes are represented by the `Node` signature, and network functions are represented by the `NF` signature. An O-Cloud instance is represented by the `oCloud` signature, and must be associated with a `DMS` signature. We represent request objects using the `Request` signature. Requests contain an NF for instantiation, a target O-Cloud, and a status which is represented by the `Response` enumerate type. The `lone` keyword enforces that a field contains at most one mapping, while the `one` keyword enforces that a field contains exactly one mapping. The `->` operator defines the relations between signatures in fields.

O2 sequence steps are represented in Alloy predicates, as shown in Listing 1.2. Such predicates define how the system may transition from one state to another. They are broken down into guard, effect, and frame conditions. Effect conditions specify which mutable fields change, while frame conditions ensure that mutable fields that should not change remain unaltered. Guard conditions must be satisfied for the predicate to take effect, and most closely represent O-RAN

**Listing 1.2.** Predicate that sends a NF Deployment Instiation request to a DMS

```
pred initiate_deployment(r:Request){
  historically r not in r.target.dms.requests and
            some r.targets.nodes //guard conditions
  requests'= requests + r.target.dms->r //effect condition
  //frame conditions
  deployment' = deployment and nodes' = nodes and responses'
      = responses and err' = err and processed' = processed
}
```

**Listing 1.3.** Predicate that simulates the NF instatiation process

```
pred sim_error(r:Request, c:oCloud){
  r in c.dms.requests and no r.error//guard
  error' = error + r->SUCCESS or error' = error + r->ERROR
  // code to update frame conditions, as in Listing 1.2
}
```

preconditions. Listing 1.2 shows a predicate representing a request being sent to a DMS. Here, the guard condition is that there must exist a worker Node that the NF can be instantiated on.

Listing 1.3 shows the transition associated with instantiating an NF Deployment. For this step, the specification only states that resources are allocated. We abstract the allocation, returning the success or failure of the instantiation. We include an error possibility is to show that NF instantiation may not be successful, as in real life.

Fact statements were written to force the solver down specific traces to explore possible real world states. That is, these statements set conditions that cannot be violated. Facts can take the form of initial conditions, as shown in Listing 1.4, or fairness conditions. Fairness conditions force system progression, preventing solver-generated counterexamples where the system *stutters* indefinitely.

### 4.2 Analysing Feasibility and Safety

Our analysis of the O2 interface consists of verification of feasibility (there exist cases where something good happens) and safety (something bad never happens in every case). To check feasibility, we instructed the Alloy analyser to find traces (instances) in which the target property holds. To verify safety, we instructed the Alloy analyser to perform bounded model checking. Due to Alloy's bounded analysis, results are necessarily scope-dependent. Increasing the number of atoms per signature or extending the trace length increases confidence in the conclusions.

**Listing 1.4.** Initial conditions in NF Deployment Instantiation

```
fact init {
    all c : oCloud | some c.nodes
    all node : Node | no node.deployment
    all r : Request |
        no r.underlying_error and
        no r.processed and
        r.payload not in Node.deployment
    no SMO.responses
    all d : DMS | no d.requests
}
```

**Listing 1.5.** Safety assertion in NF Deployment Instantiation

```
assert response_implies_added_node {
  always (all r : Request | (r->SUCCESS in SMO.responses)
      implies (once r.payload not in Node.deployment) and r.
      payload in Node.deployment)
}
```

Safety assertions were used to detect the existence of undesirable behaviour. These capture invariants that must always hold across all instances, such as consistency between SMO and O-Cloud state representations, absence of conflicting lifecycle transitions, and preservation of resource allocation constraints. For example, in the NF Deployment Instantiation protocol (Listing 1.5), we specify that a success response in the SMO's inbox implies that the corresponding NF was previously not instantiated and is now instantiated. This ensures that responses are only generated following successful completion of the associated deployment procedure. The solver then searches for traces that violate this invariant; absence of a counterexample indicates that the invariant holds under the given scope.

Feasibility assertions were written to establish the existence of valid behaviour and intended progress conditions. These properties express that certain desirable outcomes are achievable within the model. Listing 1.6 presents a property that should be feasible for the NF Deployment Instantiation protocol. For

**Listing 1.6.** Property that should be feasible in NF Deployment Instantiation

```
fact hardware_can_host_multiple_NFs_fact {
    eventually (some node : Node | (some disj a,b : NF | (a+b
        ) in node.deployment))
}
```

example, it is expected that an O-Cloud Worker Node may be able to host more than one NF instance. To evaluate feasibility, a fact statement that states that the expected behaviour eventually occurs is created. If the enclosed behaviour is valid under the model, we expect a valid trace to be generated that demonstrates the expected behaviour. If the solver produces no such trace, the intended behaviour is therefore not achievable within the given scope.

