Universidad Autónoma de Madrid

Escuela Politécnica Superior

Un lenguaje de dominio específico para mutación de modelos

Pablo Gómez Abajo

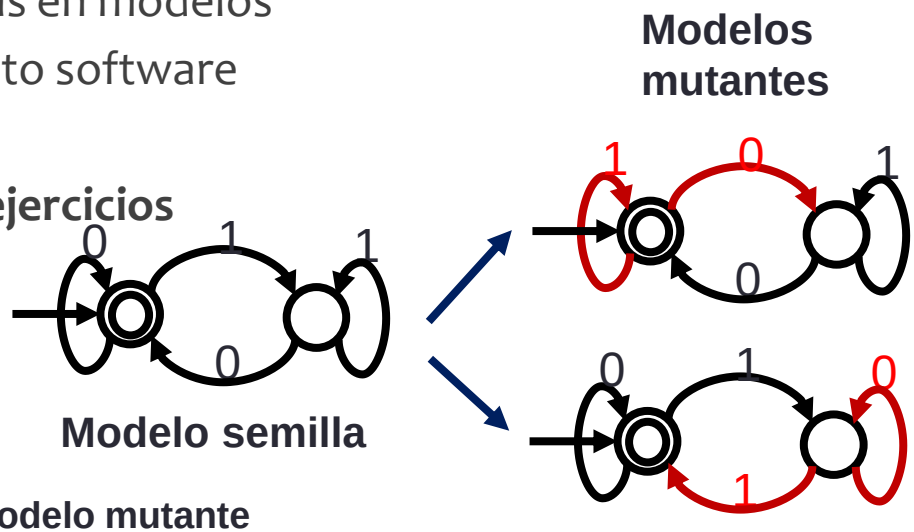
Directores: Esther Guerra y Mercedes G. Merayo

UNIVERSIDAD AUTÓNOMA DE MADRID

Doctorado en Ingeniería Informática y de Telecomunicaciones, Julio 2020

¿Qué es la mutación de modelos?

- ✓ La mutación de un modelo es la variación de un modelo semilla al que se aplica uno o más operadores de mutación
- ✓ La mutación de modelos tiene múltiples aplicaciones:
  - Pruebas de transformación de modelos
  - Pruebas de software basadas en modelos
  - Pruebas de líneas de producto software
  - Algoritmos genéticos
  - **Generación automática de ejercicios**
  - **Pruebas de mutación**
  - …



Motivación

- ✓ Los entornos existentes para mutación de modelos son:
  - Específicos para un lenguaje (p.ej., fórmulas lógicas)
  - Específicos para un dominio (p.ej., pruebas)
  - Los operadores de mutación están codificados a mano
- ✓ Se requiere un método fácil para crear aplicaciones basadas en mutación de modelos
- ✓ No proporcionan utilidades para la mutación de modelos como:
  - Generación de las trazas de los modelos mutantes
  - Primitivas potentes para crear los mutantes
  - Validación de los mutantes

Objetivos

- ✓ Lenguaje independiente del dominio para mutación de modelos- **Wodel**
  - Primitivas de mutación de alto nivel
  - Generación de modelos semilla
  - Métricas de mutación

- ✓ Extensible para aplicaciones de los mutantes
  - **Wodel-Edu:** Generación automática de ejercicios
  - **Wodel-Test:** Creación de herramientas de pruebas de mutación para lenguajes de programación o de modelado

- ✓ Utilizamos un enfoque basado en MDE

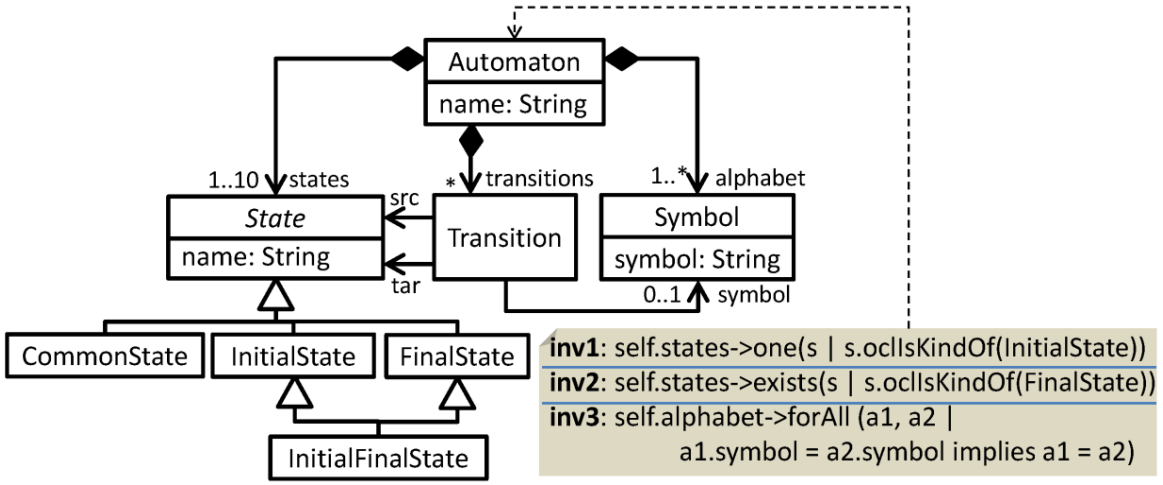
Índice

I. Conceptos previos

I. Ingeniería dirigida por modelos
II. Transformación de modelos
III. Mutación de modelos

Ingeniería dirigida por modelos

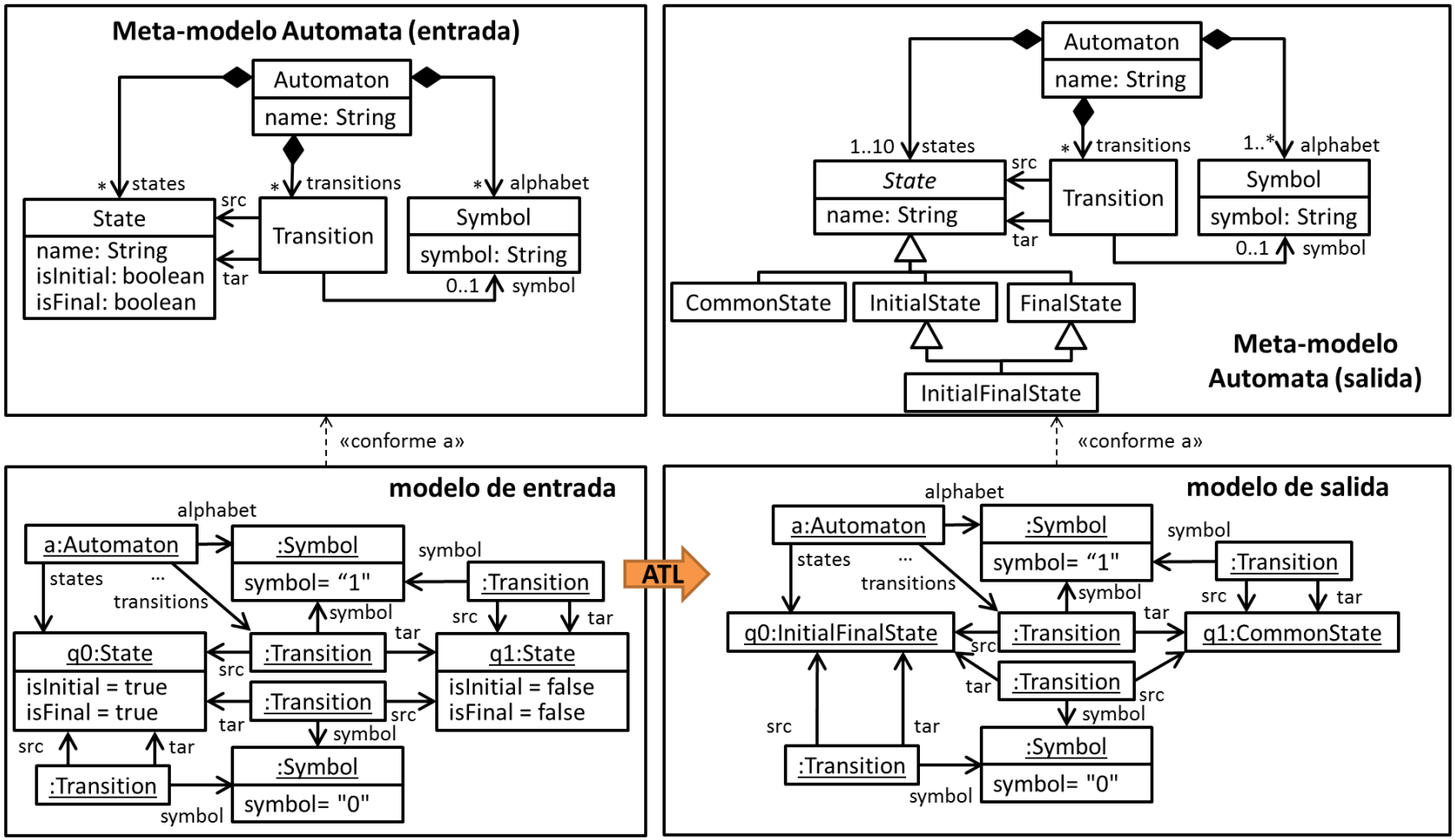
- ✓ **Modelo:** descripción de un sistema utilizando un lenguaje
- ✓ **Meta-modelo:** modelo que describe un lenguaje



- ✓ **Lenguaje de dominio específico:**
  - Ofrece los conceptos para un dominio de aplicación
  - Gráfico o textual
  - Primitivas de alto nivel
  - Orientado a expertos en el dominio
  - Traducido automáticamente a lenguajes como C, Java, etc.

