dipartimentoinformatica

Università degli Studi dell'Aquila

Source and Target Role Models for Antipattern-based Model Refactoring

Davide Arcelli, Vittorio Cortellessa, Catia Trubiani

Technical Report TRCS 005/2011

Dipartimento di Informatica
Università degli Studi dell'Aquila
Via Vetoio Loc. Coppito
I-67010 L'Aquila, Italy

http://www.di.univaq.it

Technical Report TRCS Series

Source and Target Role Models for Antipattern-based Model Refactoring

Davide Arcelli, Vittorio Cortellessa, Catia Trubiani

Dipartimento di Informatica, Università dell'Aquila
67010, L'Aquila, Italy
davide.arcelli@yahoo.it, {vittorio.cortellessa, catia.trubiani}@univaq.it

The aim of this technical report is to show the set of Source Role Models (SRMs) and Target Role Models (TRMs) that have been built as support for the antipattern-based model refactoring. Table 1 reports our current repository of Role Models: the first column contains the software performance antipatterns [Smith-Williams-2003] we considered, whereas the second column contains the SRM-TRM pairs that have been defined for the antipattern they refer to. Source and Target Role Models are applied to a case study in the e-commerce domain. This work has been inspired from a master thesis, for more details please refer to [Arcelli-2011].

Antipattern	$(SRM_i^{antipattern}, TRM_i^{antipattern})$		
Blob	$(SRM_1^{Blob}, TRM_1^{Blob})$	$(SRM_2^{Blob}, TRM_2^{Blob})$	$(SRM_3^{Blob}, TRM_3^{Blob})$
Concurrent Processing Systems	$(SRM_1^{CPS}, TRM_1^{CPS})$		
Empty Semi Trucks	$(SRM_1^{EST}, TRM_1^{EST})$		
Pipe & Filter	$(SRM_1^{P\&F}, TRM_1^{P\&F})$	$(SRM_2^{P\&F}, TRM_2^{P\&F})$	

Table 1 – Source Role Models (SRMs) and Target Role Models (TRMs) for software performance antipatterns

$(SRM_1^{Blob}, TRM_1^{Blob})$

Description of the refactoring: split the Blob software entity instance $swEx$ in n software entity instances; the new instances are deployed on the same processing node on which $swEx$ was deployed.

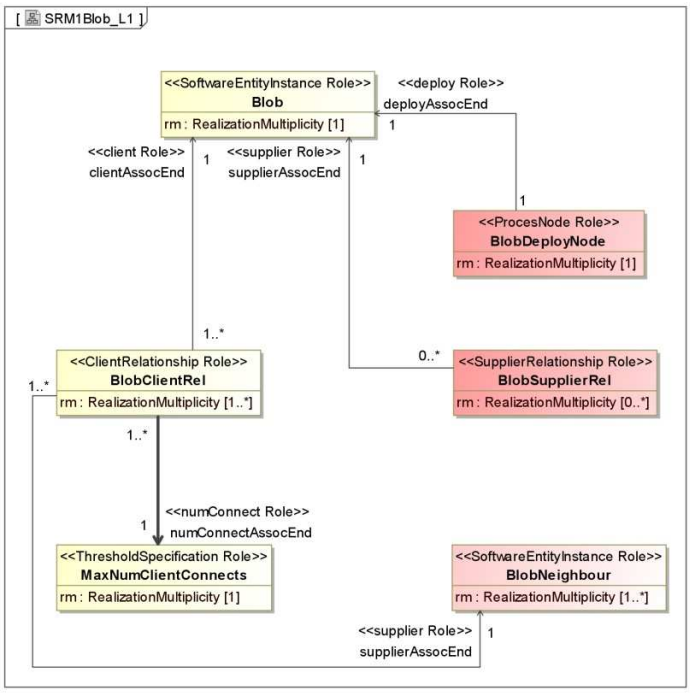


Figure 1 - SRM_1^{Blob}.

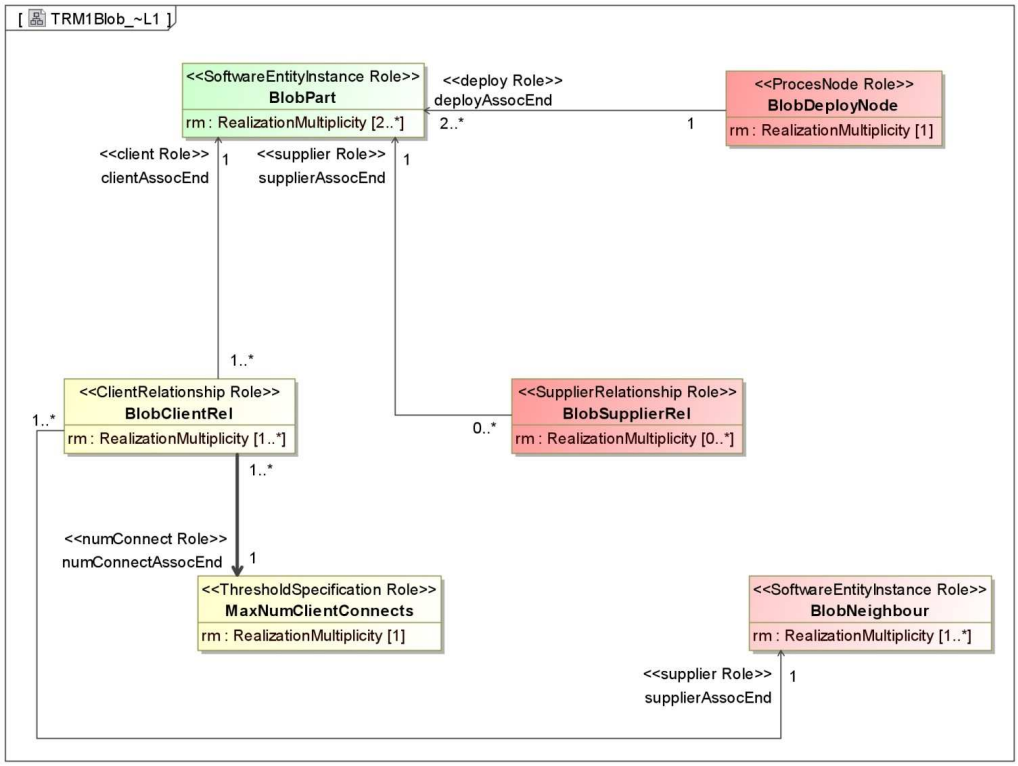


Figure 2 - TRM_1^{Blob}.

SRM_1^{Blob}

```
context Blob inv maxNumConnects:
BlobClientRel.allInstances() -> size() >= MaxNumClientConnects.allInstances() ->
first().value

context Blob inv dependenciesAndThreshold:
(BlobSupplierRel.allInstances() -> collect(getDependencies(target -> first()) ->
excluding(t | t.OclIsTypeOf(AtomicOperation)) -> size()) -> max()) <=
(MaxNumConnects.allInstances() -> first().value - 1))
```

Definition of the contextual elements

```
context Blob inv blobSupplierRel:
BlobSupplierRel.allInstances() = SupplierRelationship.allInstances() ->
select(sr | sr.source = self)

context Blob inv blobNeighbour:
BlobNeighbor.allInstance() = SoftwareEntityInstance.allInstances() -> select(sei
| BlobClientRel.allInstances() -> exists(cr | cr.supplier = sei) or
sei.required -> exists(i | Blob.allInstances() -> first().operations ->
includes(i)))

context Blob inv blobDeploy:
BlobDeployNode.allInstances() = ProcesNode.allInstances() -> select(pn |
pn.deploy -> includes(self)) -> first()
```

TRM_1^{Blob}

```
context BlobPart inv maxNumConnects:
BlobClientRel.allInstances() -> select(cr | cr.client = self) -> size() <
MaxNumConnects.allInstances() -> first().value

context BlobPart inv numBlobParts:
BlobPart.allInstances() -> size() = ((M::BlobSupplierRel.allInstances() ->
collect(getDependencies(target -> first()) -> excluding(t |
t.OclIsTypeOf(AtomicOperation)) -> size()) -> sum()) /
(M::MaxNumConnects.allInstances() -> first().value - 1)).ceiling()

context BlobPart inv dependencies:
let BlobPartSupRelsOps : Set(CallingOperation) = BlobSupplierRel.allInstances()
-> select(sr | self.operations -> includes(sr.target -> first())) ->
collect(target) in BlobPartSupRelsOps -> notEmpty() implies BlobPartSupRelsOps -
> forAll(op | getDependencies(op) -> forAll(depOp | self.operations ->
includes(depOp) or self.required -> includes(depOp)))
```

$(SRM_2^{Blob}, TRM_2^{Blob})$

Description of the refactoring: split the Blob software entity instance $swEx$ in n software entity instances; the new instances are re-deployed on the less used processing nodes.

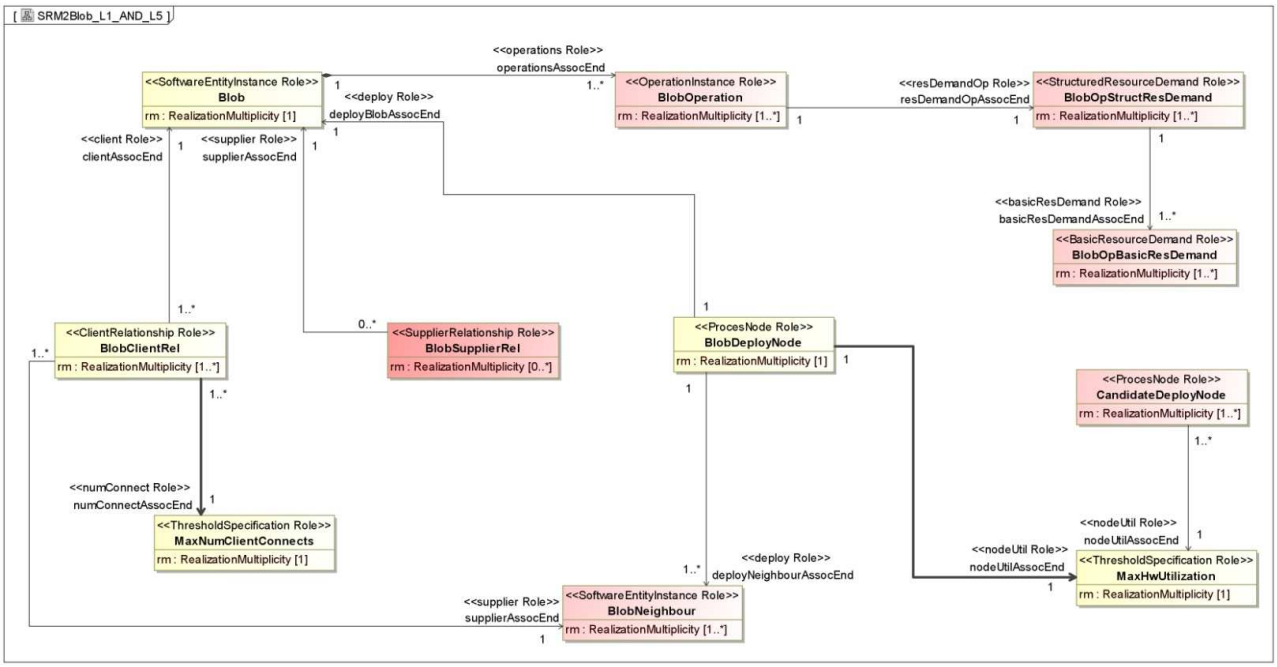


Figure 3 – SRM_2^{Blob}.

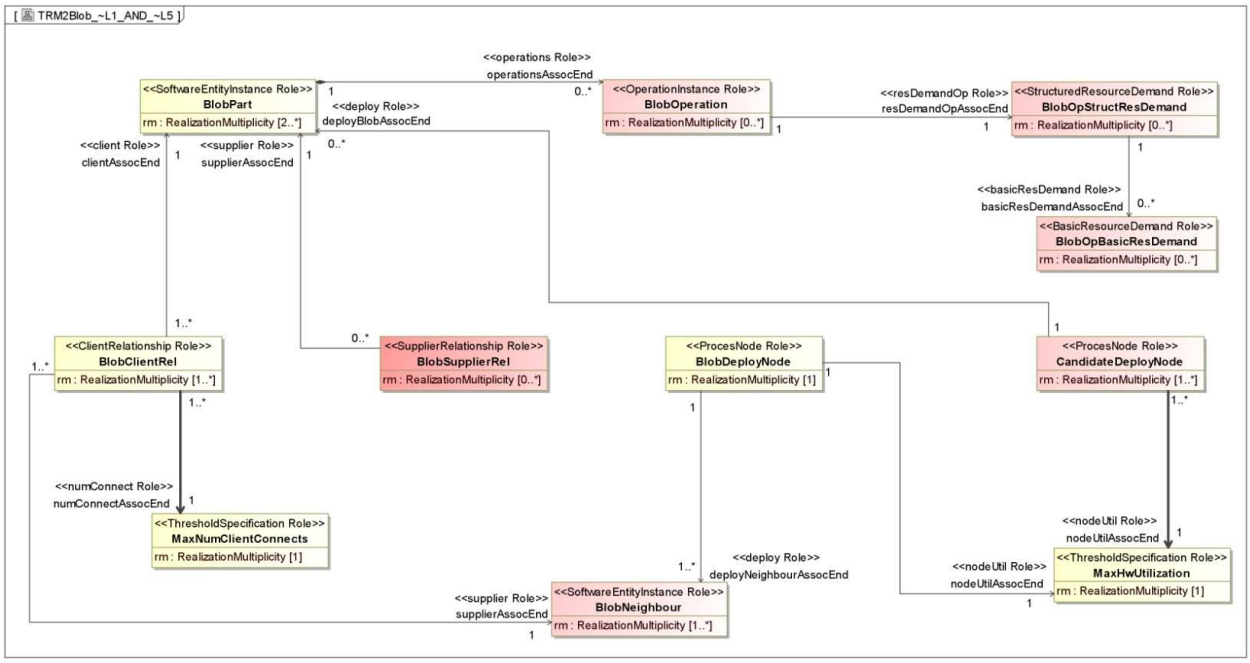


Figure 4 – TRM_2^{Blob}.

SRM_2^{Blob}

```
context Blob inv maxNumConnects:
BlobClientRel.allInstances() -> size() >= MaxNumClientConnects.allInstances() ->
first().value

context BlobDeployNode inv maxHwUtilization:
self.maxDevicesUtil >= MaxHwUtilization.allInstances()-> first().value

context Blob inv dependenciesAndThreshold:
(BlobSupplierRel.allInstances() -> collect(getDependencies(target -> first()) ->
excluding(t | t.OclIsTypeOf(AtomicOperation)) -> size()) -> max()) <=
(MaxNumConnects.allInstances() -> first().value - 1))
```

Definition of the contextual elements

```
context Blob inv blobOps:
BlobOperation.allInstances() = self.provided -> union(self.ownedOperation)

context Blob inv blobOpStructResDemand:
BlobOpStructResDemand.allInstances() = StructuredResourceDemand.allInstances() -
> select(structResDem | BlobOperation.allInstances() -> collect(resDemand) ->
includes(structResDem))

context Blob inv blobOpBasicResDemand:
BlobOpBasicResDemand.allInstances() = BasicResourceDemand.allInstances() ->
select(basicResDem | BlobOpStructResDemand.allInstances() ->
collect(basicResDemand) -> includes(basicResDem))

context Blob inv blobSupplierRel:
BlobSupplierRel.allInstances() = SupplierRelationship.allInstances() ->
select(sr | sr.source = self)

context Blob inv blobNeighbour:
BlobNeighbour.allInstance() = SoftwareEntityInstance.allInstances() ->
select(sei | BlobClientRel.allInstances() -> exists(cr | cr.supplier = sei) or
sei.required -> exists(i | Blob.allInstances() -> first().operations ->
includes(i)))

context Blob inv candidateBlobPartsDeploy:
let bitRateTypes : BitRateType = BitRateType.allInstances() -> first() in
CandidateDeployNode.allInstances() = ProcesNode.allInstances() -> excluding(pn |
pn.oclIsTypeOf(BlobDeployNode)) -> select(pn |
pn.maxDevicesUtil < MaxHwUtilization.allInstances()->first().value and
NetworkLink.allInstances() -> exists(nl | nl.endNode ->
includes({SwRxDeployNode.allInstances() -> first(), pn}) and (nl.bitRate =
bitRateTypes::Kb_sec or (nl.bitRate = bitRateTypes::Mb_sec and nl.capacity <=
100)))) -> sortedBy(maxDevicesUtil)
```

TRM_2^{Blob}

```
context BlobPart inv maxNumConnects:
BlobClientRel.allInstances() -> select(cr | cr.client = self) -> size() <
MaxNumConnects.allInstances() -> first().value

context CandidateDeployNode inv maxHwUtilization:
self.maxDevicesUtil < MaxHwUtilization.allInstances()-> first().value

context BlobPart inv numBlobParts:
BlobPart.allInstances() -> size() = ((M::BlobSupplierRel.allInstances() ->
collect(getDependencies(target -> first()) -> excluding(t |
t.OclIsTypeOf(AtomicOperation)) -> size()) -> sum()) /
(M::MaxNumConnects.allInstances() -> first().value - 1)).ceiling()

context BlobPart inv dependencies:
let BlobPartSupRelsOps : Set(CallingOperation) = BlobSupplierRel.allInstances()
-> select(sr | self.operations -> includes(sr.target -> first())) ->
collect(target) in BlobPartSupRelsOps -> notEmpty() implies BlobPartSupRelsOps -
> forAll(op | getDependencies(op) -> forAll(depOp | self.operations ->
includes(depOp) or self.required -> includes(depOp)))

context BlobPart inv BlobPartRedeploy:
let avHws : Sequence(CandidateDeployNode) = CandidateDeployNode.allInstances() -
> sortedBy(maxDevicesUtil) in
let orderedBlobParts : Sequence(BlobPart) = BlobPart.allInstances() ->
sortedBy(operations.resDemand.basicResDemand.value -> sum()) -> reverse() in
let modValue : Integer = orderedBlobParts -> indexOf(self).mod(avHws -> size())
in
if(modValue <> 0) then avHrs -> at(modValue).deploy -> includes(self) else avHrs
-> last().deploy -> includes(self) endif
```

$(SRM_3^{Blob}, TRM_3^{Blob})$

Description of the refactoring: refactor the communication pattern between the Blob software entity instance $swEx$ and the other software entity instances, thus to avoid excessive message traffic.

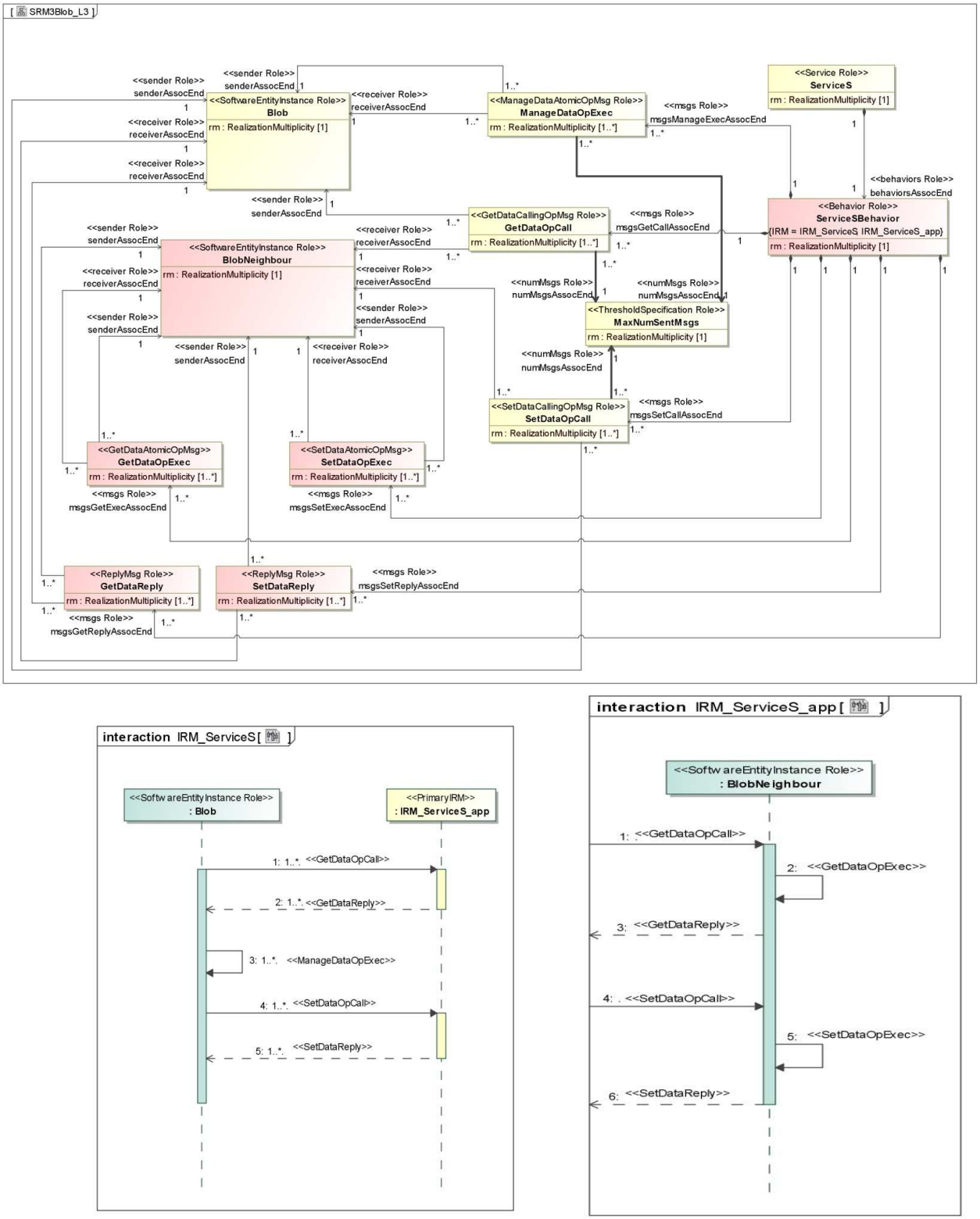
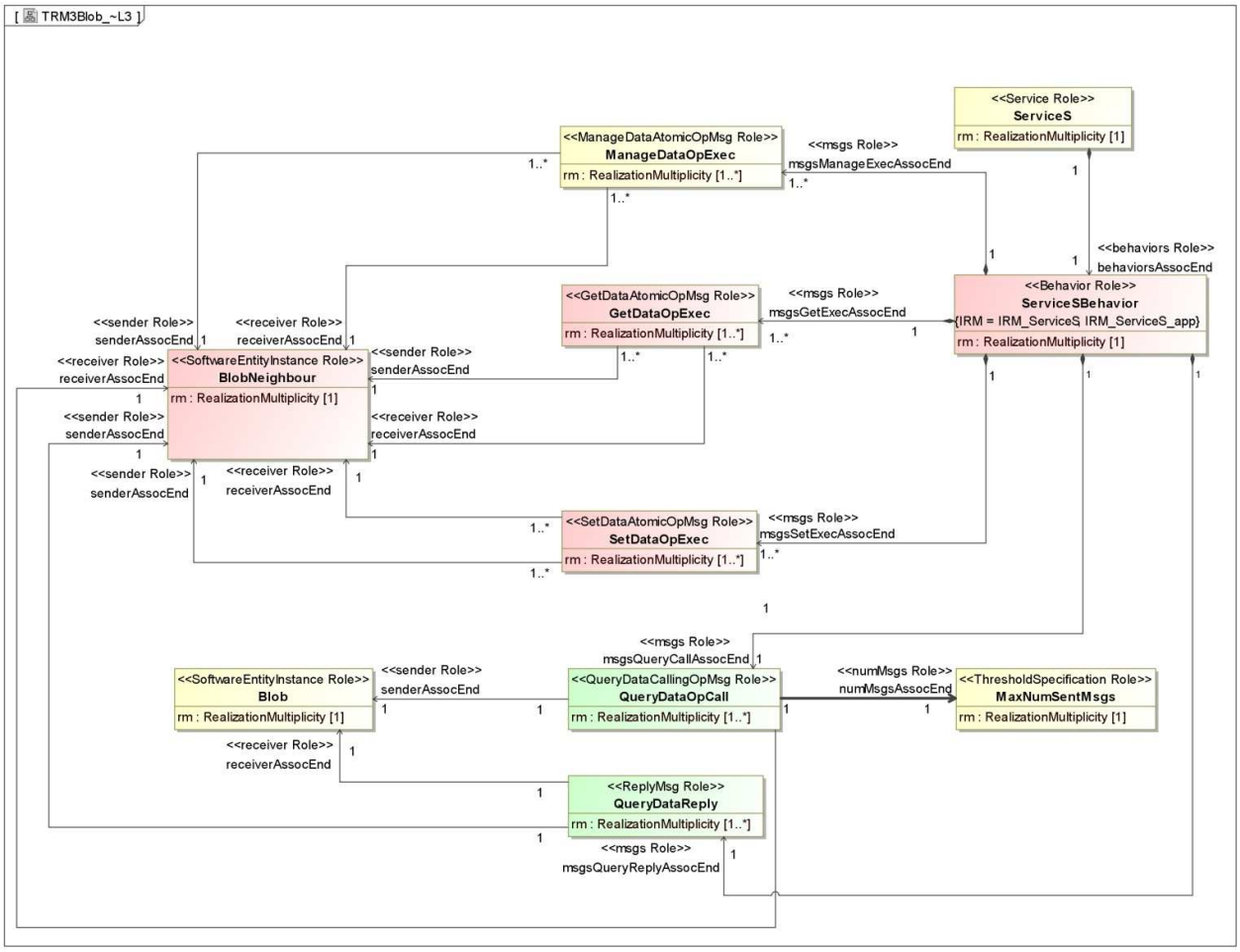
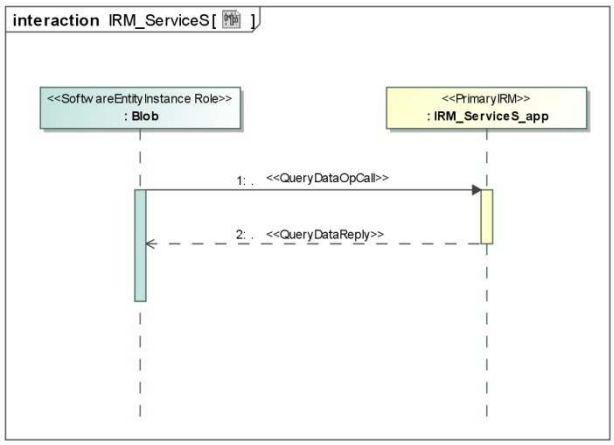


Figure 5 – SRM_3^{Blob}.





