MuEPL

Lorena Gutiérrez-Madroñal
Tutores: Juan José Domínguez-Jiménez, Inmaculada Medina-Bulo

Department of Computer Science and Engineering
University of Cádiz, Spain
{lorena.gutierrez, juanjose.dominguez,
inmaculada.medina}@uca.es

Índice

Introducción

Event Processing Language

Ejemplos

Resultados y conclusiones

Introducción

Ppal. inconveniente

El principal inconveniente de la prueba de mutaciones es el **alto coste computacional** que envuelve la ejecución de un gran número de mutantes frente sus casos de prueba.

Varias estrategias intentan solventarlo:

- Técnicas de reducción de mutantes
- Muestreo de mutantes
- Agrupamiento de mutantes
- Mutación selectiva
- Mutación de orden superior
- **Técnica de Mutación Evolutiva (EMT)**

EMT y GAmera

Esta técnica trata de generar y ejecutar algunos de los mutantes **preservando la efectividad de las pruebas**. EMT genera y selecciona los mutantes en un solo paso, reduciendo el número de mutantes a ejecutar.

La EMT se usa en la herramienta **GAmera**, con el lenguaje WS-BPEL.

GAmera puede generar, ejecutar, evaluar y clasificar los mutantes, según su calidad. Incorpora un **algoritmo genético** que solo selecciona los mutantes de alta calidad.

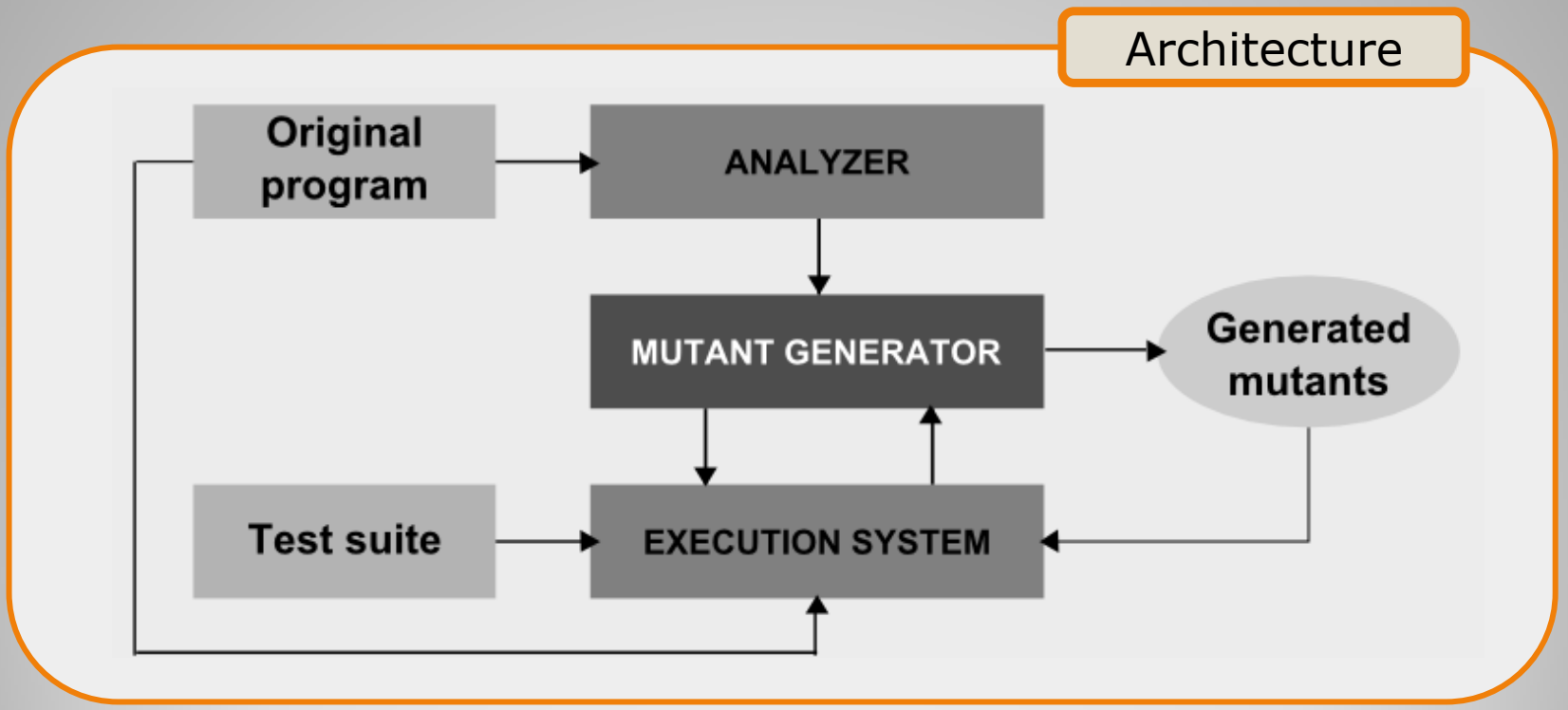
Para probar su efectividad queremos aplicarla con otro lenguaje de programación: **Event Processing Language**, EPL.

