CostoTest (un état des lieux)

Pascal ANDRE, Jean-Marie Mottu

AELOS / LINA – UMR CNRS 6241
{Firstname.Lastname}@univ-nantes.fr

Exposés AeLoS

Plan de l'exposé

Plan

Introduction / contexte et motivations

Contexte : Test d'applications à composants et services logiciels distribués

- Cible : Niveau modèle vs. niveau code
- Exécutabilité, animation
- Assistance à la construction des tests (harnais)
- Gestion de l'environnement de tests
- Partage du support de développement - banalisation
- Support : COSTO/Kmelia

$\Longrightarrow$ revoir l'exposé exposé Gencode (`gencode.pdf`) : contexte, bibliographie...

Introduction / cible

Test de modèles

- détection des erreurs en amont (moins coûteux)
- abstraction des contraintes de programmation et de déploiement (PIM)
- évolutivité et réactivité dans les premières itérations du développement
- généricité du test (unitaire, intégration, recette)
- abstraction du test (xUnit $\equiv$ JUnit $\cup$ SUnit $\cup$ ...)
- test non intrusif (encapsulation)

Harnais (TPIM) + traçabilité

Introduction / Exécutabilité

Pouvoir exécuter des cas de tests à partir de modèles

- génération de code automatique
- enchaîner les exécutions de jeu de test pour un test donné
- évolution des programmes : pas d'intervention manuelle
- évolutivité des cas de test : réutiliser les données (XML)
- variabilité : mutation, librairies de test
- portabilité ?

Harnais génération de code + bancs de test

Introduction / Assistance

Aider le testeur à construire ses applications

- partir d'une intention, même incomplète, et d'un modèle d'application
- créer des applications de test est une activité d'ingénierie et de développement
- non intrusion dans les programmes de base
- découverte des données pour l'initialisation
- découverte des services pour l'interaction
- API ouverte de manipulation de modèles
- généricité du test (unitaire, intégration, recette)
- abstraction du test (xUnit $\equiv$ JUnit $\cup$ SUnit $\cup$ ...)
- test non intrusif (encapsulation)
- qualité via la mutation

Harnais construction + vérification

Introduction / environnement

Gestion de modèles et cas de test

- gérer jeux de données et résultats
- gérer variantes d'un test (mutation)
- gérer cohérence et couverture des tests (banc de test)
- domaine du test (unitaire, intégration, recette)
- classification, réutilisation
- évolution...

Banc de test bonnes pratiques

Introduction / banalisation

Un modèle de test = un modèle

- réutiliser TOUT l'environnement de travail (outillage) : édition, vérification, transformation, génération de code, documentation...
- un test = un composant (de l'intention au cas de test)
- évolutivité et réactivité dans les premières itérations du développement
- généricité du test (unitaire, intégration, recette)
- abstraction du test (xUnit $\equiv$ JUnit $\cup$ SUnit $\cup$ ...)
- test non intrusif (encapsulation)

Harnais (TPIM) + traçabilité

Introduction / Support

COSTO/Kmelia (boîte à outils COSTO) :

- Kmelia : modèle et langage pour composants et services logiciels
  - structure : Composants/Interfaces/Service/Architectures/Composites
  - calculs : actions
  - dynamique : LTS + communication
- Boîte à outils COSTO (Famille de plugins Eclipse)
  - Editions, vérification langage
  - Vérification de propriétés de modèles (structure, dynamique, fonctionnels)
  - Génération de code pour composants et services distribués
    - Framework plain Java (Class, threads, buffers)
    - traçabilité
    - exécution, animation (Sémantique opérationnelle)
  - Documentation
- Wiki, forge

COSTO/Kmelia Maîtrise de l'environnement

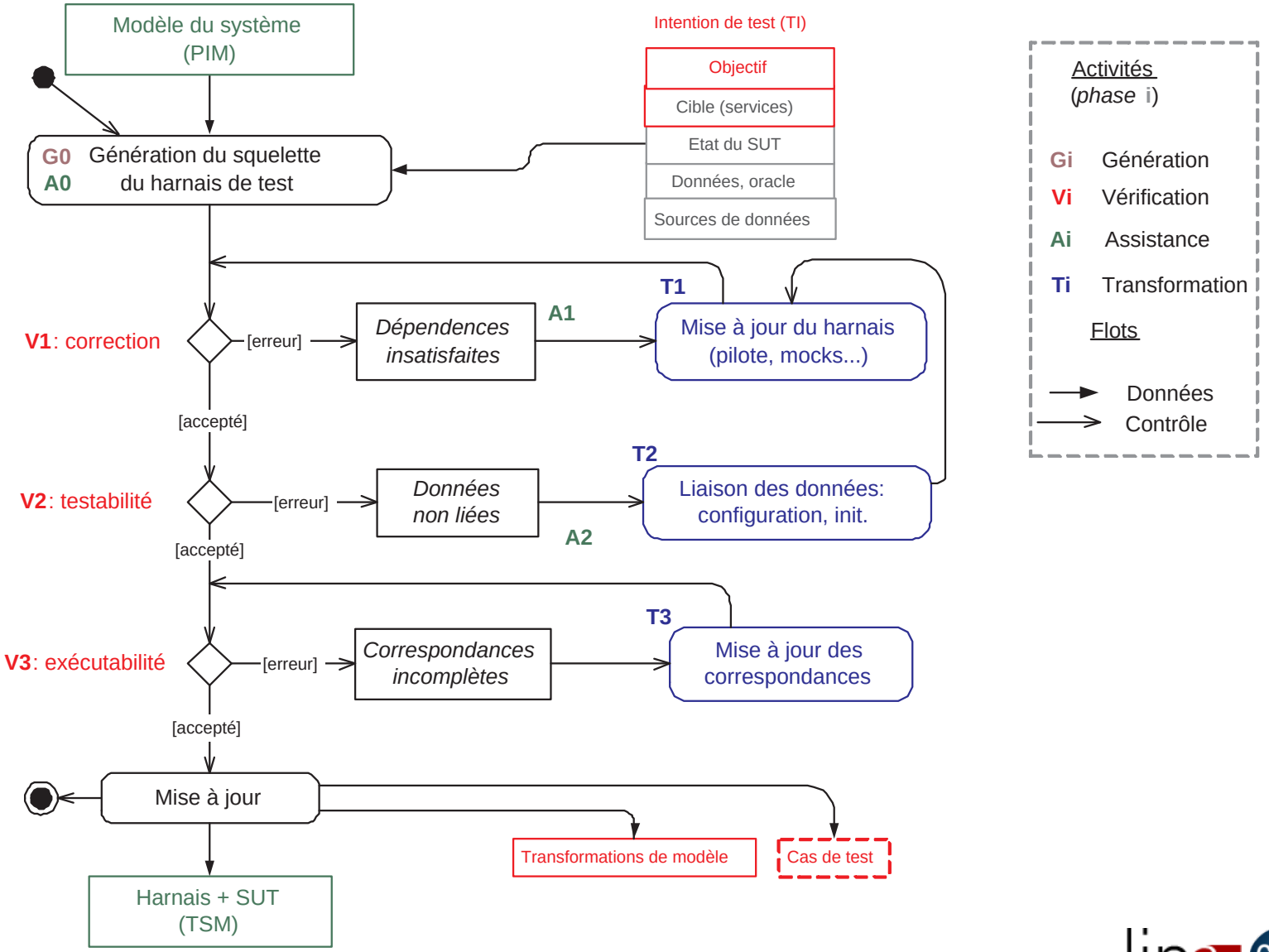
Introduction / focus

CostoTest :

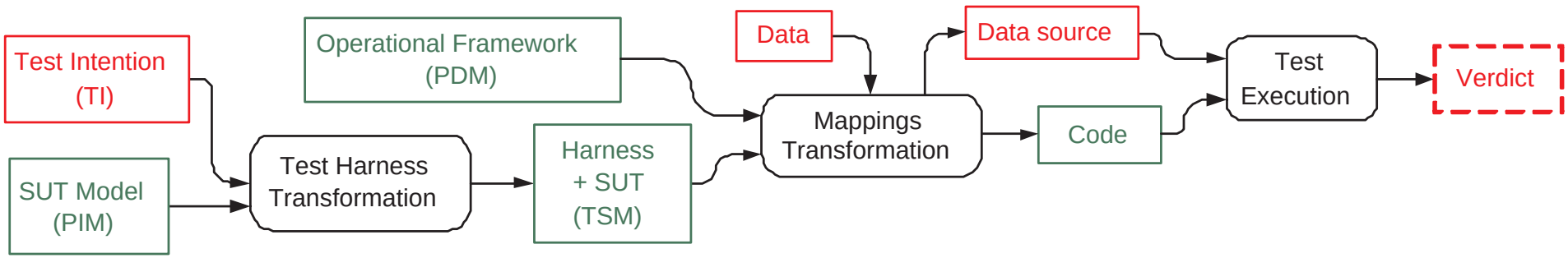
- Modèle, processus
  - Assistance au test de modèles à composants et services [**?**]
  - Building Test Harness From Service-based Component Models [**?**]
- Outils - COSTOTest : A Tool for Building and Running Test Harness for Service-based Component Models
  - modèle de test
  - processus complet implanté
  - exécutabilité (données en XML)
  - étude de cas
- Expérimentations
  - comparaison avec JUnit
  - en cours
- Forge et documentation
  - http ://www.lina.sciences.univ-nantes.fr/aelos/wiki/doku.php/costo :test :start
  - Application

Plan

CostTest - processus (spécification)

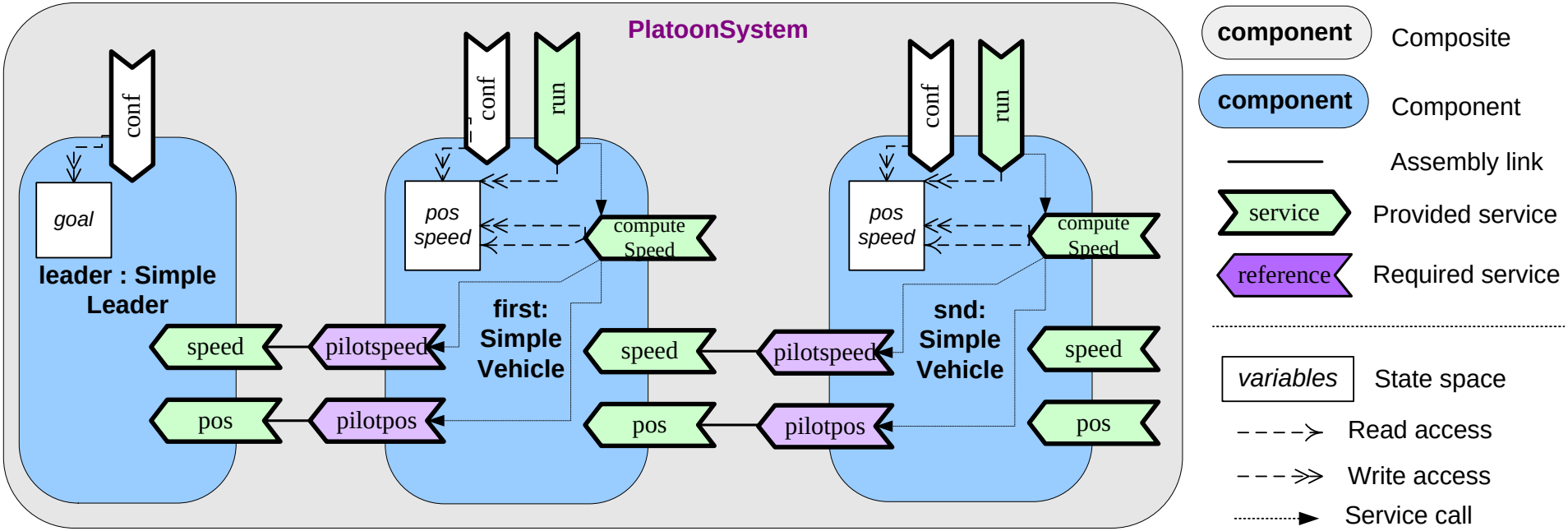


CostTest - processus (implantation)



1. construction
2. mutation (optionnel)
3. exécution

CostTest - entrées 1/2



1. assemblage de composants
2. vérification
3. exécution

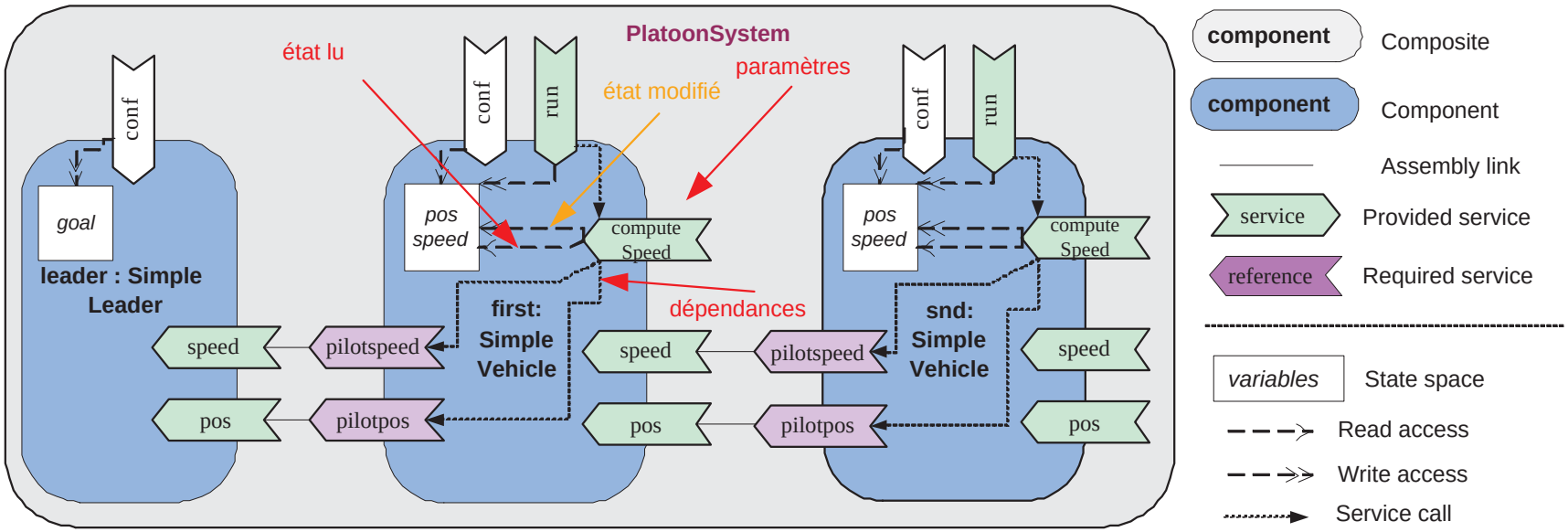
CostTest - entrées 2/2

PlatoonTestIntention.kcp

```
TEST_INTENTION PlatoonTestIntention
DESCRIPTION "test of the service computSpeed, covering control flow
USES {PLATOONTESTLIB}
INPUT VARIABLES
  lastpos:Integer;
  vspeed:Integer;
  safeDistance:Integer;
  pilotpos:Integer;
  pilotspeed:Integer;
OUTPUT VARIABLES
  speed:Integer;
  oracledata:Integer;
 ORACLE
  speed=oracledata
```