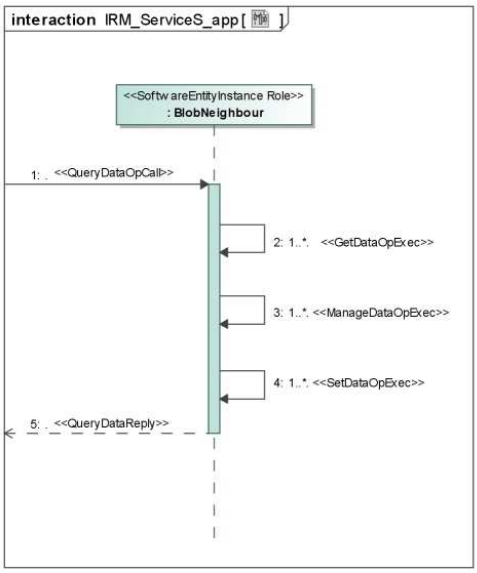


Figure 6 – TRM_3^{Blob}.

SRM_3^{Blob}

```
context ServiceSBehavior inv maxNumMsgs:
GetDataOpCall.allInstances() -> size() + ManageDataOpExec.allInstances() ->
size() + SetDataOpCall.allInstances() -> size() >= MaxNumSentMsgs.allInstances()
-> first().value
```

Definition of the contextual elements

```
context ServiceS inv serviceSBehaviour:
ServiceSBehavior.allInstances() = self.behaviors -> first()

context Blob inv blobNeighbor:
BlobNeighbour.allInstances() = GetDataOpCall.allInstances() -> first().receiver

context ServiceSBehavior inv sameData1:
GetDataOpCall.allInstances() -> forAll(getOpCall |
GetDataOpExec.allInstances() -> one(getOpExec | areInSameRealization(self.IRM,
getOpCall, getOpExec) and areOnSameData({getOpCall.operation,
getOpExec.operation})) and
GetDataReply.allInstances() -> one(getReply | areInSameRealization(self.IRM,
getOpCall, getDataReply) and getOpCall.replyMessage = getReply) and
ManageDataOpExec.allInstances() -> one(manOpExec |
areInSameRealization(self.IRM, getOpCall, manOpExec) and
areOnSameData({manOpExec.operation, getOpCall.operation})) and
SetDataOpCall.allInstances() -> one(setOpCall | areInSameRealization(self.IRM,
getOpCall, setOpCall) and areOnSameData({setOpCall.operation,
getOpCall.operation})))

context ServiceSBehavior inv sameData2:
SetDataOpCall.allInstances() -> forAll(setOpCall |
SetDataOpExec.allInstances() -> one(setOpExec | areInSameRealization(self.IRM,
setOpCall, sgetOpExec) and areOnSameData({setOpCall.operation,
setOpExec.operation})) and
SetDataReply.allInstances() -> one(setReply | areInSameRealization(self.IRM,
setOpCall, setReply) and setOpCall.replyMessage = setReply))
```

TRM_3^{Blob}

```
context ServiceSBehavior inv maxNumMsgs:
QueryDataOpCall.allInstances() -> size() < MaxNumSentMsgs.allInstances() ->
first().value

context ServiceSBehavior inv sameData1:
GetDataOpExec.allInstances() -> forAll(getOpExec |
ManageDataOpExec.allInstances() -> one(manOpExec |
areInSameRealization(self.IRM, getOpExec, manOpExec) and
areOnSameData({manOpExec.operation, getOpExec.operation})) and
SetDataOpExec.allInstances() -> one(setOpExec | areInSameRealization(self.IRM,
getOpExec, setOpExec) and areOnSameData({setOpExec.operation,
getOpExec.operation})) and
QueryDataOpCall.allInstances() -> one(queryOpCall |
areInSameRealization(self.IRM, getOpExec, queryOpCall) and
areOnSameData({queryOpCall.operation, getOpExec.operation})))

context ServiceSBehavior inv sameData2:
QueryDataOpCall.allInstances() -> forAll(queryOpCall |
QueryDataReply.allInstances() -> one(queryReply | areInSameRealization(self.IRM,
queryOpCall, queryReply) and queryOpCall.replyMessage = queryReply))
```

$(SRM_1^{CPS}, TRM_1^{CPS})$

Description of the refactoring: the most critical software entity instance is re-deployed from the over used processing node to the less used processing node.

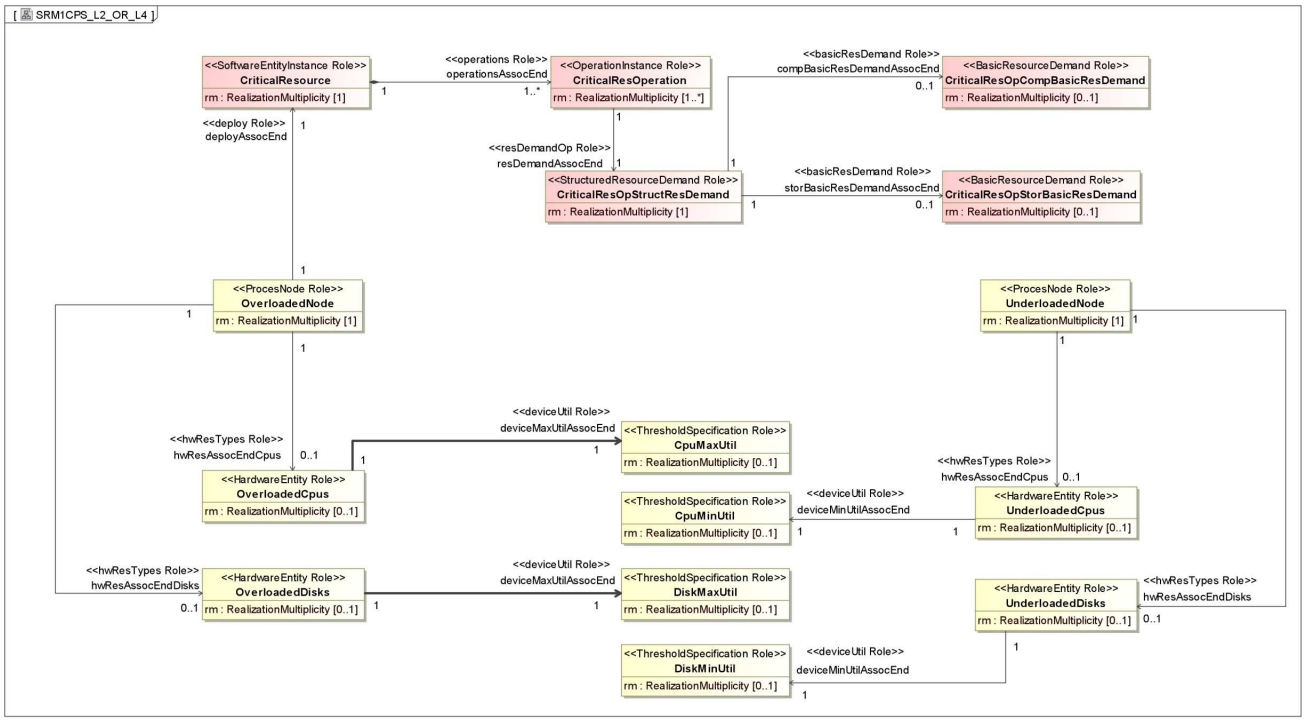


Figure 7 – SRM_1^{CPS}.

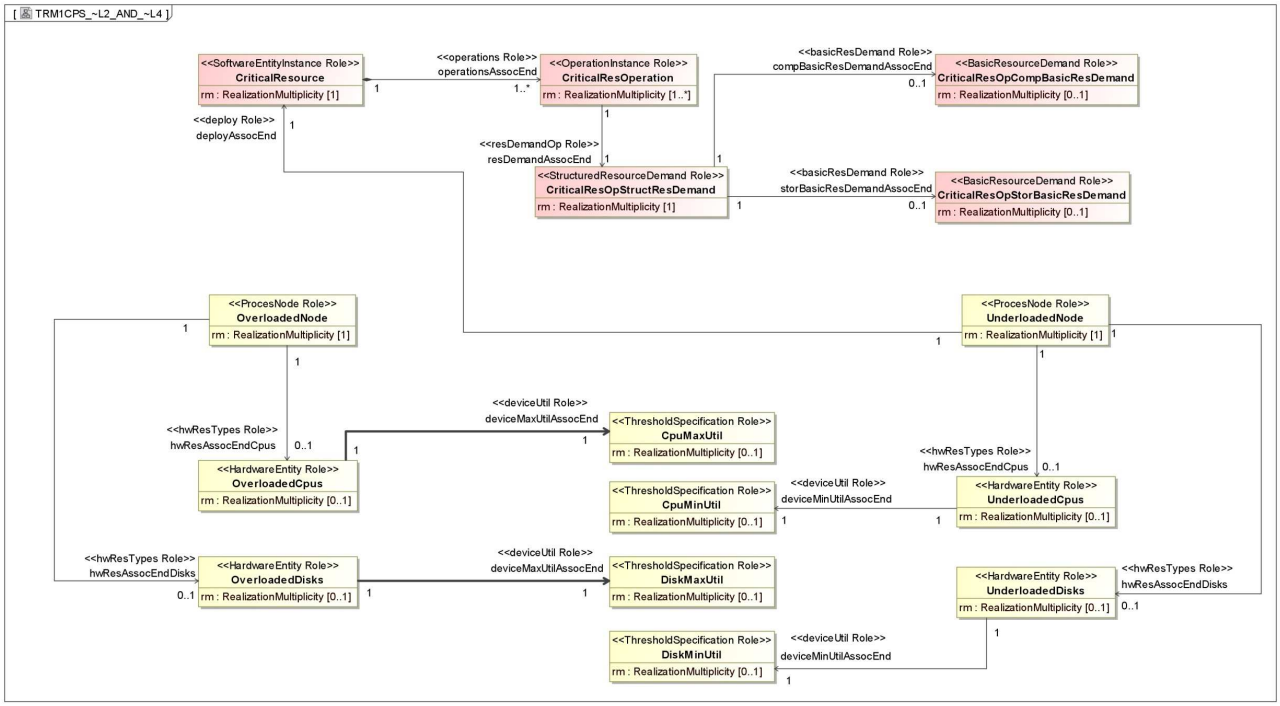


Figure 8 – TRM_1^{CPS}.

SRM_1^{CPS}

```
context OverloadedNode inv overloadedCpus_or_overloadedDisks:
self.hwRes -> select(hr | hr.oclIsTypeOf(OverloadedCpus) or
hr.oclIsTypeOf(OverloadedDisks)) -> notEmpty()

context OverloadedCpus inv overloadedCpus_maxUtil:
self.maxDevicesUtil >= CpuMaxUtil.allInstances() -> first().value

context OverloadedDisks inv overloadedDisks_maxUtil:
self.maxDevicesUtil >= DiskMaxUtil.allInstances() -> first().value

context UnderloadedNode inv underloadedCpus_or_underloadedDisks:
self.hwRes -> select(hr | hr.oclIsTypeOf(UnderloadedCpus) or
hr.oclIsTypeOf(UnderloadedDisks)) -> notEmpty()

context UnderloadedCpus inv underloadedCpus_minUtil:
self.maxDevicesUtil < CpuMinUtil.allInstances() -> first().value

context UnderloadedDisks inv underloadedDisks_minUtil:
self.maxDevicesUtil < DiskMinUtil.allInstances() -> first().value

context OverloadedNode inv olCpus_and_ulCpus_or_olDisks_and_ulDisks:
(OverloadedCpus.allInstances() -> size() = 1 and UnderloadedCpus ->
allInstances() -> size() = 1)
or
(OverloadedDisks.allInstances() -> size() = 1 and UnderloadedDisks ->
allInstances() -> size() = 1)
```

Definition of the contextual elements

```
context OverloadedNode inv criticalResource:
let resDemTypes : ResourceDemandType = ResDemandTypes.allInstances() -> first()
in
CriticalResource.allInstances() =
if(OverloadedCpus.allInstances() -> size() = 1 and
OverloadedDisks.allInstances() -> size() = 1) then (self.deploy ->
sortedBy(operations.resDemand.basicResDemand -> collect(brd.value | brd.type =
resDemTypes::computation or brd.type = resDemTypes::storage) -> sum()) ->
reverse() -> first()) else if(OverloadedCpus.allInstances() -> size() = 1) then
(self.deploy -> sortedBy(operations.resDemand.basicResDemand ->
collect(brd.value | brd.type = resDemTypes::computation) -> sum()) -> reverse()
-> first()) else (self.deploy -> sortedBy(operations.resDemand.basicResDemand ->
collect(brd.value | brd.type = resDemTypes::storage) -> sum())-> reverse() ->
first()) endif endif

context CriticalResource inv criticalResOps:
CriticalResOperation.allInstances() = self.provided ->
union(self.ownedOperation)

context CriticalResOpStructResDemand inv structResDemCompBasic:
```

```
let resDemTypes : ResourceDemandType = ResourceDemandType.allInstances() ->
first() in
CriticalResOpCompBasicResDemand.allInstances() =
if(OverloadedCpus.allInstances() -> size() = 1) then  self.basicResDemand ->
select(brd | brd.type = resDemTypes::computation) -> first() else OclUndefined
endif

context CriticalResOpStructResDemand inv structResDemStorBasic:
let resDemTypes : ResourceDemandType = ResourceDemandType.allInstances() ->
first() in CriticalResOpStorBasicResDemand.allInstances() =
if(OverloadedDisks.allInstances() -> size() = 1) then  self.basicResDemand ->
select(brd | brd.type = resDemTypes::storage) -> first() else OclUndefined endif
```

TRM_1^{CPS}

```
context OverloadedCpus inv overloadedCpus_maxUtil:
self.maxDevicesUtil < CpuMaxUtil.allInstances() -> first().value

context OverloadedDisks inv overloadedDisks_maxUtil:
self.maxDevicesUtil < DiskMaxUtil.allInstances() -> first().value

context UnderloadedCpus inv underloadedCpus_minUtil:
self.maxDevicesUtil < CpuMinUtil.allInstances() -> first().value

context UnderloadedDisks inv underloadedDisks_minUtil:
self.maxDevicesUtil < DiskMinUtil.allInstances() -> first().value
```

$(SRM_1^{EST}, TRM_1^{EST})$

Description of the refactoring: adding a new software entity instance, thus to avoid excessive remote communication.

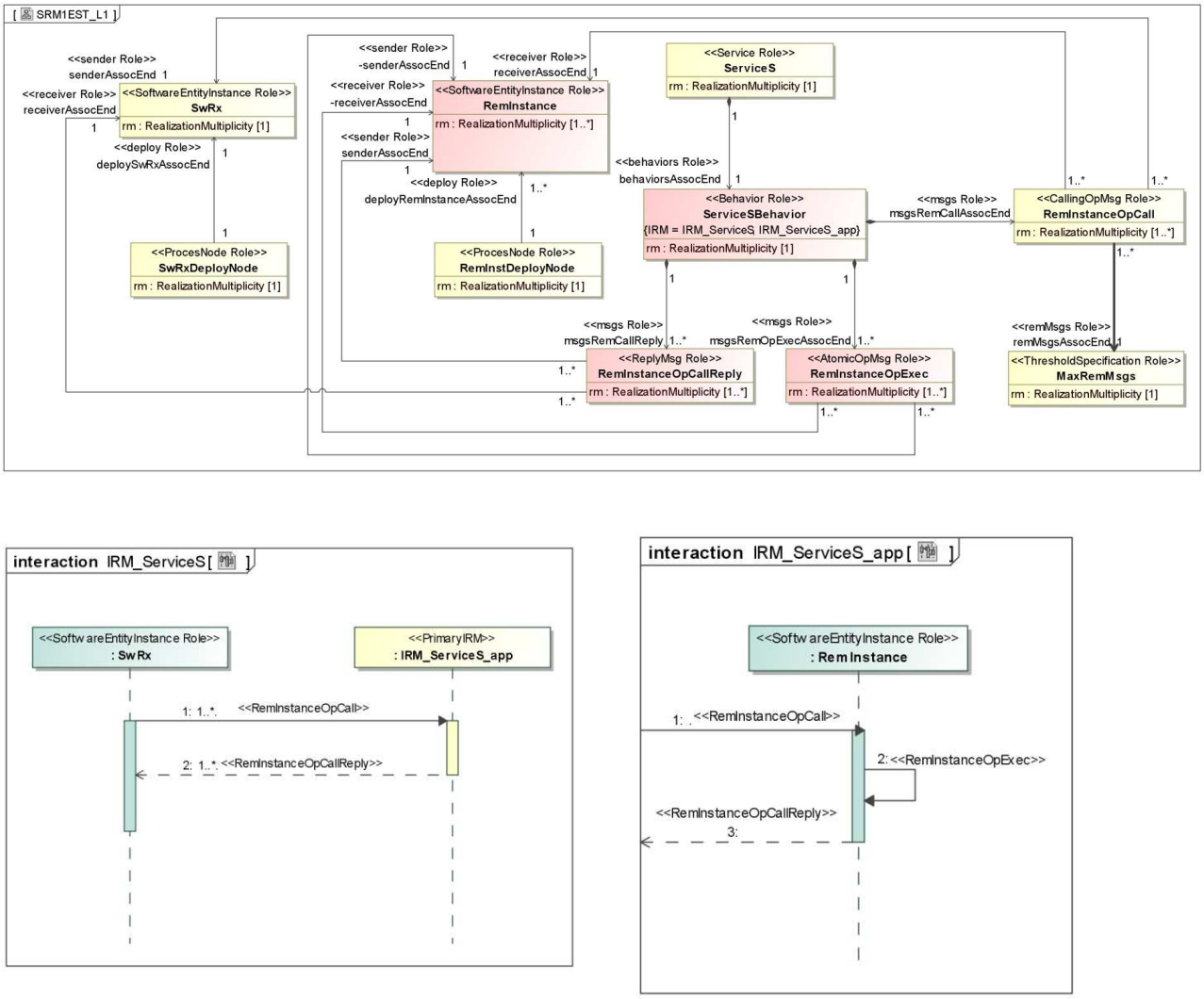
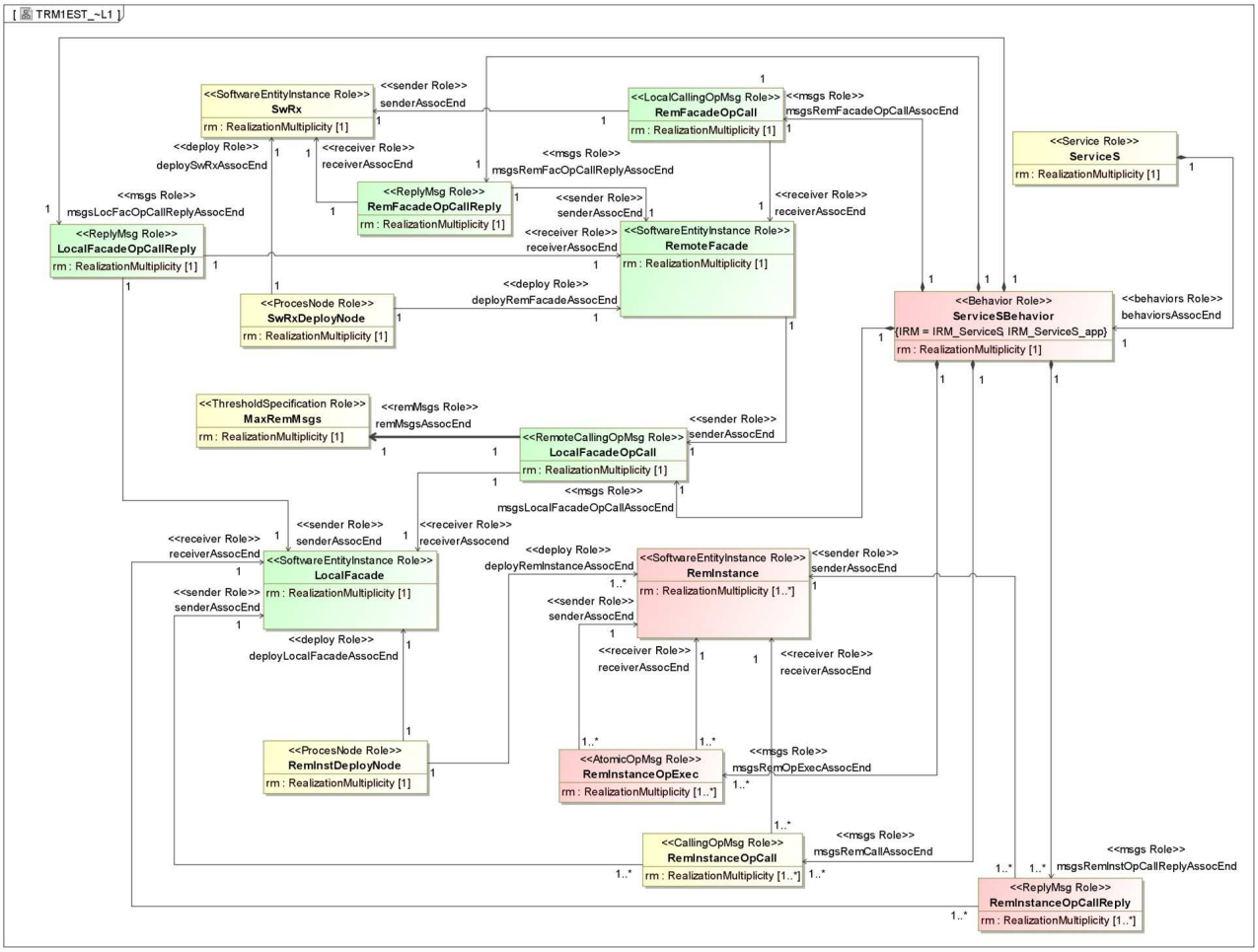
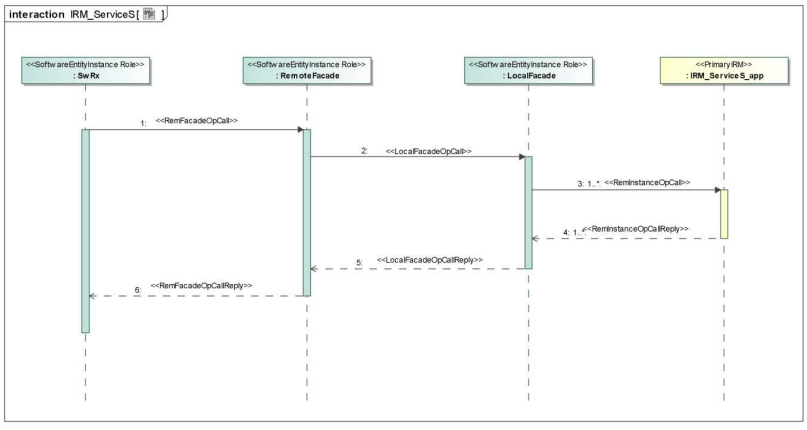


Figure 9 - SRM_1^{EST}.