Across the analysed scenarios, safety properties ensured that O2 components do not enter deployment without completion of the corresponding protocol, that responses are only generated upon legitimate completion, and that error statuses are consistent with request outcomes. Feasibility properties examined whether worker nodes can host multiple NF instantiations were permitted, whether a single NF can be instantiated across multiple nodes, whether failure messages are propagated back to the SMO, and whether concurrent requests targeting the same O2 component may both succeed. Together, these properties reflect the dual requirement in NTN contexts to prevent unsafe configurations while ensuring forward progress despite delay, concurrency, or disruption.

## 5 Results

We now discuss the results of the analysis described in Section 4.2. We also report problems we found through modelling the O2 interface.

### 5.1 Feasibility

**Hardware Scaling Post-Deployment (clause 3.1.5).** We attempted to confirm that multiple O-Cloud Nodes may be deployed in a single O-Cloud instance, and single O-Cloud Nodes may be shared between O-Cloud instances. For each of these conditions, the analyser successfully generated traces that satisfy it. In this case, there is ambiguity in the O-RAN specification. For example, despite being against best practice [27], it is unclear whether shared Nodes between O-Cloud instances is expected behaviour. Also, the hardware scaling preconditions do not state that the targeted O-Cloud Node should not be previously deployed.

Another feasibility we checked is whether O-Cloud Node boot failure stalls progression. For example, we stated an assertion that boot failure implies that the request remains in the IMS inbox. Checking this assertion did not produce a counterexample, implying that boot failure is not recoverable under the model; the expected behaviour (a configuration attempt associated with a failed boot is completed) never occurs. To further highlight this invalid behaviour, a fairness condition was introduced that forces all hardware scaling operations to complete. Upon asserting that boot failures cannot exist under this constraint, no counterexample was found. That is, for hardware scaling to complete, the Node must always successfully boot.

**NF Deployment Instantiation (clause 3.2.1).** It is commonly accepted that worker Nodes possess the ability to host multiple NF instantiations, or that a single NF Deployment may be instantiated among multiple nodes [5]. We check

this ability in Alloy. This includes that a request failure does not prevent the corresponding NF from later instantiation, or that two requests of the same NF may both yield successful outcomes. The analyser generated traces for each of these behaviours, showing they are permitted in the model, as expected.

Conversely, graceful error handling on instantiation was not observed. The analyser failed to generate a trace in which the SMO receives an error response during instantiation. Hence, this expected behaviour can never occur in the model.

**NF Deployment Scaling (clause 3.2.2, 3.2.3).** Another feasibility assertion explored is that deallocated resources can be reused. We successfully obtained a trace showing that an allocated resource is removed due to a scale in request and subsequently reallocated during a separate scale out operation. Furthermore, we checked that NF Deployments have the ability to make use of multiple resources. This returned a trace where two successive scale out requests allocate resources to the same NF Deployment.

An ambiguity in the specification concerns whether a single resource can be allocated to multiple NFs. The specification states only that resources are allocated as required, and does not mention whether currently allocated hardware resources can also be used for scaling. Best practice states that isolation is preferable to hardware sharing [27], however, in an NTN context, where resources are limited [23], shared may be required. To understand resource allocation during scaling behaviour in the specification, a resource pool was implemented in the NF Deployment scaling model to represent unused resources that are available for deployment. If a scale out request is received, then a resource from the pool is allocated to the target NF. We attempted to show that resource sharing between NF Deployments is feasible. However, no trace is returned, meaning that resource sharing is not possible assuming best practise, which can only scale out using *unallocated* resources from the pool.

**NF Deployment Software Modification (clause 3.2.4).** For software modification, the existence of failure and success responses is checked as feasibility properties. We successfully produced traces in which both response types are observed in the SMO. Hence, complete success and failure pathways exist.

It is also checked that both the desired and existing NF may be deployed on different hardware of the same O-Cloud instance. This produced a valid trace in which this property is realised. Hence, the new NF need not be deployed upon the same hardware as the NF for replacing.

**NF Deployment Termination (clause 3.2.5).** Two feasibility conditions were tested to validate the suitability of the termination model. For each of two SMO response types, the analyser returned traces in which the SMO response is generated. This implies that both termination success and failure pathways are reachable under the model.