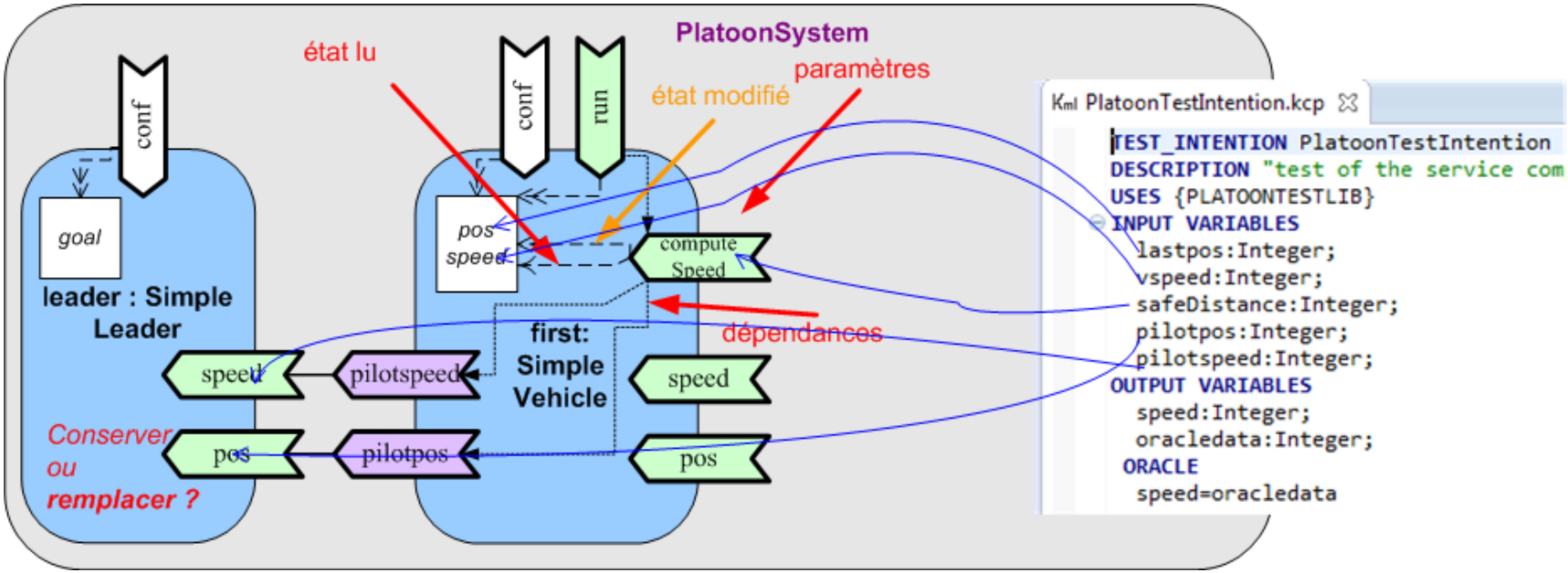
1. tester un service offert
2. déterminer le contexte
3. brancher le contexte sur les données

CostTest - fixer le périmètre 1/4



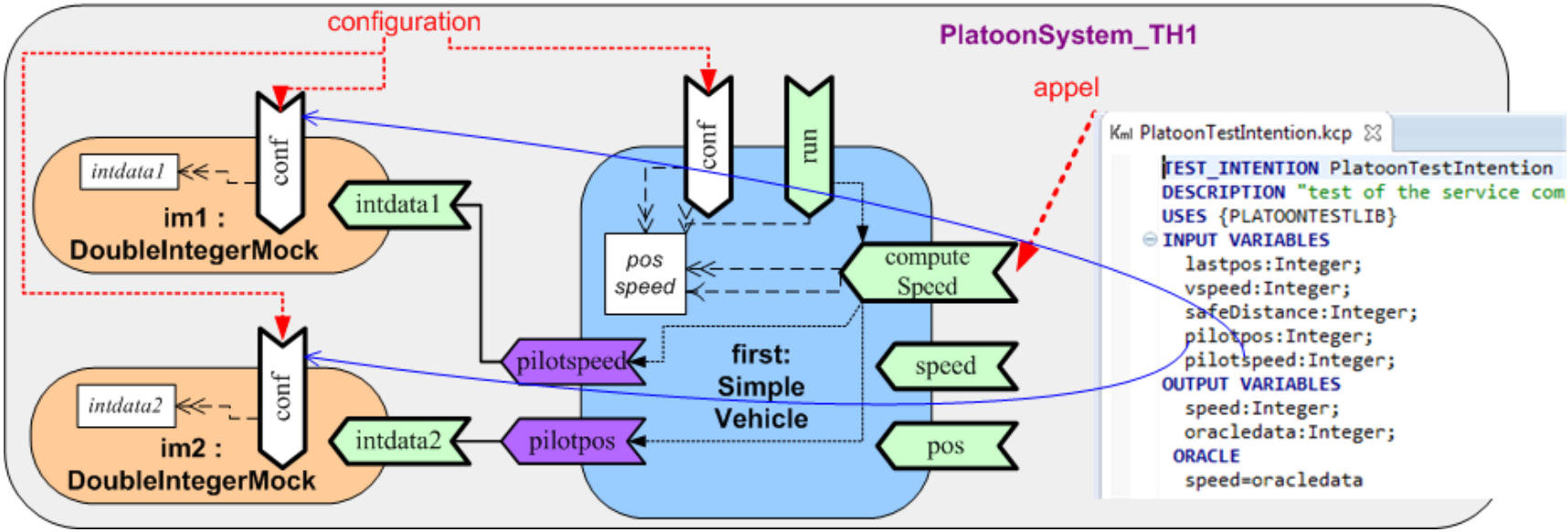
1. fixer le périmètre
2. lier à l'intention
3. cohérence, complétude

CostTest - lier à l'intention 2/4



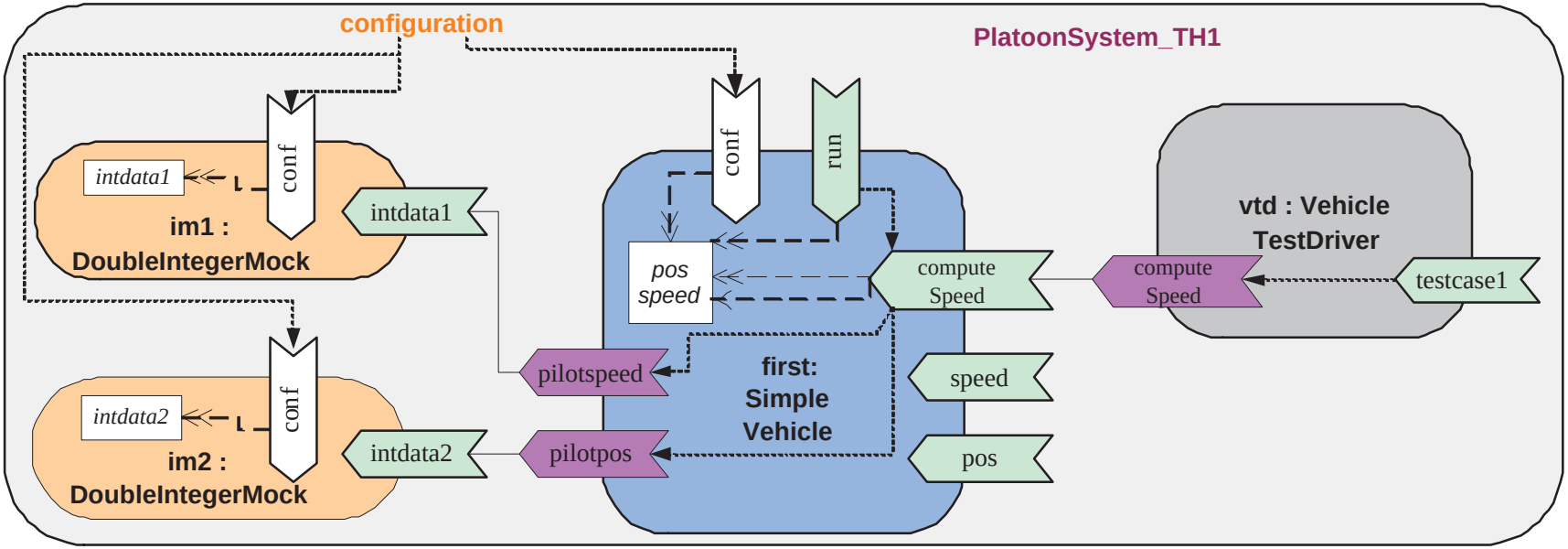
1. paramètres, variables d'états (init, accesseurs), dépendances
2. convergence intention / application de test (pourcentages)
3. cohérence, complétude

CostTest - satisfaire les besoins 3/4



1. fournisseurs de services requis
2. itérer sur l'étape 2/4 :
   paramètres, variables d'états (init, accesseurs), dépendances
3. convergence intention / application de test
4. cohérence, complétude (pourcentages)

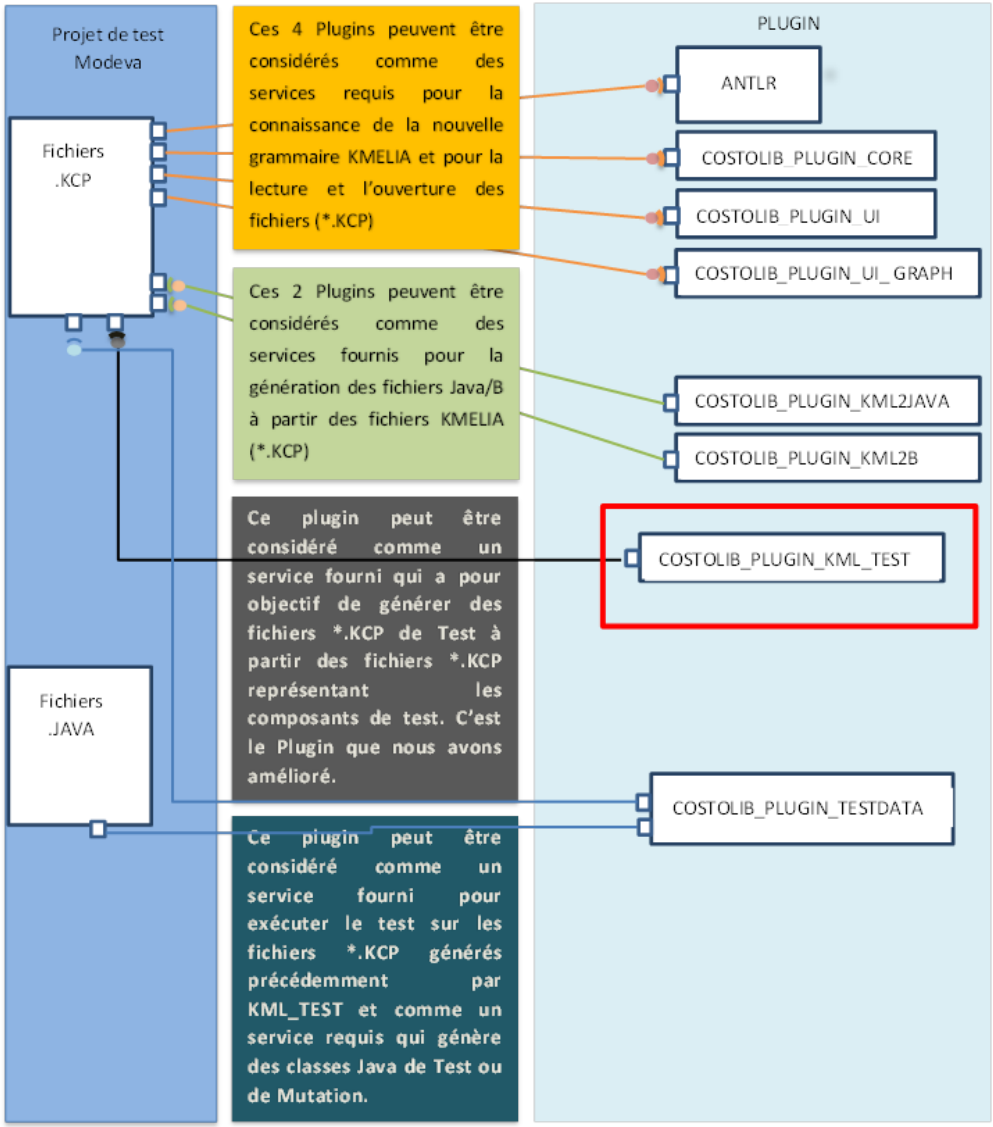
CostTest - générer le testeur 4/4



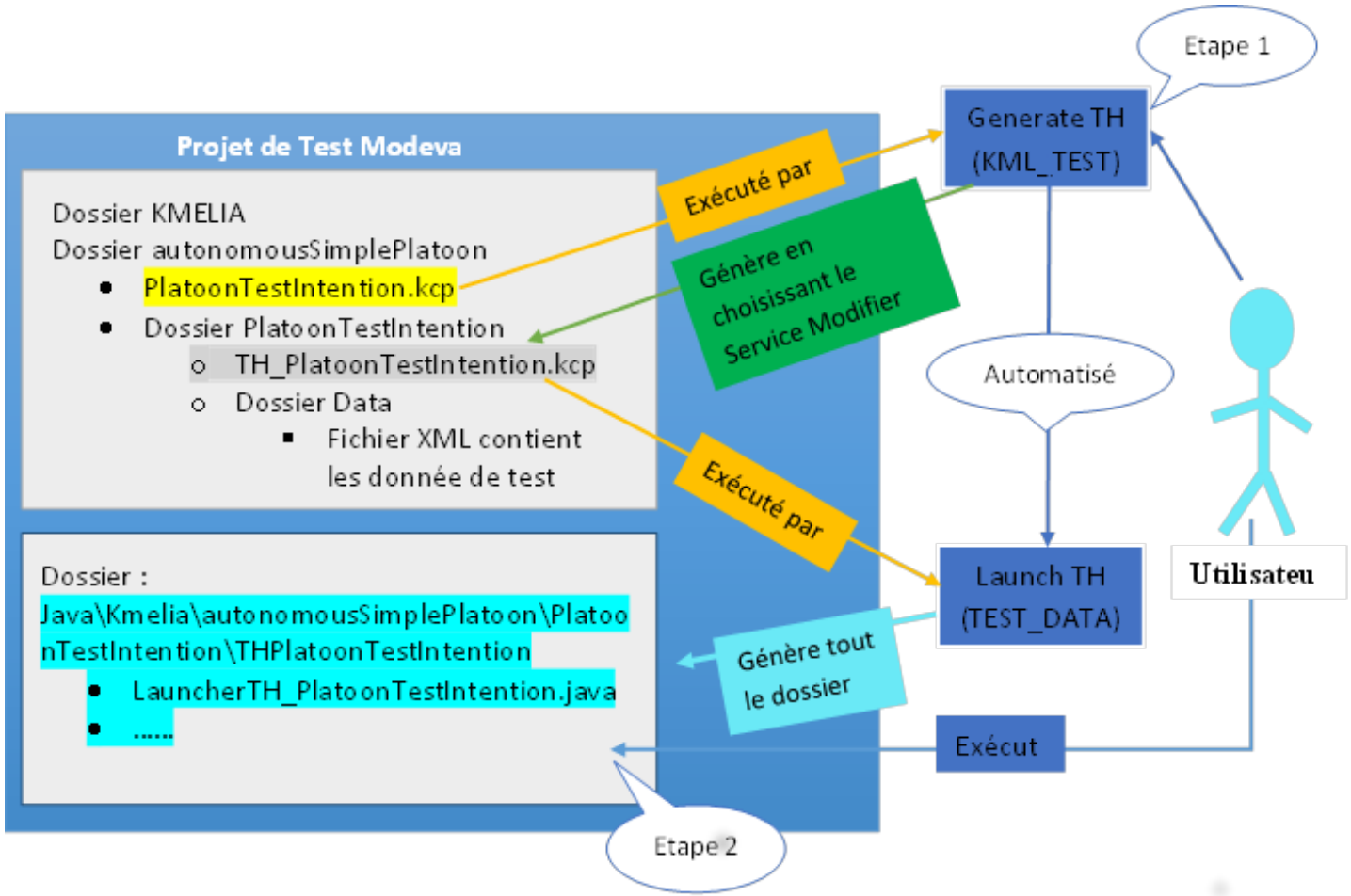
1. séquence d'initialisation
2. service de lancement du cas de test
3. application de test (composants, services, liens, configurations)