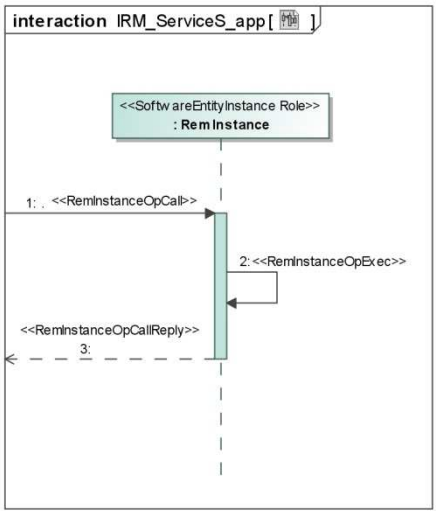


Figure 10 – TRM_1^{EST}.

SRM_1^{EST}

```
context ServiceSBehavior inv maxNumRemMsgs:
RemInstanceOpCall.allInstances() -> size() >= MaxRemMsgs.allInstances() ->
first().value
```

Definition of the contextual elements

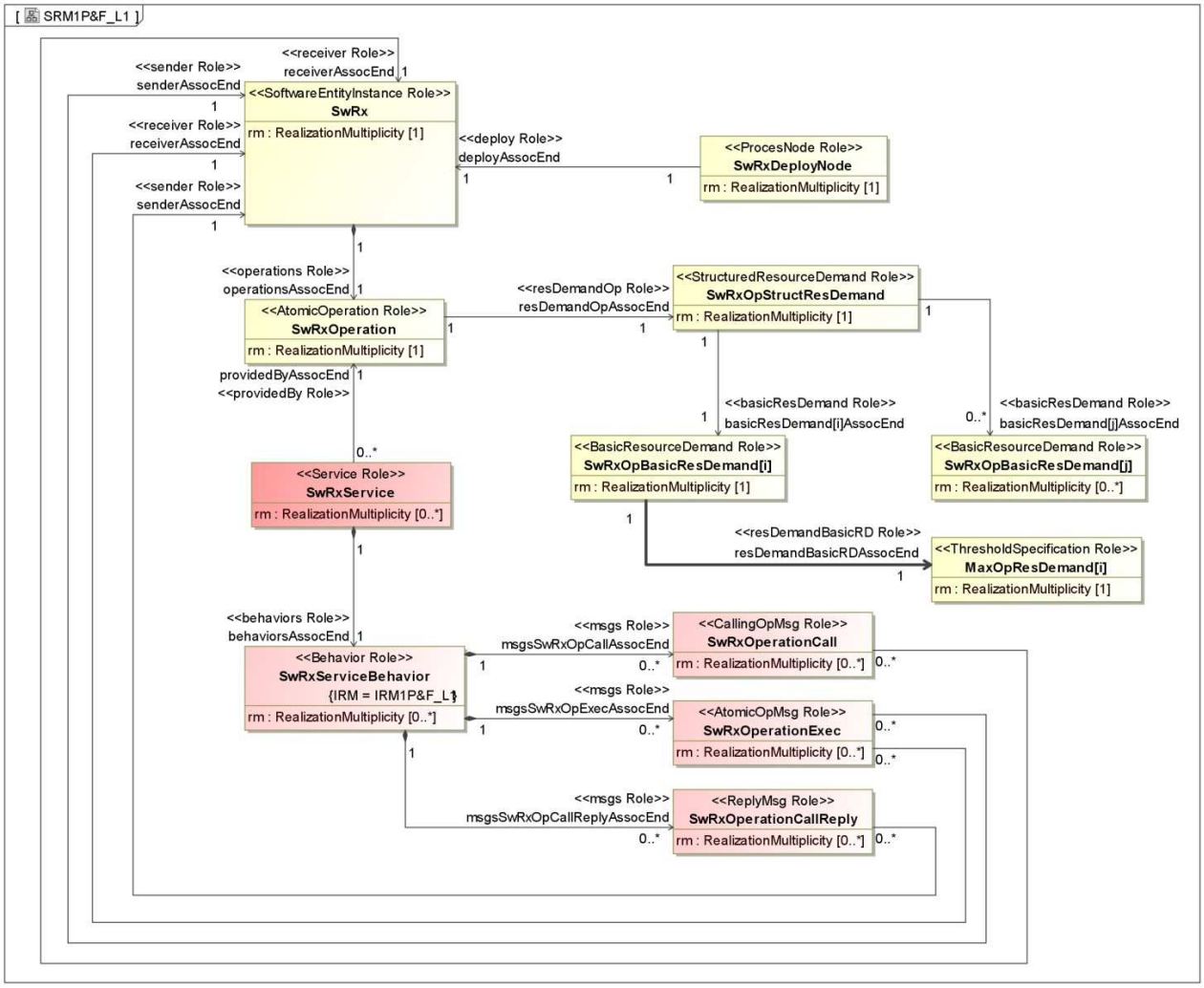
```
context ServiceS inv serviceSBehaviour:
ServiceSBehavior.allInstances() = self.behaviors -> first()

context SwRx inv remInstance:
RemInstances.allInstances() = RemInstanceOpCall.allInstances()->
collect(receiver)

context ServiceSBehavior inv sameData:
RemInstanceOpCall.allInstances() -> forAll(remOpCall |
RemInstanceOpExec.allInstances() -> one(remOpExec |
areInSameRealization(self.IRM, remOpCall, remOpExec) and
areOnSameData({remOpCall.operation, remOpExec.operation})) and
RemInstanceOpReply.allInstances() -> one(remReply |
areInSameRealization(self.IRM, remOpCall, remReply) and remOpCall.replyMessage =
remReply))
```

TRM_1^{EST}

```
context ServiceSBehavior inv maxNumRemMsgs:
LocalFacadeOpCall.allInstances() -> size() < MaxRemMsgs.allInstances() ->
first().value

context ServiceSBehavior inv sameData:
RemInstanceOpCall.allInstances() -> forAll(remOpCall |
RemInstanceOpExec.allInstances() -> one(remOpExec |
areInSameRealization(self.IRM, remOpCall, remOpExec) and
areOnSameData({remOpCall.operation, remOpExec.operation})) and
RemInstanceOpReply.allInstances() -> one(remReply |
areInSameRealization(self.IRM, remOpCall, remReply) and remOpCall.replyMessage =
remReply) and
LocalFacadeOpCall.allInstances() -> one(localFOpCall |
areInSameRealization(self.IRM, remOpCall, localFOpCall) and
areOnSameData({remOpCall.operation, localFOpCall.operation})) and
RemFacadeOpCall.allInstances() -> one(remFOpCall |
areInSameRealization(self.IRM, remOpCall, remFOpCall) and
areOnSameData({remOpCall.operation, remFOpCall.operation})))

context ServiceSBehavior inv sameData2:
LocalFacadeOpCall.allInstances() -> forAll(lFacadeOpCall |
LocalFacadeOpCallReply.allInstances() -> one(lFacadeReply |
areInSameRealization(self.IRM, lFacadeOpCall, lFacadeReply) and
lFacadeOpCall.replyMessage = lFacadeReply))

context ServiceSBehavior inv sameData2:
RemFacadeOpCall.allInstances() -> forAll(rFacadeOpCall |
RemFacadeOpCallReply.allInstances() -> one(rFacadeReply |
areInSameRealization(self.IRM, rFacadeOpCall, rFacadeReply) and
rFacadeOpCall.replyMessage = rFacadeReply))
```

$(SRM_1^{P\&F}, TRM_1^{P\&F})$

Description of the refactoring: split the operation instance OpI in n parts; the new instances are deployed on the same processing node on which OpI was deployed.



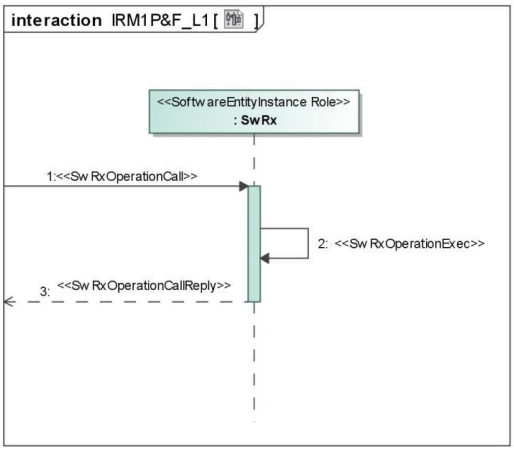


Figure 11 - $SRM_1^{P\&F}$.

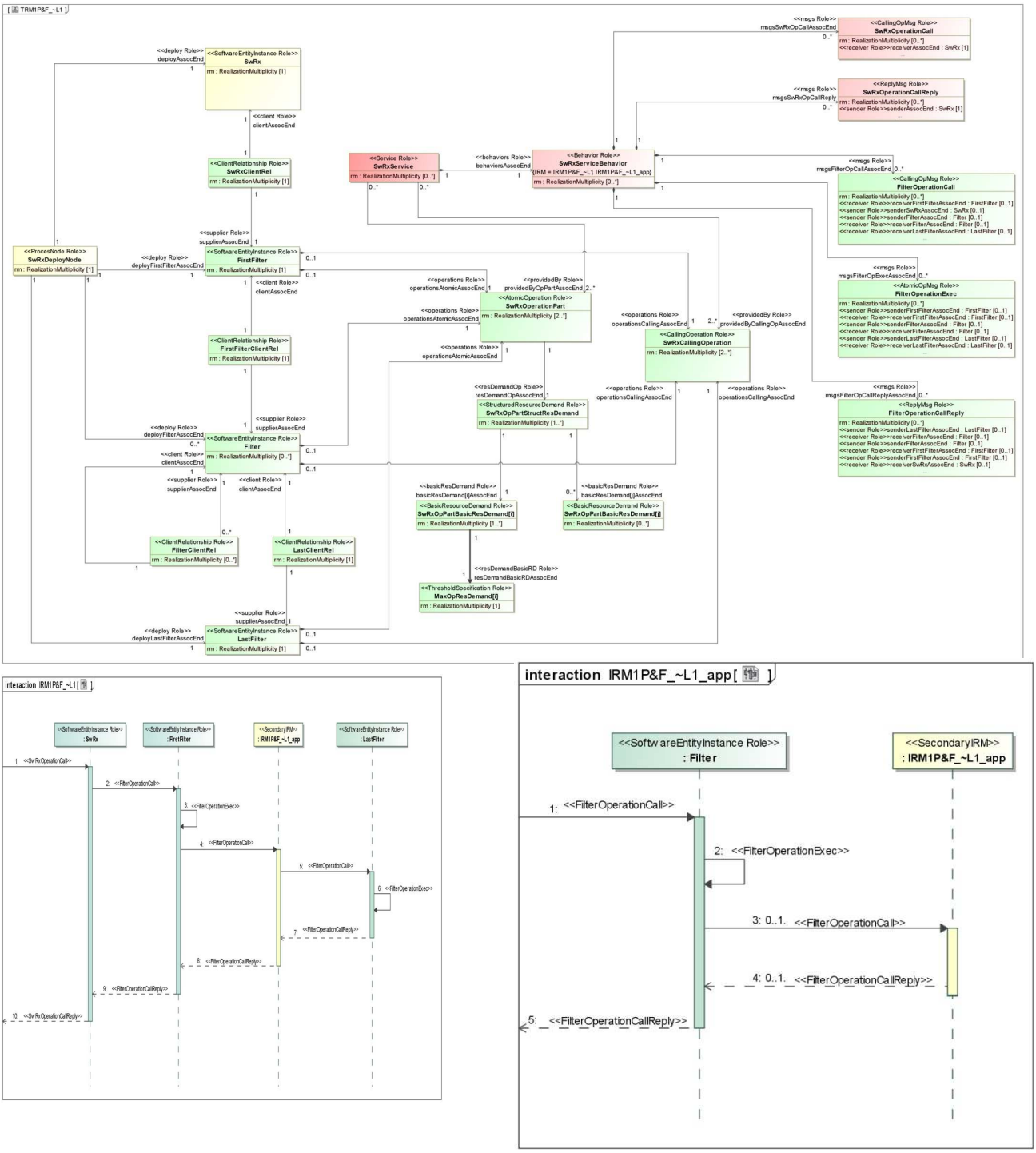


Figure 12 – $TRM_1^{P\&F}$.

$SRM_1^{P\&F}$

```
context SwRxOpBasicResDemand[i] inv maxOpResDemand[i]:
self.value >= MaxOpResDemand[i].allInstances() -> first().value
```

Definition of the contextual elements

```
context SwRx inv swRxService:
SwRxService.allInstances() = Service.allInstances() -> select(s | s.providedBy -
> includes(SwRxOperation.allInstances() -> first()))

context SwRxService inv swRxServiceBehavior:
SwRxServiceBehavior.allInstances() = self.behaviors -> first()

context SwRx inv dependencies:
CallingOperation.allInstances() -> select(op | getDependencies(op) ->
includes(self.operations -> select(swRxOp | swRxOp.oclIsTypeOf(SwRxOperation)) -
> first())) -> forAll(depOp | getDependencies(depOp) -> size() = 1)

context SwRxServiceBehavior inv sameData:
SwRxOperationCall.allInstances() -> forAll(swRxOpCall |
SwRxOperationExec.allInstances() -> one(swRxOpExec |
areInSameRealization(self.IRM, swRxOpCall, swRxOpExec) and
areOnSameData({swRxOpCall.operation, swRxOpExec.operation})) and
SwRxOperationCallReply.allInstances() -> one(swRxOpReply |
areInSameRealization(self.IRM, swRxOpCall, swRxOpReply) and
swRxOpCall.replyMessage = swRxOpReply))
```

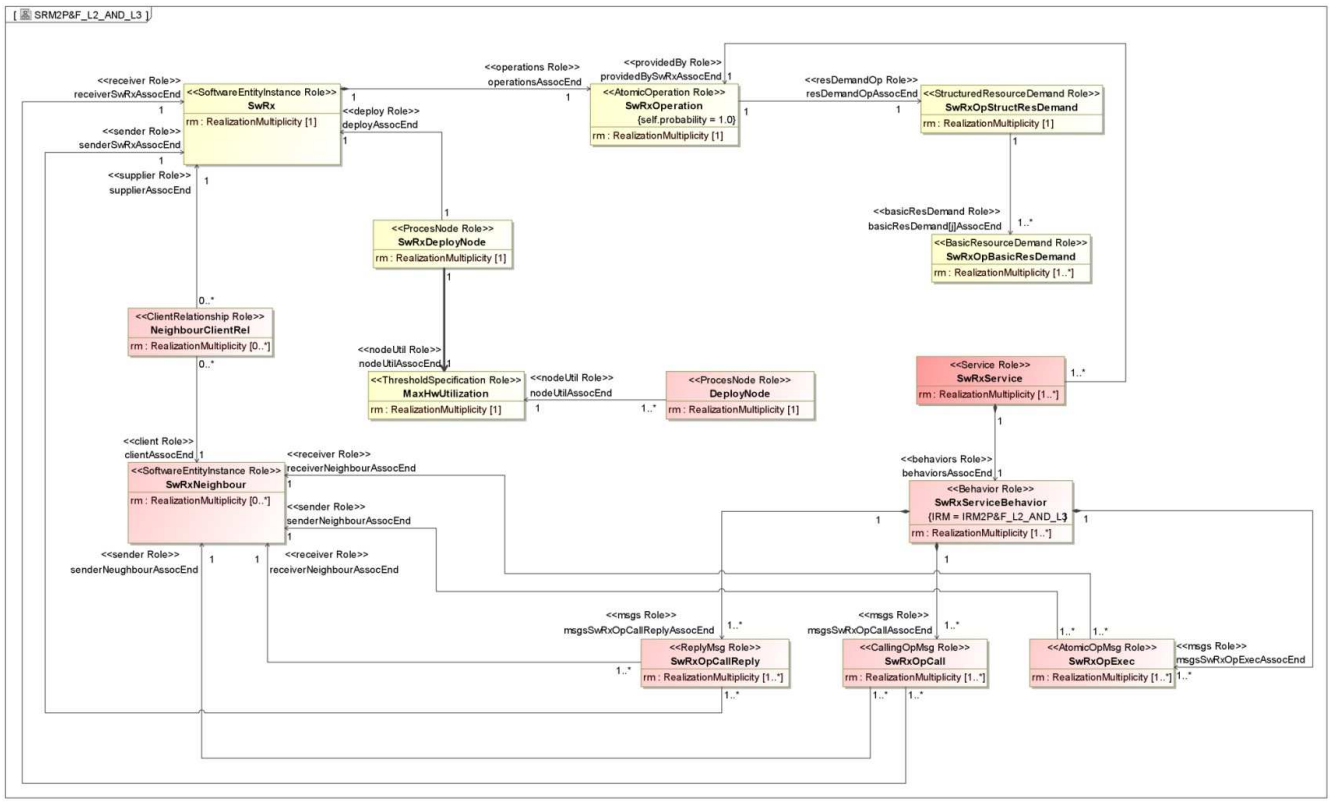
$TRM_1^{P\&F}$

```
context SwRxOpPartBasicResDemand[i] inv maxOpResDemand[i]:
self.value < MaxOpResDemand[i].allInstances() -> first().value

context SwRxOperationPart inv numSwRxOpParts:
SwRxOperationPart.allInstances() -> size() =
(M::SwRxOpBasicResDemand[i].allInstances() ->
first().value/M::MaxOpResDemand[i].allInstances() -> first()).ceiling()

context SwRxService inv numCallingOps:
SwRxCallingOperation.allInstances() -> size() = SwRxOperationPart.allInstances()
-> size()

context FirstFilter inv swRxOpPartBasicResDemSum:
SwRxOpPartBasicResDemand[i].allInstances() -> collect(brd.value()) -> sum() =
M::SwRxOpBasicResDemand[i].allInstances() -> first().value

context SwRxOpPartStructResDemand inv swRxOpPartBasicResDem[j]:
self.basicResDemand -> select(brd |
brd.oclIsTypeOf(SwRxOpPartBasicResDemand[j])) =
M::SwRxOpBasicResDemand[j].allInstances()

context SwRxClientRel inv swRxClientRelToFirstFilter:
FirstFilter.allInstances() -> first().operations -> includes(self.target)

context FirstFilterClientRel inv firstFilterClientRelToFilter:
if(Filter.allInstances() -> notEmpty()) then Filter.allInstances() -> one(f |
f.operations -> includes(self.target)) else LastFilter.allInstances() ->
first().operations -> includes(self.target) endif

context FilterClientRel inv filterClientRelToFilter:
Filter.allInstances() -> one(f | f.operations -> includes(self.target))

context LastClientRel inv filterClientRelToLastFilter:
LastFilter.allInstances() -> first().operations -> includes(self.target)

context SwRx inv dependencies:
FirstFilter.allInstances() -> first().operations -> includes(self.required ->
select(firstFilterOp | firstFilterOp.oclIsTypeOf(SwRxCallingOperation)) ->
first()) implies CallingOperation.allInstances() -> select(op |
getDependencies(op) -> includes(M::SwRx.allInstances() -> first().operations ->
select(swRxOp | swRxOp.oclIsTypeOf(M::SwRxOperation)) -> first())) ->
forAll(depOp | getDependencies(depOp) -> size() = 1)

context SwRxCallingOperation inv dependencies1:
if(LastFilter.allInstances() -> first().operations -> includes(self)) then
getDependencies(self) = LastFilter.allInstances() -> first().operations ->
select(op | op.oclIsTypeOf(SwRxOperationPart)) -> first() else
getDependencies(self) = (Filter.allInstances() -> select(f | f.operations ->
includes(self)) -> first().operations -> select(op |
```

```
op.oclIsTypeOf(SwRxOperationPart))) -> union(Filter.allInstances() -> select(f |
f.operations -> includes(self)) -> first().required) endif

context SwRxServiceBehavior inv sameData:
FilterOperationCall.allInstances() -> forAll(filterOpCall |
FilterOperationExec.allInstances() -> one(filterOpExec |
areInSameRealization(self.IRM, filterOpCall, filterOpExec) and
areOnSameData({filterOpCall.operation, filterOpExec.operation})) and
FilterOperationCallReply.allInstances() -> one(filterOpReply |
areInSameRealization(self.IRM, filterOpCall, filterOpReply) and
filterOpCall.replyMessage = filterOpReply) and
SwRxOperationCall.allInstances() -> one(swRxOpCall |
areInSameRealization(self.IRM, swRxOpCall, filterOpCall) and
areOnSameData({swRxOpCall.operation, filterOpCall.operation})))

context SwRxServiceBehavior inv sameData1:
FilterOperationCallReply.allInstances() -> forAll(filterOpReply |
FilterOperationCall.allInstances() -> one(filterOpCall |
areInSameRealization(self.IRM, filterOpCall, filterOpCallReply) and
filterOpCall.replyMessage = filterOpReply) and
FilterOperationExec.allInstances() -> one(filterOpExec |
areInSameRealization(self.IRM, filterOpCall, filterOpExec) and
areOnSameData({filterOpCall.operation, filterOpExec.operation})))
```

$(SRM_2^{P\&F}, TRM_2^{P\&F})$

Description of the refactoring: mirror a software entity instance and re-deploy it on the less used processing node.



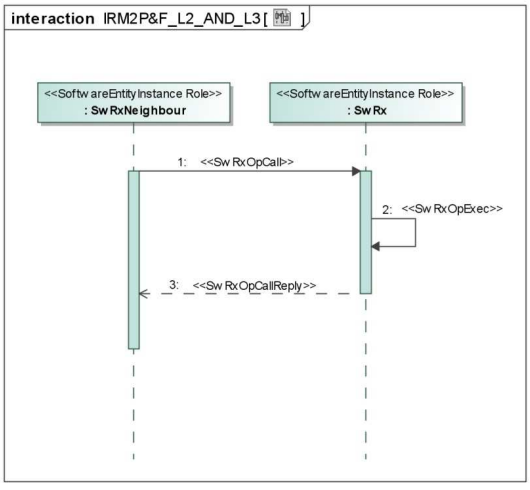
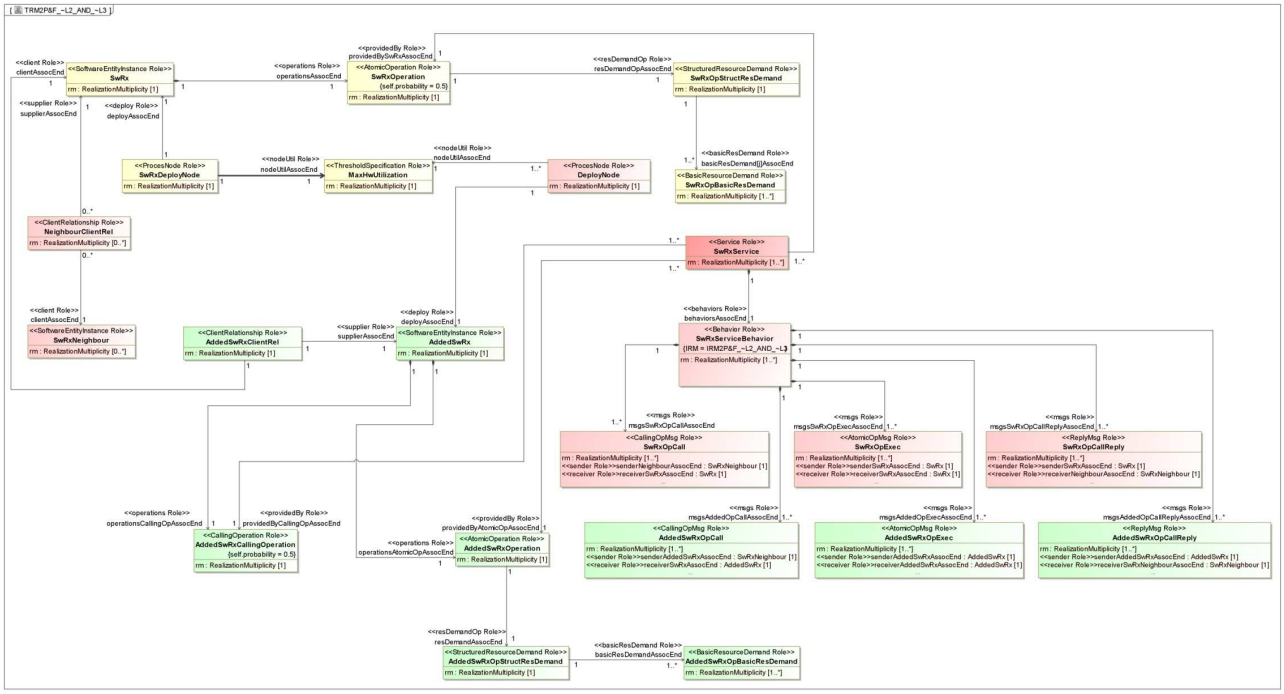


Figure 13 – $SRM_2^{P\&F}$.



