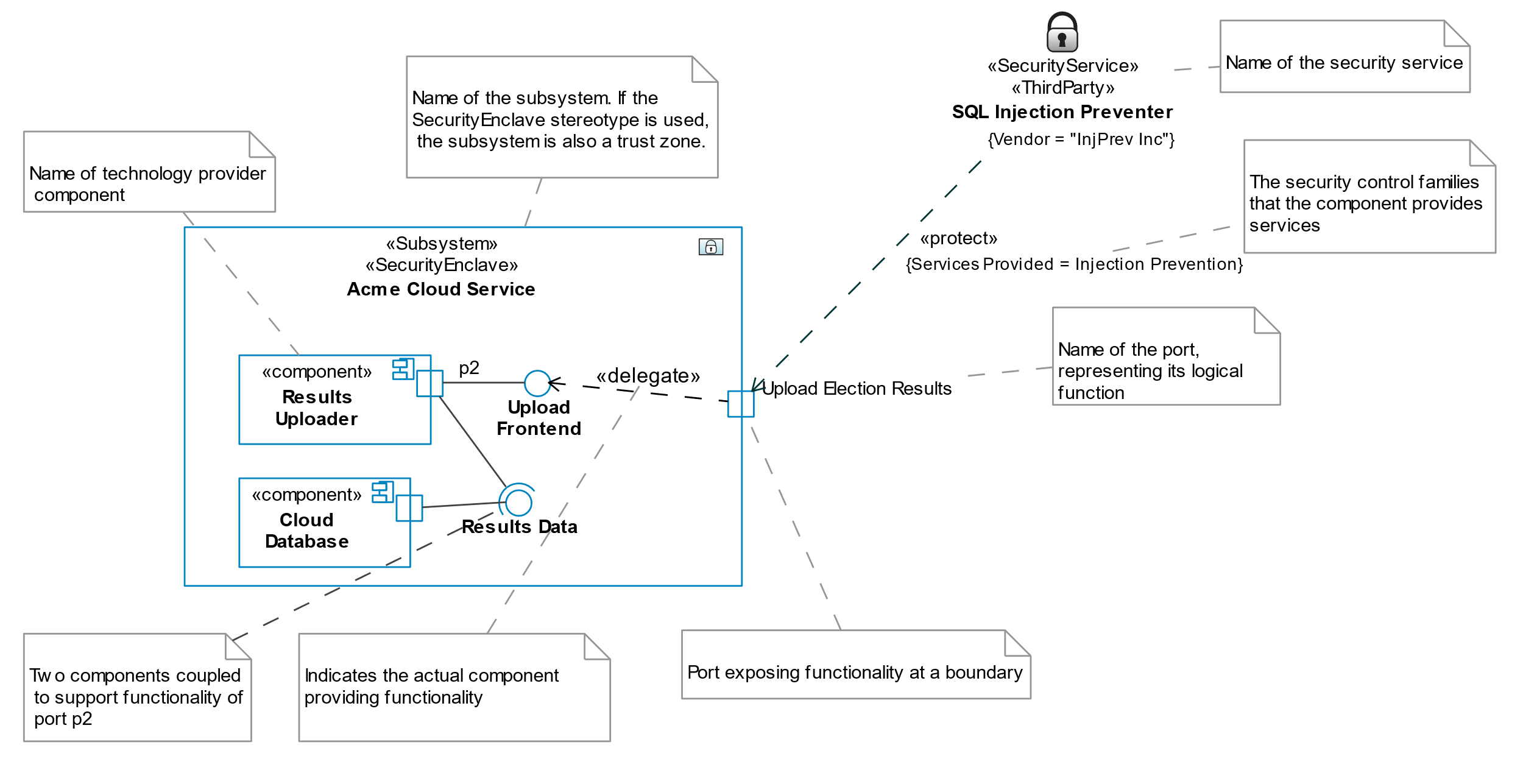
Understanding Architecture Review Diagrams

The goal behind the use of architectural diagrams is to provide detail around the registered technology provider’s product’s support for security services. This focus is novel in the area of architectural analysis, but aspects from existing notations taken together provide the necessary expressiveness. This document describes the RABET-V diagram notation.

## Diagram explanation



The following contrived example demonstrates the primary subset of [UML concepts](https://www.omg.org/spec/UML/About-UML/) used in RABET-V Component Diagrams. The key concept is that of a **component***.* Components are used to describe those of the *registered technology provider*, third party components providing security services, and the environment in which those components operate (e.g. trust zones).

Different [stereotypes](https://en.wikipedia.org/wiki/Stereotype_(UML)) are used to distinguish these uses. If a component is contained within a **trust zone** (notated with a [«SecurityEnclave»](https://docs.nomagic.com/display/UAFP190/Security+Enclave) stereotype), then its functionality must be exposed explicitly using **ports**. These ports (notated as boxes on the boundaries of a component) can expose functionality within all subcomponents, or a subset, and can be traced to particular parts of a component as well.

Each port may have interfaces that are either *required* (notated as a half circle) to support a function, or *provided* (notated as a full circle, or “lollipop”) to support another component’s function. For example, an Election Reports Reporting system may expose an *upload* *frontend* as a port on a Cloud Service. The *upload frontend* is the provided interface. However, to actually produce results, the port’s component relies on another, the Cloud Database. Thus, a port succinctly describes both the provided functionality, the means over which it is provided, and all components it depends on to support it.