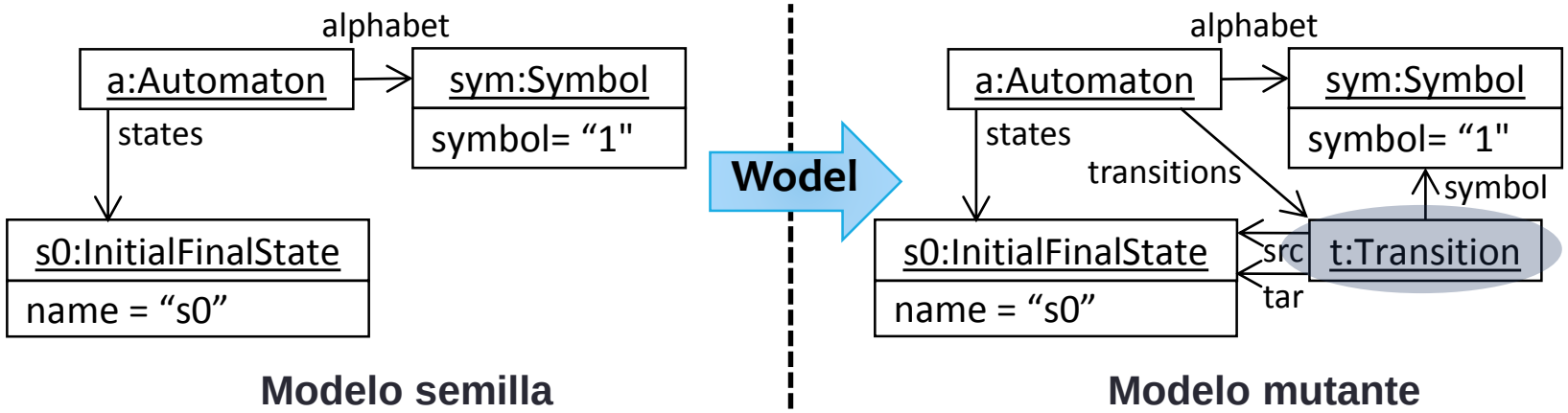
Transformación de modelos

- ✓ Necesario para la manipulación de modelos



Mutación de modelos

- ✓ Caso especial de transformación de modelos



- ✓ Las primitivas de mutación de alto nivel son más potentes que utilizar la transformación de modelos para aplicar mutación

II. Wodel

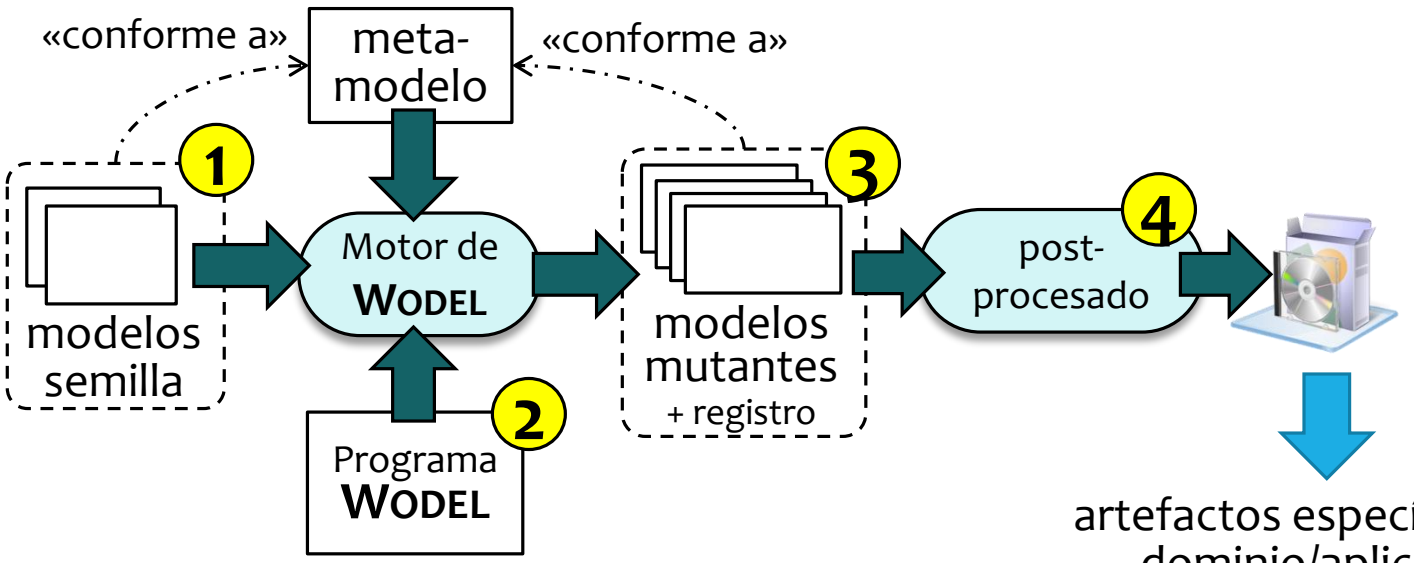
Un lenguaje de dominio específico para mutación de modelos

I. DSL de operadores de mutación
II. Servicios de ejecución
III. Servicios de desarrollo
IV. Herramienta proporcionada

Propuesta

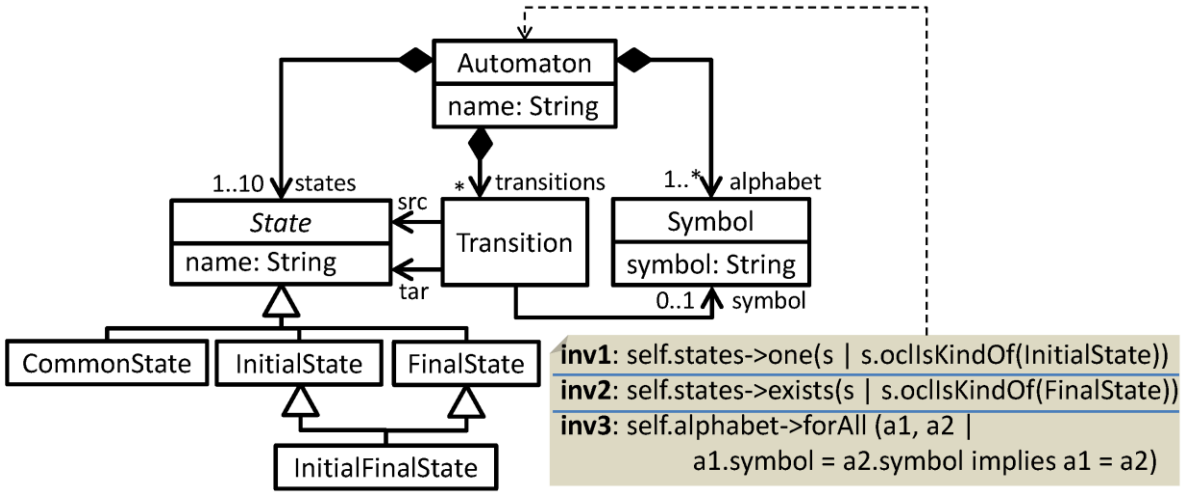
- ✓ DSL **Wodel** para mutación de modelos con:
  - Primitivas de mutación de alto nivel
  - Independiente del dominio
  - Compilado a Java
- ✓ Servicios de ejecución (extensibles)
  - Validación de los mutantes
  - Detección de los mutantes equivalentes
  - Registro de mutaciones
  - Extensible para diferentes aplicaciones
- ✓ Servicios de desarrollo
  - Síntesis de modelos semilla
  - Métricas de mutación

Esquema



pruebas de transformación de modelos
pruebas de software basadas en modelos
pruebas de líneas de producto software
algoritmos genéticos
generación automática de ejercicios
pruebas de mutación
...

Ejemplo



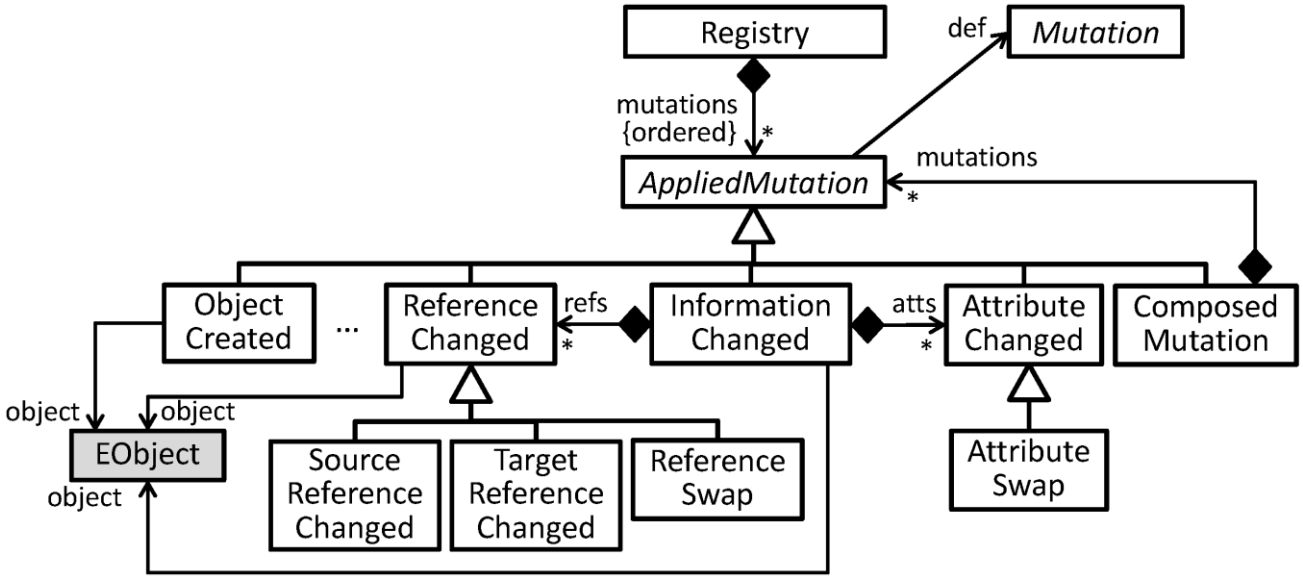

```
generate 3 mutants in "out/" from "models/"

metamodel "http://fa.com"

description "Simple Wodel Program"

with blocks {

  CNFS "Changes a final state to non-final and connects it with a new final state"
  {

     s0 = retype one FinalState as CommonState

     s1 = create FinalState

     t0 = create Transition with {src = s0, tar = s1, symbol = one Symbol}

  }}
```

Operadores de mutación

- ✓ 9 primitivas de mutación
  - crear, clonar, seleccionar, modificar, retipar y borrar información
- ✓ Asignación automática de los elementos creados en contenedores compatibles
- ✓ Eliminación de referencias vacías al borrar elementos
- ✓ Comprobación de cardinalidades y restricciones OCL
- ✓ Cardinalidades de los operadores de mutación