Plan

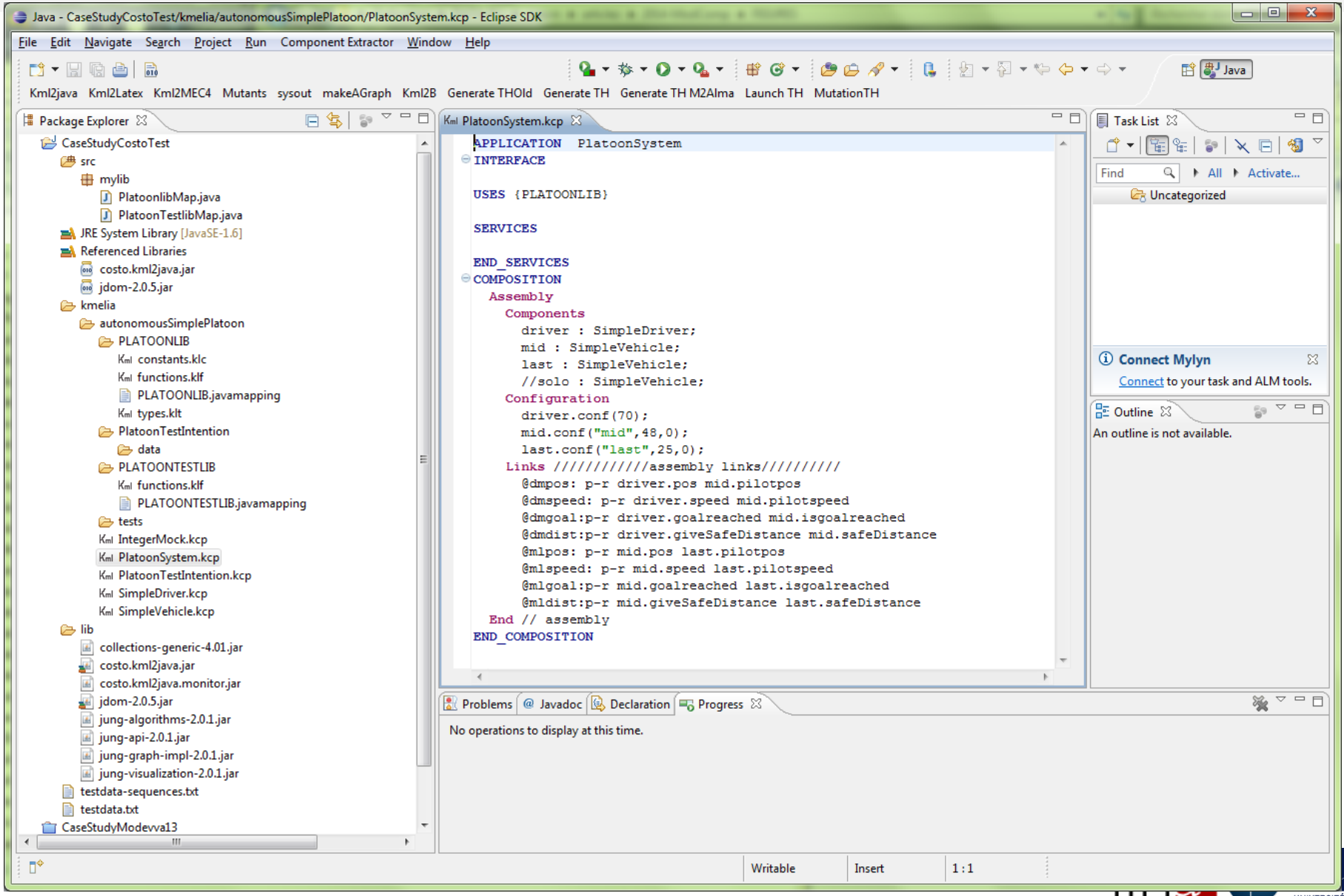
CostTest- Architecture



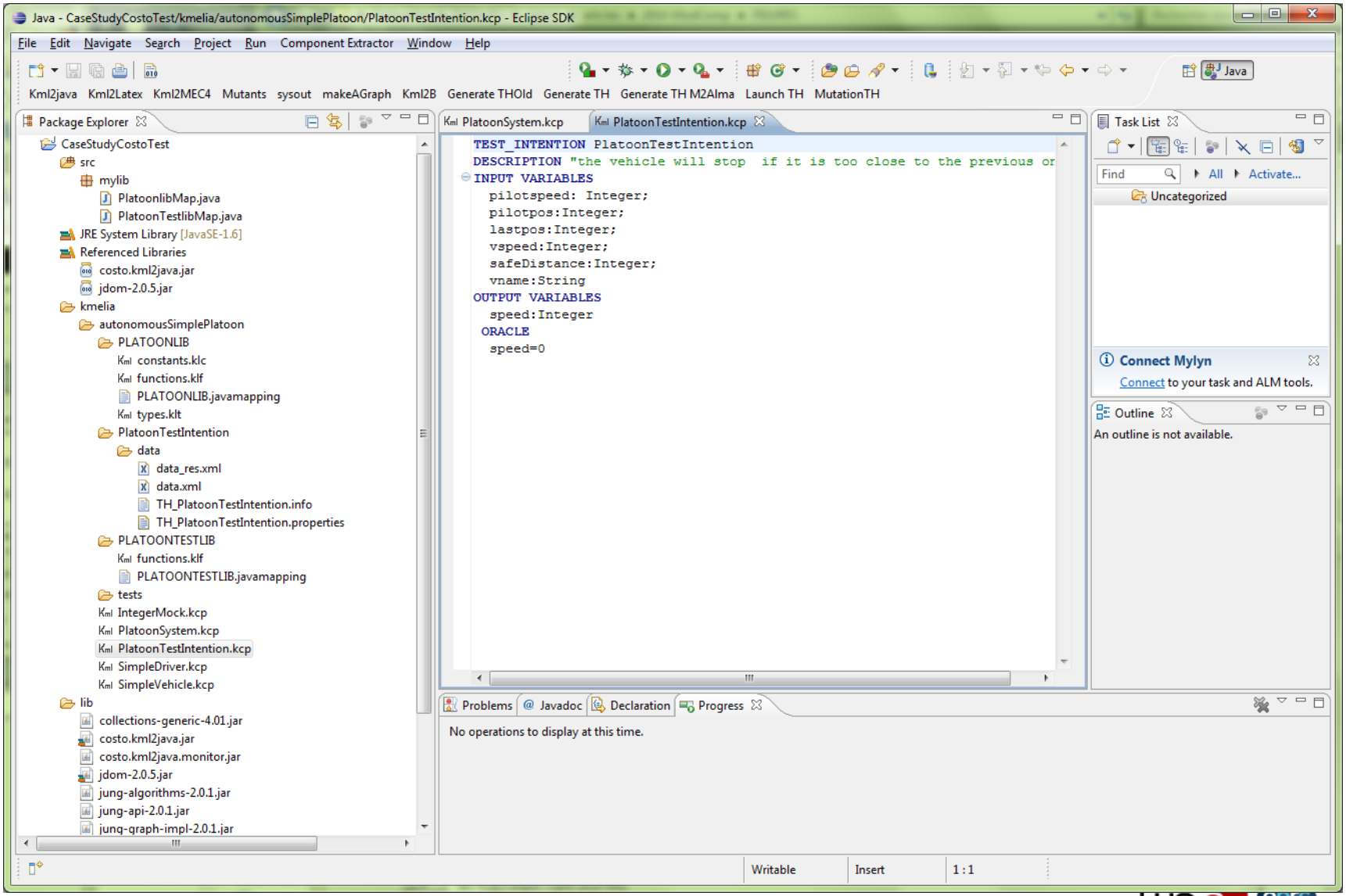
CostTest - Processus



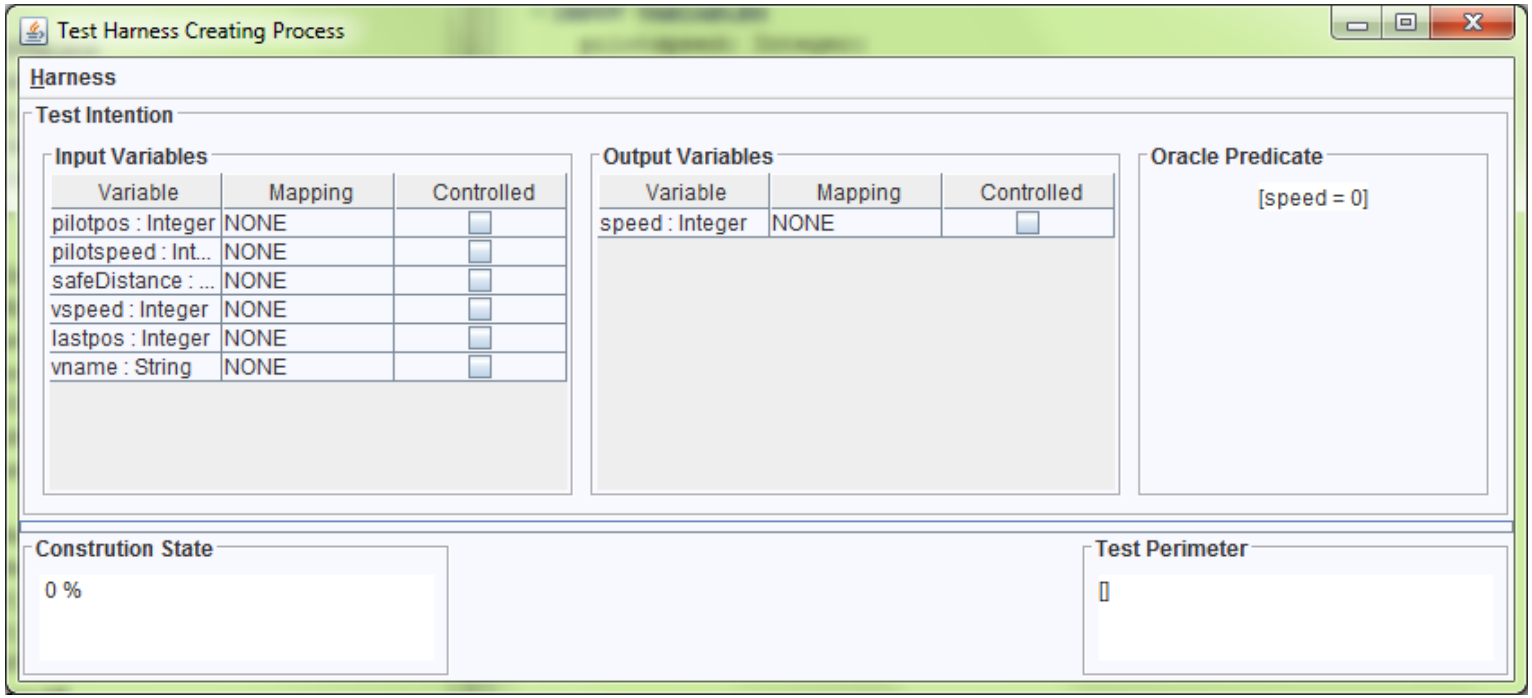
CostTest - SUT



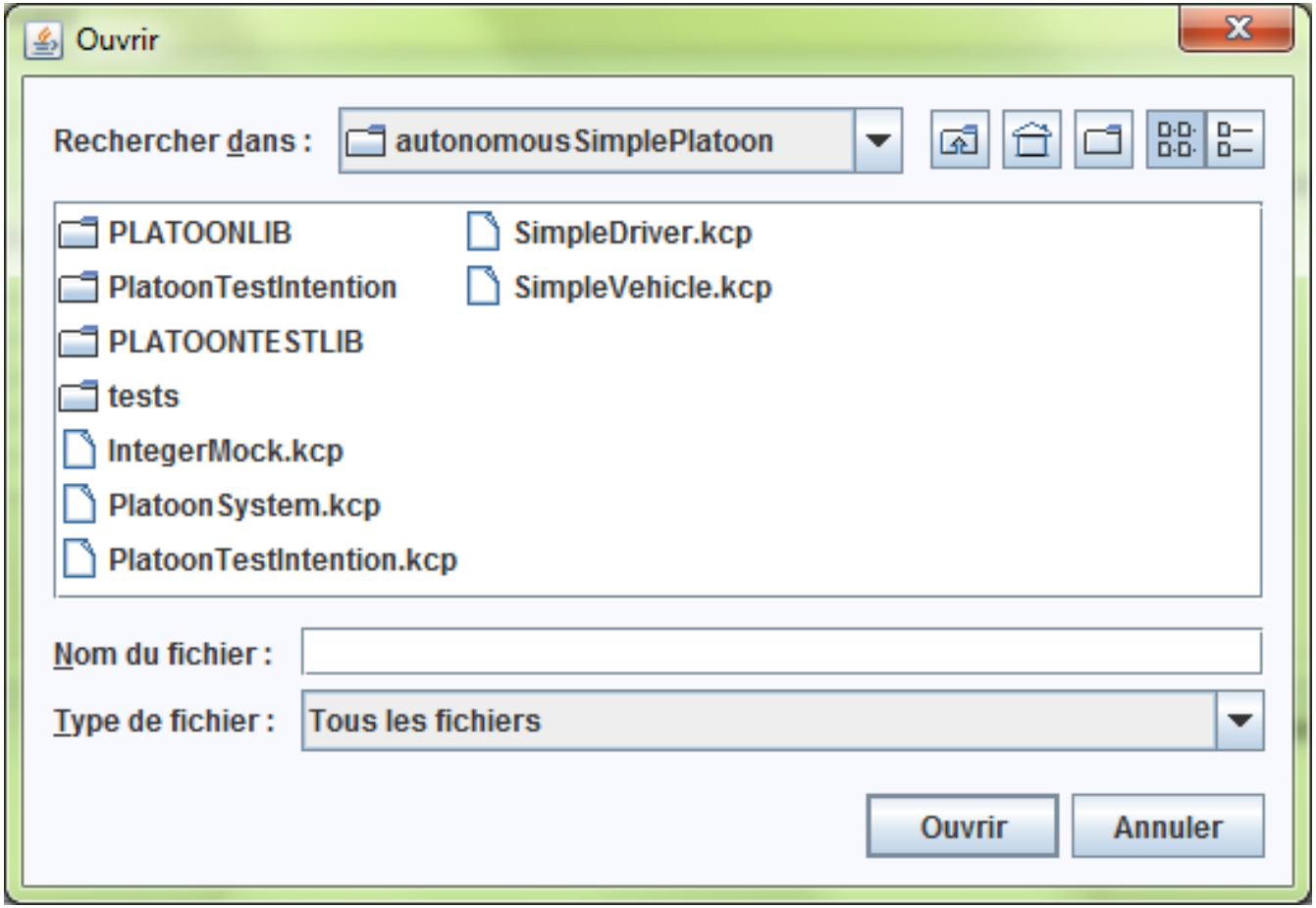
CostTest - TI


Java - CaseStudyCostoTest/kmelia/autonomousSimplePlatoon/PlatoonTestIntention.kcp - Eclipse SDK
File Edit Navigate Search Project Run Component Extractor Window Help
Kml2java Kml2Latex Kml2MEC4 Mutants sysout makeAGraph Kml2B Generate THOld Generate TH Generate TH M2Alma Launch TH MutationTH
Package Explorer
CaseStudyCostoTest
src
mylib
PlatoonlibMap.java
PlatoonTestlibMap.java
JRE System Library [JavaSE-1.6]
Referenced Libraries
costo.kml2java.jar
jdom-2.0.5.jar
kmelia
autonomousSimplePlatoon
PLATOONLIB
constants.klc
functions.klf
PLATOONLIB.javamapping
types.klt
PlatoonTestIntention
data
data_res.xml
data.xml
TH_PlatoonTestIntention.info
TH_PlatoonTestIntention.properties
PLATOONTESTLIB
functions.klf
PLATOONTESTLIB.javamapping
tests
IntegerMock.kcp
PlatoonSystem.kcp
PlatoonTestIntention.kcp
SimpleDriver.kcp
SimpleVehicle.kcp
lib
collections-generic-4.01.jar
costo.kml2java.jar
costo.kml2java.monitor.jar
jdom-2.0.5.jar
jung-algorithms-2.0.1.jar
jung-api-2.0.1.jar
jung-graph-impl-2.0.1.jar
PlatoonSystem.kcp
PlatoonTestIntention.kcp
TEST_INTENTION PlatoonTestIntention
DESCRIPTION "the vehicle will stop if it is too close to the previous or
INPUT VARIABLES
pilotspeed: Integer;
pilotpos:Integer;
lastpos:Integer;
vspeed:Integer;
safeDistance:Integer;
vname:String
OUTPUT VARIABLES
speed:Integer
ORACLE
speed=0
Task List
Find All Activate...
Uncategorized
Connect Mylyn
Connect to your task and ALM tools.
Outline
An outline is not available.
Problems Javadoc Declaration Progress
No operations to display at this time.
Writable Insert 1 : 1

