XI Jornadas de Innovación y Calidad del Software (JICS'09)
3 de setiembre ,España

Generación automática de casos de prueba para Líneas de Producto Software

Beatriz Pérez Lamancha
Centro de Ensayos de Software (CES)
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
bperez@fing.edu.uy

Macario Polo Usaola
Grupo ALARCOS
Departamento de Tecnologías y Sistemas de Información,
Universidad de Castilla-La Mancha, Ciudad Real, España
{macario.polo}@uclm.es

Agenda

- Introducción
  - Línea de Producto Software (LPS)
  - Perfil de Pruebas de UML
  - Modelo Ortogonal de Variabilidad
  - Ejemplo: Línea de Juegos de Mesa
- Propuesta
  - Marco Automatizado para pruebas en LPS
- Conclusiones
- Trabajo futuro

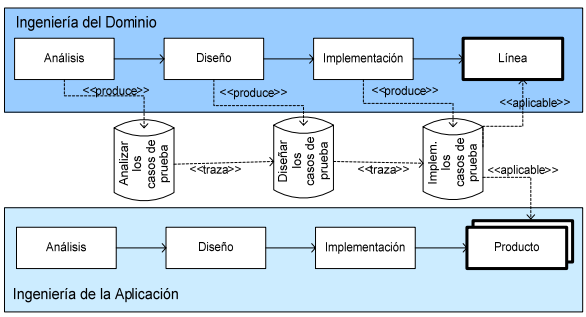
Líneas de Producto Software

- Consiste de varios sistemas que,
  - Comparten la misma arquitectura y activos base (core assets)
  - varían en las características que brindan

- Ejemplo: **Nokia Mobile Phones**
  - Sacado de la página del SEI
  - 25 a 30 nuevos productos por año
  - Los productos varían en:
    - Cantidad de teclas
    - Tamaño de la pantalla
    - Características que brindan
    - Lenguajes que soportan
    - ….

Variabilidad en la LPS

- Los productos de la línea comparten un conjunto de características comunes
- Difieren en determinados puntos de variación (PV), que representan la variabilidad entre los productos.

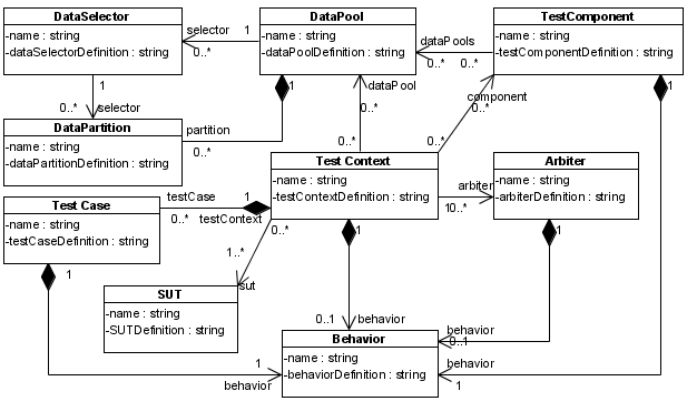


Pruebas dirigidas por modelos

- La ingeniería dirigida por modelos (IDM) utiliza modelos software para representar los elementos de un sistema y sus relaciones
- Las pruebas dirigidas por modelos (Model-driven testing) requieren la derivación sistemática y en lo posible automatizada de las pruebas a partir de modelos

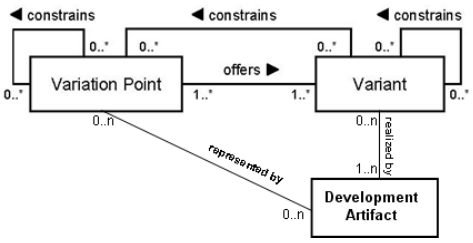
Perfil de Pruebas de UML (UML-TP)

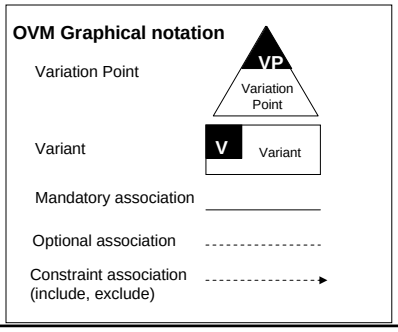
- Extiende UML 2.0 con conceptos específicos para las pruebas



Modelo Ortogonal de Variabilidad (OVM)