**Network Slices (clause 3.10.1, 3.10.2).** For Cluster creation (clause 3.10.1), it is stated in the specification that network slices may span multiple O-Cloud instances. Hence, this property was tested using a feasibility assertion.

As expected, a trace is produced showing that a slice may be associated with multiple O-Clouds.

Furthermore, we checked that the failure to create a network slice does not necessarily imply that all associated NFs are not instantiated. The analyser successfully returned traces, showing that previously deployed NFs remain unaffected by creation failures.

For network slice deletion (clause 3.10.2), a returned trace illustrated the existence if complete success and error pathways, showing appropriate SMO responses.

**O-Cloud Node Clusters (clause 3.11.2.3, 3.11.3.2).** For O-Cloud Node Cluster creation (clause 3.11.2.3) and deletion (clause 3.11.3.2) operations, we successfully generated traces showing that both error and success responses can be sent to the SMO in these use cases. All requests, no matter success or failure, can result in an SMO response. Hence, errors are handled gracefully in the model, as expected.

For the creation use case specifically, it was checked that multiple O-Cloud Node Clusters may be created on a single O-Cloud instance. This produced a trace implying that such a property occurs as expected. The possibility of multiple clusters existing in a O-Cloud instance is an important property for checking that cluster IDs are locally unique within O-Clouds, and that clusters have disjoint resources as discussed later in Section 5.2.

### 5.2 Safety

**Hardware Scaling Post-Deployment (clause 3.1.5).** As per the specification, explored properties included that O-Cloud Nodes only boot due to an on-going scaling request, deployed Nodes must have successfully booted, and that a successful boot implies eventual Node deployment using Alloy `facts` to mitigate stutter. Each assertion did not produce counterexamples, and thus, O-Cloud Node configuration does not occur without a corresponding scale request.

**NF Deployment Instantiation (clause 3.2.1).** In the NF Deployment instantiation model, various safety assertions were examined. For example, if a request is completed, then an SMO response must be generated. Conversely, if an SMO response is received, the corresponding NF must have been successfully instantiated. Neither of these assertions yield counterexamples. This implies that NFs cannot be instantiated without it being directly requested, and that responses cannot be sent to the SMO without the completion of the corresponding request. An assertion stating that NFs cannot be instantiated on undeployed Nodes was also written. This provided no counter examples, and thus, NFs can only be instantiated upon deployed Nodes, as expected.

Upon altering the model so that the allocation of resources is included, a safety condition was created to ensure that resources are only allocated to NFs that are successfully instantiated. This assertion unexpectedly revealed a counterexample in which an NF instantiation request has failed, but a required resource has been allocated. This is depicted in Figure 4. As there is no roll-back mechanism in the spec, this demonstrates a resource leak upon error.

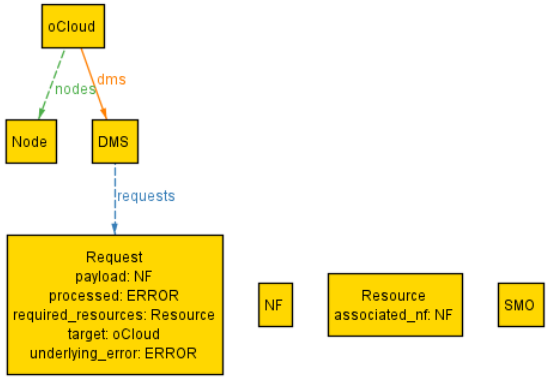


**Fig. 4.** Unhandled Error Condition: resources were allocated but the overall NF instation request failed (clause 3.2.1).

**Network Function Scaling (clause 3.2.2, 3.2.3).** For NF deployment scaling, we specified assertions requiring that resource allocation occurs only if a successful scale-out request targeting the same NF deployment previously appeared in the trace; no counterexamples were found. The dual property was verified for deallocation, ensuring resources are released only following a successful scale-in request, again yielding no counterexamples. Together, these results show that resources cannot move between pools and NF deployments without an explicit request. We further verified that any request leaving a DMS returns a final status to the SMO, confirming that requests cannot be silently discarded before completion. Finally, by constraining traces to include failed scaling operations with corresponding SMO responses, we identified a counterexample in which allocation fails but an error response is eventually returned. This demonstrates that failures do not block overall system progression.

