Agile Modeling Practises used for the Views:

1. Model with others: This modeling activity was a group activity with each group member actively contributing to the final increment of the model.
2. Iterate to Artifact: Every team member was assigned a separate use case of NginX to model over a UML diagram with different use cases serving as different artifacts to the three views.
3. Model in Small Increments: Every team member’s use case was incrementally integrated into the use case.

Agile Modeling Principles used for the Views:

1. Multiple Models: Currently showcasing 3 models and all of which have been developed post iterative feedback from the team members.
2. Rapid Feedback- Indulged in multiple peer reviews for delivery of models and content thereof.
3. Assume Simplicity- These models envision the most basic use cases while also envisioning over 90% of the use cases in which NginX is used hence these assume simplicity.
4. Incremental change- As already mentioned, based on team member feedback, these models were changed incrementally.

Structural View: [Referred: https://docs.nginx.com/nginx-service-mesh/about/architecture/]

A picture containing text, parallel, diagram, number

Description automatically generated

(Below highlighted is GPT content and can be shortened)

NginX provides fast, reliable, and low-latency network connections for modern application architectures. The architecture consists of a control plane layer and a data plane layer.

The control plane comprises multiple subsystems and follows the sidecar pattern. It uses controllers to enforce desired states across managed applications. These controllers watch Kubernetes resources and custom resources defined by the Service Mesh Interface specification. NGINX Service Mesh controllers support advanced configurations, including Kubernetes Services and Pods.

Dynamic Admission Control is applied through a process called injection, where NGINX Service Mesh mutates Pod configurations by adding an init container and a sidecar to each managed Pod. The control plane builds an internal configuration based on events in Kubernetes and sends it to all application sidecars.

The sidecar components include an init container and an NGINX Plus instance. The init container sets up networking to redirect inbound and outbound traffic to the NGINX Plus instance. The agent component accepts the control plane configuration and configures the NGINX Plus instance, including retrieving certificate information from SPIRE.

NGINX Plus serves as the "brain" of the data plane, handling mTLS, traffic routing, tracing, and metrics. It proxies inbound and outbound traffic, encrypting connections, and honoring traffic policies defined in Kubernetes custom resources. Traffic is redirected using destination NAT (DNAT) to the NGINX Plus sidecar, which then forwards it to the original destination.

NGINX Service Mesh uses an init container to set up iptables REDIRECT rules for DNAT redirection, and separate ports are used for inbound and outbound traffic. Once the sidecar receives traffic, it performs tasks such as decryption or encryption for mTLS and then proxies the traffic to the original destination using Linux getsockopt socket API.

Overall, NGINX Service Mesh architecture involves the separation of application and networking concerns through the use of controllers, sidecars, and NGINX Plus, enabling reliable and secure network connections for distributed applications.

Behavioral View: [Referred: https://aosabook.org/en/v2/nginx.html]

A picture containing text, diagram, parallel, line

Description automatically generated

The Behavioral view examines the use of FrontEnd and BackEnd Proxies with their correlation to handle HTTPs requests. We can also observe the presence of Authentication Server to authenticate HTTPS requests from client end. HPC Node is used by both the Frontend and the Backend Proxies to run client requests. Instances of applications can schedule jobs to the HPC Node instances.

Interaction View:

Below view showcases how a user interacts with NginX for different requests. These include:

* Requesting a scaleup
* Rerouting traffic through LoadBalancer
* Cache for faster delivery of content
* Using the Content Delivery Network to geolocate content and deliver the user low latency experience.

A picture containing text, diagram, parallel, line

Description automatically generated