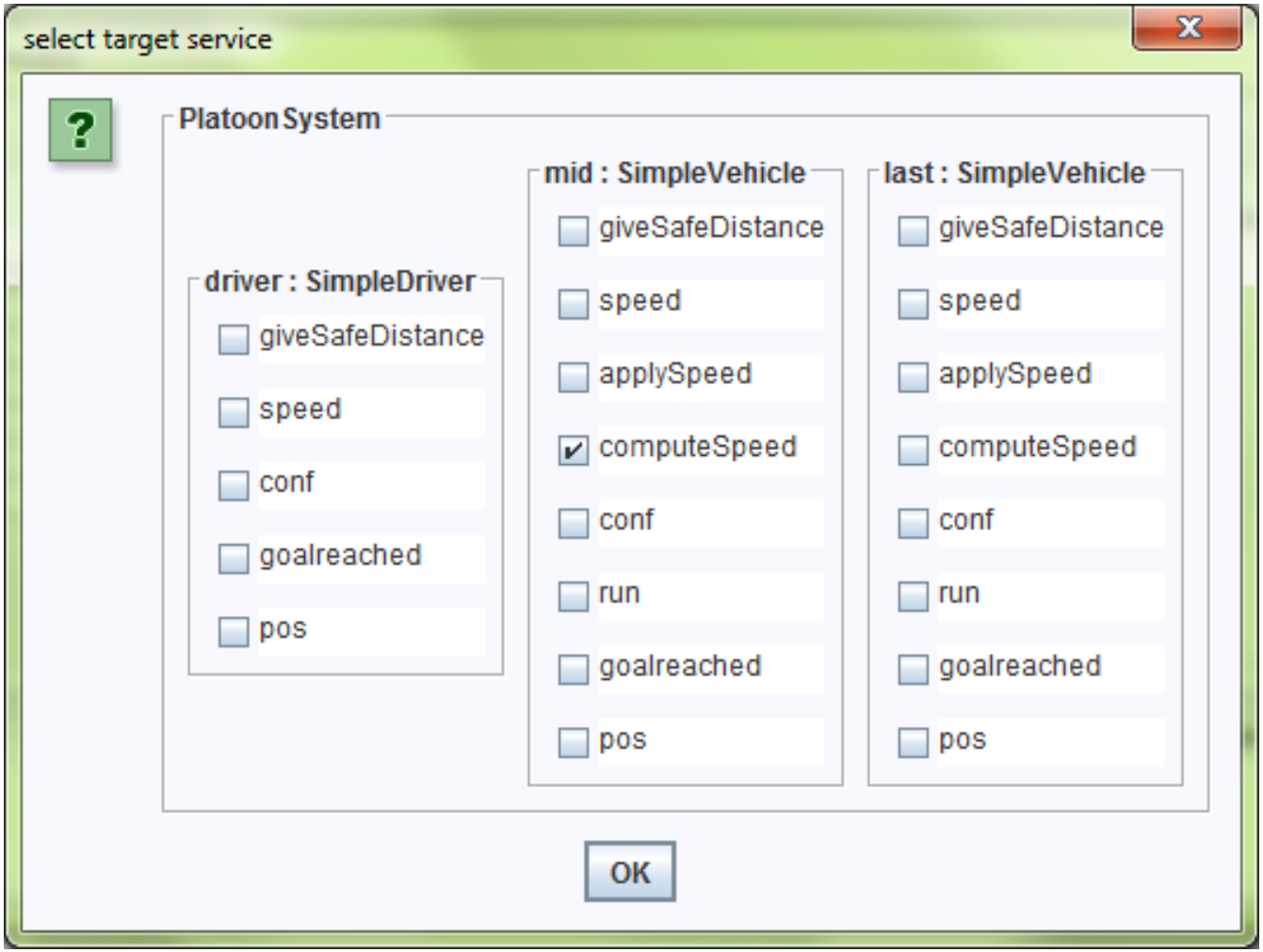
CostTest - Intention



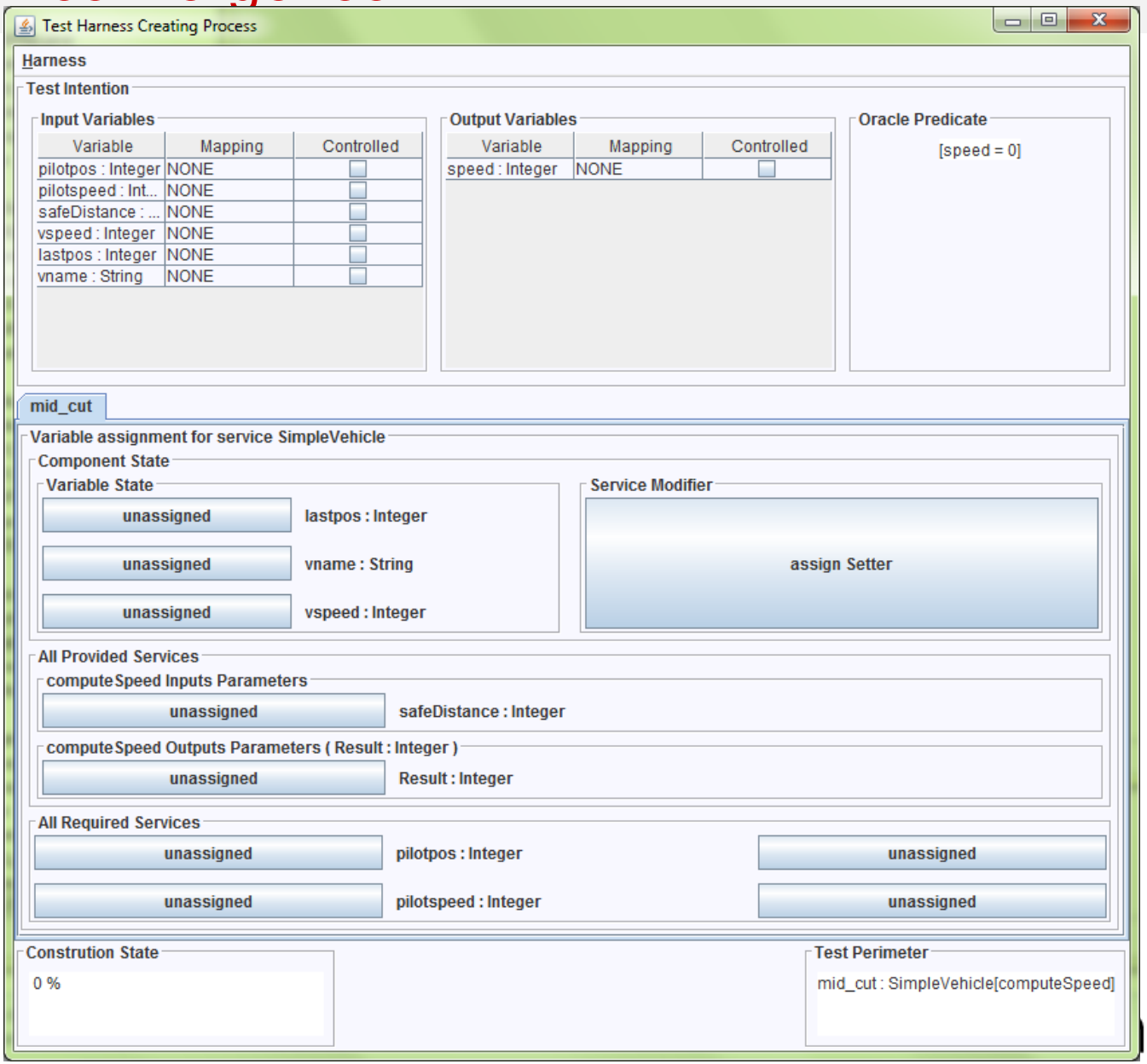
CostTest - Sélection du système



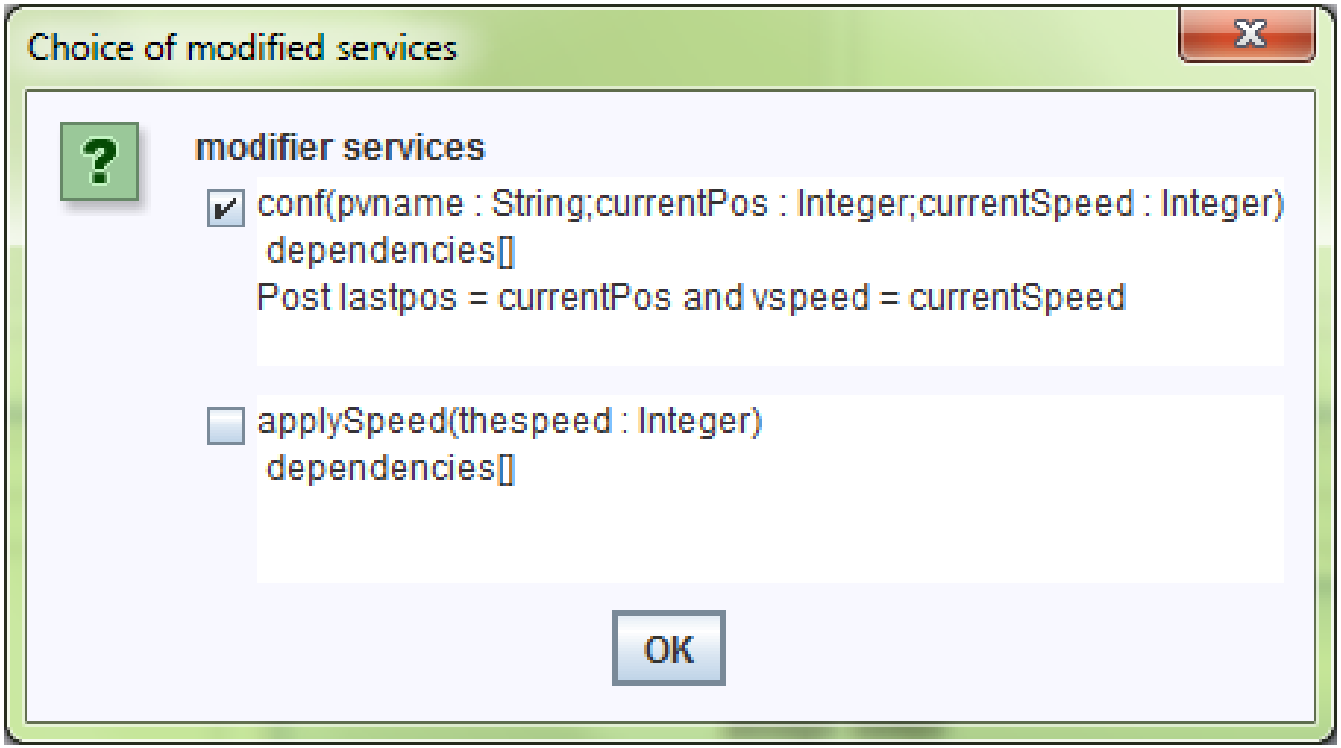
CostTest - Sélection des services cibles



