Contextual Substitutability “Car Park” Demo Application

Premek Brada <brada@kiv.zcu.cz> *1.Feb 2012*

The application consists of several bundles. It evolves in 6 application revisions between which combinations of changes to bundles happen that lead to strict and/or contextual substitutability. Descriptions here are those not clear from bundle manifests.

Abbreviations

- svc = service
- pkg = package
- v = version

Core and shared

- Namespace prefix: **cz.zcu.kiv.osgi.demo.parking** .
- All bundles except statsbase use SLF4J for logging => import package org.slf4j .
- All bundles except statsbase have activator => import org.osgi.framework

The slf4j and osgi packages are called “system” ones below.

Bundle revisions and pkg versions per application revisions

Unchanged are in normal weight, changed are in boldface, newly added are with asterisk.

Overview

Bundle	**1**	**2**	**3**	**4**	**5**	**6**	**← App Rev**	**Mmu?**
StatsBase	1 / 1.0.0	1	1	**2 / 1.0.1**	2	2	← rev / OSGi vers ← revs working in ctx given by app rev	
	1, 2	1, 2	1, 2	2	2	2		OK
Dashboard	1 / 1.0.0	1	**2 / 1.0.1**	2	2	2		
	1, 2	1, 2	1, 2	1, 2	1, 2	1, 2		OK
Gate	1 / 1.0.0	**2 / 1.0.1**	**3 / 1.1.0**	**4 / 2.0.0**	4	4		
	1, 2, 3	1, 2, 3	1, 2, 3	3, 4	4	4		SOME
CarPark	1 / 1.0.0	1	1	**2 / 1.1.0**	**3 / 2.0.0**	3		
TrafficLane			1 / 1.0.0	**2 / 1.0.1**	**3 / 1.0.2**	**4 / 2.0.0**		
RoadSign					1 / 1.0.0	1		

The last column (“Mmu?”) tells whether OSGi “Major.minor.micro” semantic bundle versions work for the given bundle.

Explanations of ctx substitutability

(Notation: “G/3” means “table cell Gate in app rev 3”, “G3” is “bundle Gate rev 3”.)

- S/1-S/3: can use S2 here because it is a superset of S1
- S/4-S/6: rev 1 cannot be used here because it lacks abstract base class used by APP/3-APP/6 statistics bundles;
- D: both versions can be in any app rev because StatsBase is always available
- G/1-G/3: cannot have G4 here because it needs S2, plus for G/1 and G/2 cases because LaneStats would be missing (=> OSGi bundle versions give false negative in G/3 case, work in G/1 and G/2)
- G/4: can have G3 here because G4's exported ParkingStatus operations are not needed in APP/4 (=> OSGi bundle versions give false negative in this case); but cannot have G1-G2 here because they don't export VehicleSink needed by TrafficLane (=> OSGi version work)
- G/5: cannot have G3 here because now TrafficLane needs ParkingStatus operations of G4 (=> OSGi bundle versions work)

TODO analysis of ctx subst for C,T,R

Details follow. The SEM-SPE differences due to the switch from plain interface implementation to extending abstract base class could be completely avoided if the implementations were hidden in non-exported internal packages...

Rev 1 – Baseline

Dashboard uses GateStats on its “gate” port only through its CountingStats subset. Gate provides LaneStatistics since there is no other bundle to do that.

All diffs are “NA” because there is no preceding revision.

***StatsBase** 1 = 1.0.0

- ex: statsbase;v=1.0.0
- im: (none)

***Dashboard** 1 = 1.0.0

imports statsbase because parent interface is used on “gate” stats port

- ex: (none)
- im:
  gate.statistics;v=[1.0.0, 2.0.0) – (!) just to create the gate stats service via GateStats factory
  lane.statistics;v=[1.0.0, 2.0.0) – ILaneStats iface and svc used
  statsbase;v=[1.0.0, 2.0.0) – ICountingStats iface used on gate stats service port

***Gate** 1 = 1.0.0

- ex:
  gate.statistics;v=1.0.0
  lane.statistics;v=1.0.0
- im:
  carpark.flow;v=[1.0.0, 2.0.0) – IVehicleFlow and provided svc used by gate statistics
  statsbase;v=[1.0.0, 1.1.0) – IGateStats and ILaneStats extend its ICountingStats

***CarPark** 1 = 1.0.0

- ex:
  carpark.flow;v=1.0.0
  carpark.status;v=1.0.0
- im: (system only)

Rev 2 – Gate observes CarPark status

Gate adds dependency on ParkingStatus, provided by CarPark.