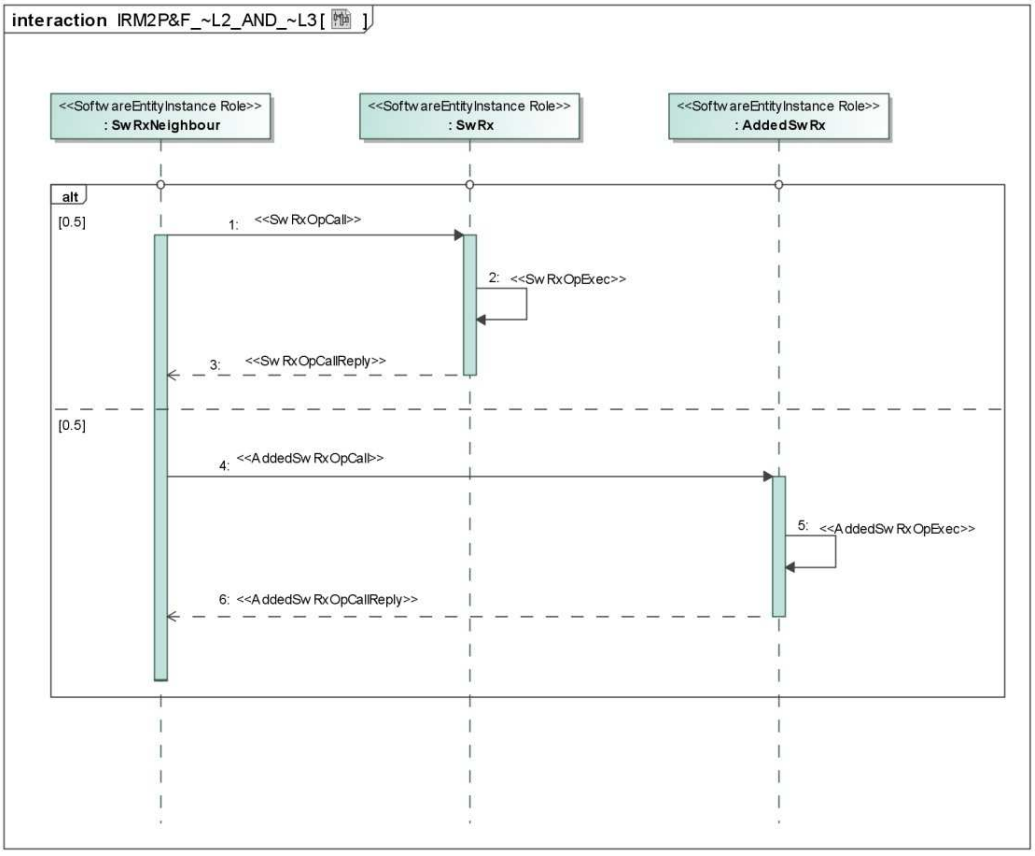


Figure 14 – $TRM_2^{P\&F}$.

$SRM_2^{P\&F}$

```
context SwRxOperation inv swRxOpProbability:
self.probability = 1.0

context SwRxDeployNode inv maxHwUtilization:
self.maxDevicesUtil >= MaxHwUtilization.allInstances()-> first().value
```

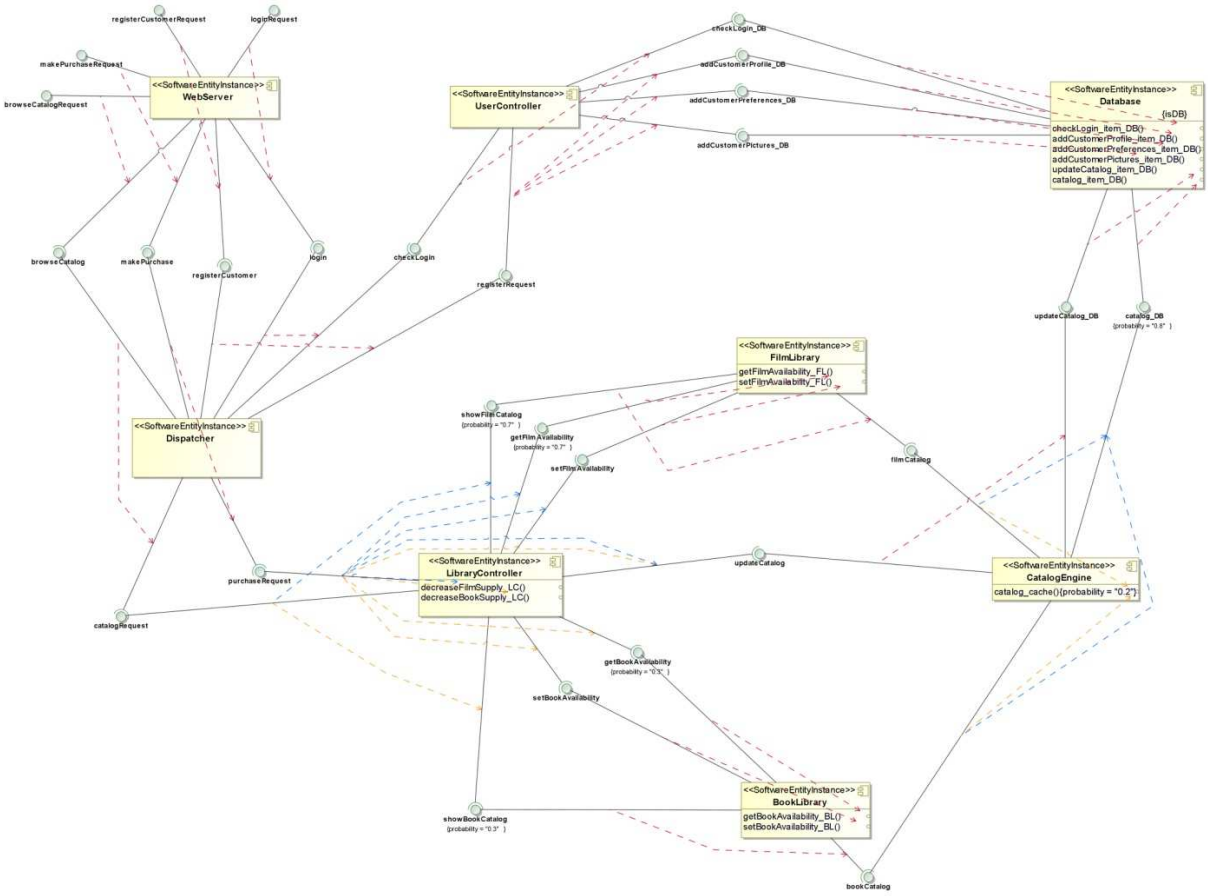
Definition of the contextual elements

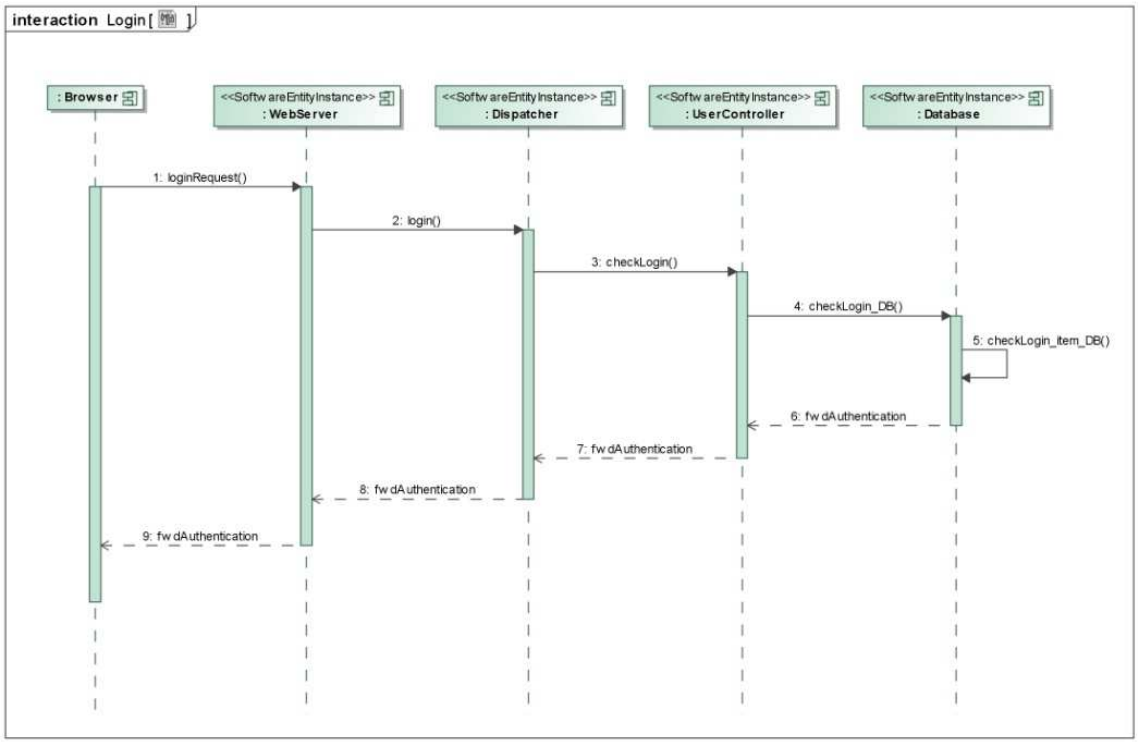
```
context SwRx inv swRxNeighbour:
SwRxNeighbour.allInstances() = SwRxOpCall.allInstances() -> first().receiver

context SwRxNeighbour inv neighbourClientRel:
NeighbourClientRel.allInstances() = ClientRelationship.allInstances() ->
select(sr | sr.source = self)

context SwRx inv swRxService:
SwRxService.allInstances() = Service.allInstances() -> select(s | s.providedBy -
> includes(SwRxOperation.allInstances() -> first()))

context SwRxService inv swRxServiceBehavior:
SwRxServiceBehavior.allInstances() = self.behaviors -> first()

context SwRx inv dependencies:
CallingOperation.allInstances() -> select(op | getDependencies(op) ->
includes(self.operations -> select(swRxOp | swRxOp.oclIsTypeOf(SwRxOperation)) -
> first())) -> forAll(depOp | getDependencies(depOp) -> size() = 1)

context SwRx inv deployNode:
DeployNode.allInstances() = ProcesNode.allInstances() -> excluding(pn |
pn.oclIsTypeOf(SwRxDeployNode)) -> select(pn | pn.maxDevicesUtil <
MaxHwUtilization.allInstances() -> first().value and NetworkLink.allInstances()
-> exists(nl | nl.endNode -> includes({SwRxDeployNode.allInstances() -> first(),
pn}))) -> sortedBy(maxDevicesUtil) -> first()

context SwRxServiceBehavior inv sameData:
SwRxOpCall.allInstances() -> forAll(swRxOpCall |
SwRxOpExec.allInstances() -> one(swRxOpExec | areInSameRealization(self.IRM,
swRxOpCall, swRxOpExec) and areOnSameData({swRxOpCall.operation,
swRxOpExec.operation})) and
SwRxOpCallReply.allInstances() -> one(swRxOpReply |
areInSameRealization(self.IRM, swRxOpCall, swRxOpReply) and
swRxOpCall.replyMessage = swRxOpReply))
```

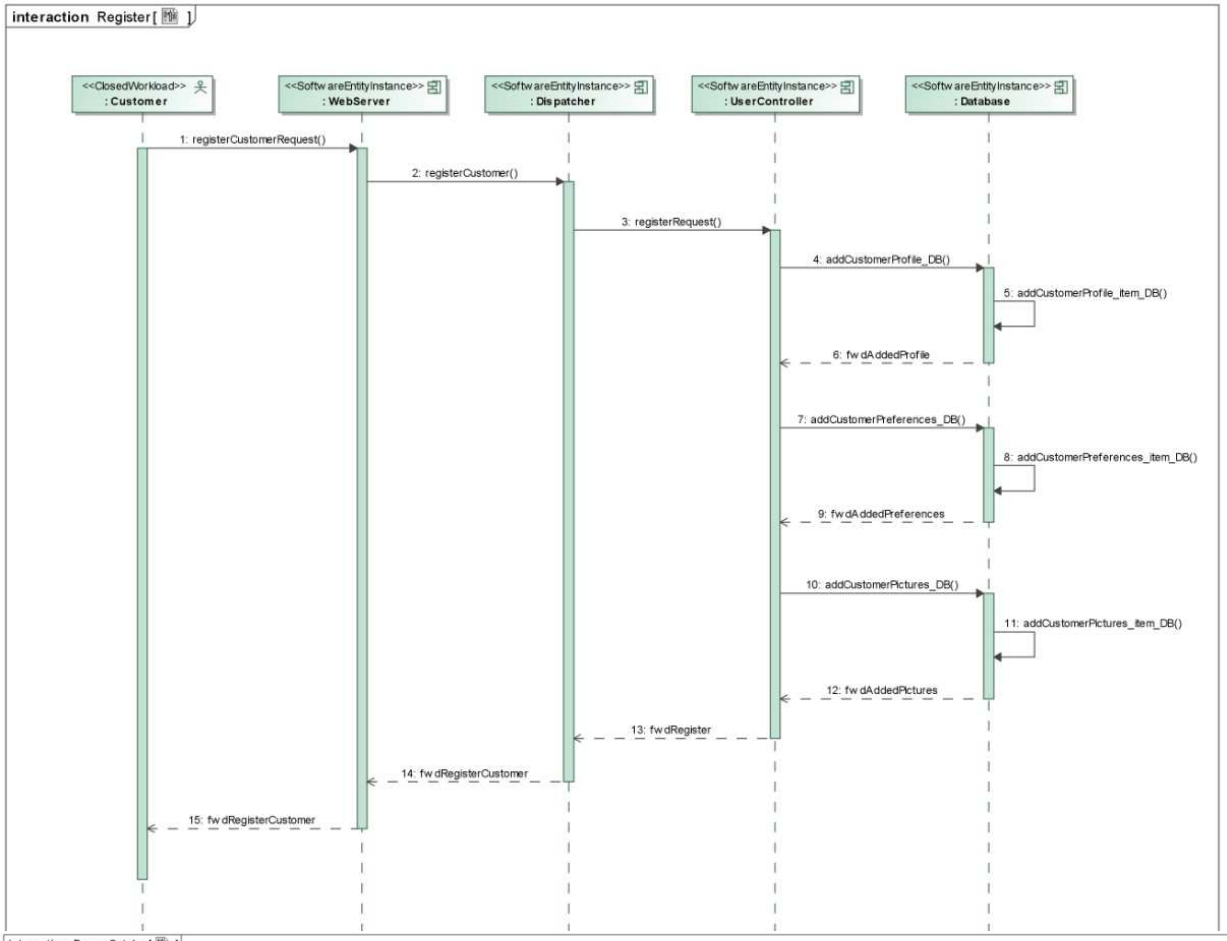
$TRM_2^{P\&F}$

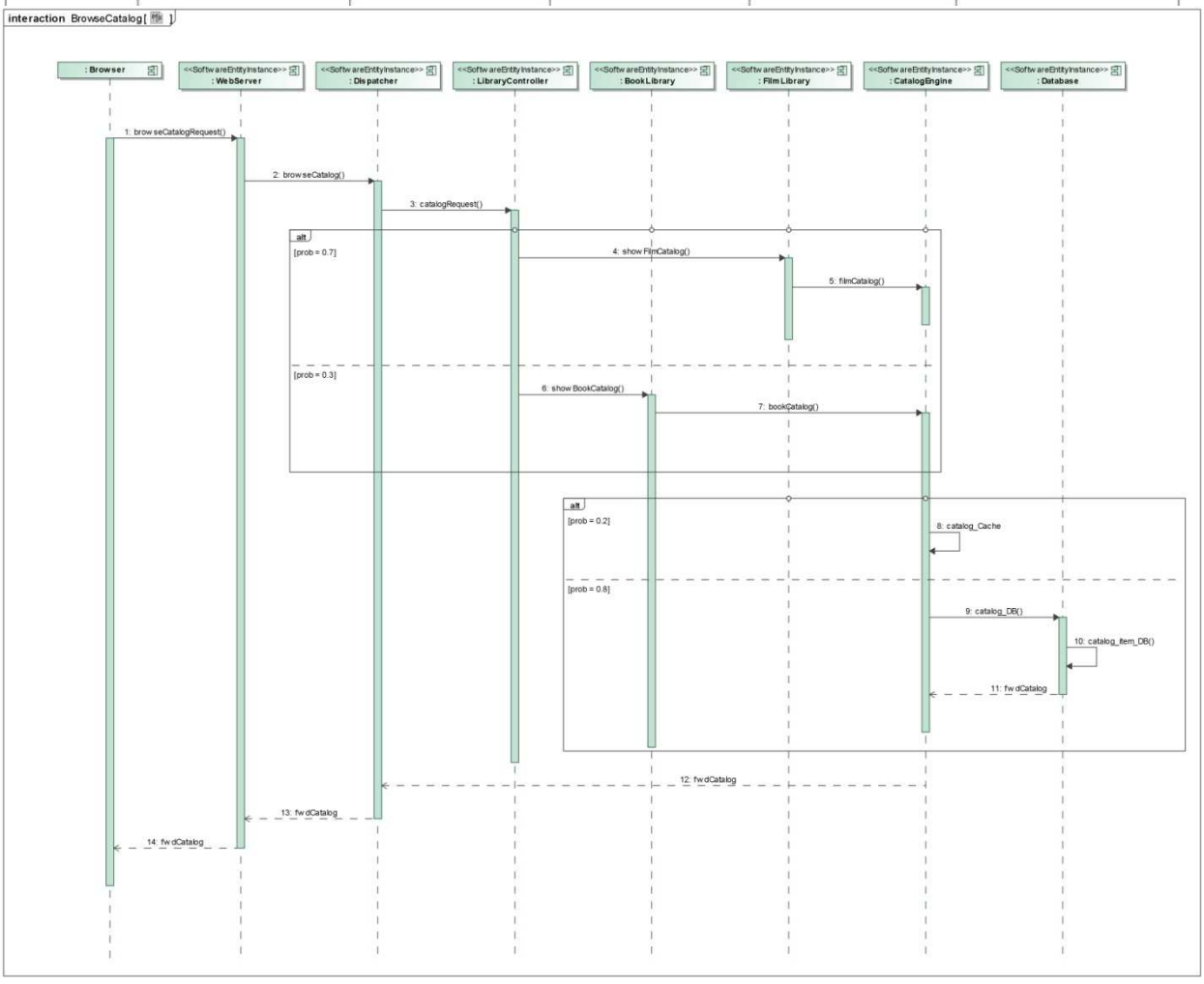
```
context SwRxOperation inv swRxOpProbability:
self.probability = 0.5

context SwRxDeployNode inv maxHwUtilization:
self.maxDevicesUtil < MaxHwUtilization.allInstances() -> first().value

context AddedSwRxCallingOperation inv addedSwRxOpProbability:
self.probability = 0.5

context AddedSwRx inv addedSwRxOpBasicResDem:
AddedSwRxOpBasicResDemand.allInstances() -> forAll(addedBrd |
SwRxOpBasicResDemand.allInstances() -> exists(brd | brd.type = addedBrd.type and
brd.value = addedBrd.value))

context AddedSwRxOpStructResDemand inv addedSwRxOpBasResDem:
self.basicResDemand -> select(brd | brd.oclIsTypeOf(AddedSwRxOpBasicResDemand))
= SwRxOpBasicResDemand.allInstances()

context AddedSwRxClientRel inv swRxClientRelToAddedSwRx:
AddedSwRx.allInstances() -> first().operations -> includes(self.target)

context SwRx inv dependencies1:
CallingOperation.allInstances() -> select(op | getDependencies(op) ->
includes(self.operations -> select(swRxOp |
swRxOp.oclIsTypeOf(AddedSwRxCallingOperation)) -> first())) -> forAll(depOp |
getDependencies(depOp) -> size() = 1)

context AddedSwRx inv dependencies2:
AddedSwRxCallingOperation.allInstances() -> select(op | getDependencies(op) ->
includes(self.operations -> select(swRxOp |
swRxOp.oclIsTypeOf(AddedSwRxOperation)) -> first())) -> forAll(depOp |
getDependencies(depOp) -> size() = 1)

context SwRxServiceBehavior inv sameData1:
AddedSwRxOpCall.allInstances() -> forAll(addedSwRxOpCall |
AddedSwRxOpExec.allInstances() -> one(addedSwRxOpExec |
areInSameRealization(self.IRM, addedSwRxOpCall, addedSwRxOpExec) and
areOnSameData({addedSwRxOpCall.operation, addedSwRxOpExec.operation})) and
AddedSwRxOpCallReply.allInstances() -> one(addedSwRxOpReply |
areInSameRealization(self.IRM, addedSwRxOpCall, addedSwRxOpReply) and
addedSwRxOpCall.replyMessage = addedSwRxOpReply))
```

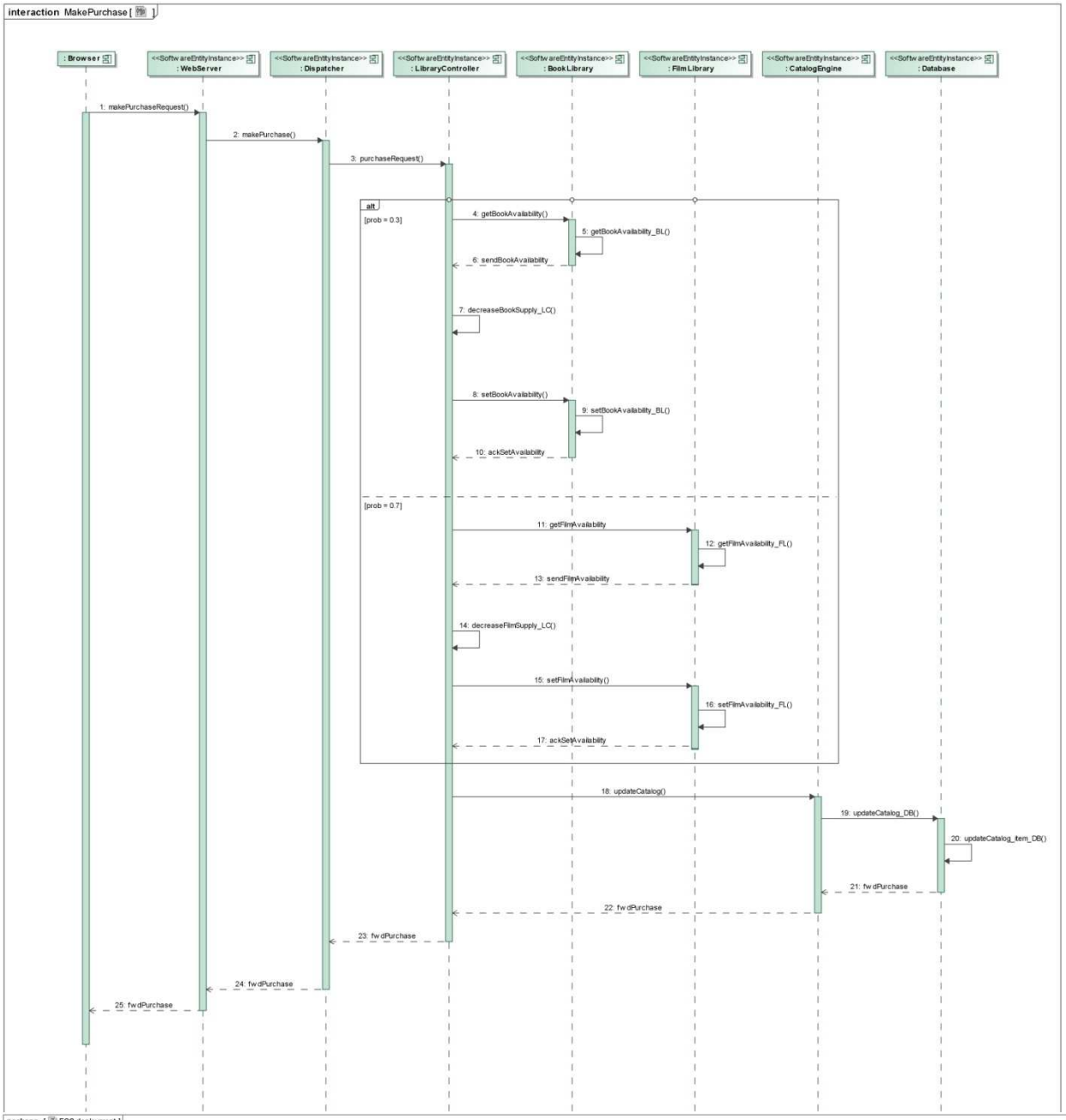
SRM-TRM pairs applied to the Electronic Commerce System (ECS)