Reutilización de **MuBPEL**, herramienta para la prueba de mutaciones para the Web Services Business Process Execution Language (WS-BPEL).

Analiza y genera **todos** los mutantes y ya está adaptada a GAmera, luego nuestro sistema será fácil de aplicar.

MuBPEL and MuEPL

Tiene tres componentes principales: analizador, generador y el motor de ejecución.



MuBPEL and MuEPL

Es un lenguage similar a SQL con cláusulas SELECT, FROM, WHERE, GROUP BY, HAVING y ORDER BY, pero difiere de éste en que utiliza vistas en vez de tablas.

Desigualdades

Los datos de prueba de SQL son un conjunto de filas presentadas en una tabla, mientras que en EPL son un conjunto de eventos presentados en una cadena.

Operadores de mutación

Mutant killing criteria

Event Processing Language

EPL

Operadores de mutación

48 operadores de mutación
8 categorías:

- Pattern expression (PEP): 6
- Window of length (WOL): 2
- Window of time (WOT): 3
- Batch processing of event (BOE): 2
- SQL injection attack (SQIJ): 4
- EPL clauses (EPLC): 16
- Operator replacement (ORP): 11
- Null mutation operator (NOP): 3

Event Processing Language

EPL

Mutant killing criteria

Category	Operator	Killing criteria
Pattern expression (PEP)	RREP, CEOP, RLOP	Núm. de eventos entre O y M
	RGEP, OEDIP, OEDDP	Latencia entre O y M
Window of length (WOL)	LINC, LDEC	Núm. de eventos entre O y M
Window of time (WOT)	TINC, TDEC, TRUN	Latencia entre O y M
Batch processing of event (BOE)	BATL	Núm. de eventos entre O y M
	BATT	Latencia entre O y M
SQL injection attack (SQIJ)	SQRC, SQNC, SQFD, SQUP	Núm. de eventos entre O y M
EPL clause (EPLC)	EAGR, ESEL, EGRUE, EGRUA, EJOI, EORDE, EORDK, EORDS, ESUBRI, ESUBRII, ESUBRIII, ESUBIBII, ESUBIBIII, ESUBIIBI, ESUBIIBIII	Núm. de eventos entre O y M
	ESIRF, ESIRL	Núm. de eventos entre O y M Latencia entre O y M
Operator replacement (ORP)	EBTW, EABS, EAOR, EROR, EUOI	Salidas entre O y M (numéricas).
	ELKEWC, ELKEWR, ELKECA, ELKECB, ELKEAE, ELKEAB	Salidas entre O y M (cadenas)
Null mutation operator (NOP)	ENLF, ENLI, ENLO	Núm. de eventos entre O y M

Event Processing Language

EPL

```
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price where timer:within(10 sec)]
```

Analizador

```
MuEPL starts…
The command analyze starts…
RREP  1  1
CEOP  0  1
RLOP  0  1
RGEP  1  1
OEDIP 1  1
…
TRUN  1  4
…
```

Ejemplos

Ejemplos

```
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price where timer:within(10 sec)]
```

operador RGEP

Remove guard expression: Elimina las expresiones "guard" de las condiciones "where" presentes en los patrones de tiempo. Éstas controlan el ciclo de vida de las expresiones de patrones.

Generador

```
MuEPL starts…
The command apply starts…
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price ~~where timer:within(10 sec)~~]
```

Ejemplos

```
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price where timer:within(10 sec)]
```

operador OEDIP

Observer expression's time delay decrement: Decrementa en una unidad el valor del tiempo del patrón.

Generador

```
MuEPL starts…
The command apply starts…
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price where timer:within(9 sec)]
```

select OrderEvent.symbol as symbol, OrderEvent.price as price from pattern[every price= OrderEvent.price where timer:within(10 sec)]

operador TRUN

Replace time unit: Reemplaza una unidad de tiempo (milisegundo, segundo, minuto, hora y día) por otra.