**NF Deployment Software Modification (clause 3.2.4).** Safety assertions were checked on the software modification operation. This included that all SMO responses correspond to a preceding operation, that the completion of requests imply SMO responses, that the termination of the new NF implies an unsuccessful operation, the termination of the existing NF implies a successful operation, that all terminated NFs have there resources deallocated, and that if both the desired and existing NF are concurrently instantiated, then they must both exist within the same O-Cloud. Each of these assertions produced no counterexamples, implying that SMO responses can only be generated via request completion, there are no partially allocated resources and that the correct NF is always terminated during software modifications.

**NF Deployment Termination (clause 3.2.5).** Several safety conditions were written to confirm the absence of unwanted states due to NF Deployment termination. This includes that success responses imply only occur if the appropriate NF has been terminated as well as that error responses imply that the corresponding NF remains instantiated. It was also tested that the termination

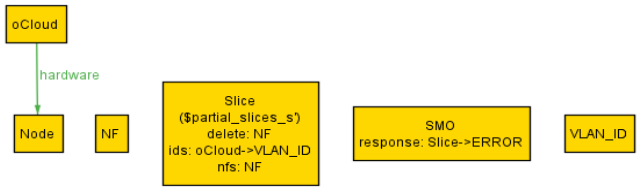


**Fig. 5.** Output of safety assertion violation for network slice deletion (clause 3.10.1). As a counterexample is generated, it has been shown that slices may exist but not have all required NFs instantiated. This is unexpected behaviour.

of an NF implies the deallocation of all appropriate resources. Each of these assertions did not produce counterexamples. This shows that certain unwanted behaviours are forbidden under the model. This includes that responses are only generated upon operation success or failure, and that resources do not remain partially allocated.

**Network Slices (clause 3.10.1, 3.10.2).** For slice creation (clause 3.10.1), it was checked that success responses imply that all required NF have been instantiated. This revealed no counterexample, as expected. Also, it was verified that upon creation failure the slice has no associated VLAN IDs. No counterexamples were produced, meaning VLAN ID clean up is always successful, as expected.

A safety assertion was written to verify that partial slices may only exist during slice deletion (clause 3.10.2). That is, if a network slice exists but an NF that should be included in the slice has since been terminated, then there must be an on-going deletion attempt. This assertion unexpectedly produces a counterexample which is visualised in Figure 5. The `ids` relation in the `Slice` atom contains a mapping to an O-Cloud and VLAN-ID, implying that this slice is currently deployed. The preceding trace that generated this counterexample attempted to delete this slice, but an error prevents this deletion as implied by the response in the `SMO` atom. However, before the error occurred, the deletion of the slice had already started, and the `NF` atom was terminated. Hence, there now exists a network slice which is no longer slated for deletion, but an NF which should be associated with the slice is no longer deployed.

**O-Cloud Node Clusters (clause 3.11.2.3, 3.11.3.2).** For O-Cloud Node Cluster creation (clause 3.11.2.3), various assertions were used to test safety properties. This includes that no two Clusters may share the same resource, all successfully created clusters contain at least one resource, no two Clusters in a single O-Cloud instance may share the same cluster ID, and that resources that reside in the same O-Cloud but different Clusters must be booked with distinct IDs. Each of these assertions produced no counterexamples. Thus, it has been shown that the inverse of these properties, which are invalid behaviours, never occur. For O-Cloud Node Cluster Deletion (clause 3.11.3.2), similar safety assertions were checked. This includes that all deletion requests deemed successful

imply that the corresponding cluster no longer exists as well as the fact that all Clusters remain non-empty and disjoint when in deployment.

### 5.3 Race Conditions

The O-RAN documentation does not prescribe a normative protocol for concurrent access. Instead, it offers illustrative examples of implementation approaches that could be used to achieve safe concurrent operation (e.g. semaphore flags, transaction tokens). This leaves specification conformance open to interpretation. An initial exploration of race conditions in Alloy found an example in hardware scaling post deployment (clause 3.1.5), however, given the absence of normative constraints, we cannot classify this as specification violation.

