FACULTAD DE INGENIERIA
Universidad de Buenos Aires

75.10 Técnicas de Diseño
Año 2018 - 1er Cuatrimestre

Trabajo Práctico - 1er Entrega
“Rule Validator”

Integrantes:

Ejberowicz, Maximiliano (95295)
Fernández Vidal, Mariano (89789)
Ponce, Julieta Belén (96375)

Grupo N° 4

Índice

Diseño de la solución

Hipótesis

Se tomaron las siguientes hipótesis:

- H1) Si un contador cumple la condición pero sus parámetros no se encuentran en los datos, el contador no aumenta de valor.
- H2) Las condiciones o expresiones que se reciben por parámetro no tienen errores de sintaxis.
- H3) Siempre se utiliza el primer dato pasado que se encuentre que cumpla la condición para chequear la validez de la misma.
- H4) Si para un contador, existen varios datos pasados que cumplen la condición, solo se elige uno para definir el subcontador. Para ilustrar mejor la situación, si por ejemplo, la regla fuera:
    - *(define-counter "past-important-count" [(past "important")] true)*

  Entonces, sólo se selecciona uno de los datos pasados para crear el subcontador o "important" true o "important" false

Ventajas y desventajas de la solución propuesta

Ventajas

- V1) Se realizó un uso necesario de las multifunciones para resolver el “polimorfismo” en distintas oportunidades.
- V2) Se modularizaron distintos componentes detectados durante el análisis. Se impuso una organización más cercana al paradigma de objetos, pero siempre respetando la parte funcional.
- V3) El query-counter no debe realizar ningún procesamiento de datos, más bien una búsqueda indexada sobre valores del contador existentes.
- V4) Se realizó la separación de datos en el initialize-processor, de manera de facilitar ciertas operaciones en pasos posteriores. Ej: counters/signals, parameters por tipo.

Desventajas

- D1) La función define-condition debe recibir los contadores, únicamente por la posibilidad de referirse a un counter-value. Es innecesario para los demás símbolos.
- D2) Debido a H3) se debe hacer un procesamiento de los datos pasados.

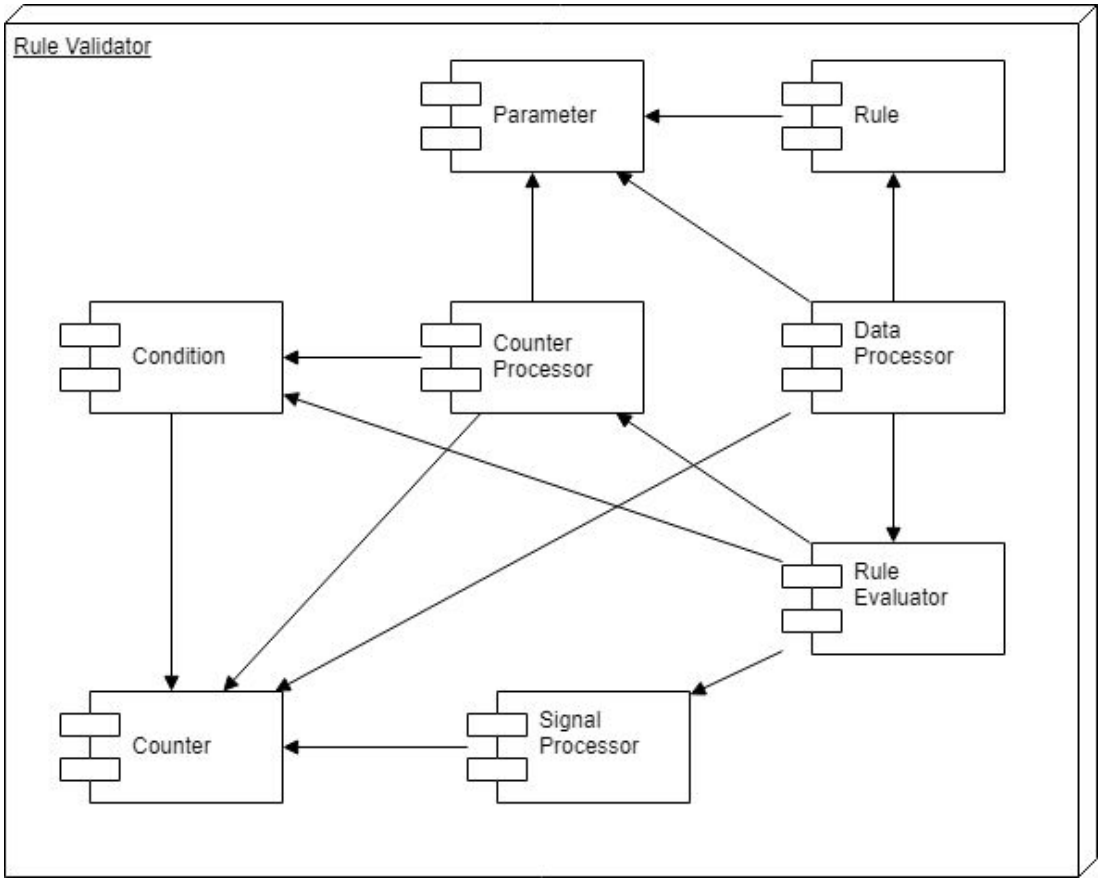
Descripción del diseño elegido

Se muestran a continuación la vista lógica y de desarrollo propuestas por el Modelo de “4+1” Vistas de la Arquitectura del Software de Kruchten.