Registro de mutaciones

- ✓ Trazas entre modelos semilla y modelos mutantes
- ✓ Opción de compactar el registro (mutaciones irrelevantes)
- ✓ Opciones de texto en Wodel-Edu
- ✓ Identificación del operador de mutación en Wodel-Test



Validación de los mutantes

- ✓ OCL en programa Wodel

```
generate 2 mutants in "out/" from "models/"
metamodel "http://fa.com"

with blocks {
  simple {
    s0 = select one InitialState
    s1 = select one State where {self not typed FinalState}
    t0 = select one Transition where {src = s0}
    modify one Transition where {tar = s1} with {swap(tar, t0->tar)}
  } [2]
}

constraints {
  context State connected:
    "oclIsKindOf(InitialState) or
    Set{self}->closure(s | Transition.allInstances()->select(t | t.tar=s)->collect(src))
        ->exists(s | s.oclIsKindOf(InitialState))"
}
```

- ✓ Punto de extensión (p.ej. modelo de autómata que acepta un lenguaje determinado)

Detección de mutantes equivalentes

- ✓ Equivalencia sintáctica

```
int inc(int value) {
          // Incrementa el valor
          return value + 1;
}
```

```
int inc(int value) {
          return value + 1;
}
```

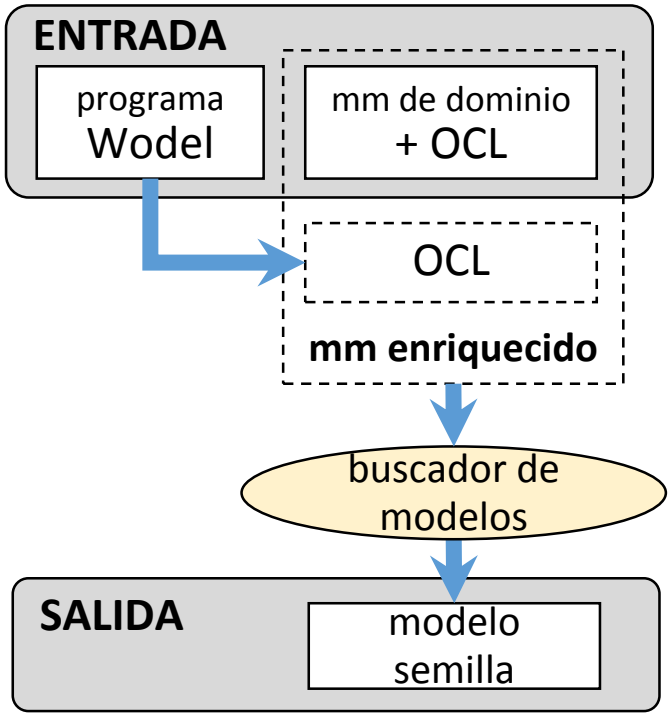
- ✓ Equivalencia semántica

```
int inc(int value) {
          // Incrementa el valor
          return value + 1;
}
```

```
int inc(int value) {
          // Incrementa el valor
          return value + 1 + 0;
}
```

Síntesis de modelos semilla I

- ✓ Comprobar que los operadores de mutación están implementados correctamente
- ✓ Basado en la búsqueda de modelos



Síntesis de modelos semilla II

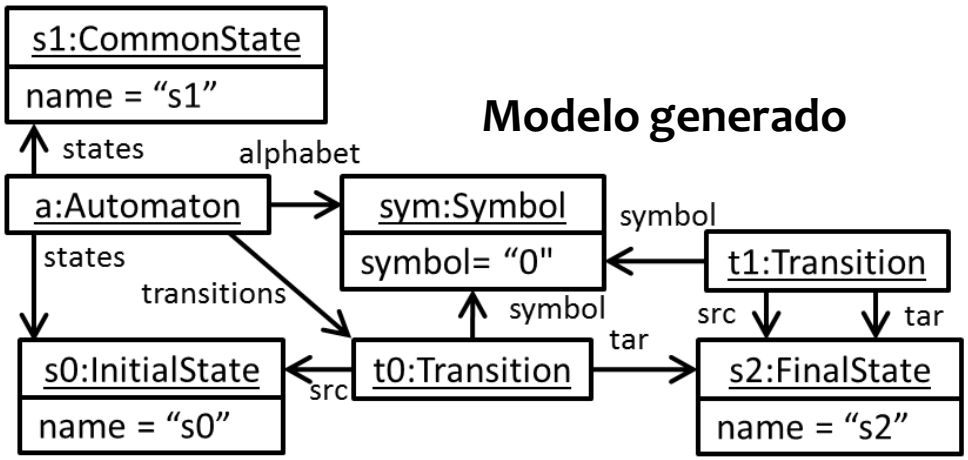
```
generate 2 mutants in "out/" from "models/"
metamodel http://fa.com

with blocks {
  cs2fs "Retypes a common state as a final state"
  {
    retype one CommonState as FinalState
  }
}
```

OCL

```
context Dummy
inv mut1 : CommonState.allInstances()->exists(s | true)
```

