Specification and Analysis of Service Oriented Architectures within the Calculus of Communicating Sequential Processes (CSP)

by

Abiar S. Al-Homaimeedi

A thesis submitted in partial fulfilment for the
degree of Doctor of Philosophy

Department of Informatics
Kings College London, University of London

Jan, 2015

A

soaCSP Code

```
-- Service-Oriented Architecture CSP (soaCSP)
-- Abiar S. Al-Homaimeedi, DEC 2014
-- This code introduces soaCSP into CSPM.
-------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------------------
-----------------------------------------START CODE----------------------------------------------
-------------------------------------------------------------------------------------------------------

-- Initialisations
---------------------
proN=10
i=0
V={0..sd}
V1={0..proN}

-- type defentions
------------------------
nametype ses= pnames.V1.V1
datatype com= diam | js | fp
datatype Marks = plus | minus | non

-- channel defentions
------------------------------
channel l,new  : V1

channel session: ses
channel cl: eeh
channel close
channel chUpdate: Marks.Mobch
channel in , out: Se
```

```
-------------------------------------------------------------------------------------------------

-- construct labels and sequences sets to the depth sd
-------------------------------------------------------

eeh1={ <>,<l.i> | i<-V1 }

eeh2={ <>,<l.i> | i<-V1 }

eeh = labelseq1(eeh2,sd)

labelseq1(seq,num)=if ((num-1) <= 0) then seq else
labelseq(union({<l.i>^G | i<-V1 , j<-V1 , G<-seq },seq) ,num-1)

labelseq(seq,num)=if ((num-1) <= 0) then seq else
labelseq(union({<l.i>^G | i<-V , j<-V1 , G<-seq },seq) ,num-1)

labels={|l.i | i<-V1  |}

labelf(labels,num)=if ((num-1) <= 0) then labels else
labelf(union({l.i.G | i<-V , G<-labels },labels) ,num-1)

labells(e)= {| G.ee, ee | G<-labelf(labels,sd) , ee<-e |}

EWL=  { G.ee | G<-labelf(labels,sd) , ee<-ourevents }

labonly=labelf(labels,sd)

Se={x.y | x<-ev ,y<-extensions(x) }

-------------------------------------------------------------------------------------------------

-- label generator
-------------------

labelGen(i) = new!(i+1) -> labelGen(i+1)


-- extra session communication
-------------------------------

diamonds(s) = diam.s

up(s)= fp.s


-- published service
-------------------------
```

```
publish(p,N) =  ( diamonds(new)?i  -> ( if (member(N,{z.z' | z<-pnames, z'<-V}))
                                        then diamonds(slabel(session,N))!i ->  SKIP
                                        else diamonds(slabel(session,N))!(0.i) ->  SKIP )  )


-- Invocation service
---------------------

inv(q,N,f) =  (
               diamonds(slabel(session,N))?((x.y)@@i) ->
                   (  ((label(l.y,q) [| {| cl |}|]  level))
                      [| union(union({|close|},{|l.y.G | x<-ourevents, G<-f , G!=(fp.x) |}),{| x |
x<-ourevents, G<-f , (G == up(x)) |}) |]
((label(l.y,process(N)) [| {| cl |}|]  level))  )
                )


-- Joint service
---------------------

jinv(q,N,f) =  (   diamonds(slabel(session,N))?((x.y)@@i) ->
                 ( q   [| union({|close|},{| x | x<-f|}) |]  process(N)  )   )


--persistent service