### 5.4 Lack of Error Conditions

Throughout the model process, it was observed that multiple sequences (clauses 3.1.5, 3.2.1) have insufficient error descriptions. As discussed in Section 5.1, there are no valid traces in which an error message is propagated back to the SMO. This is due to the complete lack of error specification in these sequences. As no exceptions are given, it is not well-defined how the system handles error. Hence, the system idles and instantiation requests remain unfinished indefinitely. It is also unclear how the O-Cloud should *clean-up* resources allocated during an NF instantiation (clause 3.2.1) that has failed to complete. As shown in Section 5.1, this lack of roll-back procedure causes resources to remain allocated to NFs that are not currently deployed. This is not a desirable state. In Section 5.2, it is shown that there exists traces in which network slices exist, but not all required NFs are instantiated. This again is due to the lack of roll-back procedures. While the specification states that error may occur during network slice deletion (clause 3.10.1), it is not stated that the slice must be restored to its initial configuration. This allows for the unwanted state in which a network slice exists, but not all of its required NFs are deployed. As undesirable states are reachable under the error conditions set out by the specification, it must be the case that such error conditions are not satisfactory to maintain valid behaviour.

### 5.5 Summary

Through analysis of the O-RAN O2 interface specification description and Alloy modelling, we have identified several issues with the robustness and correctness of the O2 interface. A combination of feasibility assertion violations (Section 5.1) and safety assertion violations (Section 5.2), show that the specification contains ambiguities in error specification. As a consequence of these ambiguities, simulated errors cause models to enter unwanted states, as discussed in Section 5.4. It was directly observed that errors in clauses 3.1.5 and 3.2.1 of the O2 specification cause requests to indefinitely idle and prevent responses from being generated. In addition, the lack of roll-back/clean up procedures in clauses 3.2.1 and 3.10.2 allow for resources to be left in partially allocated states.

These findings suggest that the specification would benefit from clearer normative concurrency semantics, explicit pre/post conditions, and a canonical lifecycle state model to reduce ambiguity and unintended non-determinism, especially in NTN deployment contexts.

## 6 Conclusions

To understand the robustness of the open radio access network (O-RAN) specification for use in non-terrestrial network environments, we use Alloy to formally model the O2 interface. While others have explored formal verification of O-RAN, this work has focused on the control applications that use the specification, as opposed to modelling the normative specification itself. We choose Alloy as it enables incremental specification for the hundreds of O-RAN use cases. Through Alloy modelling of ten use cases, we observed that multiple use cases did not contain sufficient error specification. The lack of any error conditions causes models to indefinitely stutter, never completing requests that have been deemed unsuccessful. Furthermore, the lack of clean-up procedures allow for partially allocated resources. These issues challenge the correctness and robustness of the O2 interface specification for use in NTN deployments.

## References

1. 3GPP: Management and orchestration; Architecture framework. Technical Specification V15.3.0, ETSI (2020), `https://bit.ly/3MNGOXJ`
2. Clarke, E.M.: Model checking. In: International conference on foundations of software technology and theoretical computer science. pp. 54–56. Springer (1997)
3. Cloud Native Computing Foundation: Kubernetes – production-grade container orchestration. `https://www.kubernetes.io/` (2026), last accessed 15 Feburary 2026
4. Cremers, C., Dehnel-Wild, M.: Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion. In: Network and Distributed System Security Symposium (NDSS). Internet Society (2019)
5. ETSI: Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Architectural framework. Group Specification GS NFV 006, European Telecommunications Standards Institute (ETSI) (Dec 2022), `https://bit.ly/4tXpXmf`
6. ETSI ISG ZSM: Zero-touch network and Service Management (ZSM); Reference Architecture. Group Specification GS ZSM 002, European Telecommunications Standards Institute (August 2019), `https://bit.ly/4kKKGpa`
7. Gotmanov, A., Chatterjee, S., Kishinevsky, M.: Verifying deadlock-freedom of communication fabrics. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. pp. 214–231. Springer (2011)
8. ITU-T: Network reliability in public telecommunication data networks. ITU-T Recommendation Y.2614 (08 2011), `https://www.eolss.net/sample-chapters/C05/E6-108-13-00.pdf`
9. Jackson, D.: Alloy: a lightweight object modelling notation. ACM Transactions on software engineering and methodology (TOSEM) **11**(2), 256–290 (2002)
10. Jackson, D.: Software Abstractions: logic, language, and analysis. MIT press (2012)