Buscador de modelos

Modelo generado



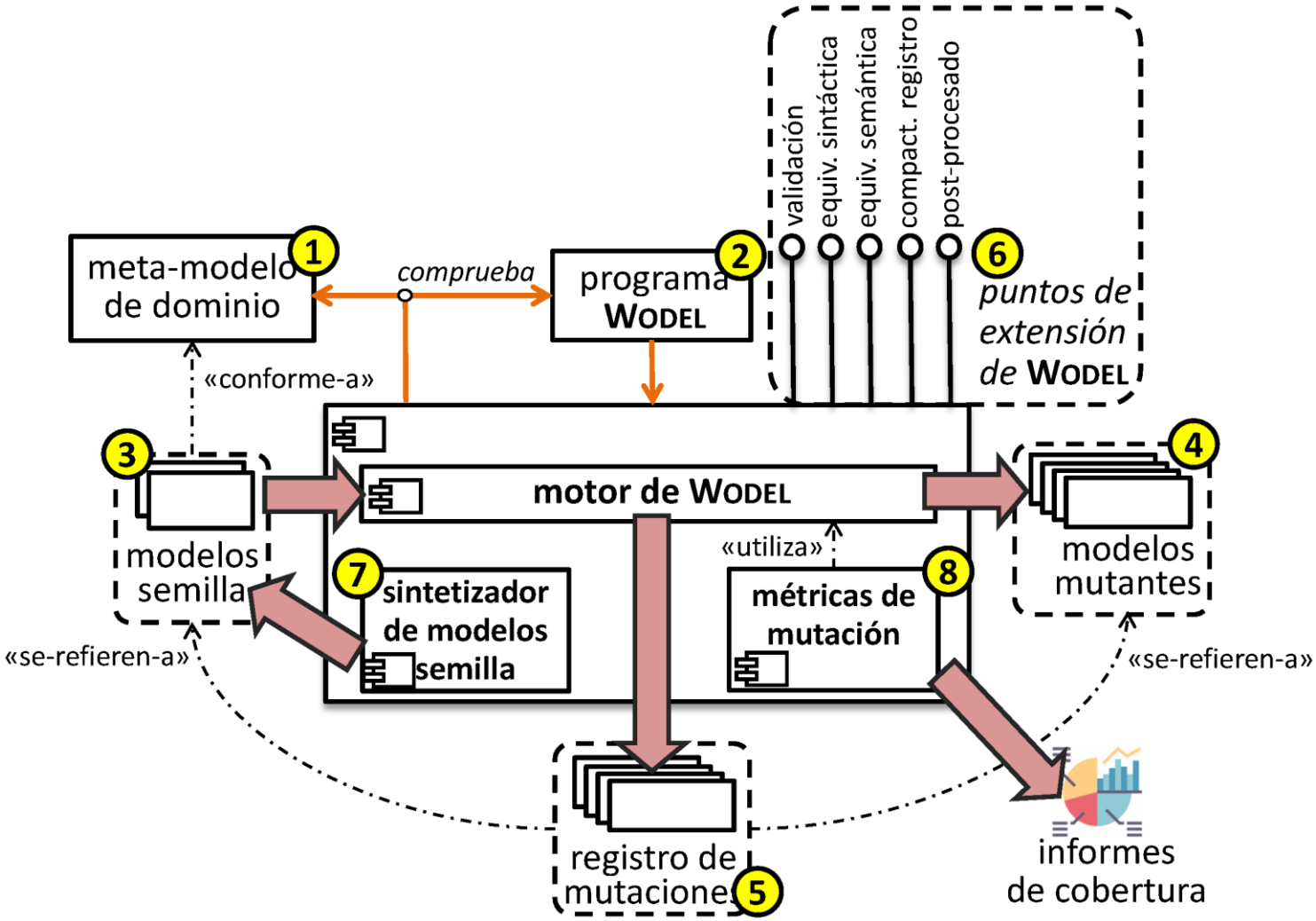
Métricas de mutación

- ✓ Cobertura de mutación de los elementos

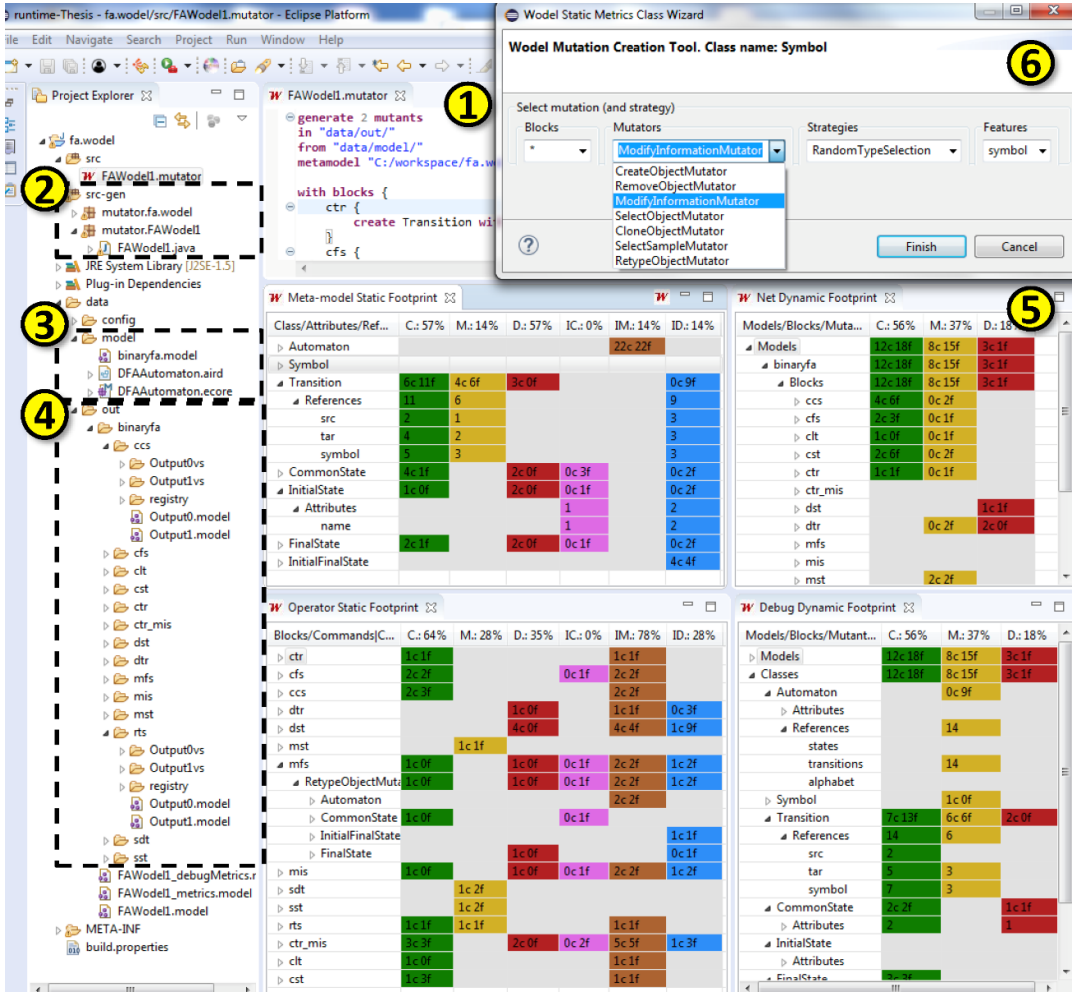
- ✓ Métricas estáticas
  - Elementos del meta-modelo afectados por las mutaciones
  - Métricas del meta-modelo
  - Métricas por operador

- ✓ Métricas dinámicas
  - Efectos reales del programa de mutación
  - Métricas netas
  - Métricas de depuración

Arquitectura de la herramienta



IDE de la herramienta



Herramienta disponible en http://gomezabajo.github.io/Wodel/ junto con ejemplos, vídeos y tutoriales

III. Wodel-Edu

Generación de ejercicios mediante mutación

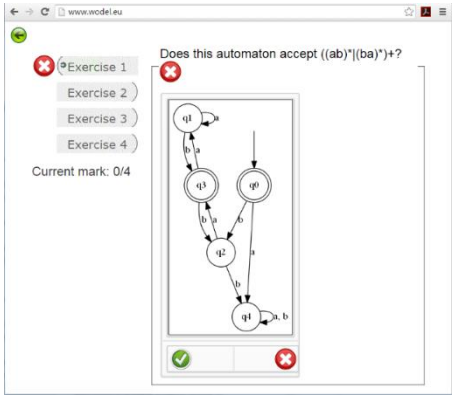
Motivación

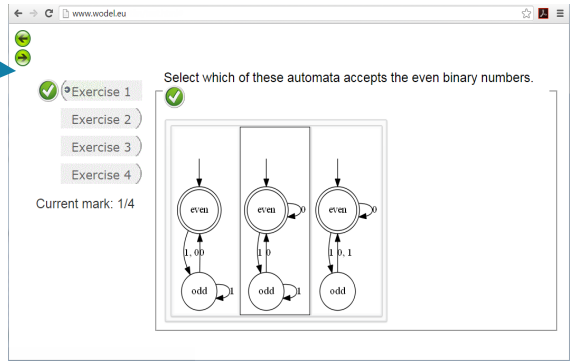
- ✓ Generación de un volumen grande de ejercicios tipo test
- ✓ Fáciles de generar
- ✓ Independientes del dominio

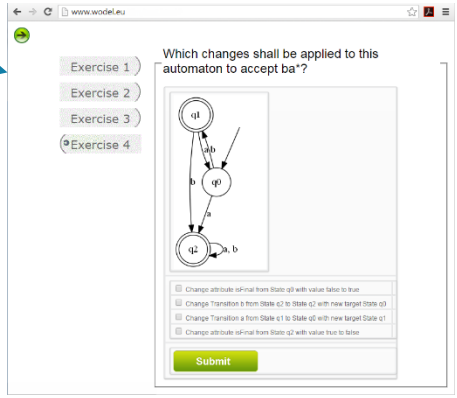
Se propone el entorno **Wodel-Edu** - extensión a Wodel

Propuesta

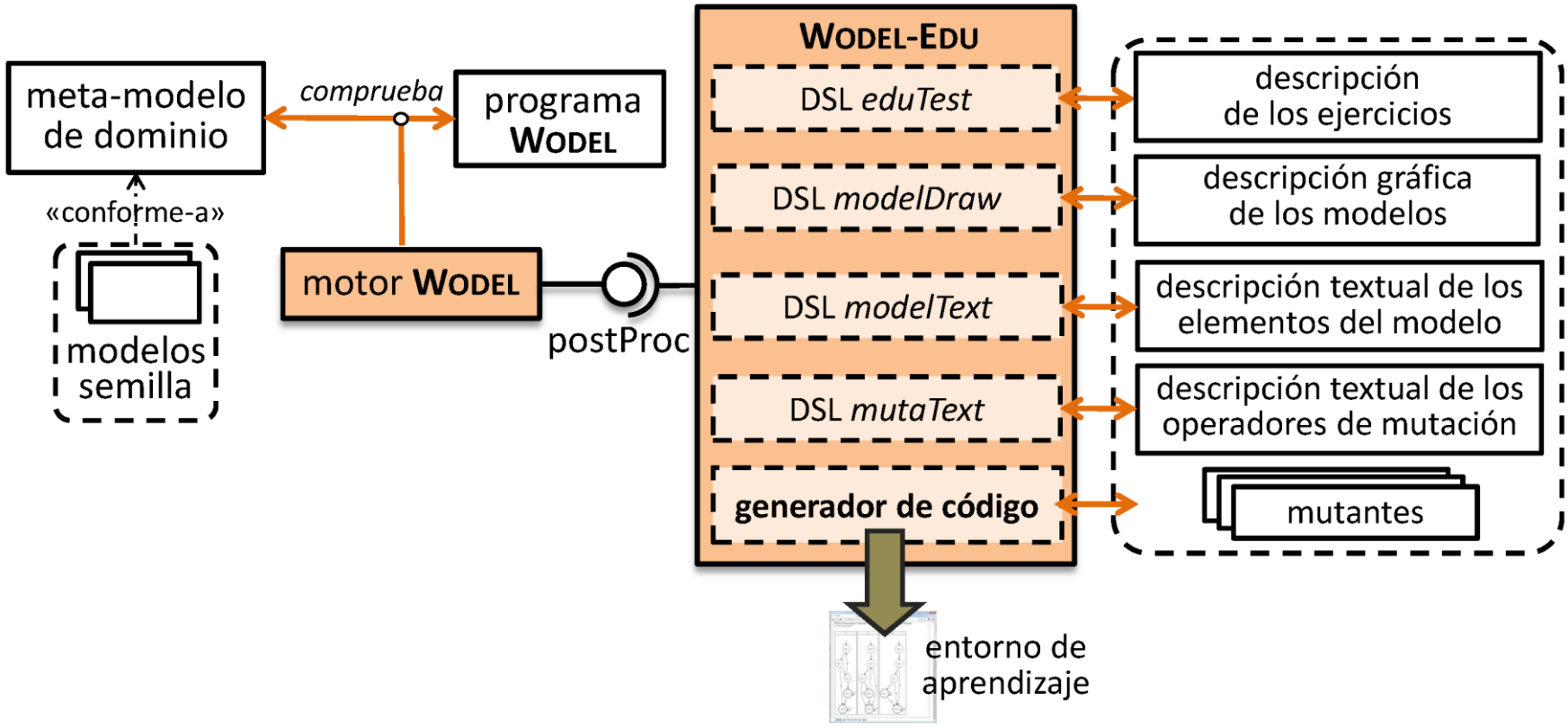
- ✓ Aplicación web con tres formatos de ejercicios:
  - Respuesta alternativa
  - Selección de un diagrama entre varios
  - Selección de opciones de texto
- ✓ Se requiere:
  - Representación gráfica de los modelos
  - Generación de las opciones de texto
  - Generación de código HTML+JavaScript