--------------------

perServ(p,N) = ||| x:{0..pn} @ publish(p,(N.x))


-- Sessios names
------------------

slabel(x,i) =  x.i


-- events labelling
---------------------

elabel(x,h) =  {|l.h.G | G<-x |}



-- process labelling
--------------------

label(L,pro) = pro [[ (cl.ll)<-(cl.(<L>^ll)) , x<-(L.x) , (diam.x)<-(diam.x)  , (fp.x)<-x
|   x<-diff(ourevents,{|cl,session|}) , ll<-eeh1 ]]
```

```
extracom(pro) = pro [[ (diam.x)<-x , (fp.x)<-x, (diam.m)<-m , (fp.m)<-m | x<-ourevents, m<-Mobch ]]


--termination
-------------------


SSKIP= cl!<> -> SKIP

level= cl?s->levelup(cl.s)

levelup(cl.s)= if ((cl.s)==(cl.<>)) then SKIP else head(s).close -> levelup(cl.tail(s))

LISTEN= (close -> STOP) [] SKIP

term_Hnr(p,q)= (p [| {| close |} |] (close -> q))


-- Mobile requlator
----------------------
reg(A) = ([] x:{| y |  y<-diff(diff(Events,union(A,{|in ,out|})),{|chUpdate|})  |} @ x -> reg(A))[]
            (chUpdate?dir?x -> if (dir==plus)
then reg(union(A,{|ch(x)|})) else (if (dir==minus) then reg(diff(A,{|ch(x)|}))
 else  reg(A)) )   --  [] SKIP


-- update interface set dynamically
---------------------------------------
parmob(p,q)= (p [| diff(Events,{|in ,out, chUpdate|}) |] q)

Mobile(p,A)= (reg(A) [| {|chUpdate|} |] p) \{|chUpdate|}


-- buffered Sigma
-----------------
bufferedEvents =  let bf(a,s) =  if null(s) then  (out.a?x -> bf(a,<x>)) [] SKIP
      else     (in.a!head(s) -> bf(a,tail(s)) []
#s<N & ( out.a?x -> bf(a,s^<x>)) ) [] SKIP

   within (||| x:ev @ bf(x,<>))



-- asynchrouns function (installing buffered Sigma)
-------------------------------------------------------
asy(p)=  p  [| {| in, out  |}|]   bufferedEvents
```

```
-------------------------------------------------------------------------------------------------
-----------------------------------------END CODE-----------------------------------------------
-------------------------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------------
-----------------------------------------START CASESTUDY----------------------------------------
-------------------------------------------------------------------------------------------------

-- system intialising
--------------------

N=2   -- buffer capacity
ev={Clist, Mlist, thisuser, thisEuser}

  -- asynchronous channel set

pn=0    -- number of parallel instances of a persistent service
sd=2   -- the depth of levels

channel mthisuser, mthisEuser --define mobile channels


Mobch={|mthisuser, mthisEuser|} -- alises for mobile channel

ch(mthisuser)= thisuser
ch(mthisEuser)= thisEuser

datatype pnames =  BankPortal | Authentication| CreditRequest | TransferMoney | BalanceValidation |
   SecAnalysis | Rating | ClerkList | MangerList -- services names

-- assign names to published services

process(BankPortal)= BankPortalp
process(Authentication)= Authenticationp
process(CreditRequest)= CreditRequestp
process(TransferMoney)= TransferMoneyp
process(BalanceValidation)= BalanceValidationp
process(SecAnalysis)= SecAnalysisp
process(Rating)= Ratingp
process(ClerkList)= ClerkListp
process(MangerList)= MangerListp

-------------------------------------------------------------------------------------------------

