Agenda

- Background: SENSORIA, SRML, COWS
- The aim
- The architecture of the implementation
- An example
- Conclusion/future work
Software engineering for SOC

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.

SOC vs CBD: our view
- There is no “system” a-priori but an evolving configuration
- Services add a layer of abstraction over a component infrastructure

The different languages and formalisms developed in SENSORIA represent each a number of aspects of SOC from different perspectives: none of them aims to be “complete”
SRML & COWS

- **SRML**: architectural
- **SRML** is declarative:
  - it supports under-specification
  - it abstracting from how the middleware provides its functionalities

- **COWS**: behavioural (lower level of abstraction)
- **COWS** its primitives explicitly model
  - orchestration
  - the functionalities provided by the middleware (e.g., publication, discovery, correlation)

**SRML (overview)**

- Module: a number of (different types of) nodes pairwise connected by edges
- Each node $n$ has a signature $sign(n)$
- Each node has a (different type of) behavioural interface. All behavioural interfaces are defined in terms of events
SRML & COWS

- **SRML**: architectural
- **SRML** is declarative:
  - it supports under-specification
  - it abstracting from how the middleware provides its functionalities
- **COWS**: behavioural (lower level of abstraction)
- **COWS** its primitives explicitly model
  - orchestration
  - the functionalities provided by the middleware (e.g., publication, discovery, correlation)

**COWS (overview)**

The invoke/receive specifies a service and an operation

Kill-protection allow to implement compensation

Pattern-matching -> correlating, by means of their same contents, different interactions logically forming a same 'session'
The Aim

- The **implementation** of those SRML models which are not underspecified in COWS...
  - ...distill minimal set of assumptions made on the middleware
  - ...provides SRML with an operational semantics
  - ...middleware modelled in a way that is operational but still abstract with respect to implementation issues with actual technologies
Architecture

- **SRML**

- The implementation of a SRML module into COWS is modular

- **COWS**
  Module\(^{(1,2,3)}\) | Middleware\(^{(4)}\) | Environment\(^{(5)}\)
  Module\(^{(1,2,3)}\)=Factory\(^{(1a)}\).(InstanceHandler \(^{(1b)}\) | Orchestrator \(^{(2)}\) | DiscoveryHandler \(^{(3)}\))

- Creates different instances of a service, each equipped with one instance handler
- Implements message correlation to support multiple instances of the same service

http://rap.dsi.unifi.it/cows/
http://rap.dsi.unifi.it/cows/papers/SRML2COWS.pdf
Discovery Process

1. intGA becomes true and triggers the discovery of GA
2. R_GA sends GA to Broker
3. Broker returns - the id of the best match for GA - information on the mapping between the names of GA and CR
4. R_GA sends a message to the factory RepairService to create a service instance

GA • trigger?(id).
( Broker • disc!(OnRoadRepair, id, "Garage is ...", carUserSLAconstraints).
  | [x_p, x_acceptBooking] OnRoadRepair • GA?(id, x_p, x_acceptBooking).
  | [id_ext] (x_p • create!(OnRoadRepair, id_ext).
  | x_p • bindingInfo!(id_ext, acceptBookingResp).
  | * [x_info] GA • acceptBooking?(id, ⊥, x_info).
  | (x_p • x_acceptBooking!(id_ext, ⊥, x_info).
  | [x_servicePrice] OnRoadRepair • acceptBookingResp?(id_ext, ⊥, x_servicePrice).
  | [x_servicePrice] OnRoadRepair • acceptBookingResp?(id_ext, ⊥, x_servicePrice).
  | [id_intra] (ProvidesInt | RequiresInt | Wires | Components)

R_GA

G_CR
Conclusion/Future Work

• We provided an implementation of SRML modules into COWS

• The aim was to provide SRML primitives with an operational semantics and clarify the assumptions on the middleware

• Focus on dynamic aspects, simplification of some static aspects

• An editor for SRML (Eclipse plugin) has been developed which represents the SRML metamodel as an EMF tree

• Ongoing work - a graphical editor for COWS (based on GMF) with an integrated interpreter

• The automation of the transformation, for example relying on the editors (by means of a transformation between the respective meta-models) would allow SRML models to benefit from the tools for analysis and reasoning made available by COWS:
  
  • a type system to check confidentiality properties [FSEN07], a temporal logic and a model checker to verify functional properties [FASE08], a static analysis to establish properties of the flow of information between services [SAS08], a stochastic extension to enable quantitative reasoning on service behaviours [ICSOC07], a symbolic representation of the operational semantics [PLACE08], bisimulation-based observational semantics to check interchangeability of services [submitted]
Thank you