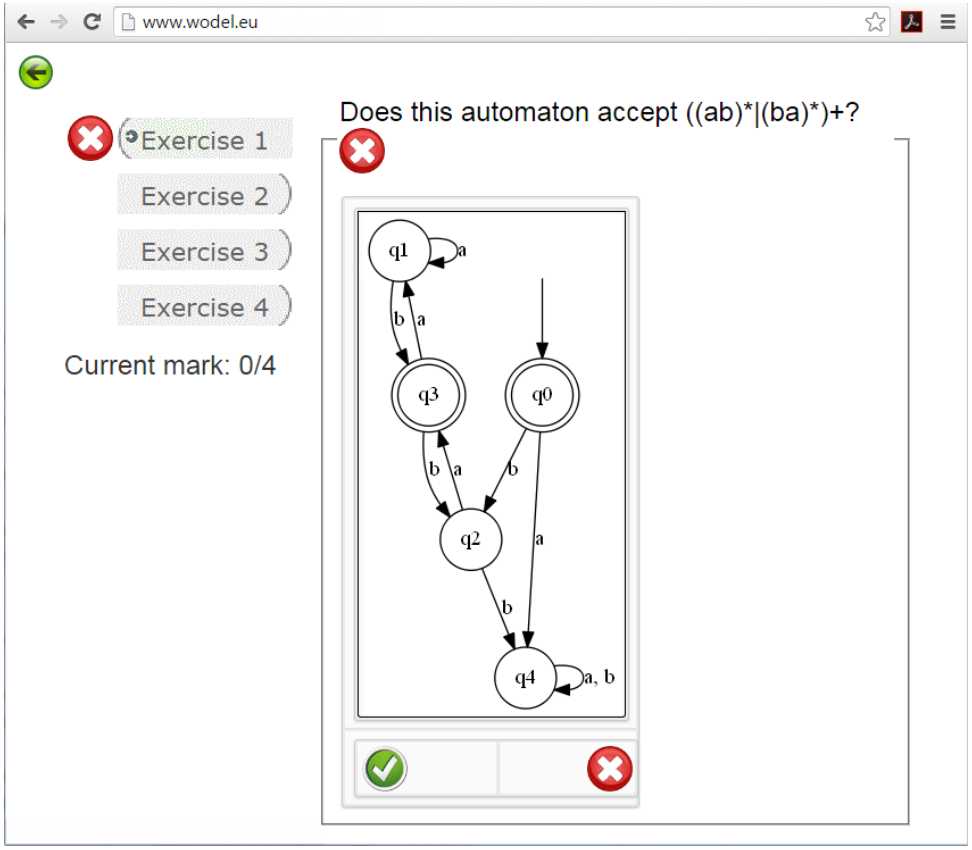




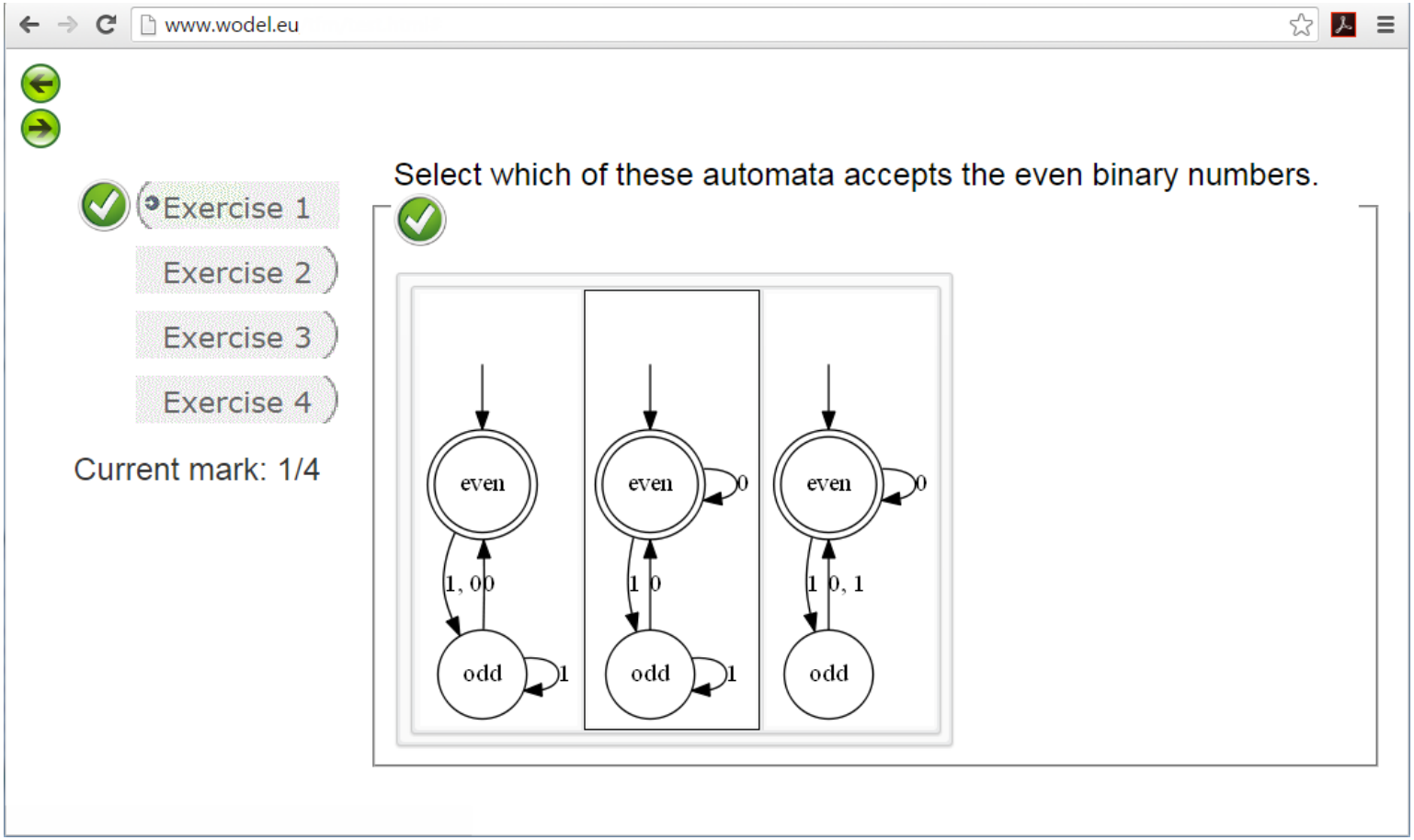
Arquitectura de Wodel-Edu



Respuesta alternativa

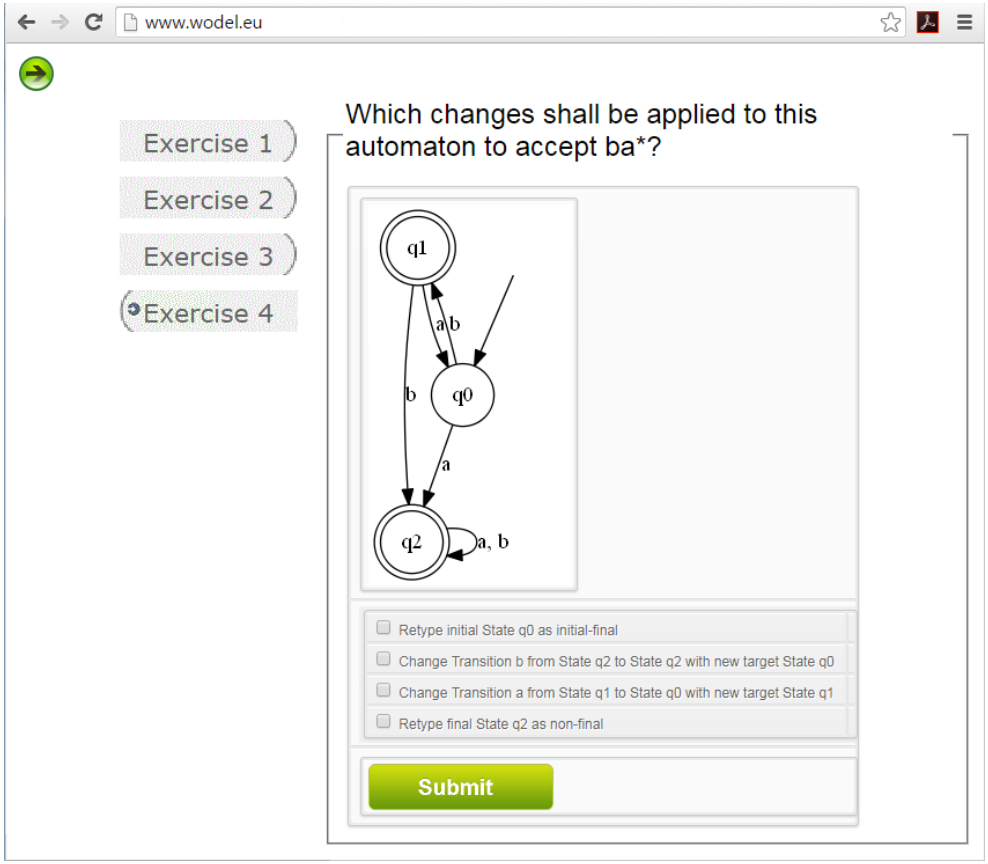


Selección de diagrama múltiple



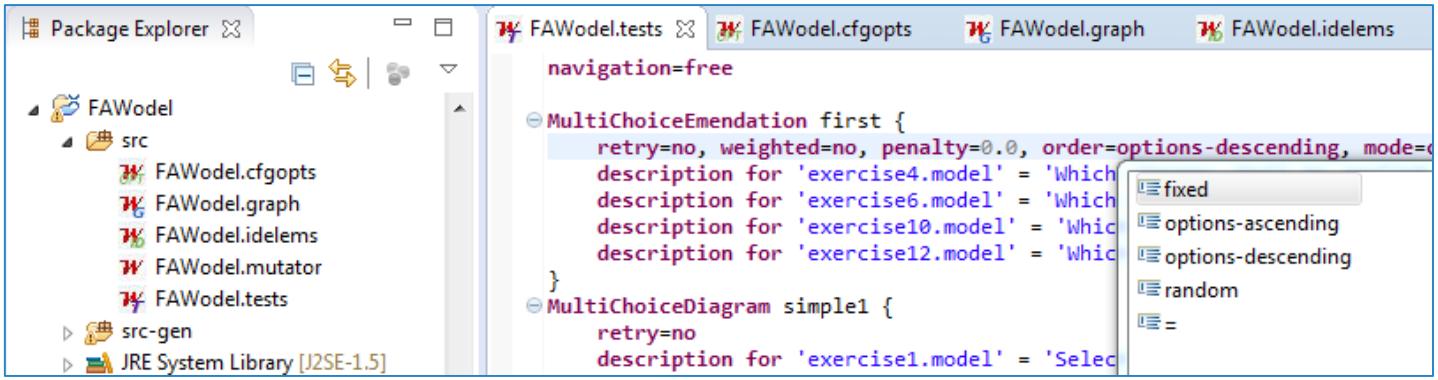
Selección de opciones de texto

- ✓ Opciones de texto a partir del registro
- ✓ Utilización de los DSLs modelText y mutaText