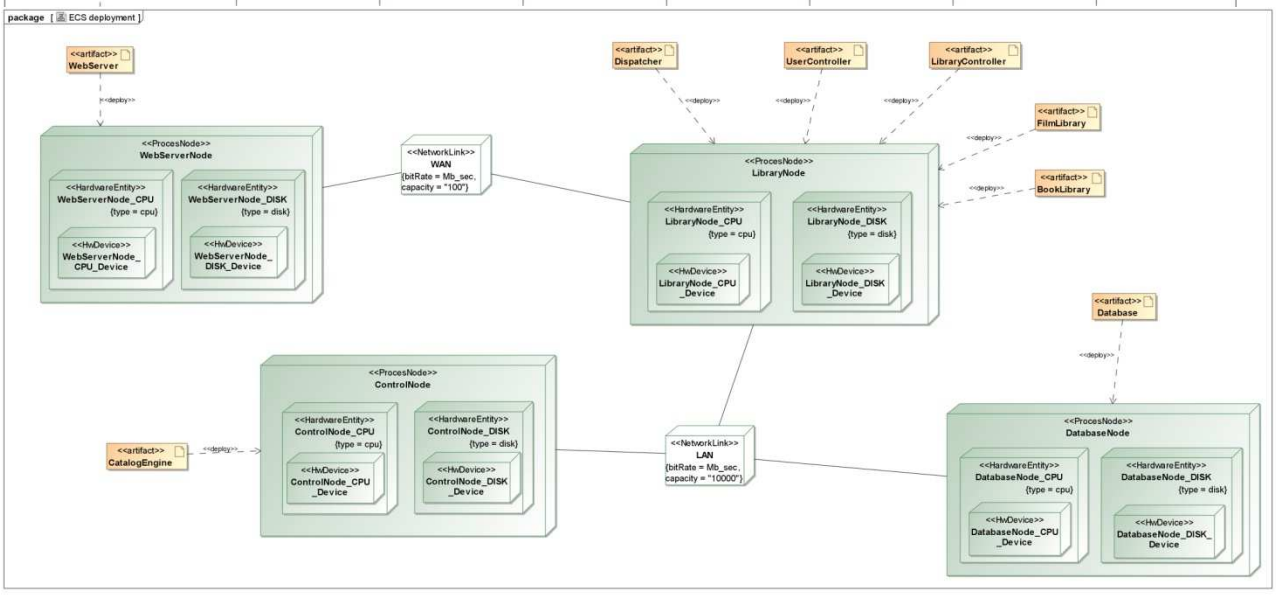




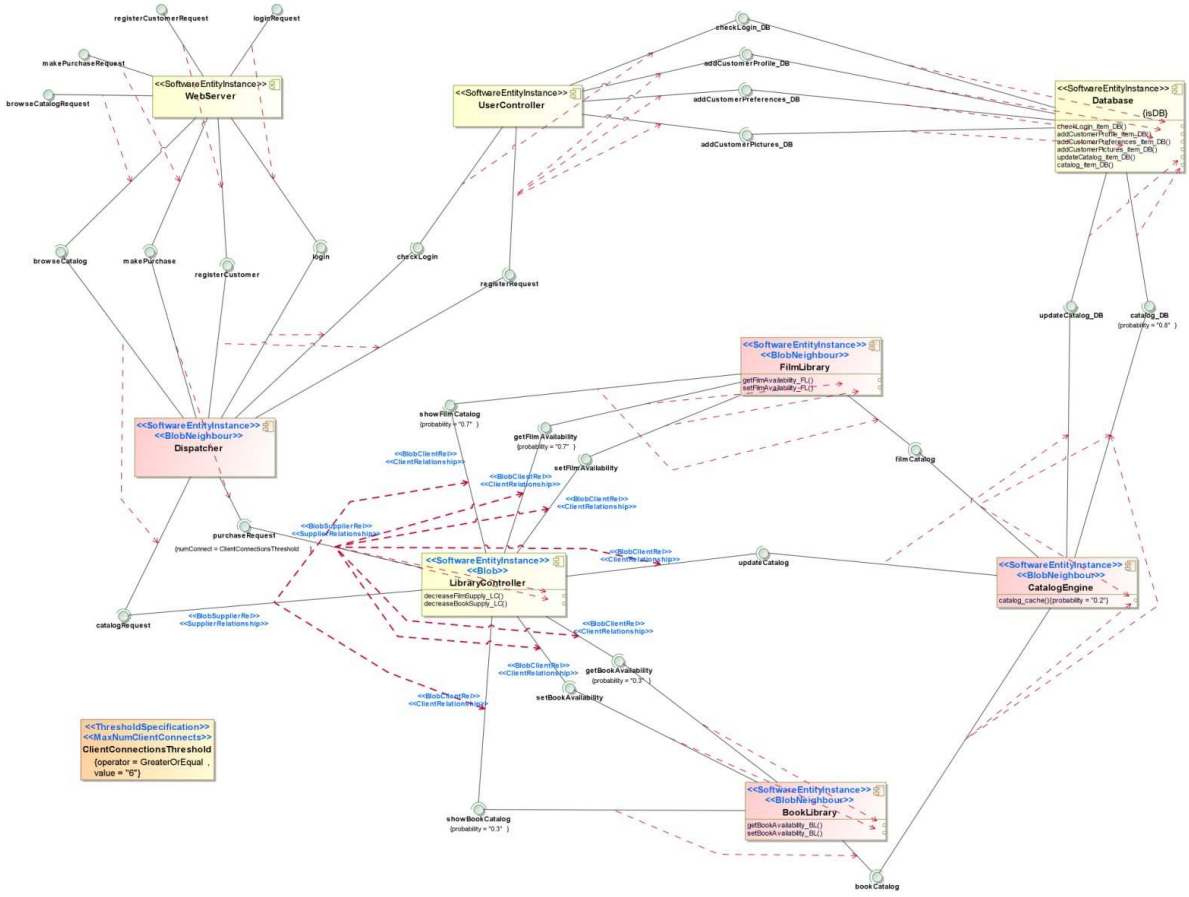

interaction Register[ ]
<<ClosedWorkload>>
: Customer
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
: UserController
<<SoftwareEntityInstance>>
: Database
1: registerCustomerRequest()
2: registerCustomer()
3: registerRequest()
4: addCustomerProfile_DB()
5: addCustomerProfile_Item_DB()
6: fwdAddedProfile
7: addCustomerPreferences_DB()
8: addCustomerPreferences_item_DB()
9: fwdAddedPreferences
10: addCustomerPictures_DB()
11: addCustomerPictures_item_DB()
12: fwdAddedPictures
13: fwdRegister
14: fwdRegisterCustomer
15: fwdRegisterCustomer

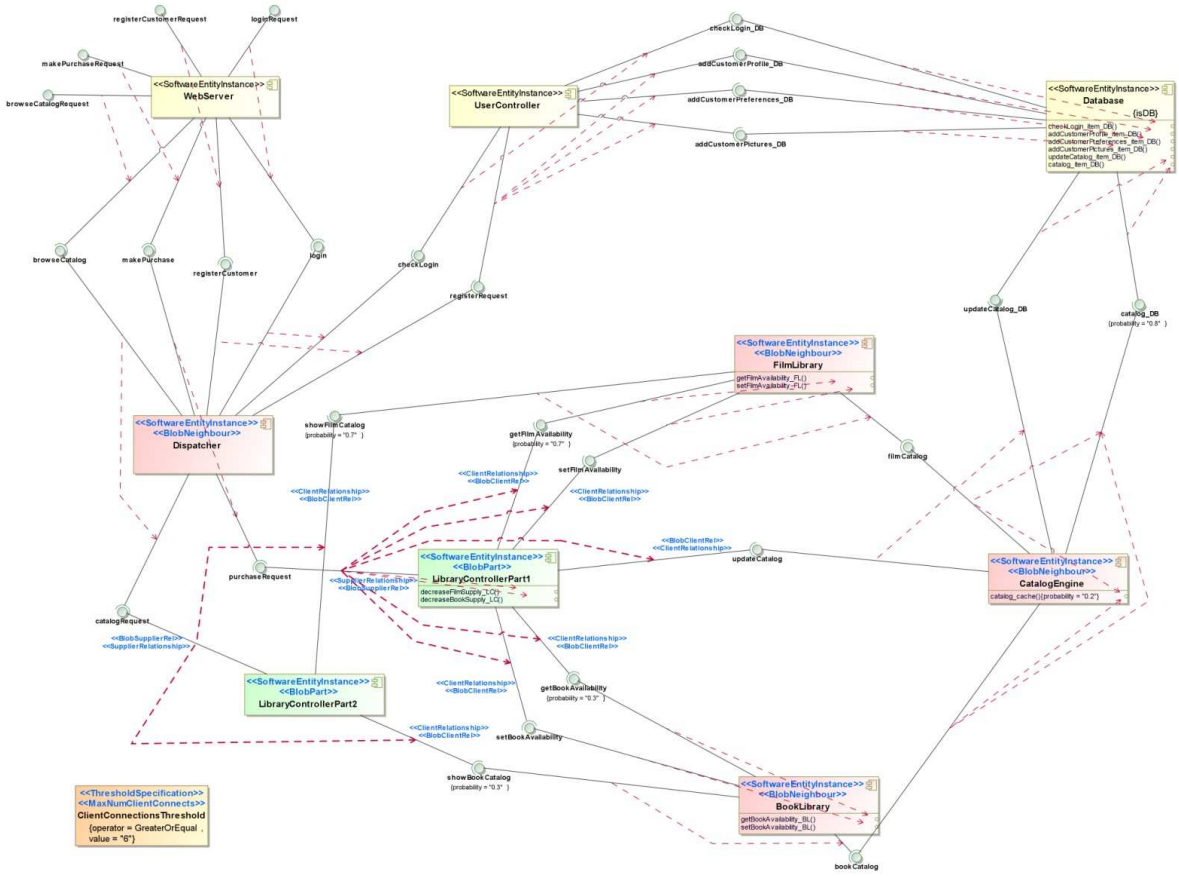

interaction BrowseCatalog[ ]
: Browser
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
: LibraryController
<<SoftwareEntityInstance>>
: BookLibrary
<<SoftwareEntityInstance>>
: FilmLibrary
<<SoftwareEntityInstance>>
: CatalogEngine
<<SoftwareEntityInstance>>
: Database
1: browseCatalogRequest()
2: browseCatalog()
3: catalogRequest()
alt
[prob = 0.7]
4: showFilmCatalog()
5: filmCatalog()
[prob = 0.3]
6: showBookCatalog()
7: bookCatalog()
alt
[prob = 0.2]
8: catalog_Cache
[prob = 0.8]
9: catalog_DB()
10: catalog_Item_DB()
11: fwdCatalog
12: fwdCatalog
13: fwdCatalog
14: fwdCatalog

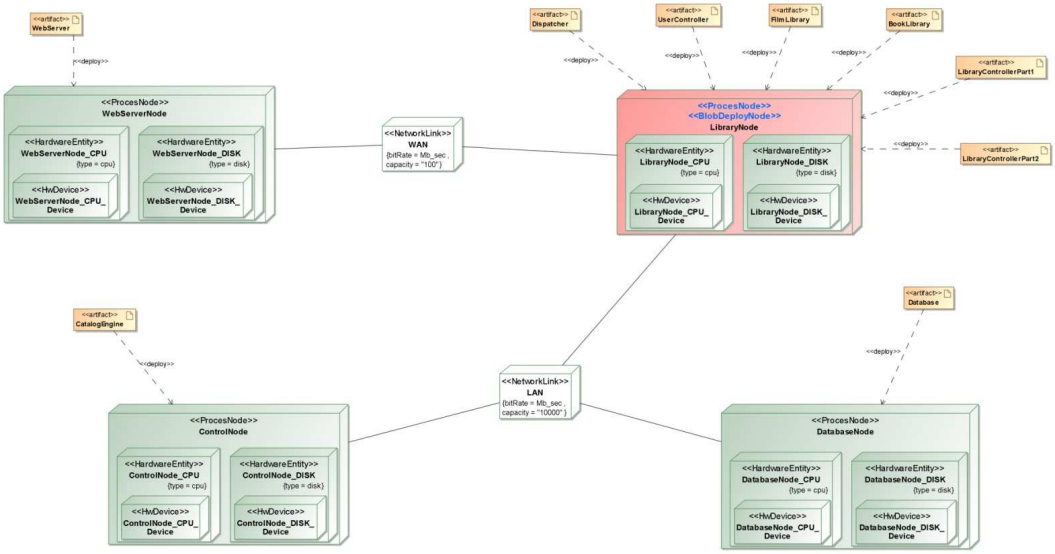

interaction MakePurchase
: Browser
<<SoftwareEntityInstance>> : WebServer
<<SoftwareEntityInstance>> : Dispatcher
<<SoftwareEntityInstance>> : LibraryController
<<SoftwareEntityInstance>> : BookLibrary
<<SoftwareEntityInstance>> : FilmLibrary
<<SoftwareEntityInstance>> : CatalogEngine
<<SoftwareEntityInstance>> : Database
1: makePurchaseRequest()
2: makePurchase()
3: purchaseRequest()
alt
[prob = 0.3]
4: getBookAvailability()
5: getBookAvailability_BL()
6: sendBookAvailability
7: decreaseBookSupply_LC()
8: setBookAvailability()
9: setBookAvailability_BL()
10: ackSetAvailability
[prob = 0.7]
11: getFilmAvailability
12: getFilmAvailability_FL()
13: sendFilmAvailability
14: decreaseFilmSupply_LC()
15: setFilmAvailability()
16: setFilmAvailability_FL()
17: ackSetAvailability
18: updateCatalog()
19: updateCatalog_DB()
20: updateCatalog_item_DB()
21: fwdPurchase
22: fwdPurchase
23: fwdPurchase
24: fwdPurchase
25: fwdPurchase


package ECS deployment
<<artifact>> WebServer
<<artifact>> Dispatcher
<<artifact>> UserController
<<artifact>> LibraryController
<<artifact>> FilmLibrary
<<artifact>> BookLibrary
<<artifact>> Database
<<artifact>> CatalogEngine
<<deploy>>
<<ProcesNode>> WebServerNode
<<HardwareEntity>> WebServerNode_CPU {type = cpu}
<<HwDevice>> WebServerNode_CPU_Device
<<HardwareEntity>> WebServerNode_DISK {type = disk}
<<HwDevice>> WebServerNode_DISK_Device
<<NetworkLink>> WAN {bitRate = Mb_sec, capacity = "100"}
<<ProcesNode>> LibraryNode
<<HardwareEntity>> LibraryNode_CPU {type = cpu}
<<HwDevice>> LibraryNode_CPU_Device
<<HardwareEntity>> LibraryNode_DISK {type = disk}
<<HwDevice>> LibraryNode_DISK_Device
<<ProcesNode>> ControlNode
<<HardwareEntity>> ControlNode_CPU {type = cpu}
<<HwDevice>> ControlNode_CPU_Device
<<HardwareEntity>> ControlNode_DISK {type = disk}
<<HwDevice>> ControlNode_DISK_Device
<<NetworkLink>> LAN {bitRate = Mb_sec, capacity = "10000"}
<<ProcesNode>> DatabaseNode
<<HardwareEntity>> DatabaseNode_CPU {type = cpu}
<<HwDevice>> DatabaseNode_CPU_Device
<<HardwareEntity>> DatabaseNode_DISK {type = disk}
<<HwDevice>> DatabaseNode_DISK_Device

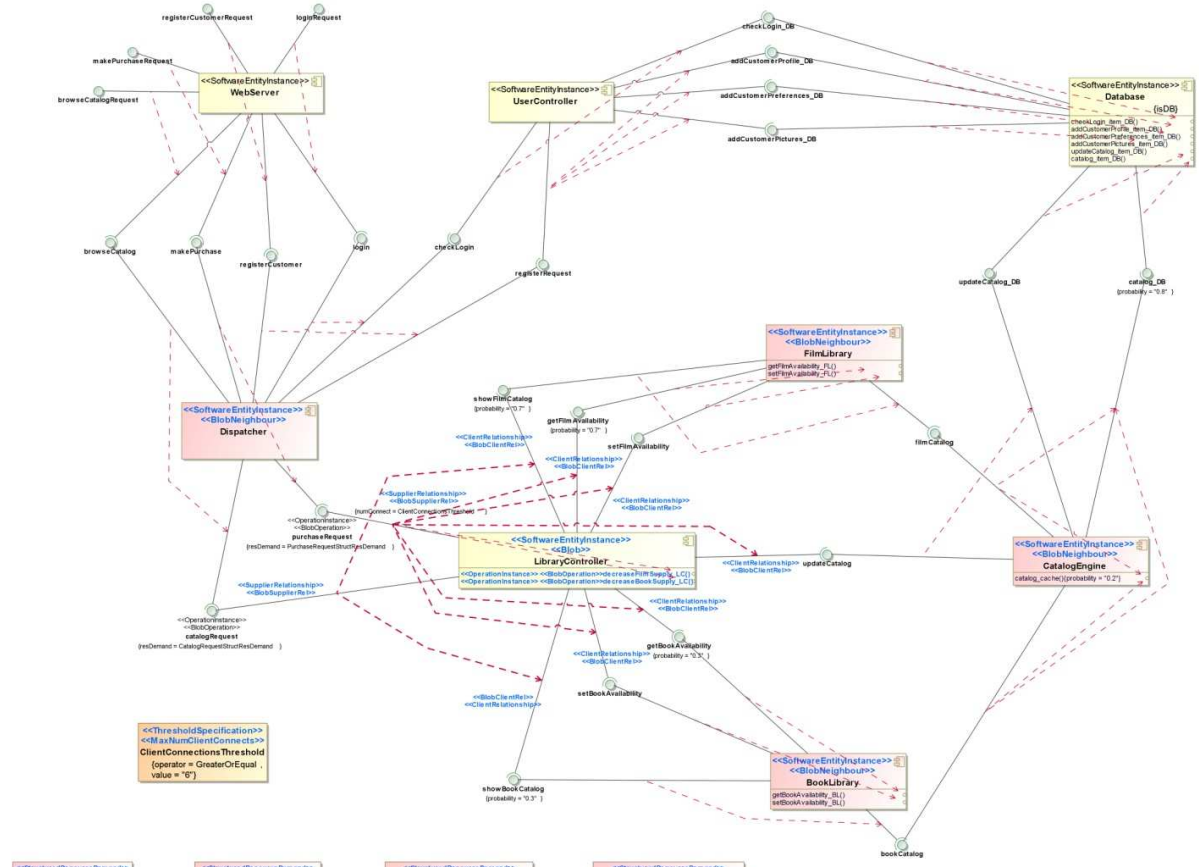
$(SRM_1^{Blob}, TRM_1^{Blob})$

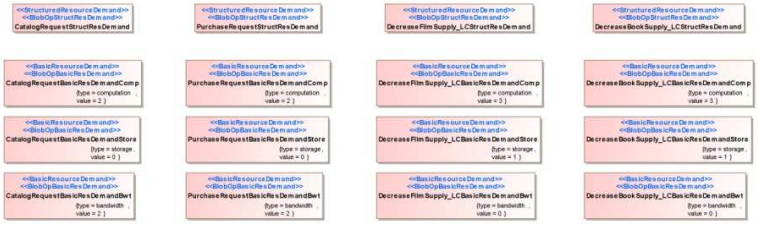


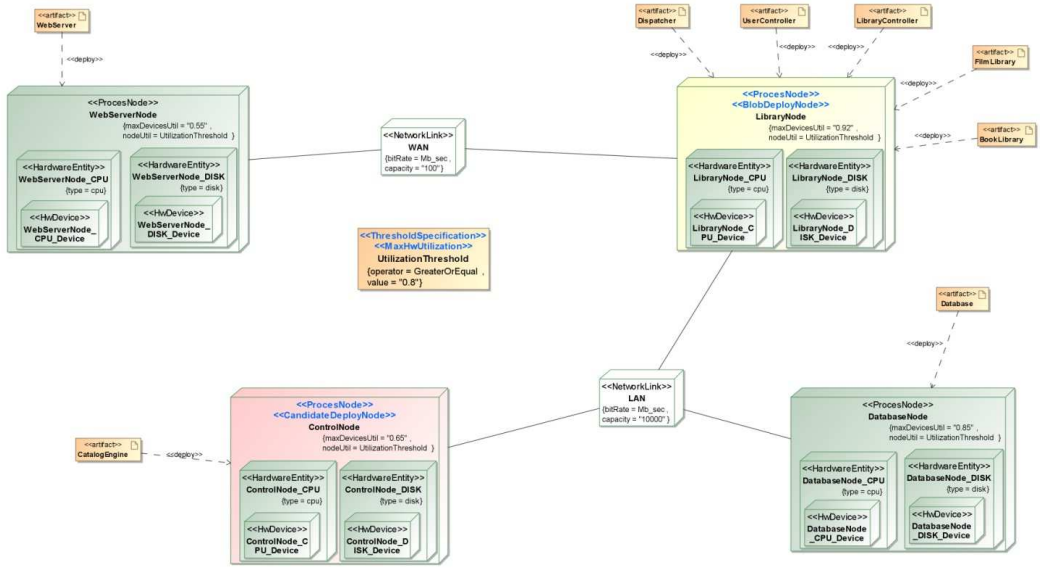

registerCustomerRequest
loginRequest
makePurchaseRequest
browseCatalogRequest
<<SoftwareEntityInstance>>
WebServer
<<SoftwareEntityInstance>>
UserController
checkLogin_DB
addCustomerProfile_DB
addCustomerPreferences_DB
addCustomerPictures_DB
<<SoftwareEntityInstance>>
Database
(isDB)
checkLogin_Item_DB()
addCustomerProfile_Item_DB()
addCustomerPreferences_Item_DB()
addCustomerPictures_Item_DB()
updateCatalog_Item_DB()
catalog_Item_DB()
browseCatalog
makePurchase
registerCustomer
login
checkLogin
registerRequest
updateCatalog_DB
catalog_DB
{probability = "0.8" }
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
FilmLibrary
getFilmAvailability_FL()
setFilmAvailability_FL()
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
Dispatcher
showFilmCatalog
{probability = "0.7" }
getFilmAvailability
{probability = "0.7" }
setFilmAvailability
filmCatalog
<<ClientRelationship>>
<<BlobClientRel>>
<<SoftwareEntityInstance>>
<<BlobPart>>
LibraryControllerPart1
decreaseFilmSupply_LC()
decreaseBookSupply_LC()
updateCatalog
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
CatalogEngine
catalog_cache()[probability = "0.2"]
purchaseRequest
catalogRequest
<<BlobSupplierRel>>
<<SupplierRelationship>>
<<SoftwareEntityInstance>>
<<BlobPart>>
LibraryControllerPart2
getBookAvailability
{probability = "0.3" }
setBookAvailability
showBookCatalog
{probability = "0.3" }
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
BookLibrary
getBookAvailability_BL()
setBookAvailability_BL()
bookCatalog
<<ThresholdSpecification>>
<<MaxNumClientConnects>>
ClientConnectionsThreshold
{operator = GreaterOrEqual ,
value = "6"}

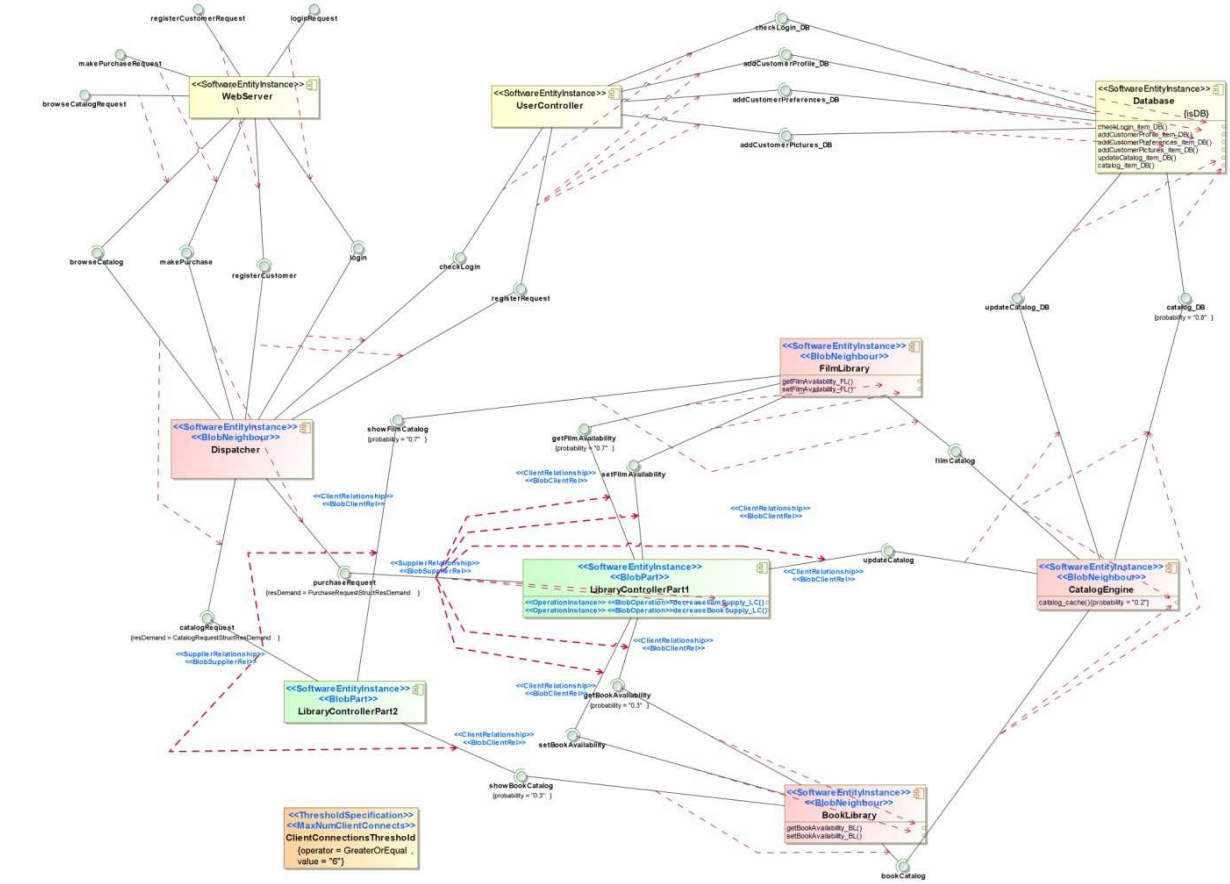

<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode_CPU_
Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_DISK_
Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<artifact>>
LibraryControllerPart1
<<artifact>>
LibraryControllerPart2
<<ProcesNode>>
<<BlobDeployNode>>
LibraryNode
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU_
Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK_
Device
<<artifact>>
CatalogEngine
<<artifact>>
Database
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000"}
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU_
Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK_
Device
<<ProcesNode>>
DatabaseNode
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU_
Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device