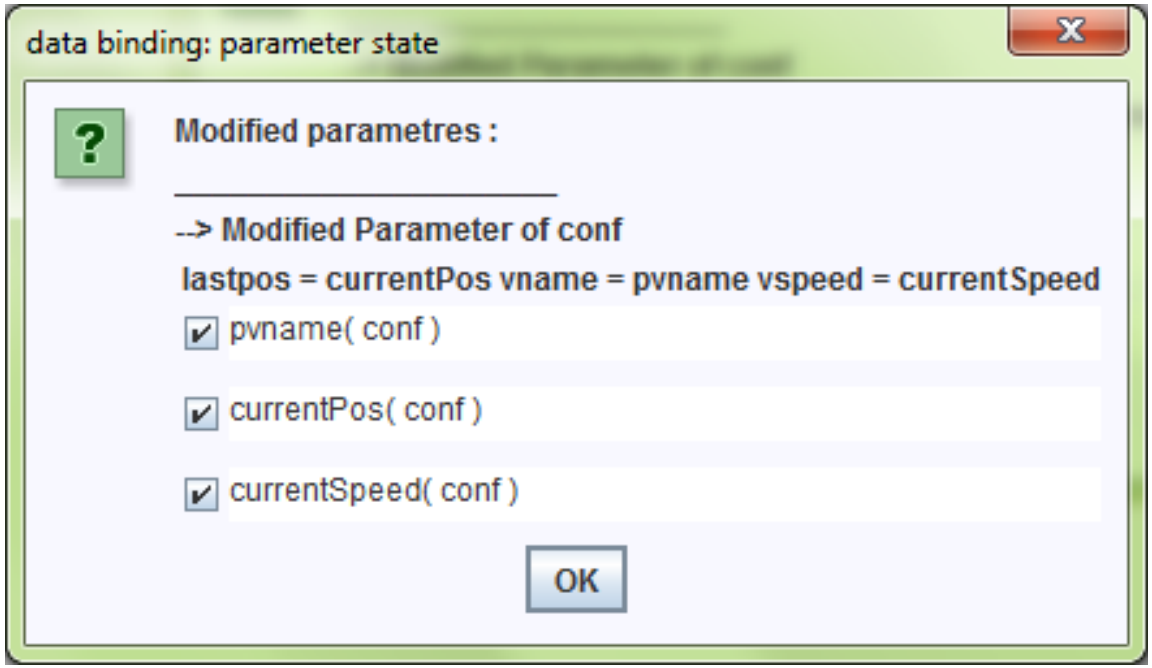
CostTest - convergence



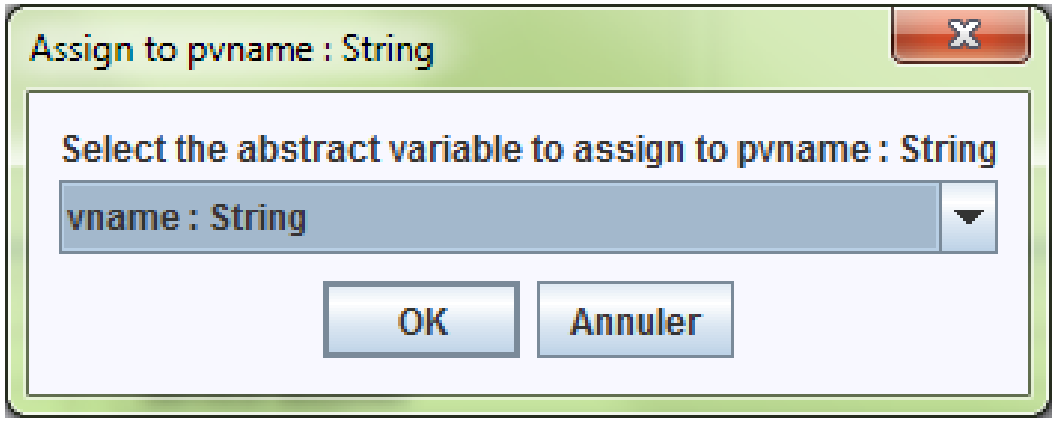
CostTest - assignSetter



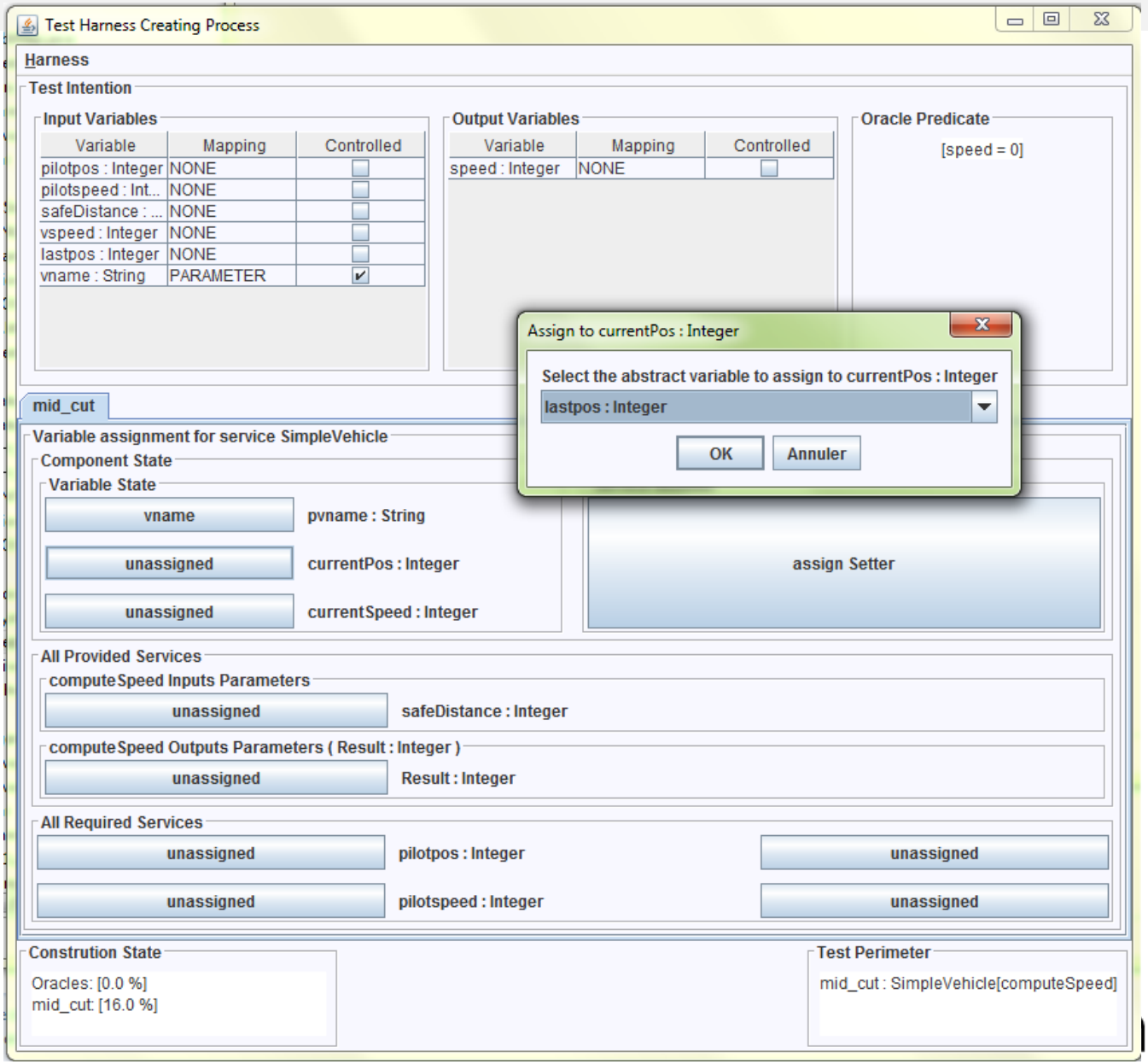
CostTest - positionnement état



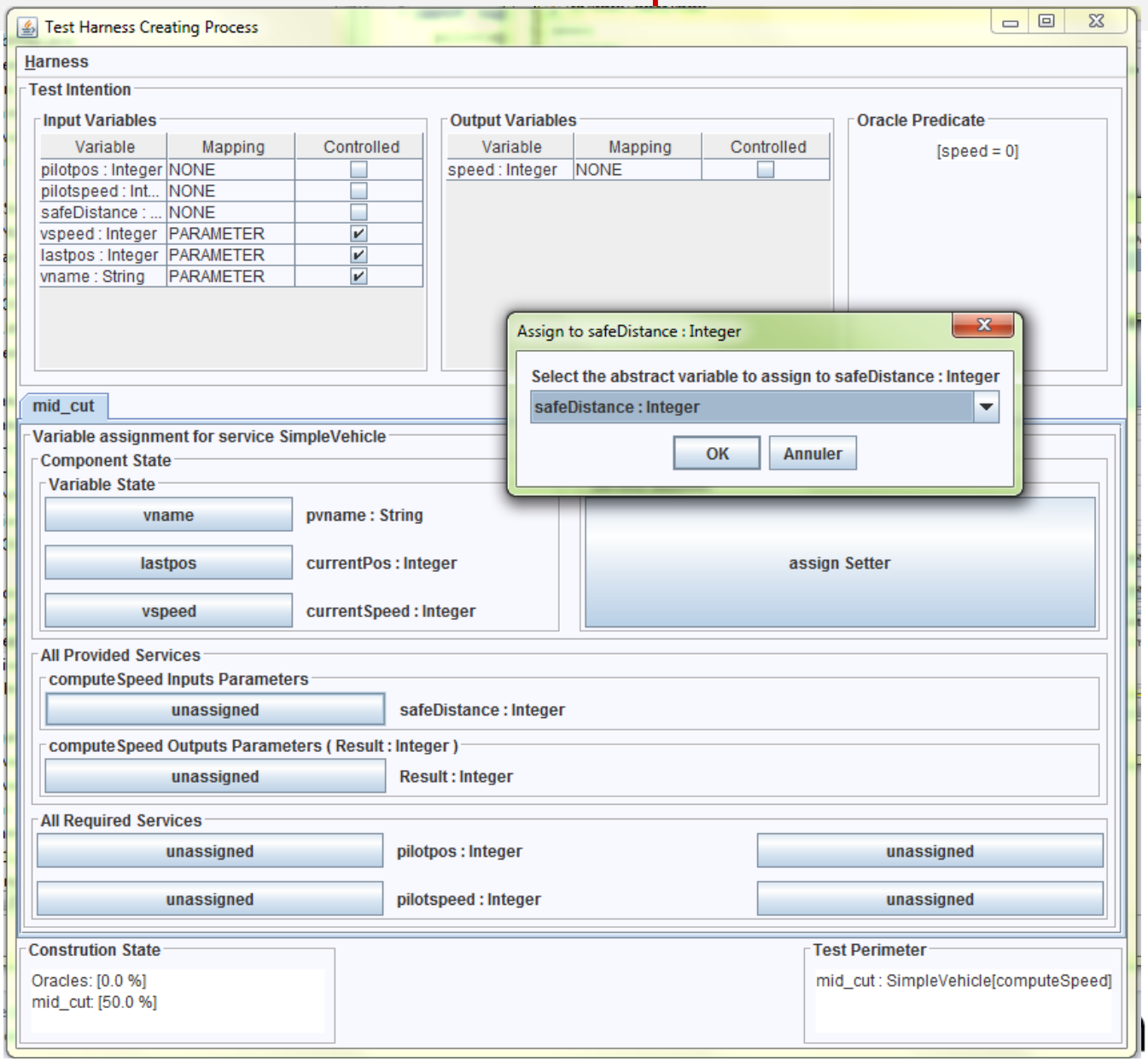
CostTest - identification intention état



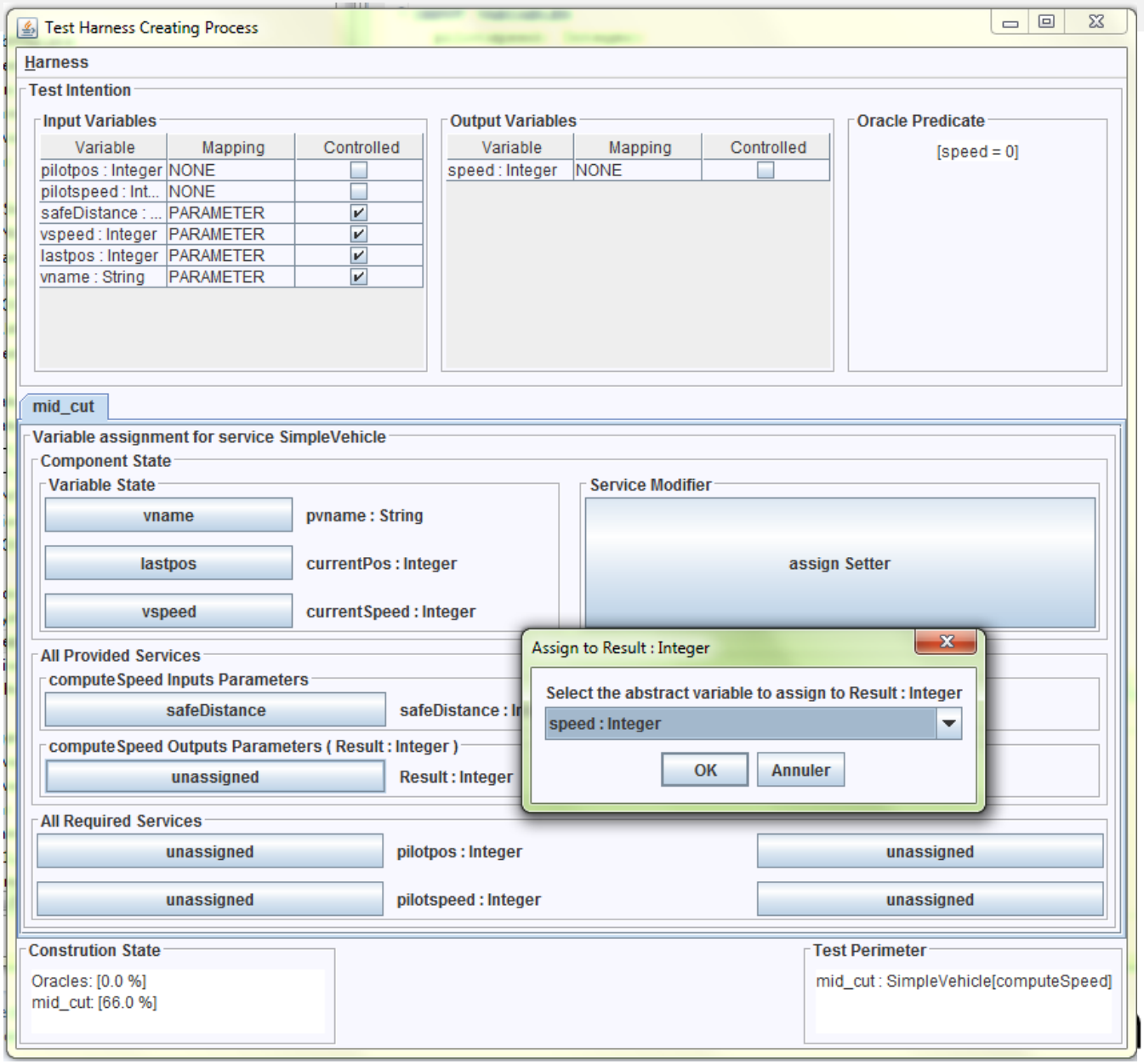
CostTest - identification intention état



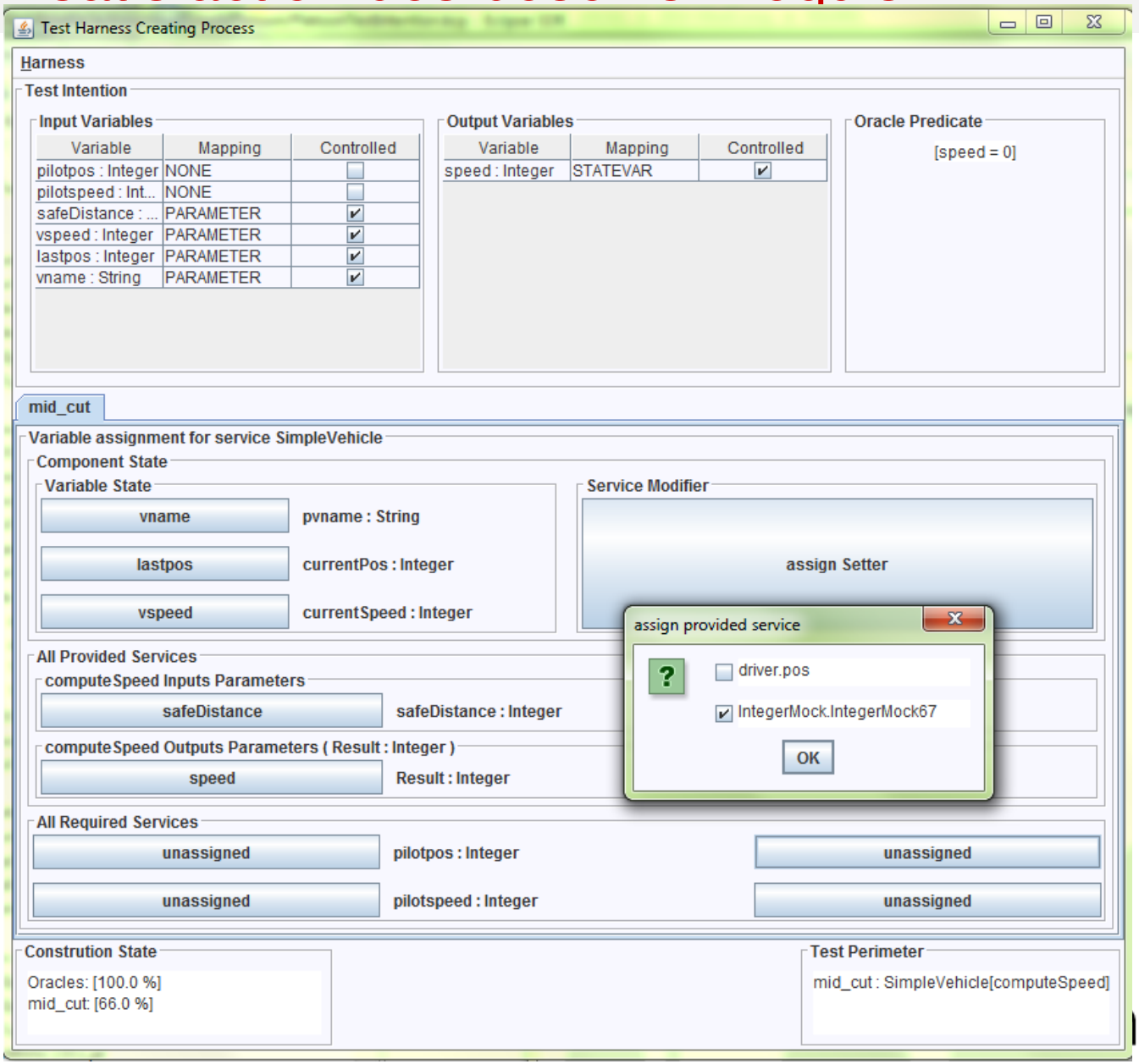
CostTest - identification intention paramètres