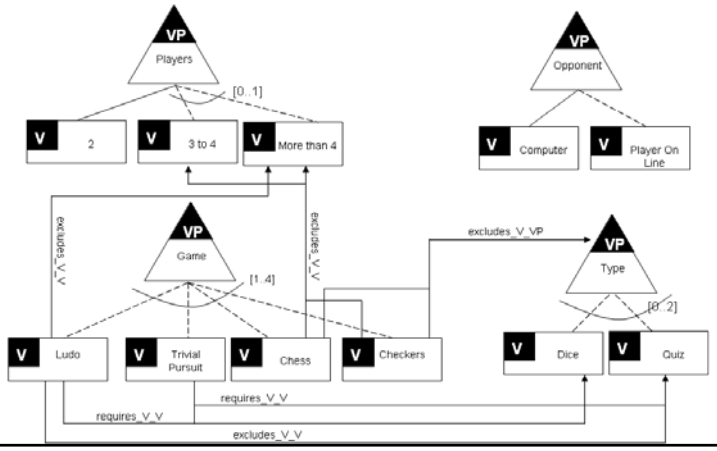
- Propuesto por Pohl et al. en el 2005
- La información sobre la variabilidad se guarda en un modelo separado
- Asocia los PV y sus variantes
  - Restringe las asociaciones entre ellos (include o exclude)





Ejemplo: LPS de Juegos de Mesa

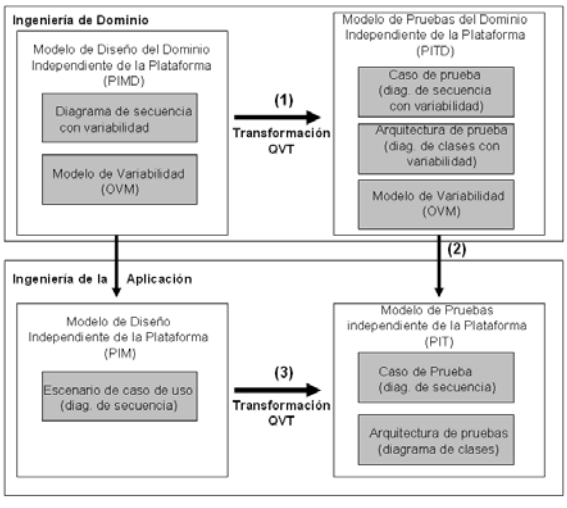
- La LPS permite cuatro tipos de juegos de mesa: Ajedrez, Damas, Parchís y Trivial.



Propuesta

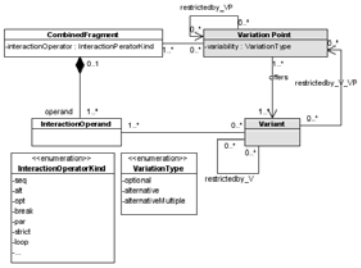
- Estrategia para la generación de casos de prueba para LPS donde:
  - se gestiona la variabilidad en los artefactos de prueba y
  - se mantiene la trazabilidad al resto de los artefactos de desarrollo de la LPS.
- En trabajos anteriores se describe cómo se pueden generar modelos de prueba basados en el Perfil de pruebas de UML (UML-TP) en forma automática a partir de diagramas de secuencia para desarrollo de software convencional, utilizando como lenguaje de transformación entre modelos Query/View/Transformation (QVT).
- En este trabajo se extienden los trabajos anteriores a las pruebas en LPS, gestionando la variabilidad en los modelos de prueba.

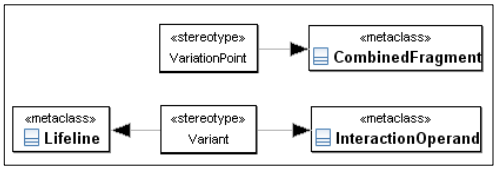
Marco automatizado para pruebas dirigidas por modelos



Extensiones a los metamodelos

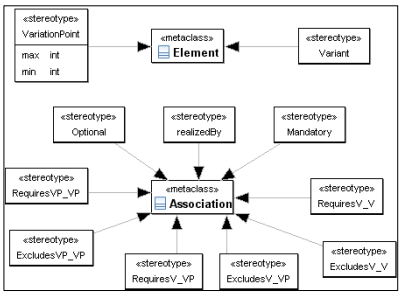
- Para gestionar la variabilidad se realizaron las siguientes extensiones:
  - Diagrama de secuencia con variabilidad
    - Extensión al CombinedFragment en los diag. de secuencia





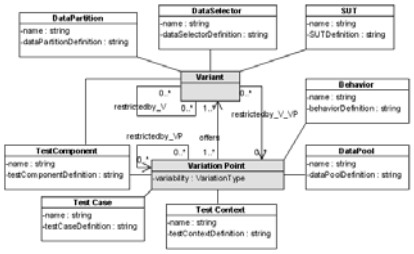
Extensiones a los metamodelos

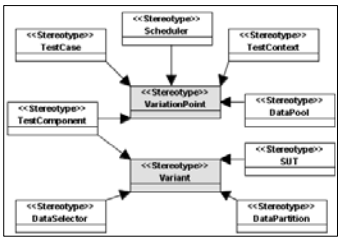
- Para gestionar la variabilidad se realizaron las siguientes extensiones:
  - Diagrama de secuencia con variabilidad
  - Perfil UML para el Modelo Ortogonal de Variabilidad