(SRM_2^{Blob}, TRM_2^{Blob})

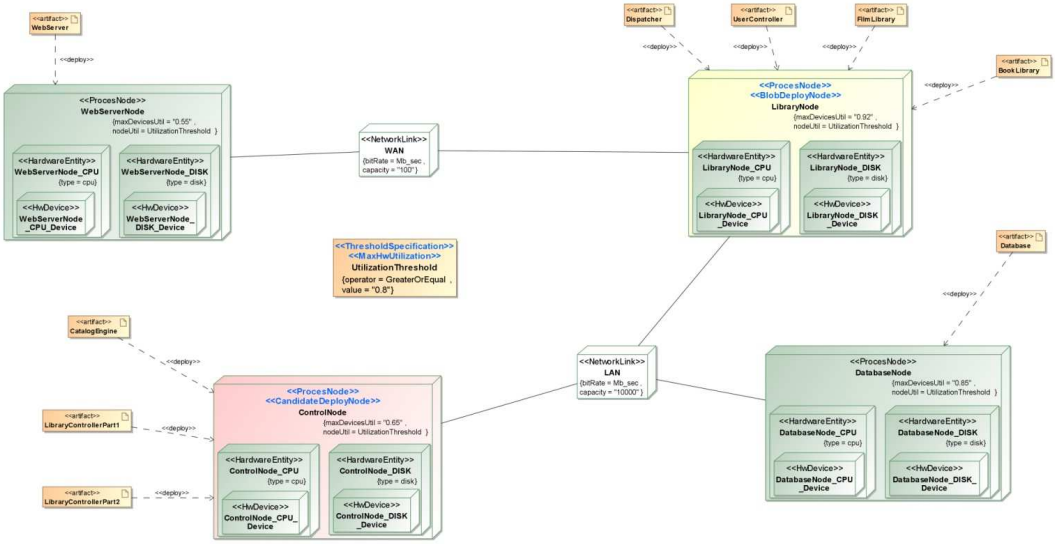




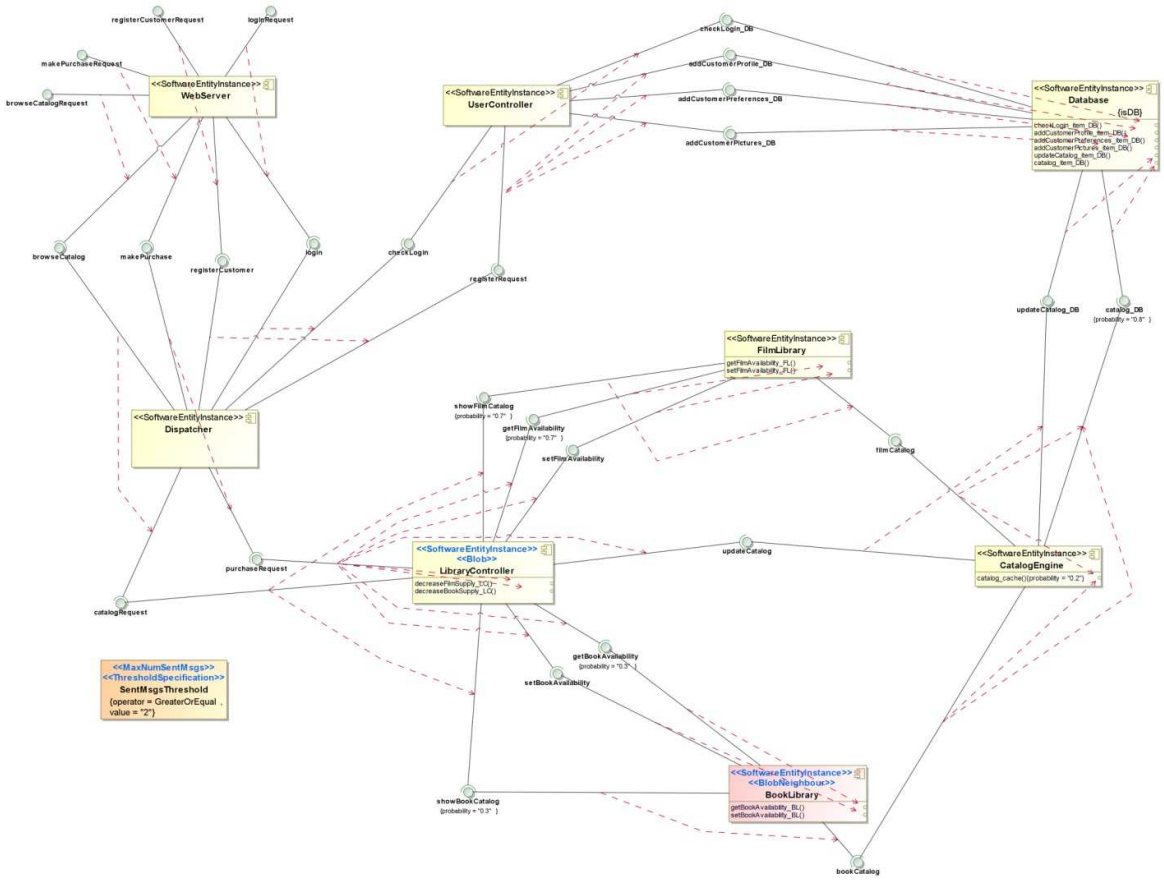


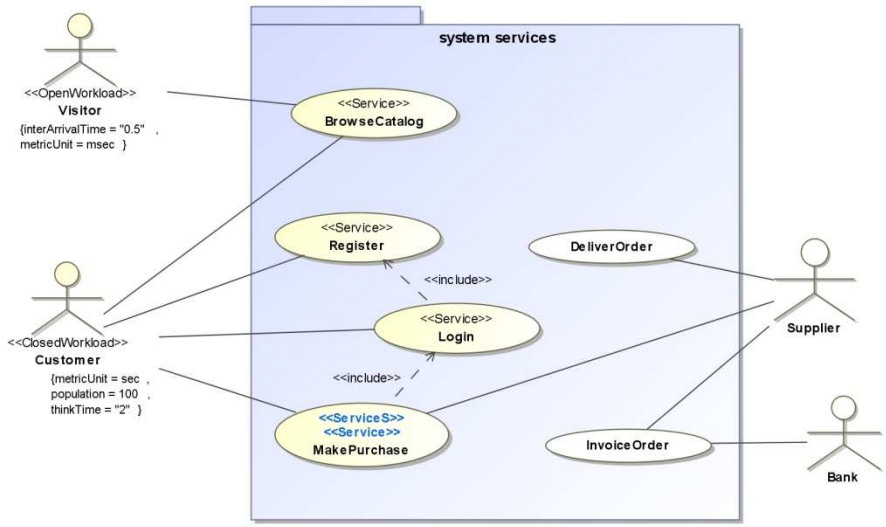

registerCustomerRequest
loginRequest
makePurchaseRequest
browseCatalogRequest
<<SoftwareEntityInstance>>
WebServer
<<SoftwareEntityInstance>>
UserController
checkLogin_DB
addCustomerProfile_DB
addCustomerPreferences_DB
addCustomerPictures_DB
<<SoftwareEntityInstance>>
Database
{isDB}
checkLogin_Item_DB()
addCustomerProfile_Item_DB()
addCustomerPreferences_Item_DB()
addCustomerPictures_Item_DB()
updateCatalog_Item_DB()
catalog_Item_DB()
browseCatalog
makePurchase
registerCustomer
login
checkLogin
registerRequest
updateCatalog_DB
catalog_DB
{probability = "0.8" }
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
FilmLibrary
getFilmAvailability_FL()
setFilmAvailability_FL()"
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
Dispatcher
showFilmCatalog
{probability = "0.7" }
getFilmAvailability
{probability = "0.7" }
setFilmAvailability
filmCatalog
<<ClientRelationship>>
<<BlobClientRel>>
<<SupplierRelationship>>
<<BlobSupplierRel>>
purchaseRequest
{resDemand = PurchaseRequestStructResDemand }
<<SoftwareEntityInstance>>
<<BlobPart>>
LibraryControllerPart1
<<OperationInstance>> <<BlobOperation>>decreaseFilmSupply_LC()
<<OperationInstance>> <<BlobOperation>>decreaseBookSupply_LC()
updateCatalog
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
CatalogEngine
catalog_cache()probability = "0.2"}
catalogRequest
{resDemand = CatalogRequestStructResDemand }
<<SoftwareEntityInstance>>
<<BlobPart>>
LibraryControllerPart2
getBookAvailability
{probability = "0.3" }
setBookAvailability
showBookCatalog
{probability = "0.3" }
<<SoftwareEntityInstance>>
<<BlobNeighbour>>
BookLibrary
getBookAvailability_BL()
setBookAvailability_BL()
bookCatalog
<<ThresholdSpecification>>
<<MaxNumClientConnects>>
ClientConnectionsThreshold
{operator = GreaterOrEqual ,
value = "6"}

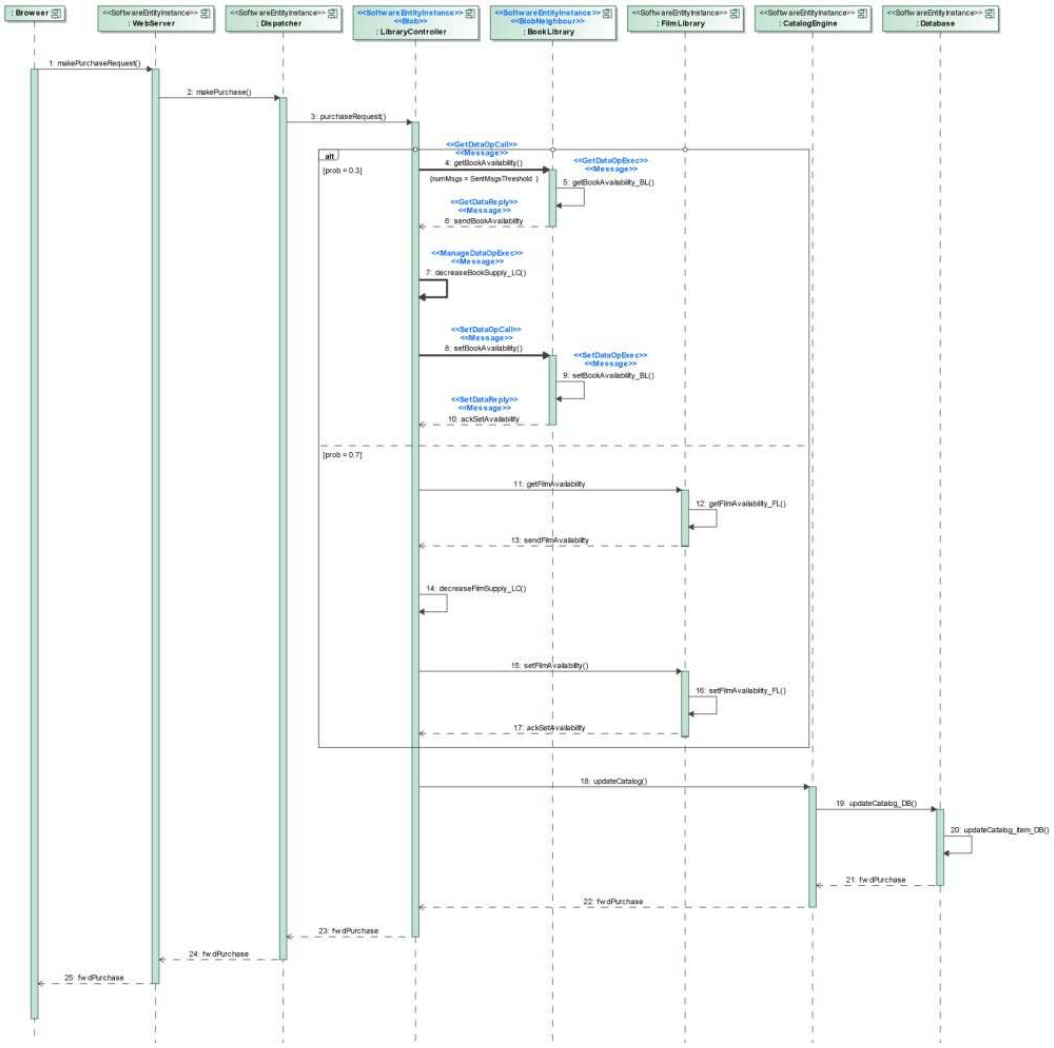

CatalogRequestStructResDemand
PurchaseRequestStructResDemand
DecreaseFilmSupply_LCStructResDemand
DecreaseBookSupply_LCStructResDemand
CatalogRequestBasicResDemandComp
{type = computation , value = 2 }
PurchaseRequestBasicResDemandComp
{type = computation , value = 2 }
DecreaseFilmSupply_LCBasicResDemandComp
{type = computation , value = 3 }
DecreaseBookSupply_LCBasicResDemandComp
{type = computation , value = 3 }
CatalogRequestBasicResDemandStore
{type = storage , value = 0 }
PurchaseRequestBasicResDemandStore
{type = storage , value = 0 }
DecreaseFilmSupply_LCBasicResDemandStore
{type = storage , value = 1 }
DecreaseBookSupply_LCBasicResDemandStore
{type = storage , value = 1 }
CatalogRequestBasicResDemandBwt
{type = bandwidth , value = 2 }
PurchaseRequestBasicResDemandBwt
{type = bandwidth , value = 2 }
DecreaseFilmSupply_LCBasicResDemandBwt
{type = bandwidth , value = 0 }
DecreaseBookSupply_LCBasicResDemandBwt
{type = bandwidth , value = 0 }

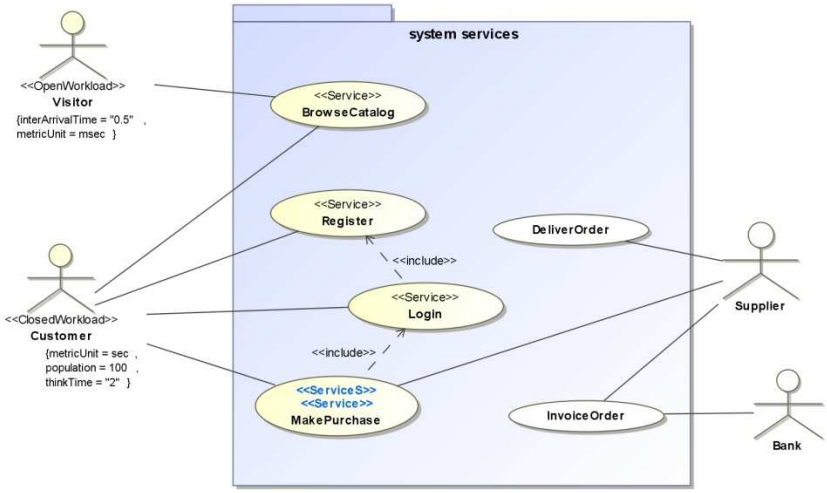

<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
{maxDevicesUtil = "0.55" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode
_CPU_Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
FilmLibrary
<<artifact>>
Book Library
<<ProcesNode>>
<<BlobDeployNode>>
LibraryNode
{maxDevicesUtil = "0.92" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU
_Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK
_Device
<<ThresholdSpecification>>
<<MaxHwUtilization>>
UtilizationThreshold
{operator = GreaterOrEqual ,
value = "0.8"}
<<artifact>>
Database
<<artifact>>
CatalogEngine
<<artifact>>
LibraryControllerPart1
<<artifact>>
LibraryControllerPart2
<<ProcesNode>>
<<CandidateDeployNode>>
ControlNode
{maxDevicesUtil = "0.65" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU_
Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK
_Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000" }
<<ProcesNode>>
DatabaseNode
{maxDevicesUtil = "0.85" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU
_Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device

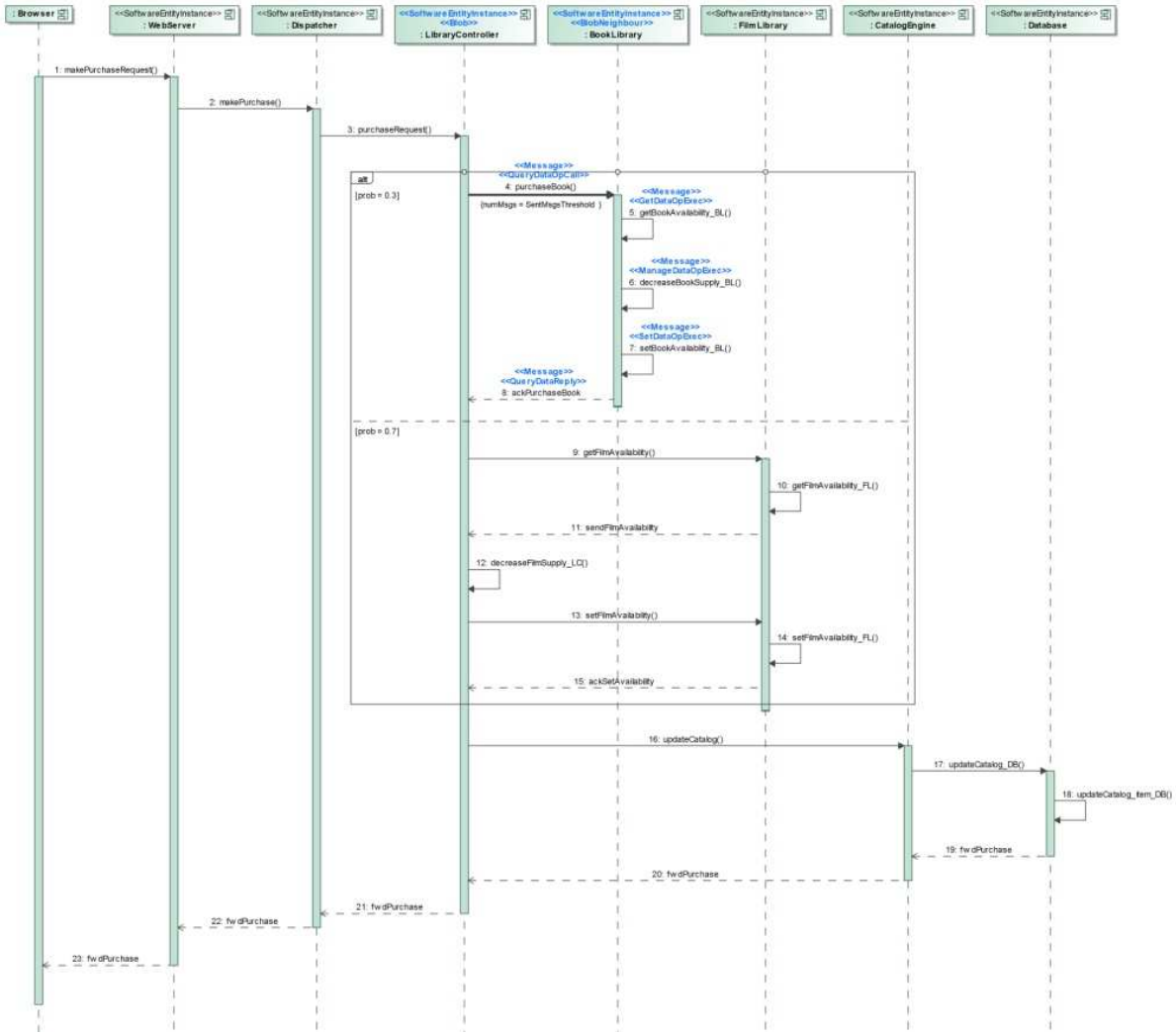
(SRM_3^{Blob}, TRM_3^{Blob})