StatsBase 1 = 1.0.0 from rev 1

Dashboard 1 = 1.0.0 from rev 1

Gate 2 = 1.0.1 (NON)

INS on imported packages (ParkingStatus)

- ex:
  NON gate.statistics;v=1.0.0
  NON lane.statistics;v=1.0.0
- im:
  INS carpark.status;v=[1.0.0, 2.0.0) – used by GateStats to get "carpark full" info
  NON carpark.flow;v=[1.0.0, 2.0.0)
  NON statsbase;v=[1.0.0, 1.1.0)

CarPark 1 = 1.0.0 from rev 1

Rev 3 – TrafficLane added

LaneStatistics evolved by adding a method (which however is not used by clients in app rev 3-5). TrafficLane component added to the system, provides LaneStatistics but the Dashboard is still connected to the Gate which still provides LaneStatistics as well. Gate consequently also runs the traffic simulation internally. Dashboard now uses full GateStatistics on its "gate" port.

StatsBase 1 = 1.0.0 from rev 1

Dashboard 2 = 1.0.1 (NON)

SPE on required svc for gate stats results in imported pkg no longer needed.

- ex: (none)
- im:
  NON gate.statistics;v=[1.0.0, 2.0.0) – (used for real now, IGateStats iface needed)
  NON lane.statistics;v=[1.0.0, 2.0.0)
  DEL statsbase – ICountingStats no longer used (svc variable typed to IGateStats instead)

Gate 3 = 1.1.0 (SPE)

Diffs as follows, plus internal change: activator spawns traffic lane simulation thread.

- ex:
  NON gate.statistics;v=1.0.0
  SPE lane.statistics;v=1.1.0 – a getVehiclesPerInterval() method added
  INS gate.vehiclesink;v=1.0.0 – newly added
- im:
  NON carpark.flow;v=[1.0.0, 2.0.0)
  NON carpark.status;v=[1.0.0, 2.0.0)
  NON statsbase;v=[1.0.0, 1.1.0)

CarPark 1 = 1.0.0 from rev 1

***TrafficLane 1 = 1.0.0**

- ex:
  NA lane.statistics;v=1.1.0 – bundle exports the same pkg as Gate
- im:
  NA gate.vehiclesink;v=[1.0.0, 2.0.0) – the lane needs to put generated vehicles to the sink
  NA statsbase;v=[1.0.0, 1.1.0) – ILaneStats inherits from ICountingStats

Rev 4 – TrafficLane Statistics used

TrafficLane component's interface did not change, but its LaneStatistics are now used by Dashboard, because Gate stopped providing LaneStatistics. Internally the TrafficLane implementation changed to run the traffic simulation. CarPark now provides ParkingStatistics which however are not used by anybody. StatsBase changed, now provides an abstract implementation of CountingStatistics interface, all statistics classes (gate, lane, carpark) inherit from this abstract base.

StatsBase 2 = 1.0.1 (SEM-SPE)

Exported pkg evolved but ... see below for detailed reasoning behind the bundle version change.

