Service-Oriented Modelling of Automotive Systems

Laura Bocchi  bocchi@mcs.le.ac.uk
José Luiz Fladeiro  jwf4@mcs.le.ac.uk
Antónia Lopes  mal@di.fc.ul.pt
Agenda

- Positioning
- Service-Oriented Computing (SOC) vs Component Based Development (CBD)
- SRML and an automotive case study
- Ongoing work
The aim is to develop a novel approach to the engineering of software systems for service-oriented overlay computers where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach.
WP1 provides support for **service-oriented modelling** at high levels of abstractions, i.e., independently of the hosting middleware and hardware platforms, and the languages in which services are programmed.
SOC vs CBD: our View

• In CBD, software components are “taken out of a box” and plugged into a system (possibly with the addition of some “glue” code) to provide a “service” (see article by Broy et al, TOSEM February 2007)

  CBD assumes early binding: the “architecture” is defined at design time.

  Composition is relatively controlled. (performed “manually”)

• In SOC, each time a service is invoked, a different provider may be chosen to negotiate terms and conditions, and then the service is finally bound (see article by Elfatatry, CACM August 2007)

  SOC adopts late binding: binding is deferred to run time, enabling the choice of provision each time and change in the quality of the requirements.

  Service composition is dynamic and unpredictable. (needs to be automated)
• There is no “system” a-priori but an evolving configuration

• We organize a configuration in three layers

  - interfaces with entities that use the service layer by launching business activities
  - dynamic configurations of components that perform volatile business activities
  - components that correspond to stable business entities and offer persistency
A SRML Activity Module

An Activity Module is launched by the top layer in a traditional way (no discovery)

One serves-interface: a sensor in a vehicle detects an engine failure

A number of component-interfaces: describe the orchestration

A number of uses-interfaces: persistent resources (no discovery)

A number of requires-interfaces: describe the properties expected by external services discovered at run-time

An event in the orchestration might trigger the discovery for one requires-interface
An **Service Module** is published, discovered and invoked by a service requester.

One **provides-interface**: a description of the properties provided to the requester.

**SLA**: description of SLA constraints (relied on c-semirings)


*Semiring-based constraint satisfaction and optimization.*

Journal of the ACM (JACM) 44(2): 201-236
On Road Repair & SRML

compatibility

entailment
Language of Interactions

- Description of behaviour based on
  - events
  - connectives (branching time logic with linear past)
- Events concern the occurrence of interactions
- Several related events may be associated with an interaction name
**Orchestration**

**BUSINESS ROLE** Orchestrator(carID:vehicleId)

**INTERACTIONS**

- **tll** handleEngineFailure
- **ask** currentLocation:location
- **s&r** askUsrDetails
  - cust:customerId
  - card:paydata
  - workRelated:boolean
  - destination:location
  - apointmentTime:time
- **s&r** bookGarage
  - collectPnt:location
  - servPrice:moneyVal
  - card:paydata

... 

**ORCHESTRATION**

```plaintext
local  s: [INIT, FAILURE_DETECTED, ..., FINAL],
       l: location

transition StartProcess
   triggeredBy handleEngineFailure
   guardedBy s=INIT
   effects s'=FAILURE_DETECTED
   l'=currentLocation
   sends askUsrDetails

transition getContextData
   triggeredBy askUsrDetails
   guardedBy s=FAILURE_DETECTED
   effects s'=CONTEXT_RECEIVED
   sends bookGarage
   bookGarage.collectPnt=l
```
**BUSINESS PROTOCOL**  Garage is

**INTERACTIONS**

- `r&s` acceptBooking
  - ` véhiculed` : `vehicleId`
  - `collectPnt:location`
  - `servicePrice:moneyVal`
  - `card:paydata`

- `snd` confirmation

**BEHAVIOUR**

- `initiallyEnabled` acceptBooking
- acceptBooking `✓` **ensures** confirmation


SLA
Ongoing work

- model checking
- cows
Ongoing work

- hybrid system as plugs? quote automotive languages