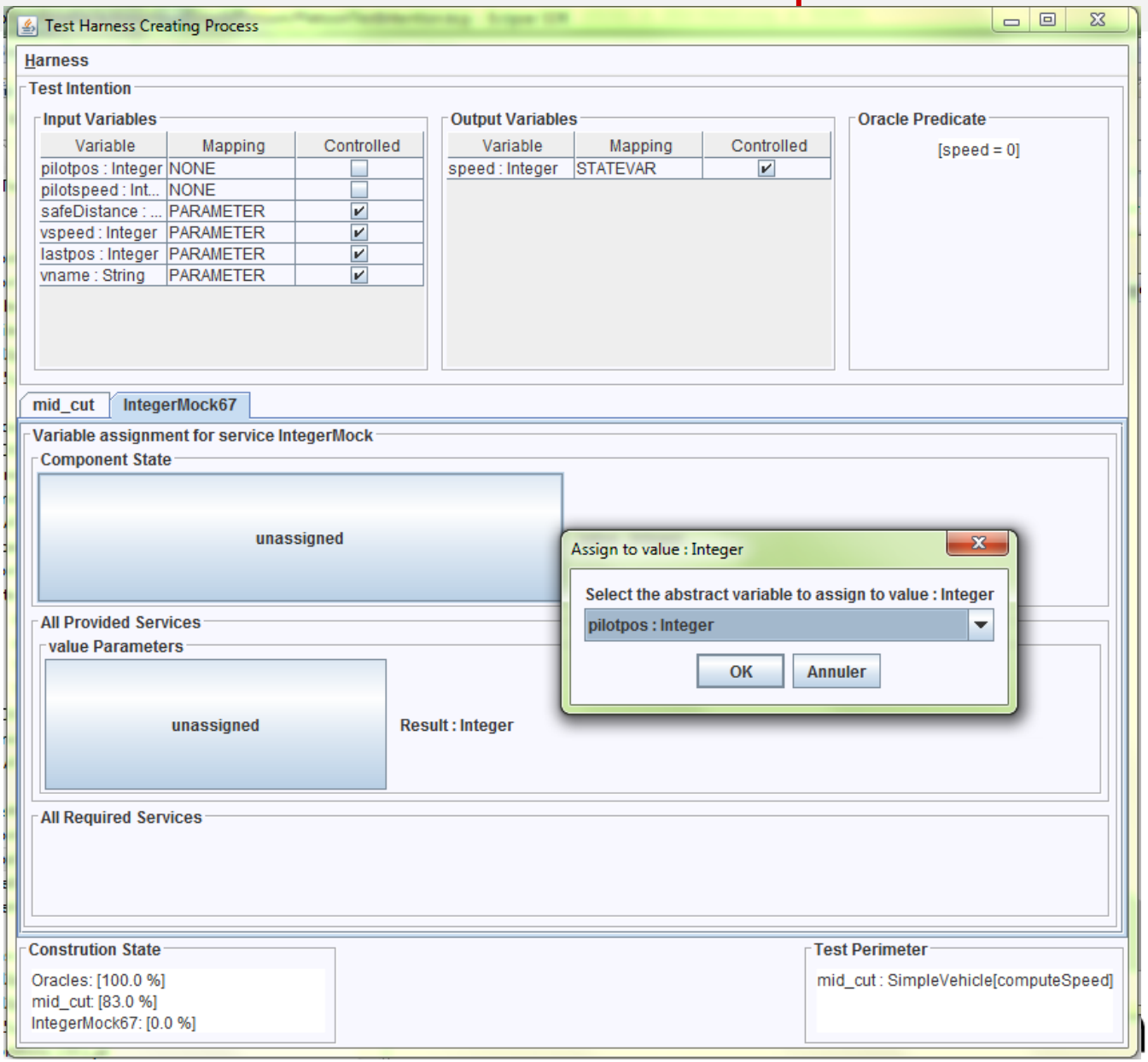
CostTest - identification intention résultat



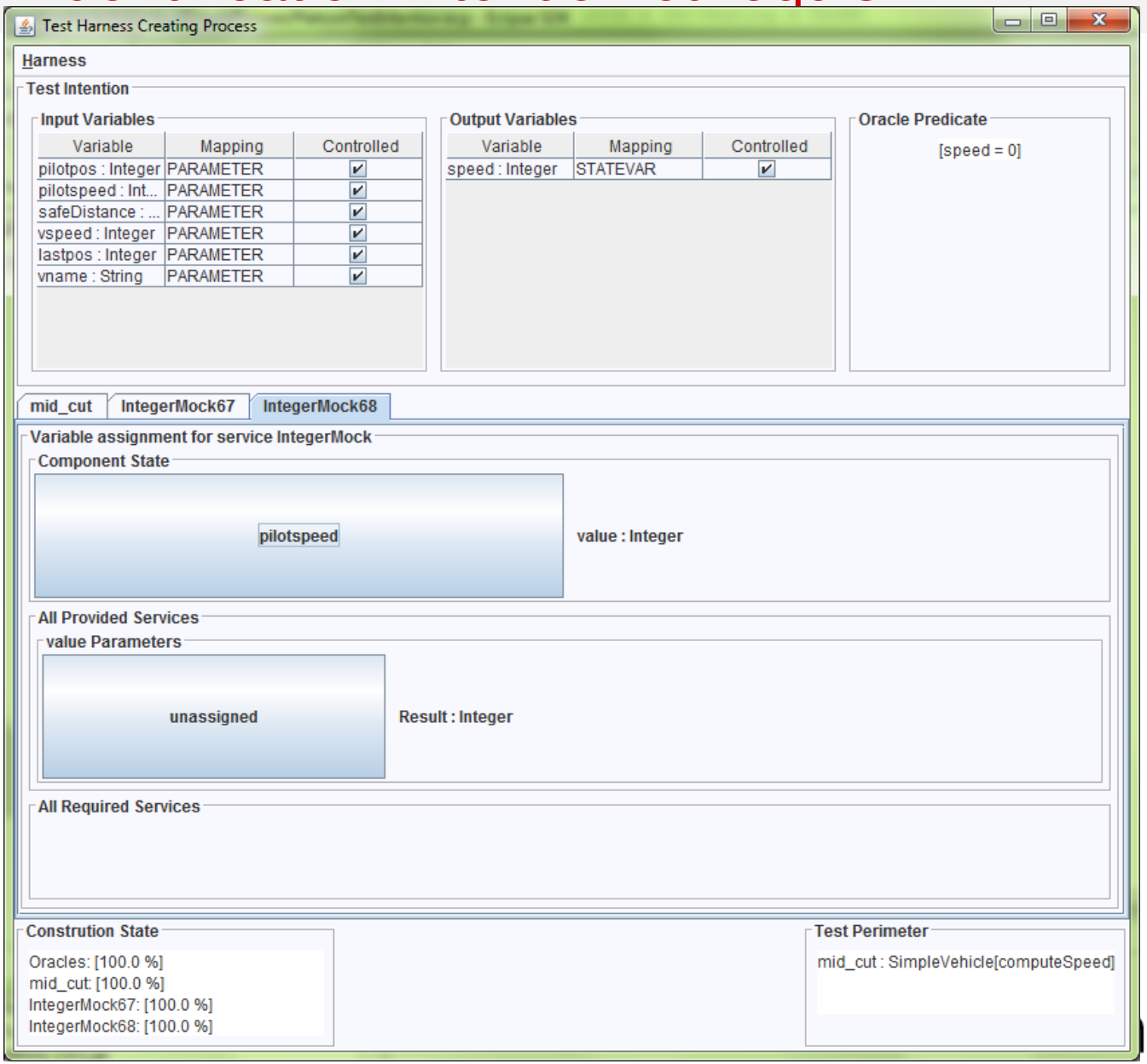
CostTest - satisfaction des besoins - requis 1



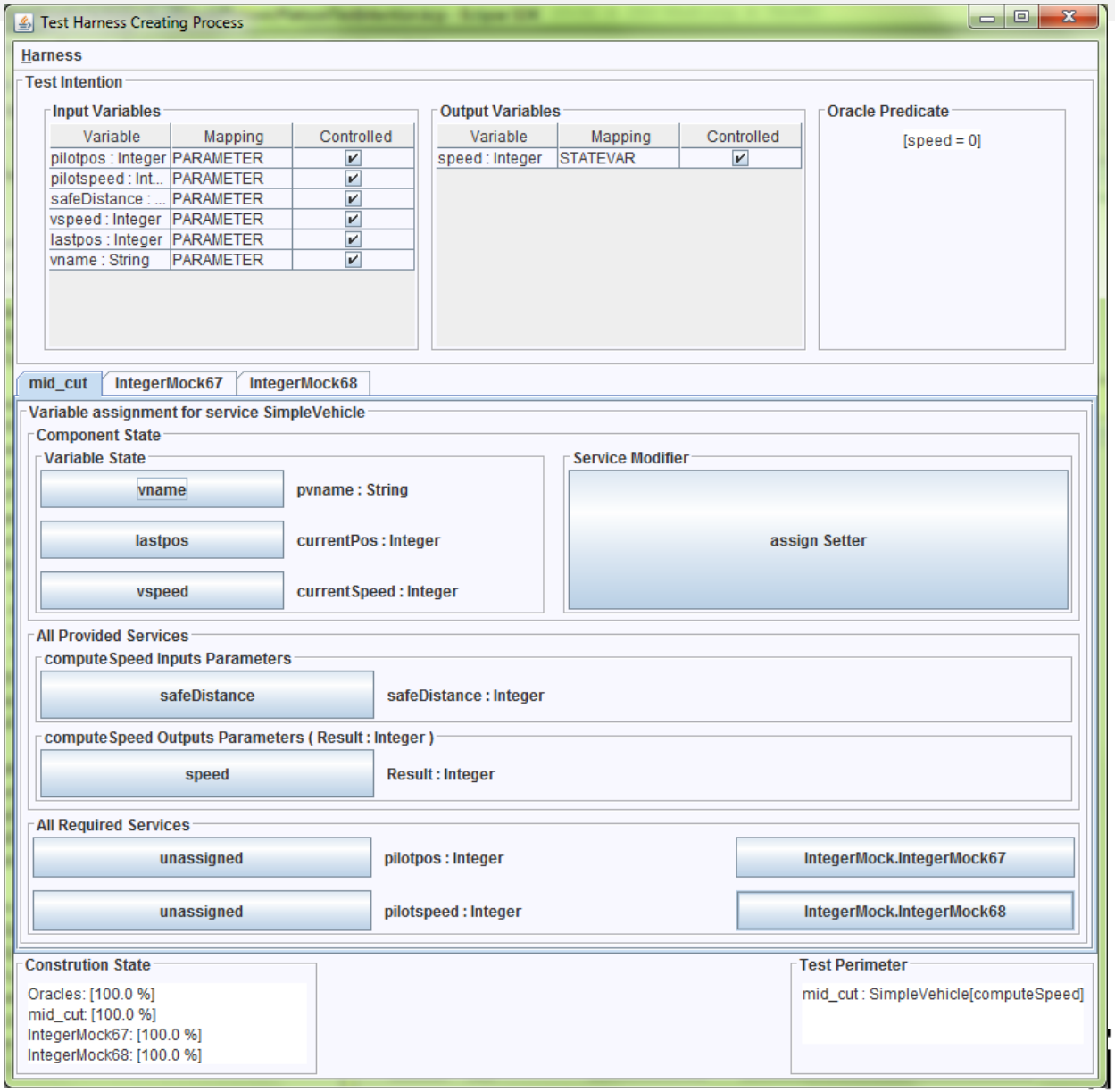
CostTest - identification intention et requis 1



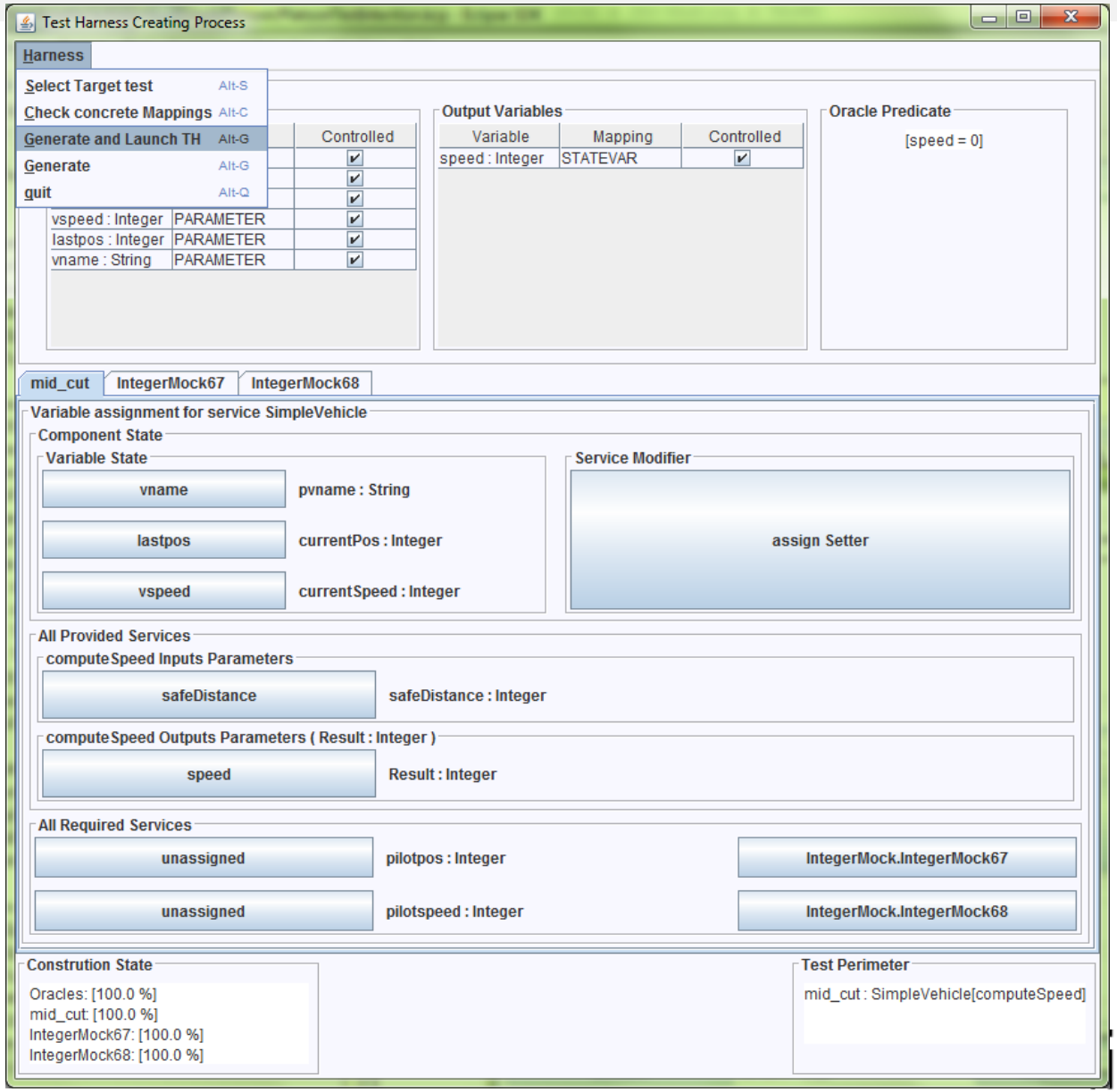
CostTest - identification intention et requis 2