-- 1. Cridet  Request senario
---------------------------------

ourevents={thisuser, thisuser, notValid, getDBresult, returnRate,assessRisk, Mdecision,analysed,
Valid,validBalance,requestDenied, RejectOffer, AcceptOffer,login, request, ValidID,
```

```
searchDB, ValidateBC, getRate, calculaterate, ordertransfer,addtoClist,
    addtoMlist,Clist,Mlist,offer, transMoney, transferDate, analysis,scheduletransfer,
EvalauteRisk, secanalysisE, evaluateM, evaluateC,in,out,l,new,session,chUpdate,
notvalidBalance,cancel}


datatype riskDT = HIGH | LOW
datatype ErrorMSGDT = InvalidName | InvalidPass
datatype serDT = exists | notexists
datatype RDT= AAA | BBB | CCC
datatype decDT= approve | disapprove
datatype valDT= OK | notOK

channel notValid :ErrorMSGDT
channel getDBresult: serDT
channel returnRate: RDT
channel assessRisk, thisuser: riskDT.V
channel Mdecision, thisEuser: decDT
channel analysed: valDT

channel cancel,Valid,validBalance,requestDenied,RejectOffer, AcceptOffer, notvalidBalance

channel login, request, ValidID, searchDB, ValidateBC, getRate, calculaterate, ordertransfer: V.V
channel addtoClist: V.V.RDT
channel addtoMlist: V.V.RDT.V

channel Clist: Mobch.V.V.RDT
channel Mlist: Mobch.V.V.RDT.V

channel offer, transMoney, transferDate, analysis: V

channel scheduletransfer, EvalauteRisk, secanalysisE, evaluateM, evaluateC


----------------------------------------

client= (
 login!2!2 ->
(
(
   (notValid?ErrorMSG -> SSKIP) []
(
  Valid -> (
       jinv(  (
       let CRrec = (
   ( request!2!2 ->
(
(
(validBalance -> (( userOffer [] (requestDenied -> SSKIP)) |~| (cancel->SKIP)) )
 [] (notvalidBalance -> CRrec)
```

```
)
|~| (cancel->SKIP) )
   )
   |~| (cancel->SKIP) )   within CRrec
)
,CreditRequest,{ request, validBalance, requestDenied, notvalidBalance, offer, AcceptOffer,
 RejectOffer, transferDate }) |~| (cancel->SKIP))
)
)
|~| (cancel->SKIP) )
|~| (cancel->SKIP) )
```

```
BankPortalp = (
      login?ID?pass -> inv( ( ValidID!ID!pass ->( ( (up(notValid)?ErrorMSG -> SSKIP) []
(up(Valid) -> SSKIP) ) [] (up(cancel)->SKIP)) ),
Authentication, { ValidID, up(notValid), up(Valid) })
      [] (up(cancel)->SKIP))
```

```
CreditRequestp = (
let Rrec=(
 ( request?amount?SEC ->
      inv(
 (
ValidateBC!amount!SEC ->
(
(
(
     up(validBalance) ->
jinv(
           (
getRate!amount!SEC -> returnRate?R ->
if (R == AAA) then generateOffer(amount)
           else if (R == CCC) then ((up(requestDenied) -> SSKIP)
[] (up(cancel)->SKIP))
           else ProcessRequest(amount,SEC,R)
    )
, Rating, {getRate,returnRate})
) [] (up(notvalidBalance) -> Rrec)
       ) [] (up(cancel)->SKIP)
)
  )
  ,BalanceValidation,{ ValidateBC, up(validBalance), up(notvalidBalance)} )
  ) [] (cancel->SKIP)
  )  within Rrec
  )
```

```
userOffer= offer?2 -> (
(
( AcceptOffer ->
( (transferDate?date -> SSKIP) |~| (cancel->SKIP) )
        |~| ( RejectOffer -> SSKIP)
)
)
|~| (cancel->SKIP))

generateOffer(amount)= up(offer)!amount ->( ( ( up(AcceptOffer) ->
(jinv( (transMoney!amount -> SSKIP), TransferMoney, {transMoney})
[] (up(cancel)->SKIP)) )
 [] ((up(RejectOffer) ->  SSKIP)[] (up(cancel)->SKIP)) ) [] (up(cancel)->SKIP))