- ex:
  SEM-SPEC statsbase;v=1.0.1 – added abstract base class to the package
- im: (none)

Dashboard 2 = 1.0.1 from rev 3

Gate 4 = 2.0.0 (MUT)

- ex:
  SPE gate.statistics;v=1.1.0 – adds ParkingStatus operations
  SEM-SPE gate.vehiclesink;v=1.0.1 – adds throw exception
  DEL lane.statistics – because lane stats now provided by traffic lane bundle
- im:
  SPE carpark.status;v=[**1.1.0**, 2.0.0) – GateStats re-exports evolved IParkStatus operations
  SEM-SPE statsbase;v=[**1.0.1**, 1.1.0) – abstract base class extended by GateStats etc
  NON carpark.flow;v=[1.0.0, 2.0.0)

CarPark 2 = 1.1.0 (SPE)

- ex:
  SEM-SPE carpark.flow;v=1.0.1 – impl of VehicleFlow adds throw exception
  SPE carpark.status;v=1.1.0 – new IParkingStatusUpdate, methods added to IParkingStatus
  INS carpark.statistics;v=1.0.0 – whole new pkg exported
- im:
  INS statsbase;v=[1.0.1, 2.0.0) – the (evolved) statsbase needed for carpark.statistics

TrafficLane 2 = 1.0.1 (SEM-SPE)

- ex:
  SEM-SPE lane.statistics;v=1.1.1 – LaneStats now extends abstract base class
- im:
  NON gate.vehiclesink;v=[1.0.0, 2.0.0)
  SEM-SPE statsbase;v=[**1.0.1**, 1.1.0) – now using evolved statsbase with abstract base

Note on the version of "statsbase" package and whole bundle: the changes introduced are NON on the API part (the ICountingStatistics iface unchanged) and INS on the implementation part (abstract base

class added). These changes don't break existing clients (who only used the interface so far) neither existing providers (because the interface hasn't changed). In this fairly singular case therefore, the change should IMHO be classified as internal.

OTOH, it makes versioned dependency difficult, needing to use the 1.0.1 as lower bound in gate, carpark etc which somehow feels odd. But if 1.0.0 was used as pkg vers in the import, linkage errors would result, and if 1.1.0 was used as pkg version, it would render the bundle incompatible with previous revisions of other bundles in app revs 1-3 which clearly would be wrong.

Lastly: implementation of statistics classes extend the abstract base, which makes its public methods removed from their public interface; we do not classify this as DEL → GEN difference because clients will not notice the difference.

Rev 5 – RoadSign added

RoadSign component added, driven by TrafficLane; shows a sign about car park's free capacity. CarPark still provides ParkingStatistics which got changed by removing a method but which are still not used by anybody.

StatsBase	2 = 1.0.1	from rev 4
Dashboard	2 = 1.0.1	from rev 3
Gate	4 = 2.0.0	from rev 4

CarPark 3 = 2.0.0

- ex:
  NON carpark.flow;v=1.0.1
  NON carpark.status;v=1.1.0
  GEN carpark.statistics;v=2.0.0 – method removed from IParkingStatistics
- im:
  NON statsbase;v=[1.0.1, 2.0.0)

TrafficLane 3 = 1.0.2 (NON)

- ex:
  NON lane.statistics;v=1.1.1
- im:
  NON gate.vehiclesink;v=[1.0.0, 2.0.0)
  NON statsbase;v=[1.0.1, 2.0.0)
  INS gate.statistics;v=[1.1.0, 2.0.0) – to show free parking places on road sign
  INS sign.roadsign;v=[1.0.0, 2.0.0) – to invoke road sign, introduced in this app revision

***RoadSign 1 = 1.0.0**

- ex:
  sign.roadsign;v=1.0.0
- im: (system only)

Rev 6 – TrafficLane Statistics modified

LaneStatistics interface evolved by replacing a method with another one; that method however was not used by anybody in previous app revisions.

StatsBase	2 = 1.0.1	from rev 4
Dashboard	2 = 1.0.1	from rev 3
Gate	4 = 2.0.0	from rev 4
CarPark	3 = 2.0.0	from rev 5
RoadSign	1 = 1.0.0	from rev 5
TrafficLane	**4 = 2.0.0**	**(MUT)**

MUT on exported package/interface LaneStatistics

- ex:
  MUT lane.statistics;v=2.0.0 – method vehPerInterval() replaced by vehPerSecond()
- im:
  NON gate.vehiclesink;v=[1.0.0, 2.0.0)
  NON statsbase;v=[1.0.1, 2.0.0)
  NON gate.statistics;v=[1.1.0, 2.0.0)
  NON sign.roadsign;v=[1.0.0, 2.0.0)