Extensiones a los metamodelos

- Para gestionar la variabilidad se realizaron las siguientes extensiones:
  - Diagrama de secuencia con variabilidad
  - Perfil UML para el Modelo Ortogonal de Variabilidad
  - Perfil de Pruebas UML con variabilidad

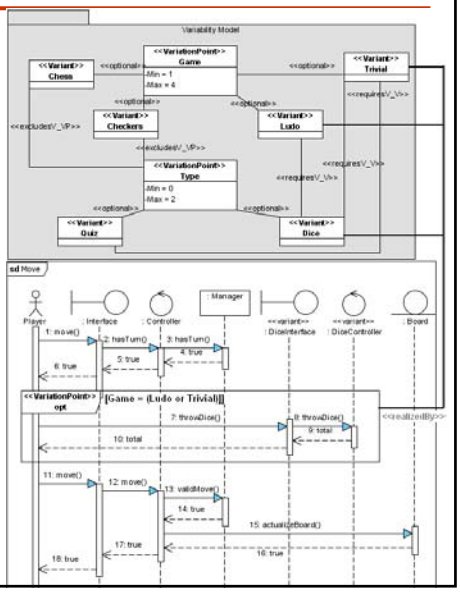




Extensiones a OVM y Diagrama de secuencia

- Modelo OVM siguiendo Perfil UML

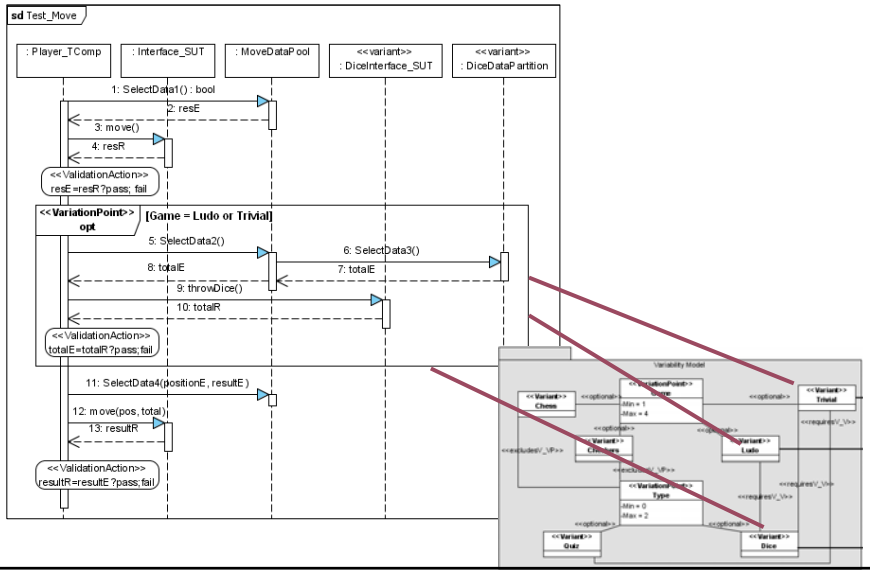
- Diagrama de secuencia con extensión para LPS :
  - Mover pieza



Pasos para la generación automática del caso de prueba con UML-TP

- Obtener los datos de prueba
  - El *testComponent* invoca el *DataSelector* en el *dataPool* que retorna los datos necesarios para probar la funcionalidad.
- Ejecutar el caso de prueba en el SUT
  - El *testComponent* simula al actor y éste a su vez invoca la funcionalidad a probar en el *SUT*.
- Obtener el resultado del caso de prueba
  - El *testComponent* es el responsable de comprobar si el resultado retornado por el SUT es correcto, para esto utiliza las acciones de validación (*ValidationActions*) que son las encargadas de informar al árbitro (*arbiter*) el resultado de la prueba.

Generación automática del Caso de Prueba



Conclusiones y trabajo a futuro

- Se ha presentado un marco para las pruebas dirigidas por modelos aplicable a Líneas de Producto Software
- Se basa en estándares:
  - UML 2.0
  - Perfil de Pruebas de UML (UML-TP)
  - Lenguaje de transformación Query/View/Transformation (QVT).
- Mantiene la trazabilidad entre los distintos modelos en LPS:
  - Modelo Ortogonal de Variabilidad (OVM) como un Perfil de UML.
- Trabajo a futuro
  - Derivar los modelos de prueba de cada producto a partir de los modelos de prueba de la LPS

¿Preguntas?
Gracias por su atención

Beatriz Pérez Lamancha
Centro de Ensayos de Software (CES)
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
bperez@fing.edu.uy

Macario Polo Usaola
Grupo ALARCOS
Departamento de Tecnologías y Sistemas de Información,
Universidad de Castilla-La Mancha,
Ciudad Real, España
{macario.polo}@uclm.es