Authenticationp = ValidID?ID?pass -> diamonds(searchDB)!ID!pass -> diamonds(getDBresult).exists ->
          if (exists == exists)
then ( (up(Valid) ->  SSKIP) [] (up(cancel)->SKIP))
          else ((up(notValid)!InvalidName -> SSKIP) [] (up(cancel)->SKIP))



TransferMoneyp = transMoney?amount -> scheduletransfer ->
((up(transferDate)!2 -> SSKIP) [] (up(cancel) -> SKIP))


BalanceValidationp = ValidateBC?amount?SEC  ->
jinv(  (analysis!SEC  -> analysed?val -> if (val == OK)
then (( up(validBalance) ->  SSKIP) [] (up(cancel)->SKIP))
else (( up(notvalidBalance) -> SSKIP)[] (up(cancel)->SKIP)) ),
 SecAnalysis, {analysis,analysed})

SecAnalysisp = analysis?SEC  -> secanalysisE -> analysed!OK -> SSKIP

Ratingp = getRate?amount?SEC -> calculaterate.amount.SEC -> returnRate!BBB -> SSKIP


ProcessRequest(amount,SEC,R)= jinv(
(( addtoClist!amount!SEC!R -> assessRisk?risk?eval -> if (risk == HIGH)
then ((up(requestDenied) -> SSKIP) [] (up(cancel)->SKIP))
           else jinv( ((addtoMlist!amount!SEC!R!eval ->
                   Mdecision?dec -> if (dec == approve)
           then generateOffer(amount)
                                  else ((up(requestDenied) -> SSKIP)
                   [] (up(cancel)->SKIP)) )   [] (up(cancel)->SKIP))
   ,MangerList,{addtoMlist,Mdecision})
  ) [] (up(cancel)->SKIP))
,ClerkList,{|addtoClist,assessRisk|})
```

```
ClerkListp = addtoClist?amount?SEC?R -> (
diamonds(out.Clist)!mthisuser.amount.SEC.R ->
(
      (diamonds(in.thisuser)?risk?eval -> assessRisk!risk!eval -> SSKIP) []
( up(cancel) -> let del= (diamonds(in.Clist)?x.amount.SEC.R ->
     if (x == mthisuser) then SKIP else
(diamonds(out.Clist)!x.amount.SEC.R -> del)) within del
)
)
[] (up(cancel)->SKIP))


Clerk= in.Clist?x.amount.SEC.R -> evaluateC  -> out.ch(x)!LOW!2  -> Clerk


MangerListp = addtoMlist?amount?SEC?R?eval -> (
diamonds(out.Mlist)!mthisEuser.amount.SEC.R.eval ->
(
(diamonds(in.thisEuser)?dec -> Mdecision!dec -> SSKIP) []
(up(cancel) -> let mdel= (diamonds(in.Mlist)?x.amount.SEC.R.eval ->
 if (x == mthisEuser) then SKIP else
(diamonds(out.Mlist)!x.amount.SEC.R.eval -> mdel)) within mdel )
)
[] (up(cancel)->SKIP))

Manger= in.Mlist?x.amount.SEC.R.eval ->  evaluateM  -> out.ch(x)!approve -> Manger

---------------------------------

system = (
extracom(perServ(TransferMoneyp,TransferMoney)) |||
 extracom(perServ(BalanceValidationp,BalanceValidation)) |||
extracom(perServ(SecAnalysisp,SecAnalysis)) |||
extracom(perServ(Ratingp,Rating)) |||
extracom(perServ(ClerkListp,ClerkList)) |||
extracom(perServ(MangerListp,MangerList))
||| extracom(perServ(BankPortalp,BankPortal)) |||
extracom(perServ(Authenticationp,Authentication)) |||
extracom(perServ(CreditRequestp,CreditRequest))
) [| {|session, cancel|} |]
extracom(inv(client,BankPortal,{| login, Valid, notValid |}))

finance= asy(system ||| (Manger ||| Clerk) )  [|{| new |}|]  labelGen(0)


-------------------------------------------------------------------------------------------------
-----------------------------------------END CASESTUDY-------------------------------------------
-------------------------------------------------------------------------------------------------
```