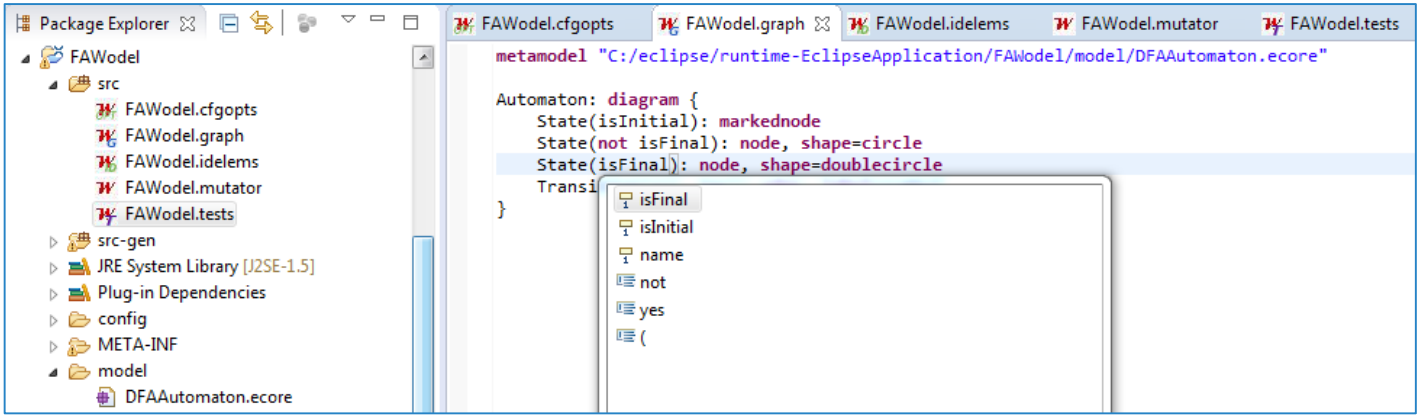


IDE de Wodel-Edu I

DSL eduTest

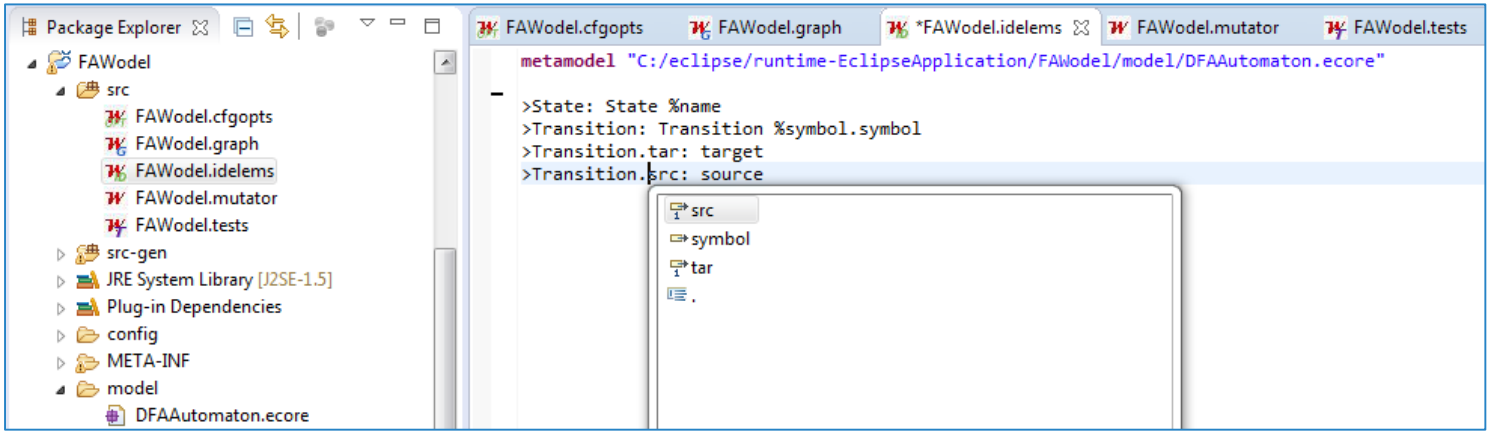


DSL modelDraw

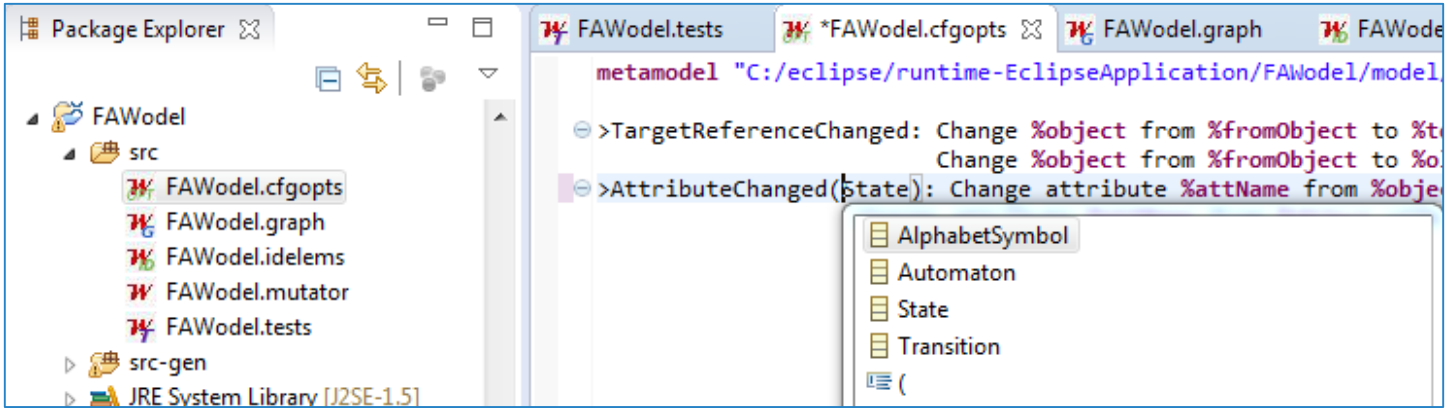


IDE de Wodel-Edu II

DSL modelText



DSL mutaText



Evaluación de Wodel-Edu

- ✓ Aplicación web http://www.wodel.eu en el dominio de los autómatas con tres secciones
  - Opciones de texto
  - Selección de un diagrama entre varios
  - Respuesta alternativa
- ✓ Tres dimensiones, además de la nota obtenida (opcional)
  - El ejercicio se entiende bien
  - La dificultad del ejercicio es adecuada
  - El ejercicio es útil para aprender autómatas
- ✓ 10 participantes entre 22 y 41 años (1 sin formación en autómatas)

Conclusiones de la evaluación

- ✓ Se perciben como muy útiles para aprender autómatas
- ✓ Dificultad percibida razonable
- ✓ El ejercicio de selección de opciones de texto es el más complicado de entender (la nota promedio fue de un 50%)

IV. Wodel-Test

Generación de herramientas de pruebas de mutación

Motivación

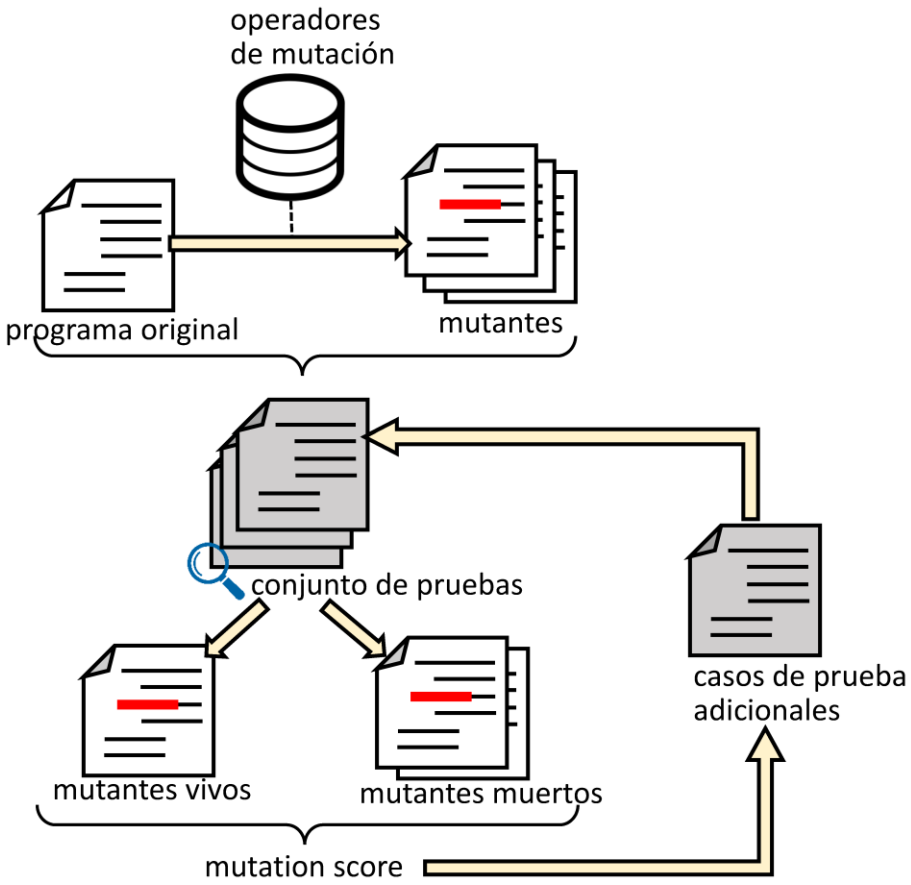
- ✓ Las herramientas de pruebas de mutación son
  - específicas para un lenguaje
  - se desarrollan manualmente
  - alto coste de mantenimiento

Se propone el entorno **Wodel-Test** - extensión a Wodel para la generación de herramientas de pruebas de mutación