Se omiten la vista física ya que el proceso se ejecuta en un único procesador y la vista de procesos ya que se ejecuta un único proceso.

Vista de Desarrollo

A continuación, se muestra la organización del software en su entorno de desarrollo ilustrando las dependencias de cada módulo.



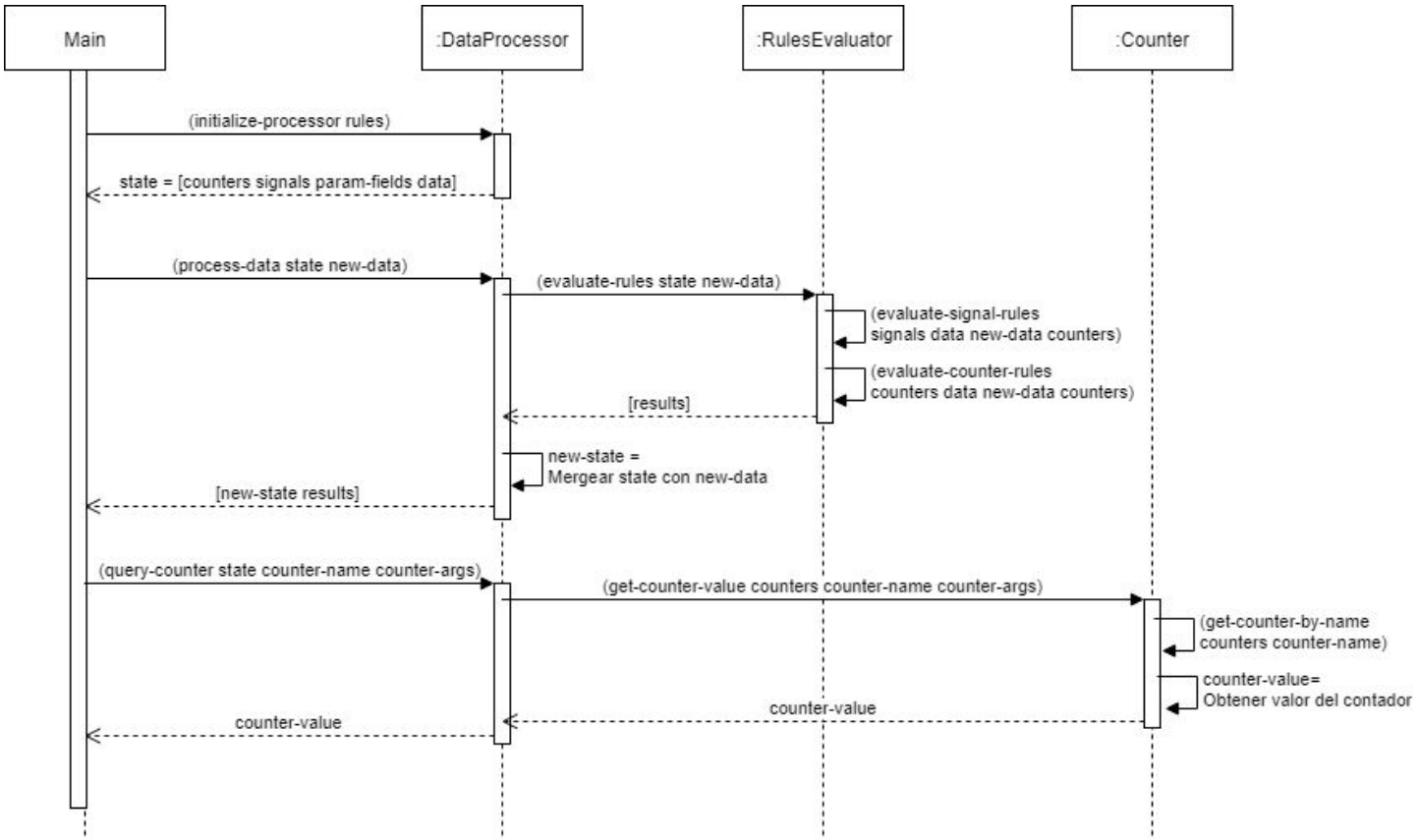
Vista Lógica

Se muestran las principales funciones del sistema en el siguiente diagrama lógico:

El procesamiento de nuevos datos necesita de un sistema inicializado para poder operar y es el que recibe un dato nuevo para que luego se validen sus condiciones y así generar un nuevo estado del sistema, con los contadores y señales propiamente actualizados. La consulta de contadores se realiza sobre un estado del sistema ya generado y validado.

Para visualizar la funcionalidad que se le presenta al usuario final y el detalle de su funcionamiento, se eligió un diagrama de secuencia.

Se muestra en el siguiente diagrama la interacción simplificada entre los módulos principales que ejecutan las funciones *initialize-processor*, *process-data* y *query-counter*.


Main
:DataProcessor
:RulesEvaluator
:Counter
(initialize-processor rules)
state = [counters signals param-fields data]
(process-data state new-data)
(evaluate-rules state new-data)
(evaluate-signal-rules
signals data new-data counters)
(evaluate-counter-rules
counters data new-data counters)
[results]
new-state =
Mergear state con new-data
[new-state results]
(query-counter state counter-name counter-args)
(get-counter-value counters counter-name counter-args)
(get-counter-by-name
counters counter-name)
counter-value=
Obtener valor del contador
counter-value
counter-value