11. Kaveh, N.: Using model checking to detect deadlocks in distributed object systems. In: Engineering Distributed Objects: Second International Workshop, EDO 2000 Davis, CA, USA, November 2–3, 2000 Revised Papers. pp. 116–128. Springer (2001)
12. Leathrum, J., Morsi, R., Leathrum, T.: Formal Verification of Communication Protocols. In: IASTED Eighth International Conference on Parallel and Distributed Computing and Systems. Citeseer (1996)
13. Li, Z., Xiao, M., Xu, R.: Formal analysis of signal protocol based on logic of events theory. Scientific Reports **14**(1), 20606 (2024)
14. Metere, R., Ye, K., Gu, Y., Zhang, Z., Alrajeh, D., Sevegnani, M., Yadav, P.: Towards Achieving Energy Efficiency and Service Availability in 6G O-RAN via Formal Verification. In: Czekster, R.M., Milazzo, P. (eds.) From Data to Models and Back. pp. 137–157. Springer Nature Switzerland, Cham (2025)
15. Mumtaz, T., Muhammad, S., Bouali, F.: Formal Verification- and AI/ML-Assisted Radio Resource Allocation for Open RAN Compliant 5G/6G Networks. IEEE Access **13**, 96198–96212 (2025). https://doi.org/10.1109/ACCESS.2025.3575021
16. Nguyen, C.T., Saputra, Y.M., Huynh, N.V., Nguyen, T.N., Hoang, D.T., Nguyen, D.N., Pham, V.Q., Voznak, M., Chatzinotas, S., Tran, D.H.: Emerging Technologies for 6G Non-Terrestrial-Networks: From Academia to Industrial Applications. IEEE Open Journal of the Communications Society **5**, 3852–3885 (2024). https://doi.org/10.1109/OJCOMS.2024.3418574
17. Niknam, S., Roy, A., Dhillon, H.S., et al.: Intelligent O-RAN for Beyond 5G and 6G Wireless Networks. arXiv preprint arXiv:2005.08374 (2020), https://arxiv.org/abs/2005.08374
18. O-RAN Alliance: O-RAN Minimum Viable Plan and the Acceleration towards Commercialization. https://bit.ly/4rWbIw3, accessed: 2025-01-08
19. O-RAN Alliance: O-RAN Software Community (SC) Documentation, https://docs.o-ran-sc.org/en/j-release/index.html, last accessed 15 Feburary 2026
20. O-RAN Alliance: O-RAN Use Cases, https://specifications.o-ran.org/specifications, last accessed 22 March 2026
21. O-RAN Alliance: O-RAN Use Cases and Deployment Scenarios. https://bit.ly/4qFZ2YY, accessed: 2025-01-08
22. O-RAN Alliance: O-RAN White Papers and Resources. https://www.o-ran.org/o-ran-resources, accessed: 2025-01-08
23. O-RAN Alliance White Paper Contributors: Deployments of o-ran-based non-terrestrial networks. White Paper O-RAN.WP.IEFG.O-RAN_NTN, O-RAN ALLIANCE (May 2025), https://bit.ly/4aVybn5, accessed 2025-02-05
24. ONAP Project: Open Network Automation Platform, https://www.onap.org/
25. Polese, M., et al.: Open RAN: A Concise Overview. IEEE Communications Magazine (2024). https://doi.org/10.1109/MCOM.005.2300467, https://ieeexplore.ieee.org/document/10601697/
26. Rodgers, P., Harvey, P.: The xApp Store: A Framework for xApp Onboarding and Deployment in O-RAN. In: 2025 IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW). pp. 647–652 (2025). https://doi.org/10.1109/ICDCSW63273.2025.00121
27. StarlingX Project: O-RAN Specification Compliant O2 Interfaces. https://bit.ly/4ar6ssY (2022), starlingX stx-8.0 Approved Specification, Story 2010278
28. Thimmaraju, K., Shaik, A., Flück, S., Mora, P.J.F., Werling, C., Seifert, J.P.: Security testing the O-RAN near-real time RIC & A1 interface. In: Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks. pp. 277–287 (2024)

29. Wang, W., Wang, K., Zhang, M., Khurshid, S.: Learning to Optimize the Alloy Analyzer. In: 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). pp. 228–239 (2019). https://doi.org/10.1109/ICST.2019.00031
30. Wray, T., Wang, Y.: 5G Specifications Formal Verification with Over-the-Air Validation: Prompting is All You Need. In: MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM). pp. 412–418 (2024). https://doi.org/10.1109/MILCOM61039.2024.10773849
31. Yang, T., Rashid, S.M.M., Ranjbar, A., Tan, G., Hussain, S.R.: ORANalyst: Systematic testing framework for open RAN implementations. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 1921–1938. USENIX Association, Philadelphia, PA (Aug 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/yang-tianchang
32. Zave, P.: Using lightweight modeling to understand chord. SIGCOMM Comput. Commun. Rev. **42**(2), 49–57 (Mar 2012). https://doi.org/10.1145/2185376.2185383, https://doi.org/10.1145/2185376.2185383