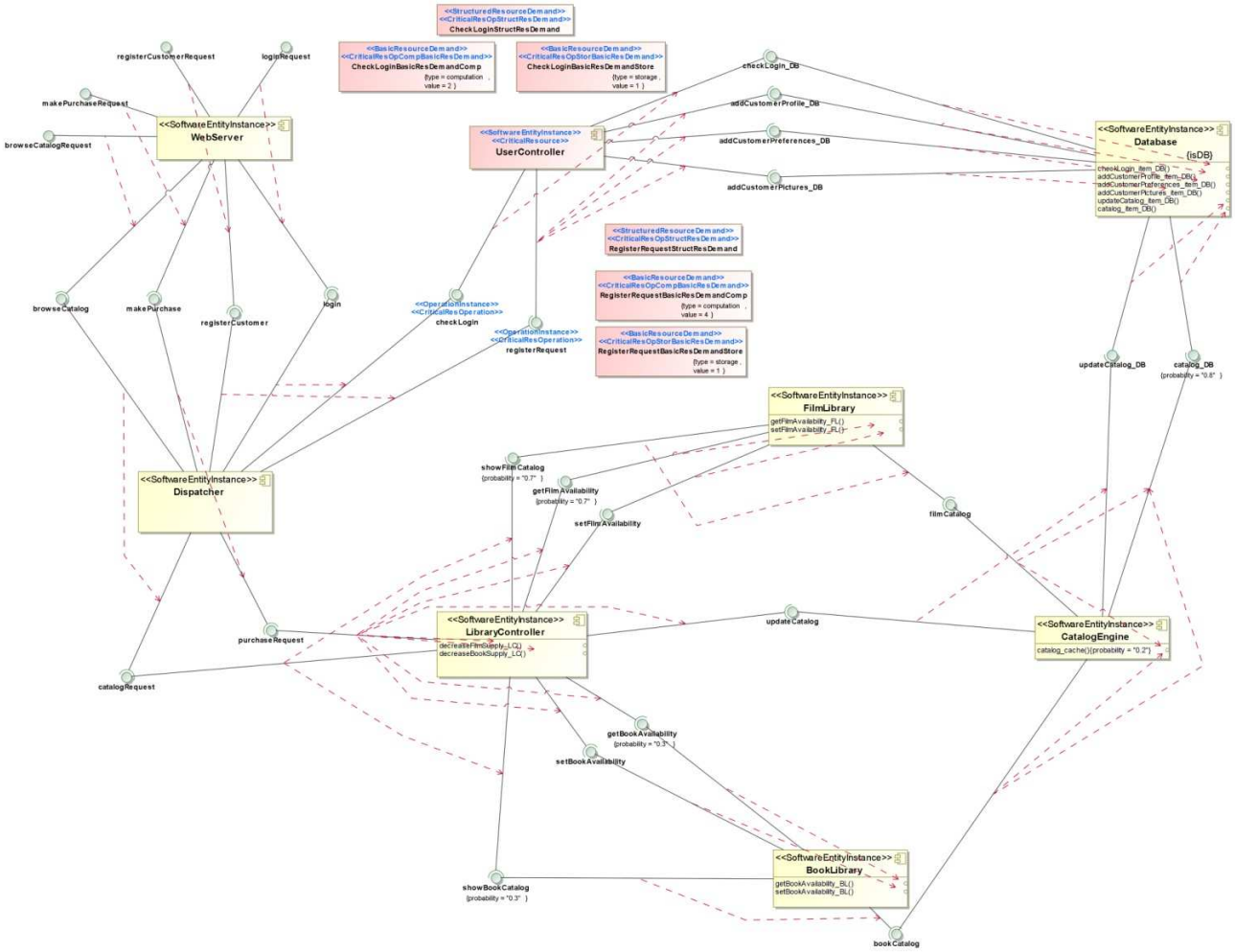


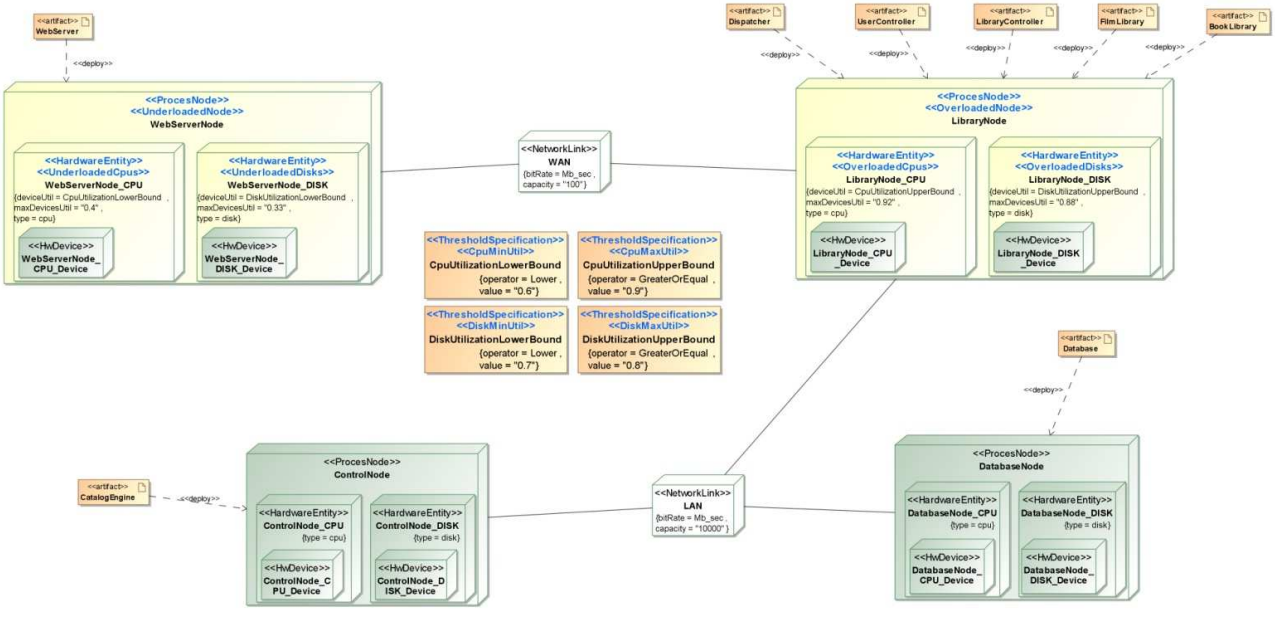

: Browser
<<SoftwareEntityInstance>> : WebServer
<<SoftwareEntityInstance>> : Dispatcher
<<SoftwareEntityInstance>> <<Blob>> : LibraryController
<<SoftwareEntityInstance>> <<BlobNeighbour>> : BookLibrary
<<SoftwareEntityInstance>> : FilmLibrary
<<SoftwareEntityInstance>> : CatalogEngine
<<SoftwareEntityInstance>> : Database
1: makePurchaseRequest()
2: makePurchase()
3: purchaseRequest()
alt
[prob = 0.3]
<<GetDataOpCall>>
<<Message>>
4: getBookAvailability()
{numMsgs = SentMsgsTreshold }
<<GetDataOpExec>>
<<Message>>
5: getBookAvailability_BL()
<<GetDataReply>>
<<Message>>
6: sendBookAvailability
<<ManageDataOpExec>>
<<Message>>
7: decreaseBookSupply_LC()
<<SetDataOpCall>>
<<Message>>
8: setBookAvailability()
<<SetDataOpExec>>
<<Message>>
9: setBookAvailability_BL()
<<SetDataReply>>
<<Message>>
10: ackSetAvailability
[prob = 0.7]
11: getFilmAvailability
12: getFilmAvailability_FL()
13: sendFilmAvailability
14: decreaseFilmSupply_LC()
15: setFilmAvailability()
16: setFilmAvailability_FL()
17: ackSetAvailability
18: updateCatalog()
19: updateCatalog_DB()
20: updateCatalog_item_DB()
21: fwdPurchase
22: fwdPurchase
23: fwdPurchase
24: fwdPurchase
25: fwdPurchase

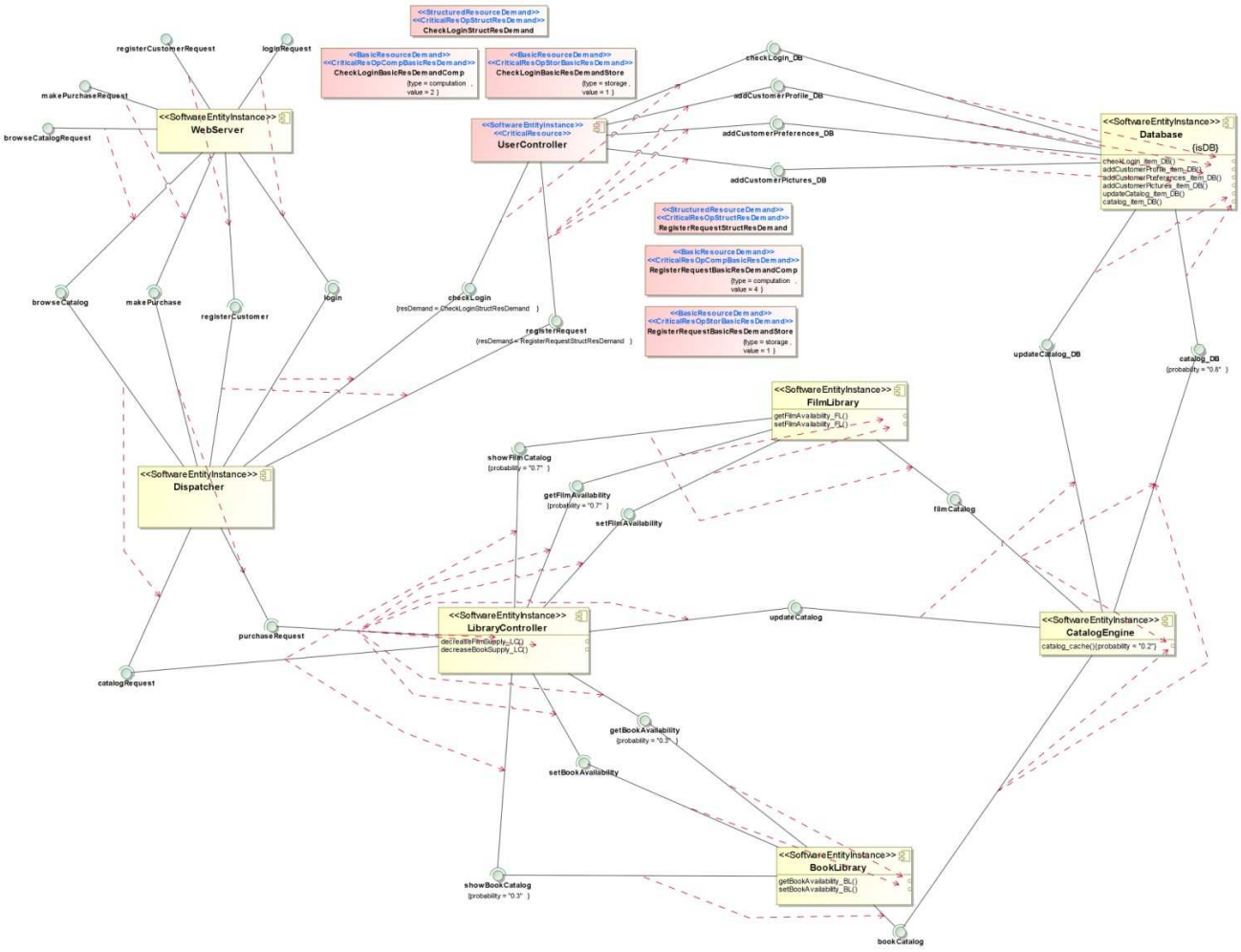

system services
<<OpenWorkload>>
Visitor
{interArrivalTime = "0.5" ,
metricUnit = msec }
<<Service>>
BrowseCatalog
<<Service>>
Register
DeliverOrder
Supplier
<<include>>
<<ClosedWorkload>>
Customer
{metricUnit = sec ,
population = 100 ,
thinkTime = "2" }
<<Service>>
Login
<<include>>
<<ServiceS>>
<<Service>>
MakePurchase
InvoiceOrder
Bank

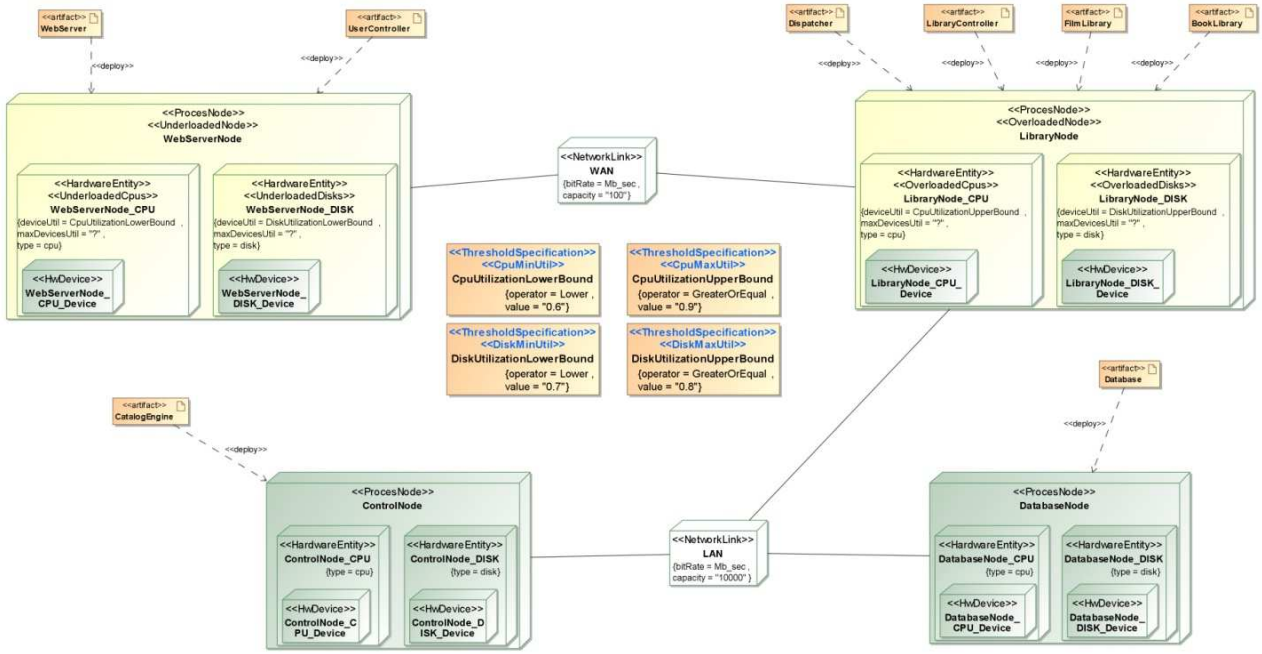

: Browser
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
<<Bob>>
: LibraryController
<<SoftwareEntityInstance>>
<<BobNeighbour>>
: BookLibrary
<<SoftwareEntityInstance>>
: FilmLibrary
<<SoftwareEntityInstance>>
: CatalogEngine
<<SoftwareEntityInstance>>
: Database
1: makePurchaseRequest()
2: makePurchase()
3: purchaseRequest()
alt
[prob = 0.3]
<<Message>>
<<QueryDataOpCall>>
4: purchaseBook()
[numMsgs = SentMsgsThreshold]
<<Message>>
<<GetDataOpExec>>
5: getBookAvailability_BL()
<<Message>>
<<ManageDataOpExec>>
6: decreaseBookSupply_BL()
<<Message>>
<<SetDataOpExec>>
7: setBookAvailability_BL()
<<Message>>
<<QueryDataReply>>
8: ackPurchaseBook
[prob = 0.7]
9: getFilmAvailability()
10: getFilmAvailability_FL()
11: sendFilmAvailability
12: decreaseFilmSupply_LC()
13: setFilmAvailability()
14: setFilmAvailability_FL()
15: ackSetAvailability
16: updateCatalog()
17: updateCatalog_DB()
18: updateCatalog_Item_DB()
19: fwdPurchase
20: fwdPurchase
21: fwdPurchase
22: fwdPurchase
23: fwdPurchase

(SRM_1^{CPS}, TRM_1^{CPS})

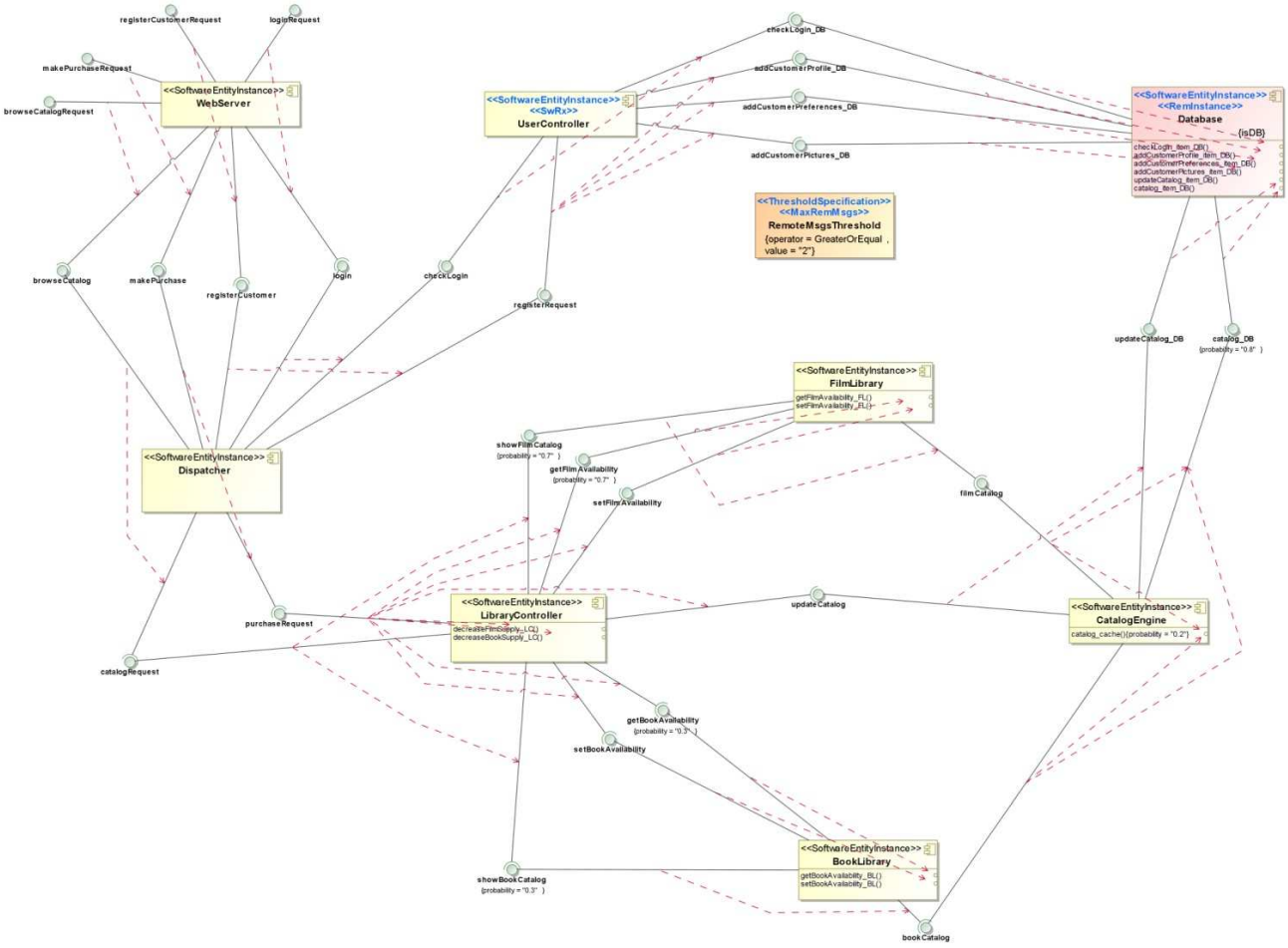


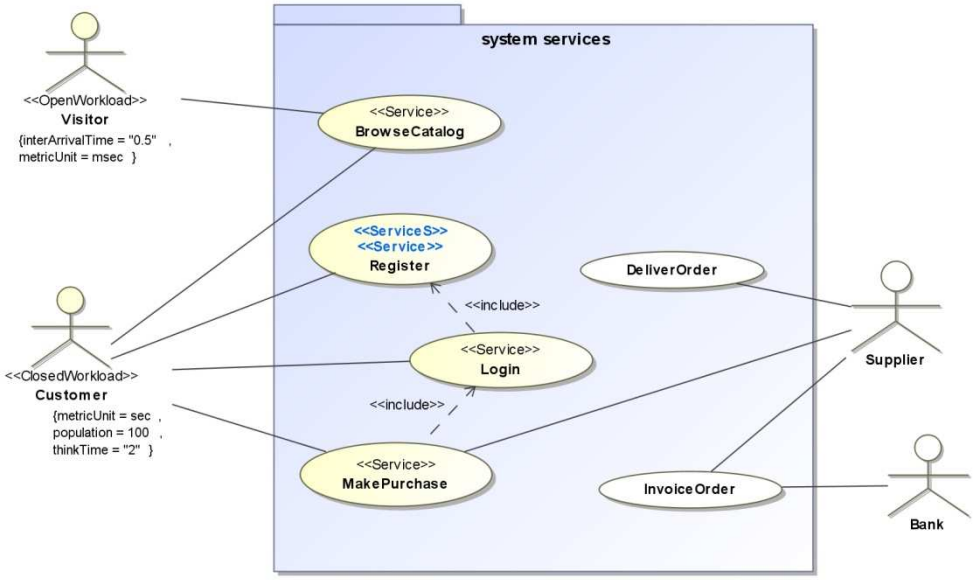


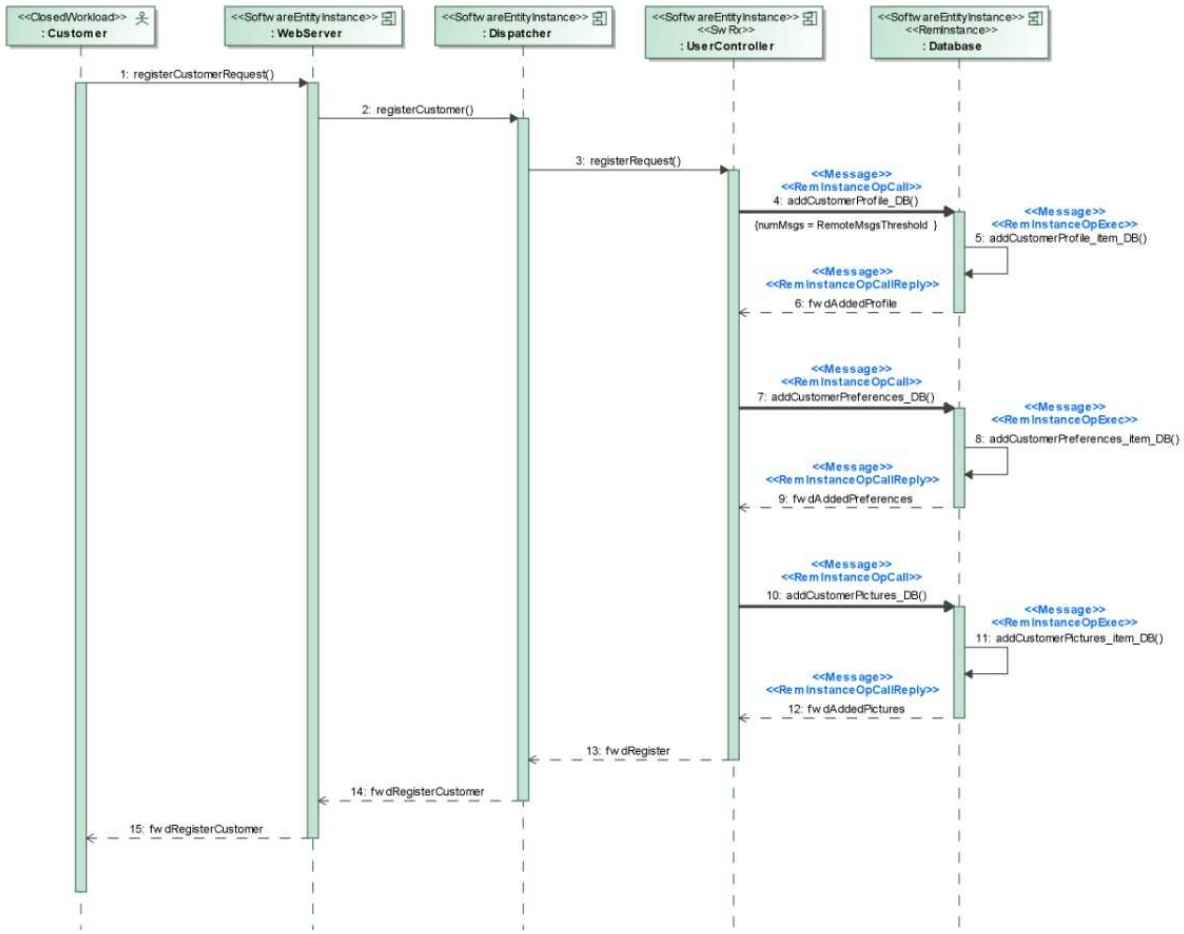

<<StructuredResourceDemand>>
<<CriticalResOpStructResDemand>>
CheckLoginStructResDemand
registerCustomerRequest
loginRequest
<<BasicResourceDemand>>
<<CriticalResOpCompBasicResDemand>>
CheckLoginBasicResDemandComp
{type = computation , value = 2 }
<<BasicResourceDemand>>
<<CriticalResOpStorBasicResDemand>>
CheckLoginBasicResDemandStore
{type = storage , value = 1 }
checkLogin_DB
addCustomerProfile_DB
addCustomerPreferences_DB
addCustomerPictures_DB
makePurchaseRequest
browseCatalogRequest
<<SoftwareEntityInstance>>
WebServer
<<SoftwareEntityInstance>>
<<CriticalResource>>
UserController
<<SoftwareEntityInstance>>
Database
{isDB}
checkLogin_Item_DB()
addCustomerProfile_Item_DB()
addCustomerPreferences_Item_DB()
addCustomerPictures_Item_DB()
updateCatalog_Item_DB()
catalog_Item_DB()
<<StructuredResourceDemand>>
<<CriticalResOpStructResDemand>>
RegisterRequestStructResDemand
<<BasicResourceDemand>>
<<CriticalResOpCompBasicResDemand>>
RegisterRequestBasicResDemandComp
{type = computation , value = 4 }
<<BasicResourceDemand>>
<<CriticalResOpStorBasicResDemand>>
RegisterRequestBasicResDemandStore
{type = storage , value = 1 }
browseCatalog
makePurchase
registerCustomer
login
checkLogin
{resDemand = CheckLoginStructResDemand }
registerRequest
{resDemand = RegisterRequestStructResDemand }
updateCatalog_DB
catalog_DB
{probability = "0.8" }
<<SoftwareEntityInstance>>
FilmLibrary
getFilmAvailability_FL()
setFilmAvailability_FL()
showFilmCatalog
{probability = "0.7" }
getFilmAvailability
{probability = "0.7" }
setFilmAvailability
filmCatalog
<<SoftwareEntityInstance>>
Dispatcher
purchaseRequest
<<SoftwareEntityInstance>>
LibraryController
decreaseFilmSupply_LC()
decreaseBookSupply_LC()
updateCatalog
<<SoftwareEntityInstance>>
CatalogEngine
catalog_cache(){probability = "0.2"}
catalogRequest
getBookAvailability
{probability = "0.3" }
setBookAvailability
<<SoftwareEntityInstance>>
BookLibrary
getBookAvailability_BL()
setBookAvailability_BL()
showBookCatalog
{probability = "0.3" }
bookCatalog