Generador

MuEPL starts…
The command apply starts…
select OrderEvent.symbol as symbol, OrderEvent.price as price from pattern[every price= OrderEvent.price where timer:within(10 **min**)]

Ejemplos

Ejemplos

```
select OrderEvent.symbol as symbol, OrderEvent.price as price from
pattern[every price= OrderEvent.price where timer:within(10 sec)]
```

operador ESEL

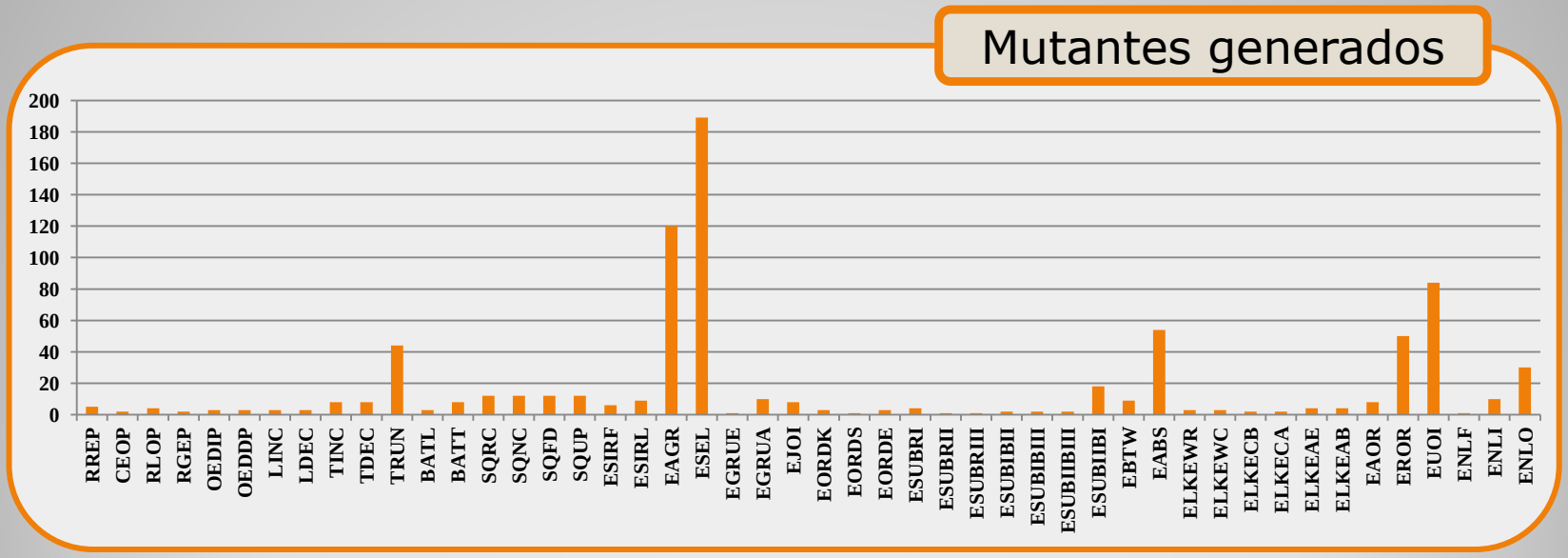
SELECT stream clause: Tenemos tres tipos de cadenas: istream, rstream e irstream. Cada una es mutada por la otra. Además existen las palabras reservadas distinct y all. SELECT [DISTINCT | ALL] (ISTREAM | RSTREAM | IRSTREAM) ...

Generador

```
MuEPL starts…
The command apply starts…
select rstream OrderEvent.symbol as symbol, OrderEvent.price as price
from pattern[every price= OrderEvent.price where timer:within(10 sec)]
```

Se ha utilizado **MuEPL** para generar mutantes de cada una de las consultas del conjunto de casos de prueba.

Ninguno de los trabajos relacionados ha desarrollado un conjunto de casos de prueba para evaluar consultas EPL.



Resultados y conclusiones

Resultados y conclusiones

De los operadores definidos, hemos identificado algunos que generan **mutantes equivalentes** (EABS y ESEL).

Los resultados indican que los operadores propuestos y el mutant killing criteria pueden ser usados para poner a prueba los **problemas comunes** de calidad de los **programas en tiempo real** tales como: la notificación de fallos y retrasos, secuencia incorrecta de eventos y ataques de introducción de código.

Se propone un enfoque basado en la prueba de mutaciones para evaluar la calidad de consultas de procesamiento de eventos.

Resultados y conclusiones

Se han desarrollado nuevos operadores y el mutant killing criteria para consultas EPL. También se han creado un conjunto de casos de prueba específico para EPL.

Utilizando la arquitectura de MuBPEL, se desarrolla MuEPL, para el análisis y generación automática de mutantes de EPL. MuEPL genera todos los mutantes, por lo que, al tener la misma arquitectura que MuBPEL, podemos adaptarlo a GAmera y probar la efectividad de la EMT.

To Do

1. Adaptación de MuEPL a GAmera
2. BD para el conjunto de casos de prueba
3. Comprobar y comparar resultados

¡Gracias!

¿Preguntas?