Se implementan dos herramientas de pruebas de mutación

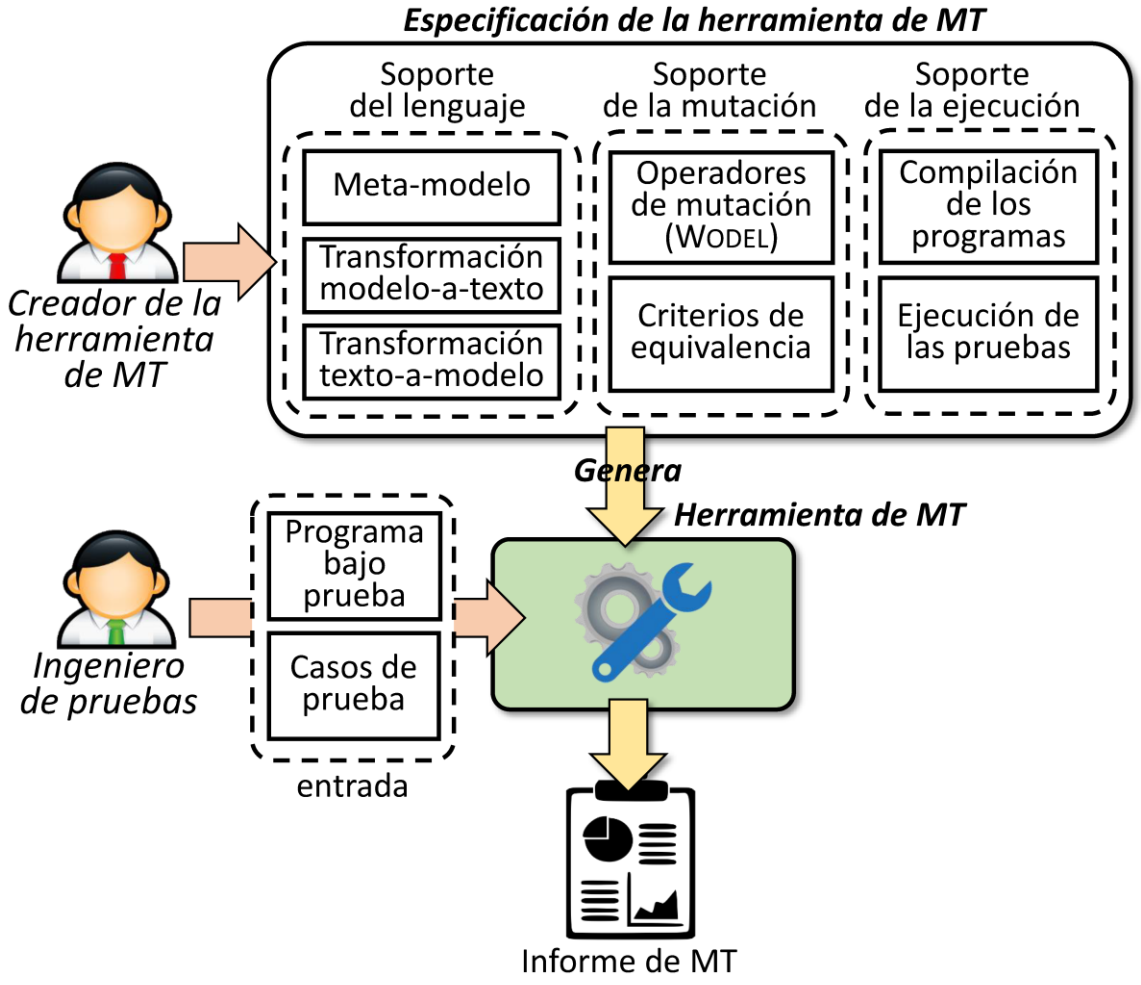
Wodel-Test para Java y Wodel-Test para ATL

¿Qué son las pruebas de mutación?

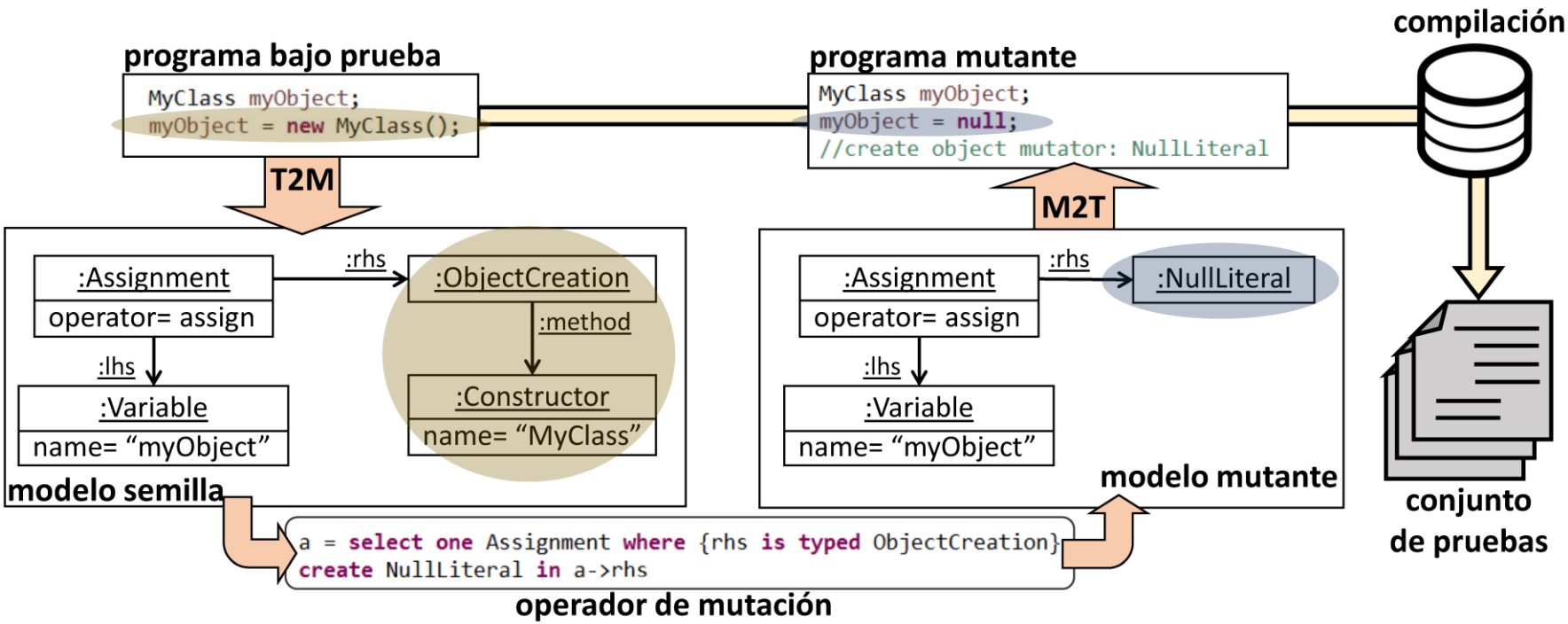


- ✓ Enfoque de pruebas de software para evaluar la calidad de los conjuntos de pruebas
- ✓ Introducción de cambios sintácticos en un programa mediante operadores de mutación
- ✓ Las mutaciones introducidas emulan fallos comunes de programación
- ✓ Facilita mejorar la calidad de los conjuntos de pruebas

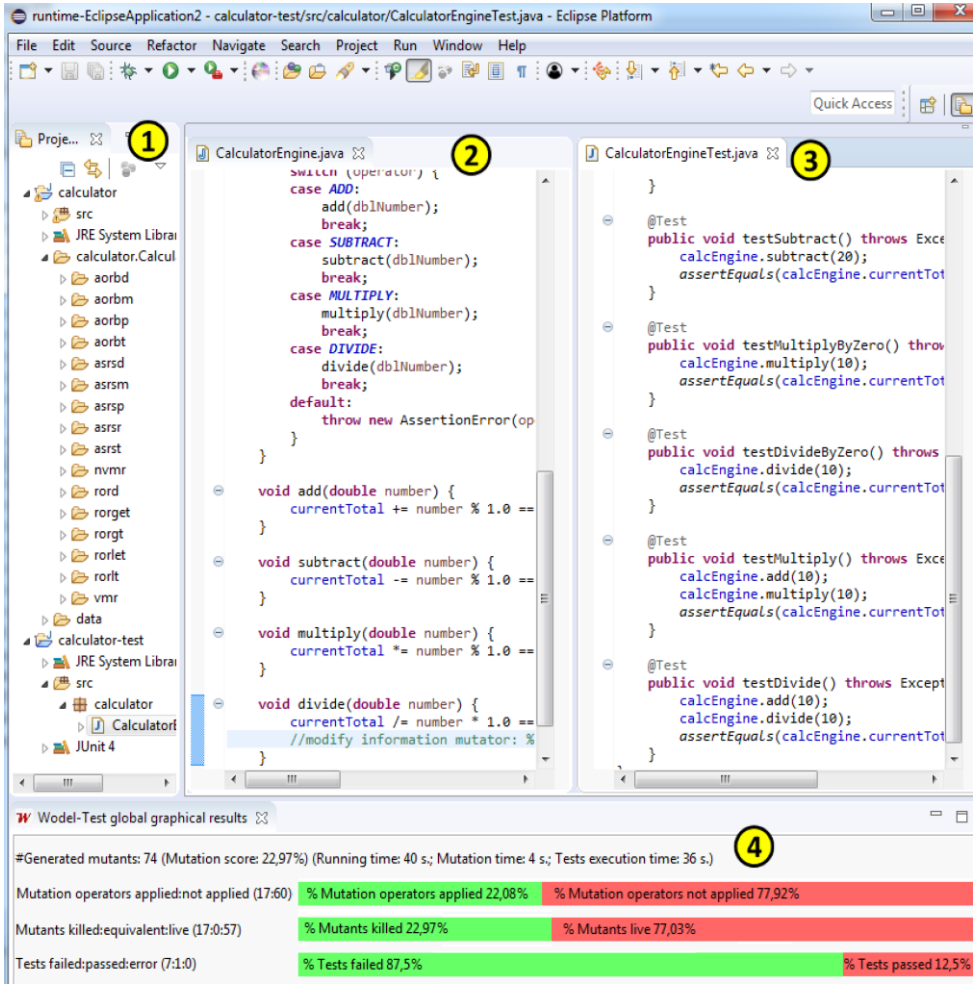
Proceso de Wodel-Test



Herramienta de pruebas de mutación para Java



Herramienta de pruebas de mutación generada



1) Explorador de proyectos

2) Ejemplo de programa mutante

3) Conjunto de pruebas

4) Resultados globales del proceso de pruebas de mutación

Evaluación de Wodel-Test

RQ1

¿Permite Wodel-Test crear herramientas de pruebas de mutación con capacidades similares a las herramientas de pruebas de mutación creadas manualmente?

✓Comparativa de Wodel-Test para Java con herramientas existentes de pruebas de mutación para Java

RQ2

¿Wodel-Test es efectivo para especificar herramientas de pruebas de mutación?

