A semi-automatic, behavioral mediation approach based on models@runtime

Runze HAO, Brice MORIN, Arne J. BERRE

www.remics.eu
import "API1.thingml"
import "API2.thingml"

thing Mediator includes API1, API2{

  property password: String // Use to store the password

  provided port Client {
    receives log
    sends ack
  }

  required port Server {
    sends login, password
    receives access
  }

  statechart Authentication init Ready {

    state Ready {
      on entry { print "Mediator: Ready, Waiting for credential ..." }

      transition -> CheckPassword
        event e: Client?log
        action do
          password = e.password
          Server!login(e.login)
        end
    }

    state CheckPassword {
      on receipt success {
        Server!postAuth
        Ready!
      }
    }

    state PostAuth {
      on receipt access {
        Server!permit
        Ready!
      }
    }

    state Permit {
      on receipt login {
        Server!grant
        Ready!
      }
    }

    state Grant {
      on receipt password {
        Server!deny
        Ready!
      }
    }

    state Deny {
      on receipt login {
        Server!require
        Ready!
      }
    }

    state Require {
      on receipt password {
        Server!deny
        Ready!
      }
    }

    state Error {
      on receipt {
        Server!error
        Ready!
      }
    }

  }
}
Conclusion & Perspective

• Service Interoperability
  ➔ difficult (even impossible?) to fully automate

• Our approach
  ➔ a set of light weight tools to help and guide designers
  ➔ tools and resulting mediators deployed on M@RT platform