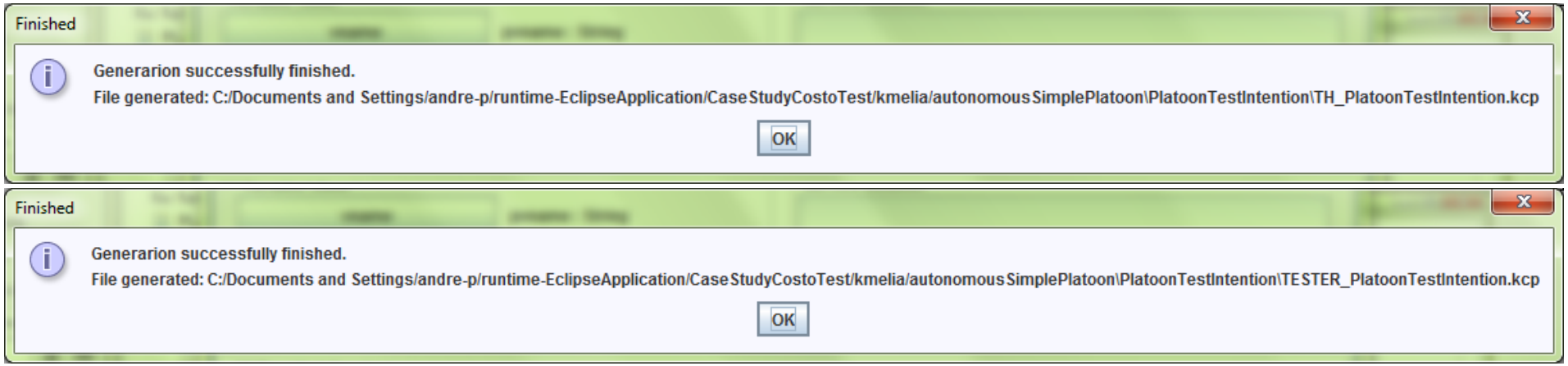
CostTest - contexte du harnais établi



CostTest - Génération du harnais

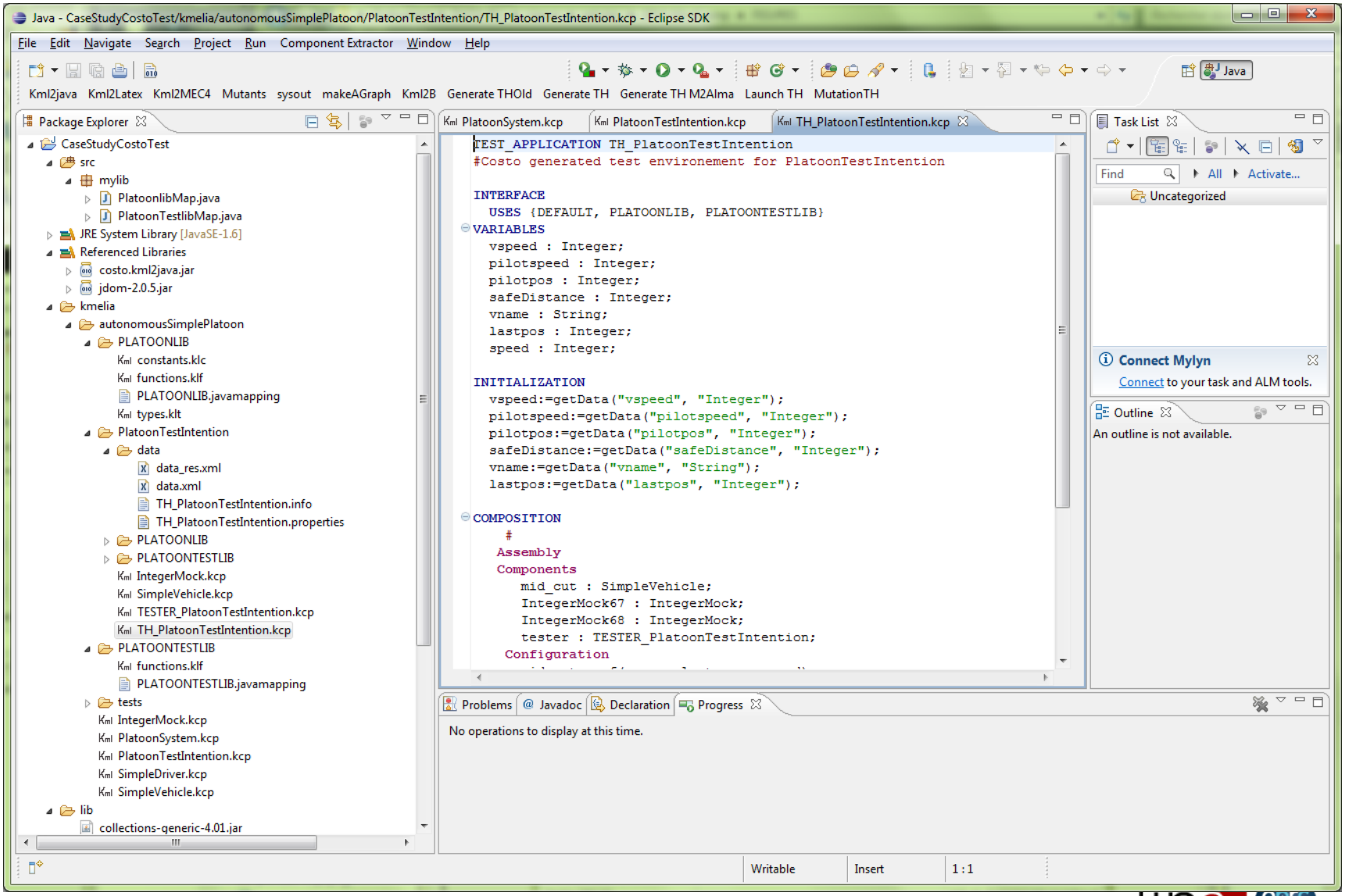


CostTest - Génération du harnais

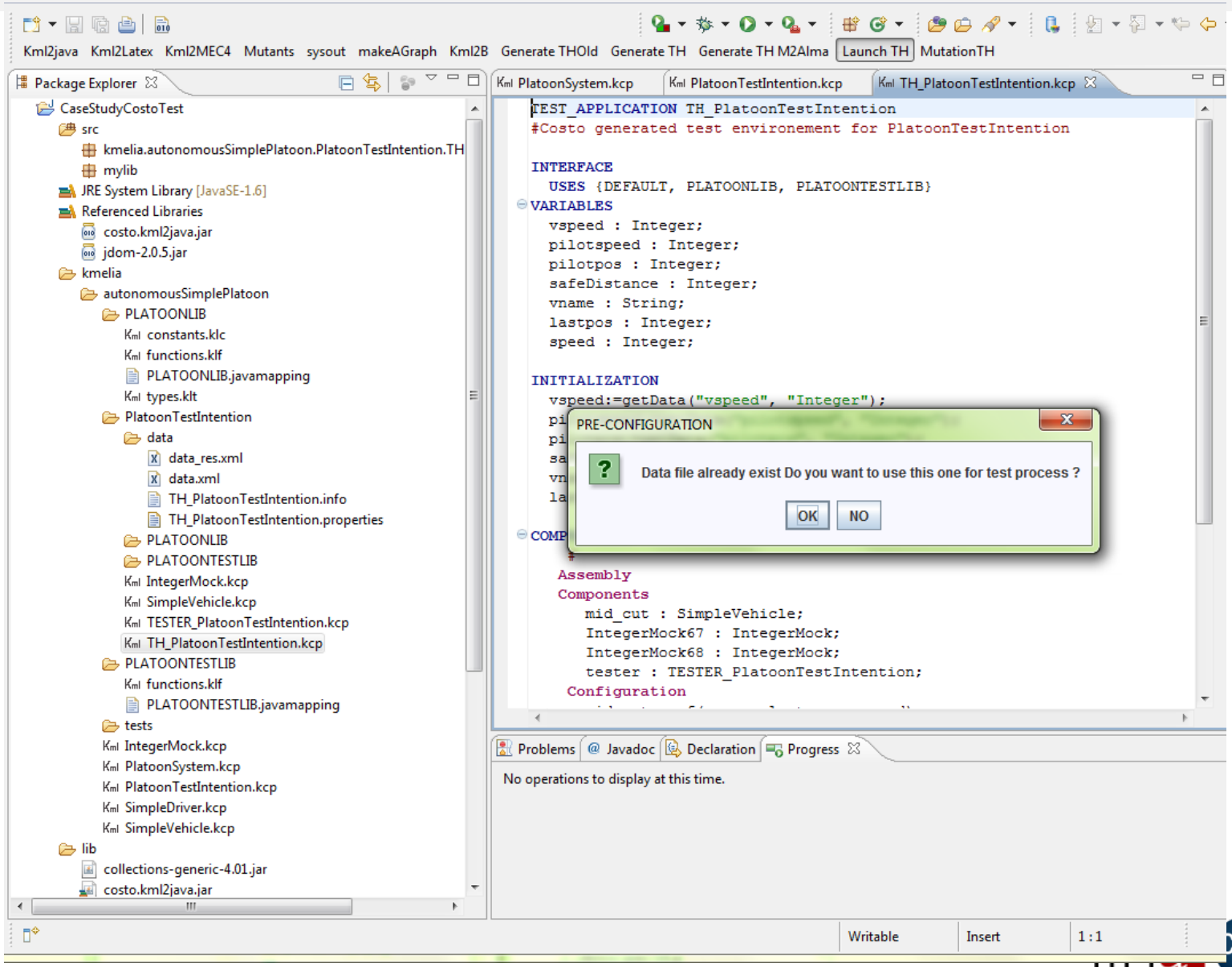


mutation possible

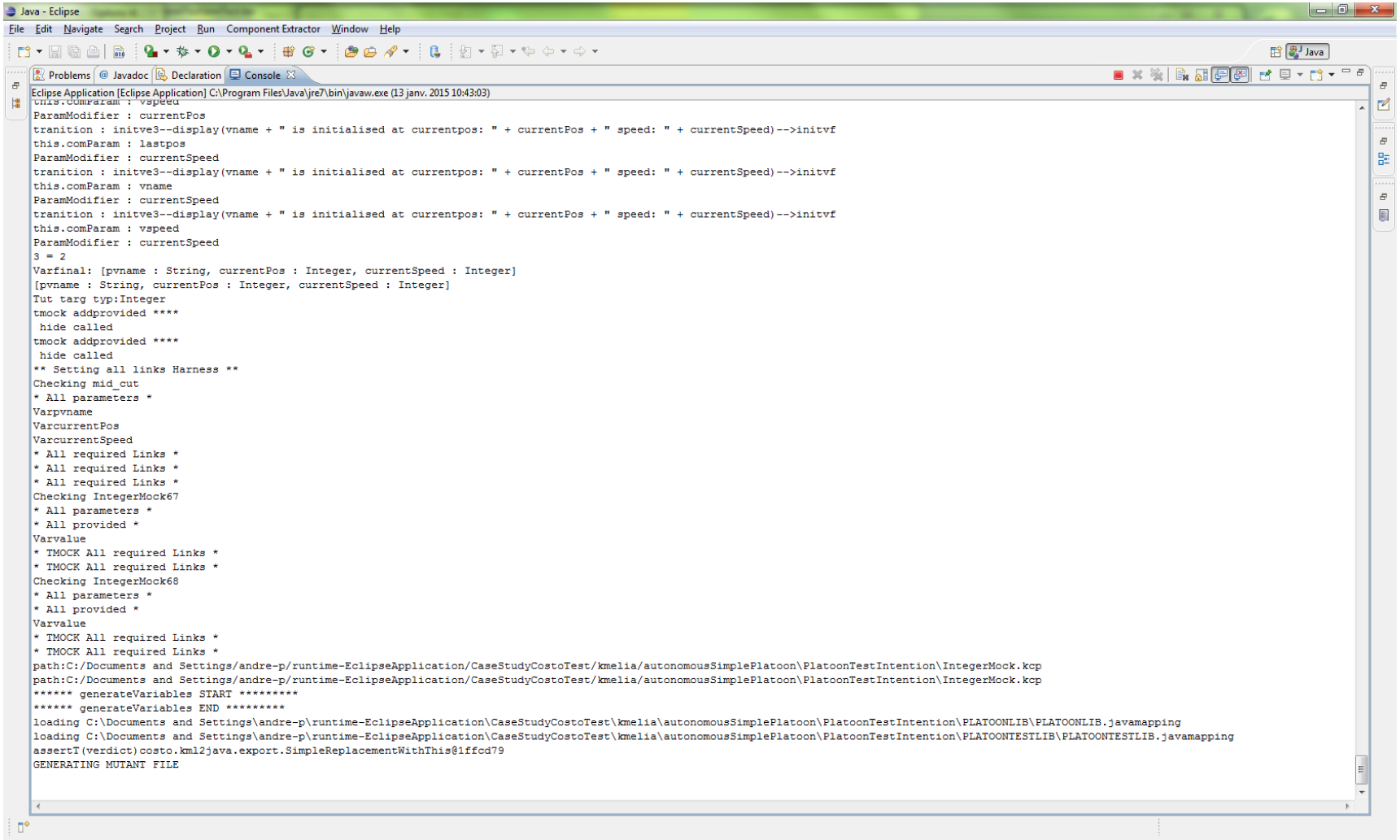
CostTest - Application de test



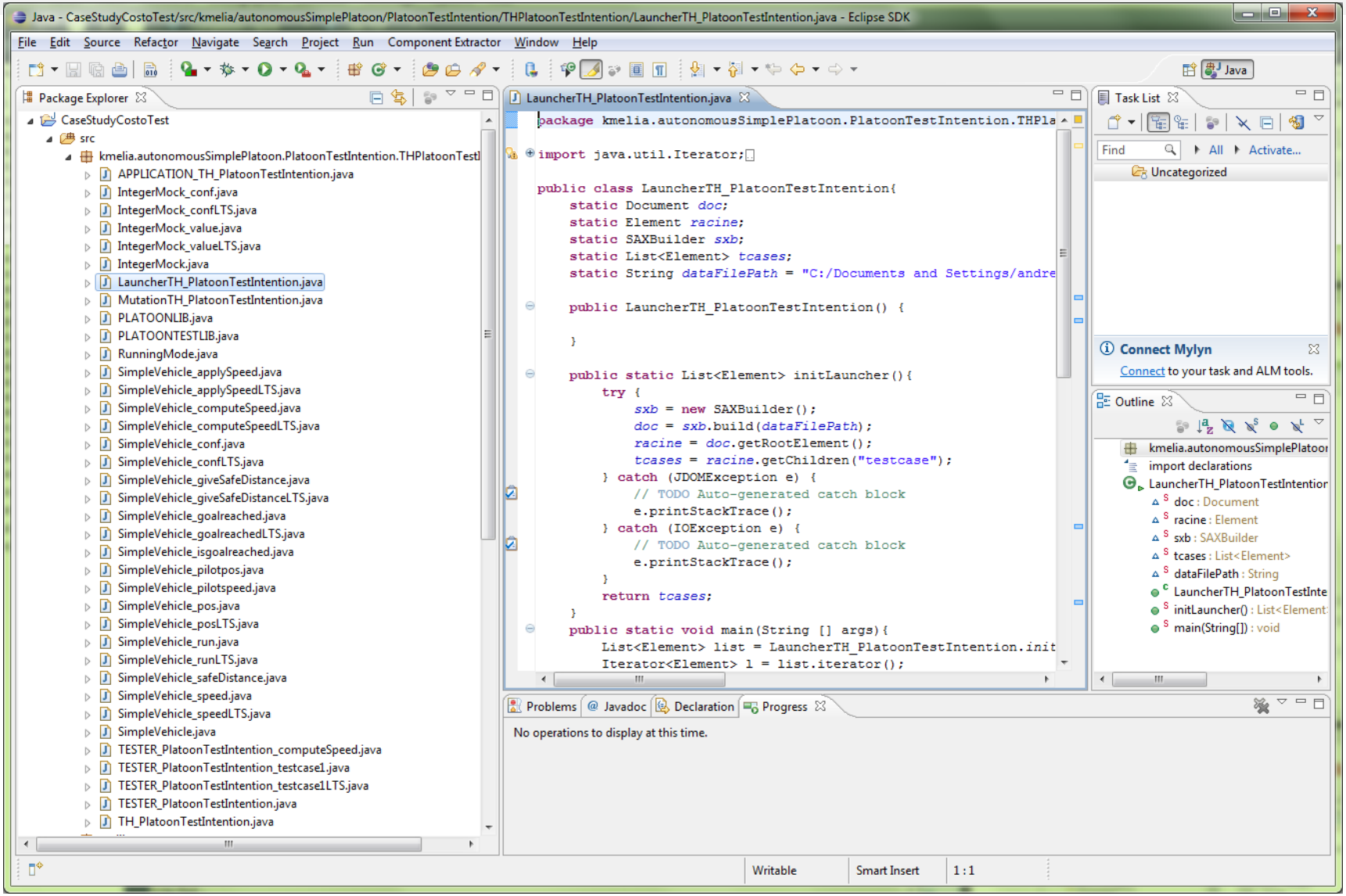
CostTest - Génération du code et lien aux données