✓Implementación de Wodel-Test para ATL con los operadores de mutación introducidos en Troya et al.[1]

[1] Troya, J., Bergmayr, A., Burgueño, L. y Wimmer, M.: Towards systematic mutations for and with ATL model transformations. En International conference on software testing, verification and validation workshops (ICSTW), págs. 1–10, 2015

Evaluación de Wodel-Test RQ1

	Major	Javalanche	PITest	LittleDarwin	Wodel-Test/Java
N. de operadores	30 (por defecto)	19	40	28	77 (por defecto)
Extensibilidad de ops.	Sí (DSL)	No	API	No	Sí (DSL)
Artefacto mutado	Bytecode	Bytecode	Bytecode	AST	Modelo
Código del mutante	No	No	Sí	Sí	Sí
Detección equivalentes	No	Sí (invariantes dinámicas)	No	No	Sí (TCE)
Tipo de informe	CSV	HTML	HTML	HTML	Vistas interactivas
Número de mutantes	✓	✓	✓	✓	✓
Mutation score	✓	✓	✓	✓	✓
Mutantes muertos/vivos	Número	Número	Número	Número	Número, lista
Cobertura de operadores	Número	Número	Número	✗	Número, lista
Mutantes por clase	✗	✗	✗	✓	✓
Pruebas por mutante	✗	✗	✗	✗	✓
Mutantes por prueba	✗	✗	✗	✗	✓

Evaluación de Wodel-Test RQ1

✓Aplicación del proceso de pruebas de mutación sobre el proyecto functional-matrix[1]:

Herramienta	Mutantes (muertos/vivos)	Mutation score	Tiempo ejecución	Tiempo por mutante
Major	1638 (331/864)	20,21%	11h 21min 40seg	24,99seg
PITest	918 (321/597)	34,97%	56min 9seg	3,67seg
LittleDarwin	439 (130/309)	29,61%	2h 45min 27seg	22,61seg
Wodel-Test/Java	4756 (985/3771)	20,71%	2h 40min 27seg	2,02seg

[1]https://github.com/soursop/functional-matrix-operator

Evaluación de Wodel-Test RQ2

- ✓ Wodel-Test para ATL con los operadores de mutación introducidos en el trabajo de Troya et al.

- ✓ El prototipo de Troya et al. es un generador de mutantes

- ✓ Wodel-Test para ATL proporciona la funcionalidad completa de una herramienta de pruebas de mutación para este lenguaje

Evaluación de Wodel-Test RQ2

- ✓ Operadores de mutación para ATL

Concepto	Operador de mutación	LOC en Wodel	LOC en ATL
Regla	Creación Borrado Cambio de nombre	1 1 1	- - -
Elemento de patrón de entrada	Creación Borrado Cambio de tipo Cambio de condición	6 1 4 1	14 - - -
Filtro de regla	Creación Borrado Cambio de condición	10 1 4	- - -
Elemento de patrón de salida	Creación Borrado Cambio de tipo Cambio de nombre	6 1 4 1	- 6 - -
Binding	Creación Borrado Cambio de valor Cambio de propiedad	6 1 2 6	- 3 - -

Conclusiones de la evaluación

- ✓ Wodel-Test proporciona una funcionalidad comparable a la de las herramientas de pruebas de mutación existentes para Java

- ✓ Wodel-Test puede ser una mejor opción:
  - Acceso al código fuente de los mutantes
  - Razonar sobre qué mutantes reducen el mutation score y por qué
  - Experimentar con nuevos operadores de mutación

- ✓ Las herramientas de pruebas de mutación existentes requieren dos órdenes de magnitud más de código que la especificación de un entorno similar utilizando Wodel-Test

V. Conclusiones y trabajo futuro

Conclusiones

- ✓ La mutación está presente en múltiples dominios en la ingeniería del software
- ✓ Las soluciones existentes son específicas para cada caso
- ✓ Se ha propuesto **Wodel** que es un DSL para mutación de modelos
- ✓ Se han hecho dos aplicaciones:
  - Generación de ejercicios
  - Pruebas de mutación

Trabajo futuro

- ✓ Servicios de Wodel
  - generación de modelos near misses: verificación de la implementación correcta de los operadores de mutación
  - técnicas de análisis estático para detectar conflictos y dependencias entre los operadores
- ✓ Nuevas aplicaciones de Wodel
  - ingeniería basada en búsqueda
- ✓ Wodel-Edu
  - entornos de aprendizaje más complejos (gamificación)
  - ejercicios (p. ej., que sean interactivos)
  - diferentes plataformas (móviles o tablets)
  - ejercicios para Moodle
- ✓ Wodel-Test
  - abordar programas más grandes
  - optimizaciones del proceso de pruebas de mutación
  - estudio con usuarios para analizar la usabilidad

Publicaciones I

Revistas (3):

- P. Gómez-Abajo, E. Guerra, y J. de Lara. ***A domain-specific language for model mutation and its application to the automated generation of exercises***. **Computer Languages, Systems & Structures**, 49:152 – 173, 2017. Elsevier. Índice de impacto: JCR 2017: 1,840. Q2 en Computer Science / Software Engineering

- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***A tool for domain independent model mutation***. **Science of Computer Programming**, 163:85–92, 2018. Elsevier. Índice de impacto: JCR 2018: 1,088 Q3 en Computer Science / Software Engineering

- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***Systematic Engineering of Mutation Operators***. **Journal of Object Technology Special Issue dedicated to Martin Gogolla**

Revistas en proceso de revisión (1):

- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***A model-based framework for language-independent mutation testing***. **Software and Systems Modeling**. Springer. Índice de impacto: JCR 2018: 2,660. Q1 en Computer Science / Software Engineering. En segunda ronda de revisión

Publicaciones II

Congresos internacionales y workshops (4):

- P. Gómez-Abajo, E. Guerra, y J. de Lara. ***Wodel: a domain-specific language for model mutation***. 31st ACM/SIGAPP Symposium on Applied Computing, SAC, páginas 1968–1973. ACM, 2016. Porcentaje de aceptación: 24,07%
- P. Gómez-Abajo. ***A DSL for model mutation and its application to different domains***. Doctoral Symposium at the 19th ACM/IEEE International Conference of Model-Driven Engineering Languages and Systems, MoDELS. ACM/IEEE, 2016
- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***Mutation Testing for DSLs (Tool Demo)***. 17th ACM SIGPLAN International Workshop on Domain-Specific Modeling, DSM, páginas 60–62. ACM, 2019
- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***Seed Model Synthesis for Testing Model-based Mutation Operators***. 32nd International Conference on Advanced Information Systems Engineering, CAiSE Forum. ACM, 2020

Congresos nacionales (1):

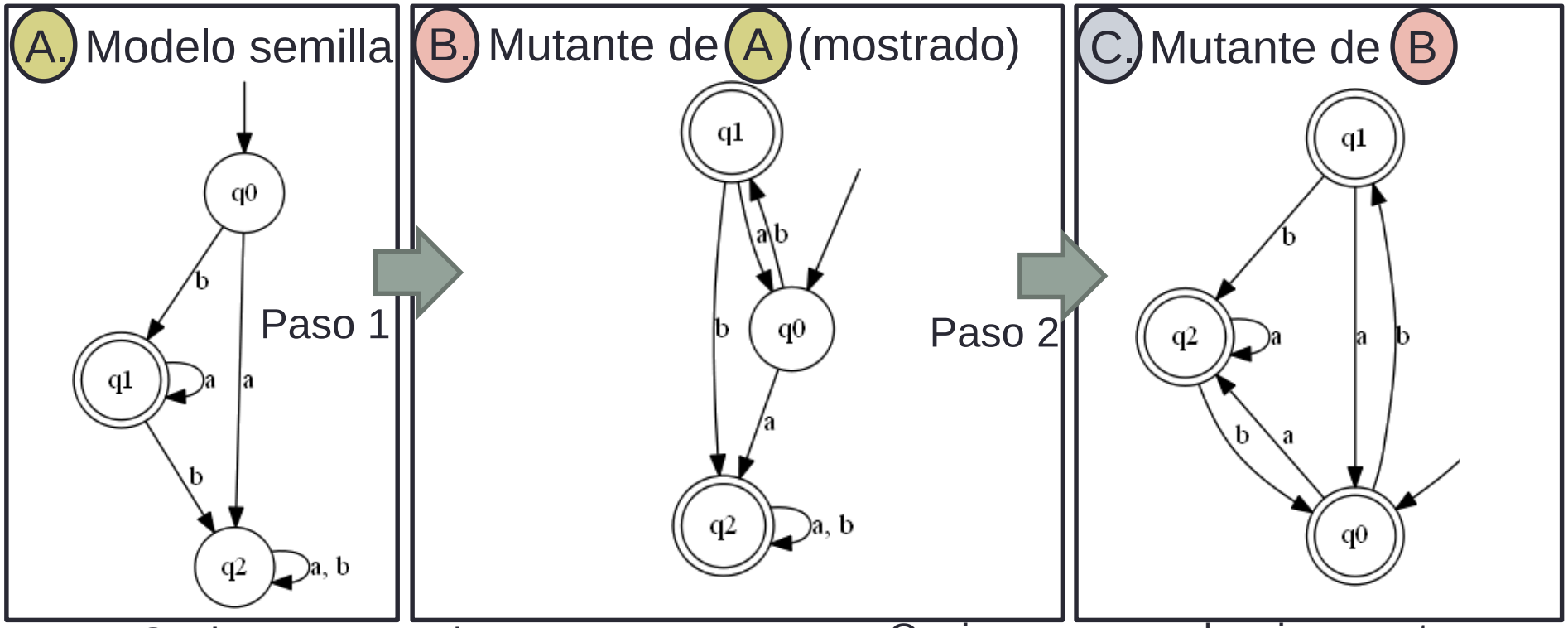
- P. Gómez-Abajo, E. Guerra, J. de Lara, y M. G. Merayo. ***Towards a model-driven engineering solution for language independent mutation testing***. En Jornadas de Ingeniería del Software y Bases de Datos (JISBD), página 4pps. Biblioteca digital SISTEDES, 2018

Presentaciones en cursos y seminarios internacionales (1):

- P. Gómez-Abajo, E. Guerra, y J. de Lara***. Wodel: a DSL for model mutation; and Wodel- Edu: its application to the automated generation of exercises***. En 7th International Summer School on Domain-Specific Modeling, DSM-TP, 2016


Gracias

Selección de opciones de texto



- Opciones generadas: correctas

1. Retype final state q2 as common state
2. Change transition a from state q1 to state q0 with new target state q1

- Opciones generadas: incorrectas

1. Retype initial state q0 as initial final state
2. Change transition b from state q2 to state q2 with new target state q0

Mutaciones de los pasos 1 y 2:

```
modify target tar from one Transition to other State
retype one CommonState as FinalState
```

Arquitectura de Wodel-Test