## Purpose of subsystems

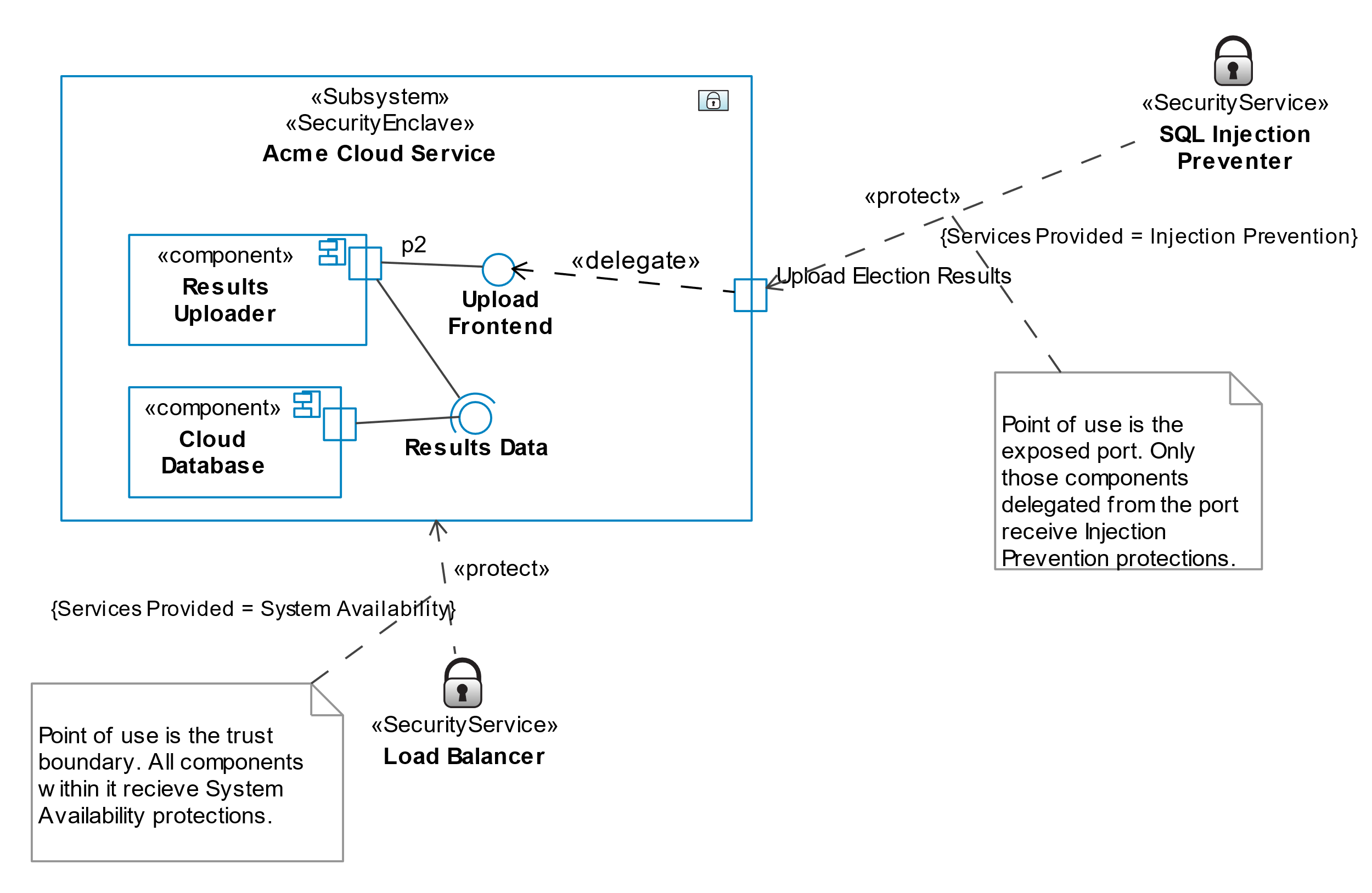
Many election technology products are complex “system of systems”. These smaller “subsystems” may use different technology stacks that do not interoperate at all levels. This is particularly true of software stacks, where it is understood that a library for the Ruby on Rails platform will not work with one designed for Microsoft’s ASP.NET.

A subsystem is a grouping of components identified from the threat modeling exercise and other architectural artifacts.

## Security services

Security services (notated «SecurityService») are considered components as well and are usually provided by a third party (notated «ThirdParty»). In the initial iteration of RABET-V diagrams, all security services appear outside any trust boundaries. This is to provide space in the diagrams; these services may or may not reside within a particular trust boundary or component.

Each security service may provide one or more services at a **point of use**. This is indicated by the application of a dependency (an arrow point from the security service to the protected element) with a «protect» stereotype. A security service may *protect* different components at various points in the architecture.



The following table describes the different points of use and their implication.

|  |  |  |
| --- | --- | --- |
| Point of Use | Meaning | Example |
| Trust Boundary | The trust boundary and all components inside it are protected | Firewall |
| Component | The entire component is protected | Logging and Alerting |
| Interface | The interface is protected | N/A |
| Port | The functionality provided by the port is protected | Authentication |

A security service that protects an individual port speaks for that port, and the target of its delegate (if any), only. Others may indicate a larger surface of the product is protected.

**Gap Analysis**

The diagrams are constructed to automate the process of performing a gap analysis. For our purposes, the gap analysis determines if any component does not receive, direct or indirect protection from a security service providing mitigations from all ten control families. The gap analysis uses the point of use relationship to determine if there are gaps.

Future iterations of gap analysis may be further refined by attaching sensitives to functions.