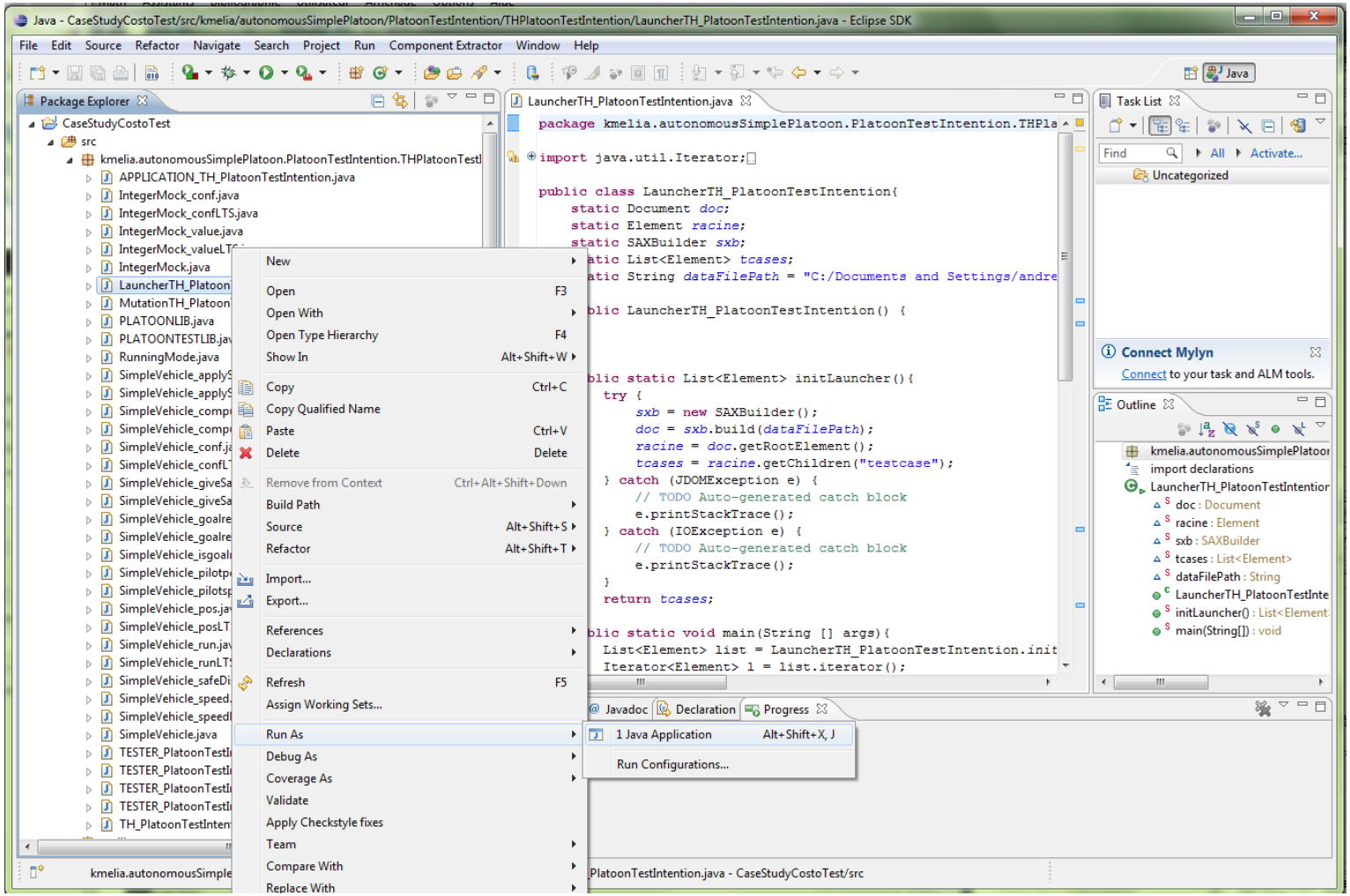
CostTest - Génération du code et lien aux données (console)



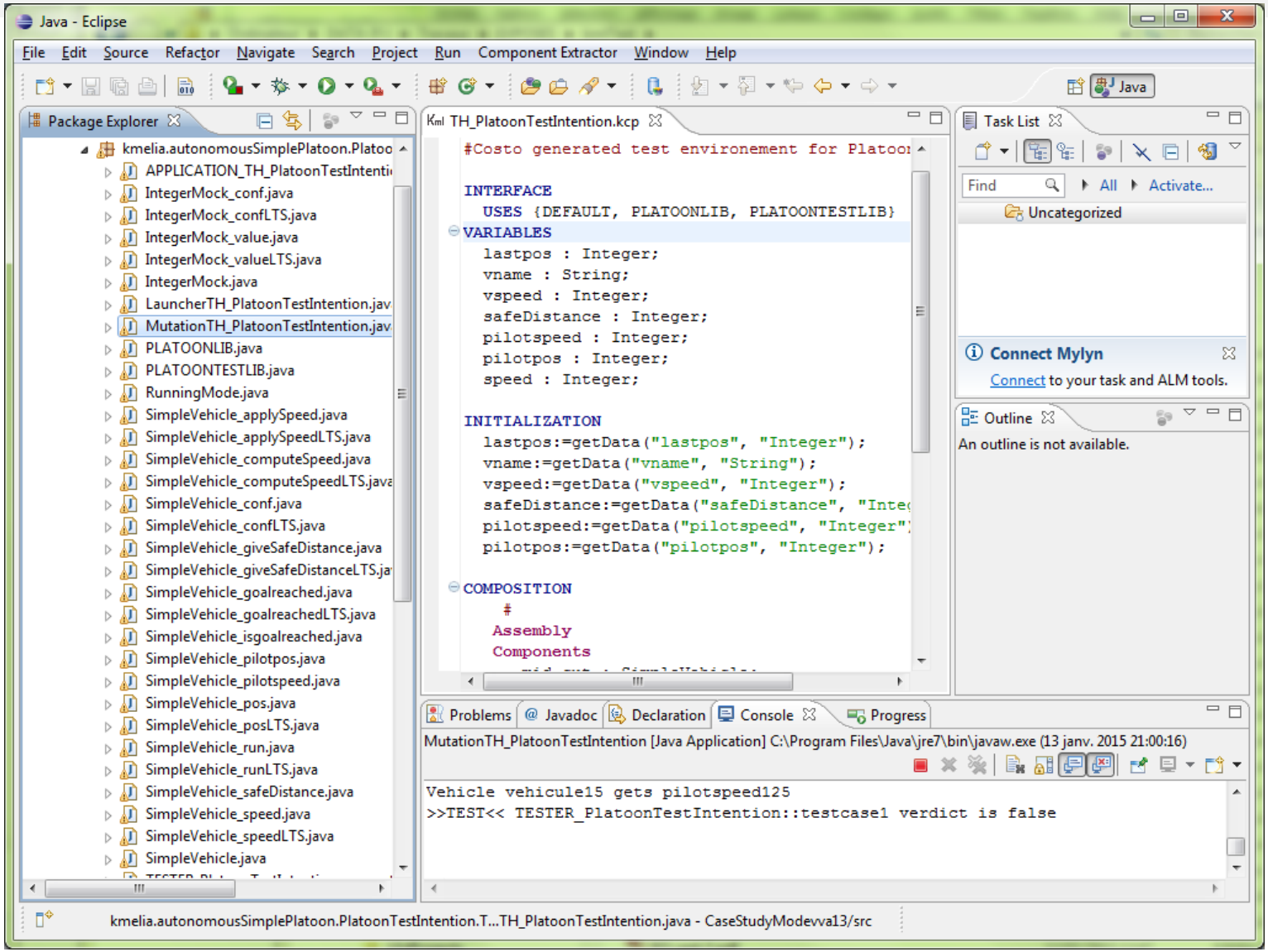
CostTest - code généré



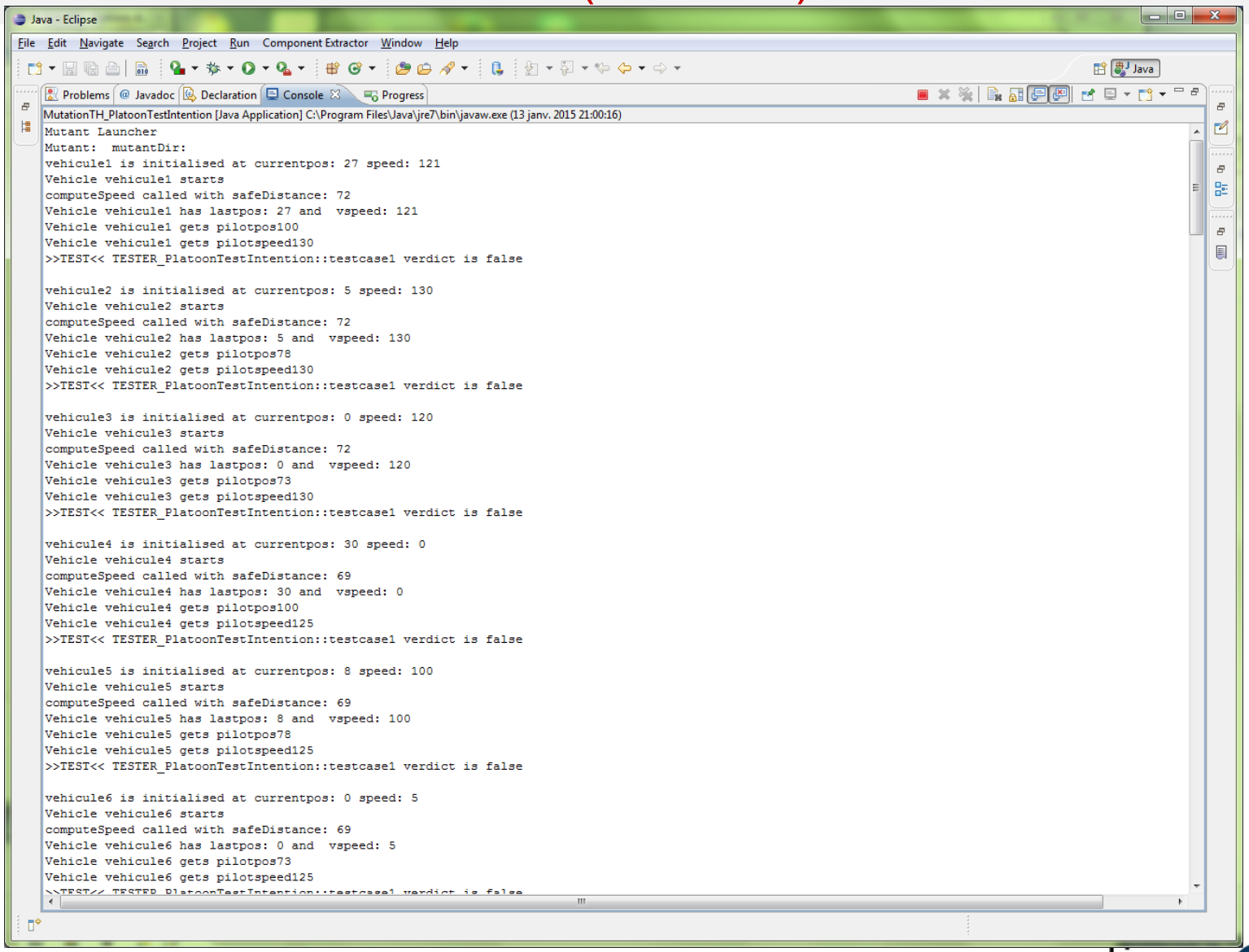
CostTest - exécution du test



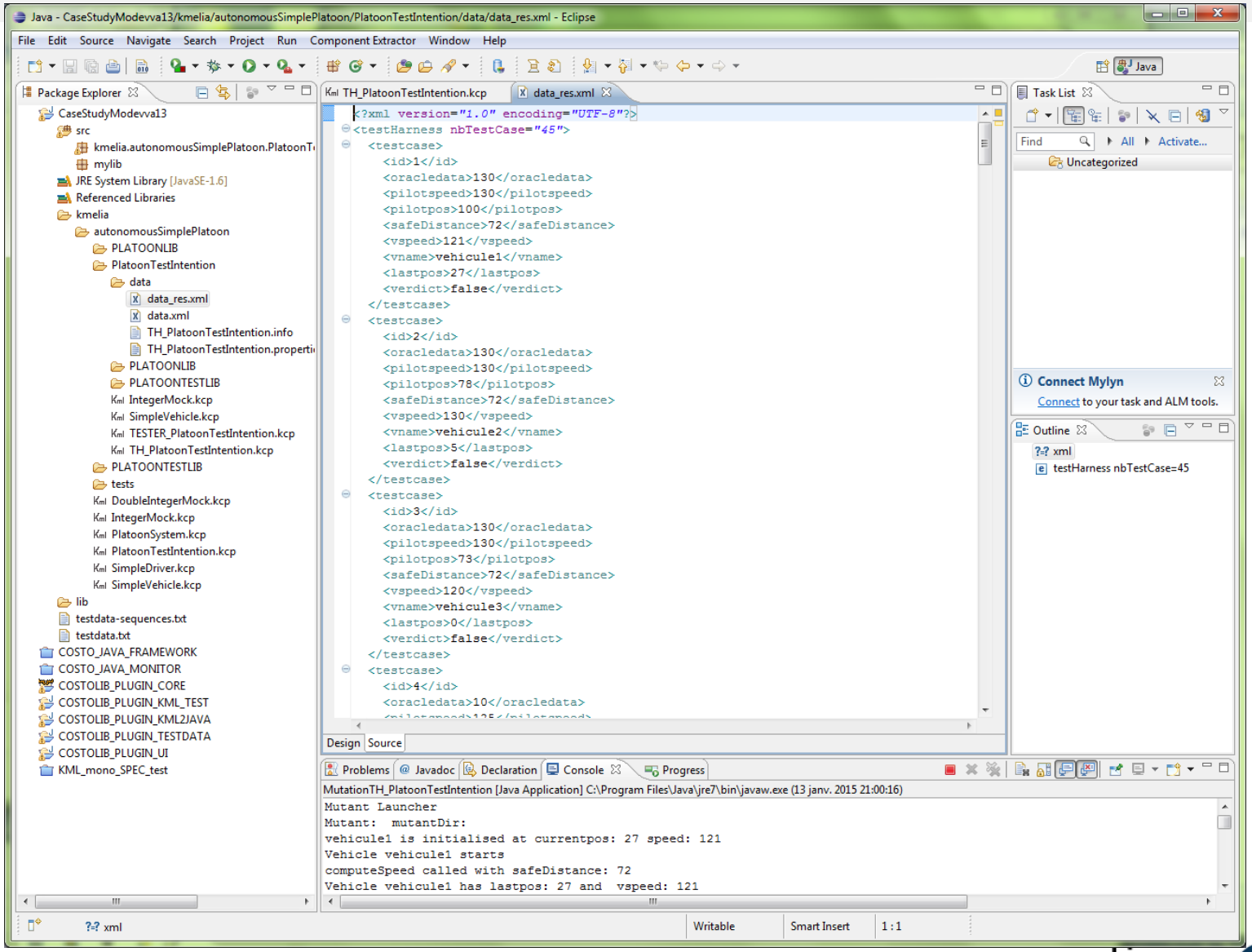
CostTest - exécution du test (lancement manuel)



CostTest - exécution du test (console)



CostTest - résultat du test



Plan

CostTest - Historique

1. V1 (GA) : fonctions d'assistance, début construction, API de base
2. V2 (M1/Alma) : construction, exécution, documentation wiki
   - `http://www.lina.sciences.univ-nantes.fr/aelos/wiki/doku.php/costo:test:userguide:costotestuginstall`
   - `http://www.lina.sciences.univ-nantes.fr/aelos/wiki/doku.php/costo:test:results:start`
3. V3 (M2/Alma) : intégration poursuivie, documentation web
4. ...

forges : CIE, Lina

CostTest - En cours

1. Maintenance des plugins COSTO
   - Builder
   - Interface
   - Refactoring
2. Intégration des plugins CostoTest
   - Installer
   - Console améliorée
   - Gestion des données de test
3. Expérimentations
   - comparaison xUnit (Alma ?)
   - études de cas
4. Documentation wiki
   - site Web, wiki
   - Forges
5. Evolutions CostoTest
   - Mutation
   - Gestion de bancs de test
   - Méthodologie