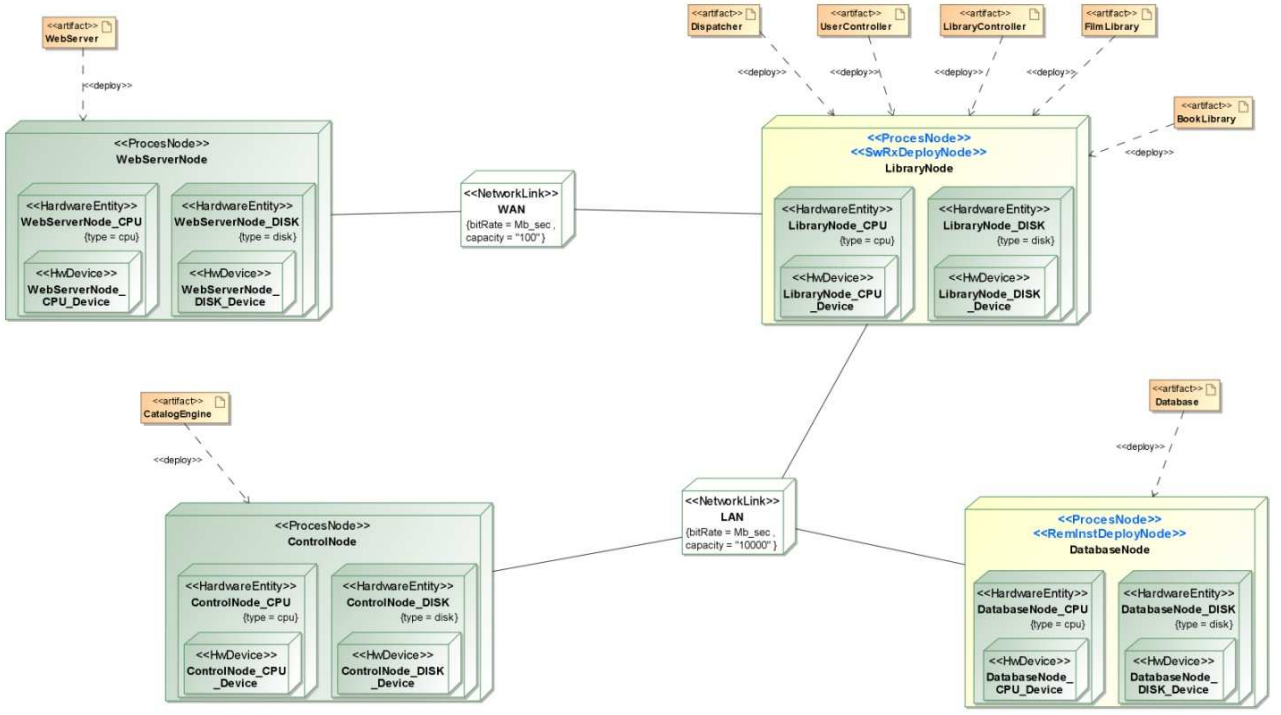

<<artifact>>
WebServer
<<artifact>>
UserController
<<deploy>>
<<ProcesNode>>
<<UnderloadedNode>>
WebServerNode
<<HardwareEntity>>
<<UnderloadedCpus>>
WebServerNode_CPU
{deviceUtil = CpuUtilizationLowerBound , maxDevicesUtil = "?" , type = cpu}
<<HardwareEntity>>
<<UnderloadedDisks>>
WebServerNode_DISK
{deviceUtil = DiskUtilizationLowerBound , maxDevicesUtil = "?" , type = disk}
<<HwDevice>>
WebServerNode_CPU_Device
<<HwDevice>>
WebServerNode_DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec , capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<ProcesNode>>
<<OverloadedNode>>
LibraryNode
<<HardwareEntity>>
<<OverloadedCpus>>
LibraryNode_CPU
{deviceUtil = CpuUtilizationUpperBound , maxDevicesUtil = "?" , type = cpu}
<<HardwareEntity>>
<<OverloadedDisks>>
LibraryNode_DISK
{deviceUtil = DiskUtilizationUpperBound , maxDevicesUtil = "?" , type = disk}
<<HwDevice>>
LibraryNode_CPU_Device
<<HwDevice>>
LibraryNode_DISK_Device
<<ThresholdSpecification>>
<<CpuMinUtil>>
CpuUtilizationLowerBound
{operator = Lower , value = "0.6"}
<<ThresholdSpecification>>
<<CpuMaxUtil>>
CpuUtilizationUpperBound
{operator = GreaterOrEqual , value = "0.9"}
<<ThresholdSpecification>>
<<DiskMinUtil>>
DiskUtilizationLowerBound
{operator = Lower , value = "0.7"}
<<ThresholdSpecification>>
<<DiskMaxUtil>>
DiskUtilizationUpperBound
{operator = GreaterOrEqual , value = "0.8"}
<<artifact>>
CatalogEngine
<<artifact>>
Database
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_CPU_Device
<<HwDevice>>
ControlNode_DISK_Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec , capacity = "10000"}
<<ProcesNode>>
DatabaseNode
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_CPU_Device
<<HwDevice>>
DatabaseNode_DISK_Device

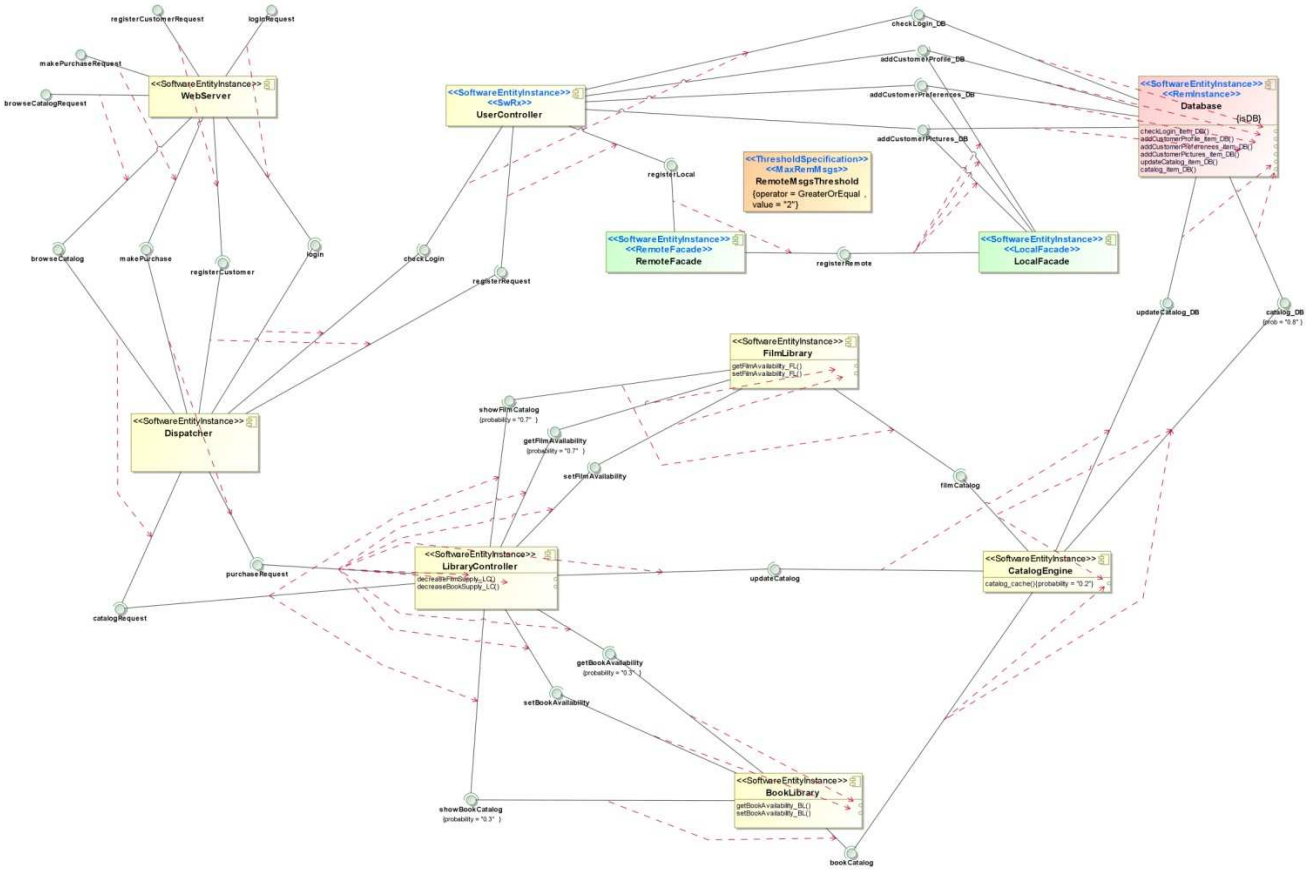
(SRM_1^{EST}, TRM_1^{EST})

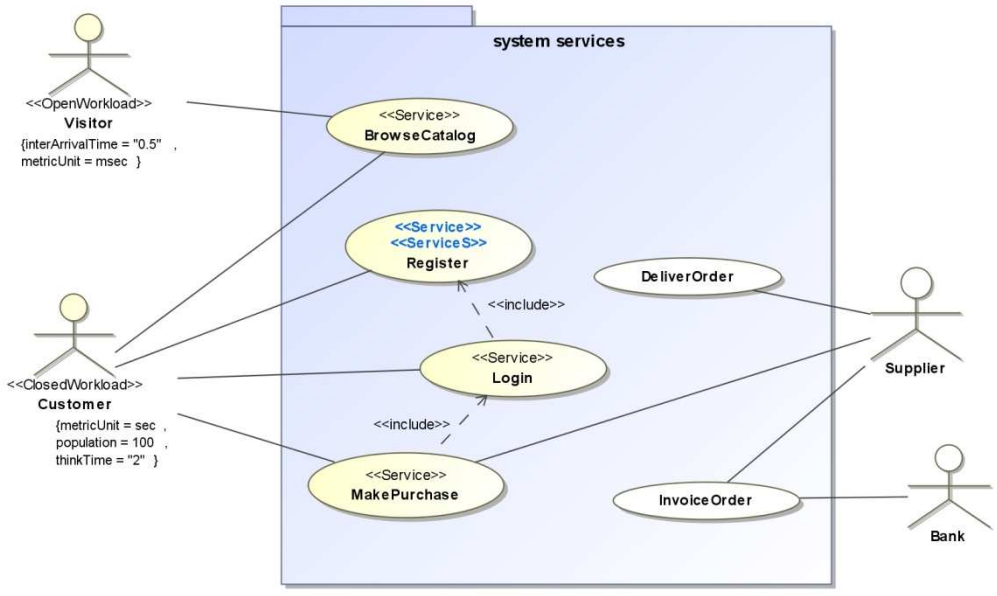


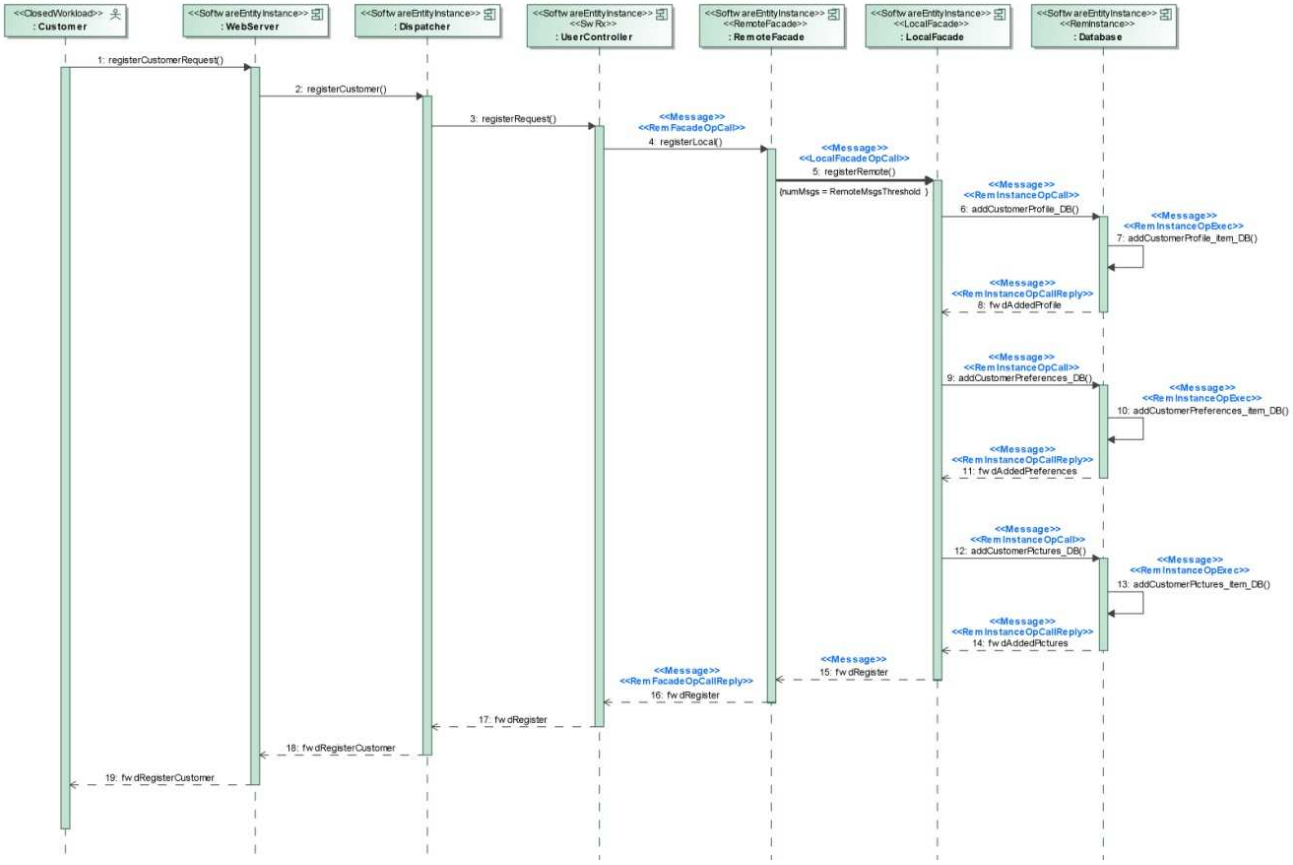


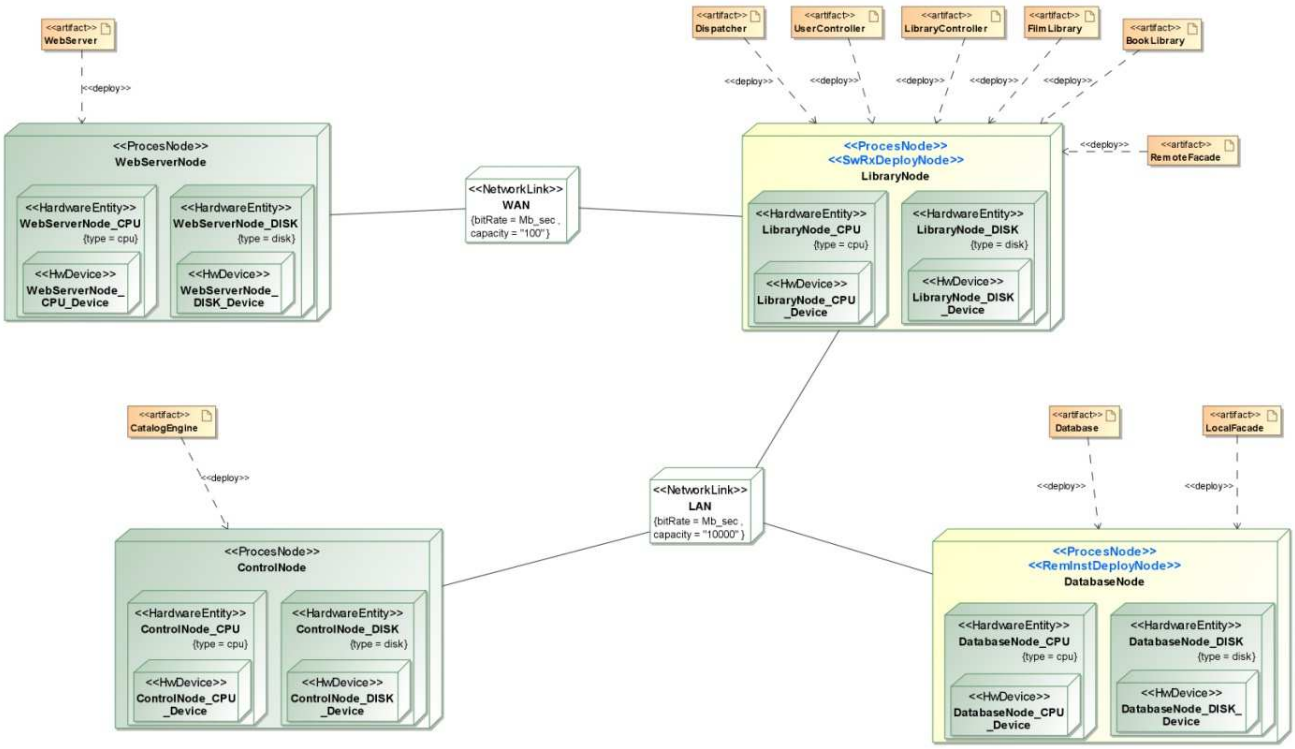

<<ClosedWorkload>>
: Customer
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
<<SwRx>>
: UserController
<<SoftwareEntityInstance>>
<<RemInstance>>
: Database
1: registerCustomerRequest()
2: registerCustomer()
3: registerRequest()
<<Message>>
<<RemInstanceOpCall>>
4: addCustomerProfile_DB()
{numMsgs = RemoteMsgsThreshold }
<<Message>>
<<RemInstanceOpExec>>
5: addCustomerProfile_item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
6: fwdAddedProfile
<<Message>>
<<RemInstanceOpCall>>
7: addCustomerPreferences_DB()
<<Message>>
<<RemInstanceOpExec>>
8: addCustomerPreferences_item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
9: fwdAddedPreferences
<<Message>>
<<RemInstanceOpCall>>
10: addCustomerPictures_DB()
<<Message>>
<<RemInstanceOpExec>>
11: addCustomerPictures_item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
12: fwdAddedPictures
13: fwdRegister
14: fwdRegisterCustomer
15: fwdRegisterCustomer


<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
CPU_Device
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100" }
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<ProcesNode>>
<<SwRxDeployNode>>
LibraryNode
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_CPU
_Device
<<HwDevice>>
LibraryNode_DISK
_Device
<<artifact>>
CatalogEngine
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_CPU
_Device
<<HwDevice>>
ControlNode_DISK
_Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000" }
<<artifact>>
Database
<<ProcesNode>>
<<RemInstDeployNode>>
DatabaseNode
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_
CPU_Device
<<HwDevice>>
DatabaseNode_
DISK_Device

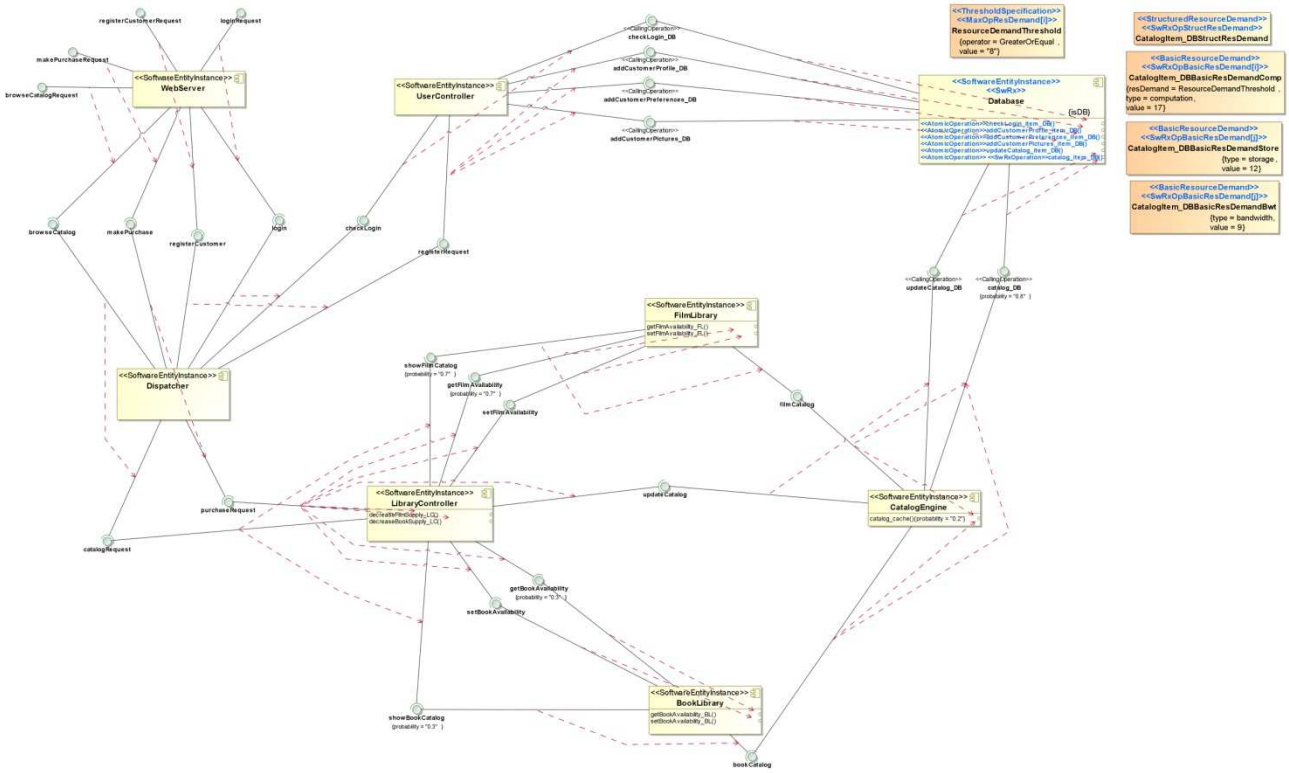

registerCustomerRequest
loginRequest
makePurchaseRequest
browseCatalogRequest
<<SoftwareEntityInstance>>
WebServer
<<SoftwareEntityInstance>>
<<SwRs>>
UserController
checkLogin_DB
addCustomerProfile_DB
addCustomerPreferences_DB
addCustomerPictures_DB
<<SoftwareEntityInstance>>
<<RmInstance>>
Database
{isDB}
checkLogin_Item_DB()
addCustomerProfile_Item_DB()
addCustomerPreferences_Item_DB()
addCustomerPictures_Item_DB()
updateCatalog_Item_DB()
catalog_Item_DB()
registerLocal
<<ThresholdSpecification>>
<<MaxRemMsgs>>
RemoteMsgsThreshold
{operator = GreaterOrEqual ,
value = "2"}
<<SoftwareEntityInstance>>
<<RemoteFacade>>
RemoteFacade
registerRemote
<<SoftwareEntityInstance>>
<<LocalFacade>>
LocalFacade
browseCatalog
makePurchase
registerCustomer
login
checkLogin
registerRequest
updateCatalog_DB
catalog_DB
{prob = "0.8" }
<<SoftwareEntityInstance>>
FilmLibrary
getFilmAvailability_FL()
setFilmAvailability_FL()
showFilmCatalog
{probability = "0.7" }
getFilmAvailability
{probability = "0.7" }
setFilmAvailability
filmCatalog
<<SoftwareEntityInstance>>
Dispatcher
purchaseRequest
catalogRequest
<<SoftwareEntityInstance>>
LibraryController
declareBookBuy_LC()
decreaseBookSupply_LC()
updateCatalog
<<SoftwareEntityInstance>>
CatalogEngine
catalog_cache(){probability = "0.2"}
getBookAvailability
{probability = "0.3" }
setBookAvailability
<<SoftwareEntityInstance>>
BookLibrary
getBookAvailability_BL()
setBookAvailability_BL()
showBookCatalog
{probability = "0.3" }
bookCatalog

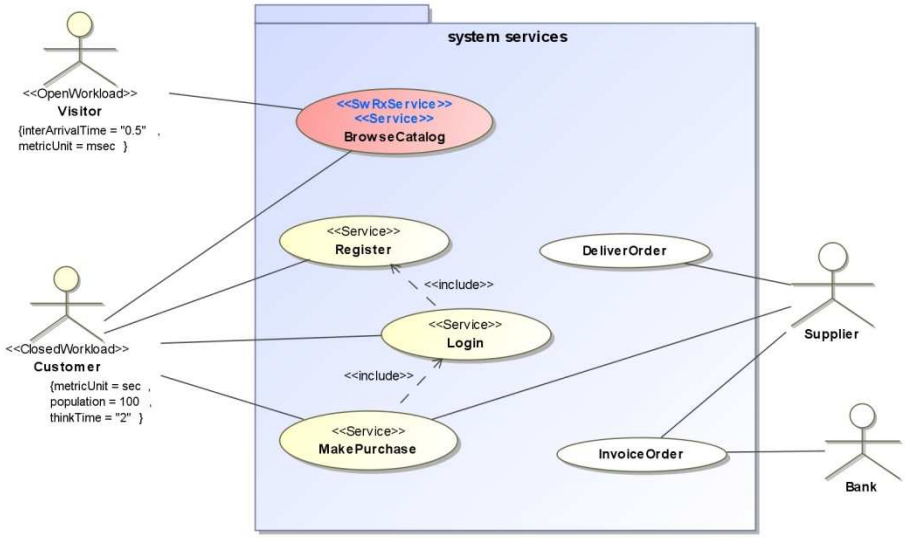

system services
<<OpenWorkload>>
Visitor
{interArrivalTime = "0.5" ,
metricUnit = msec }
<<Service>>
BrowseCatalog
<<Service>>
<<ServiceS>>
Register
DeliverOrder
<<include>>
<<ClosedWorkload>>
Customer
{metricUnit = sec ,
population = 100 ,
thinkTime = "2" }
<<Service>>
Login
Supplier
<<include>>
<<Service>>
MakePurchase
InvoiceOrder
Bank

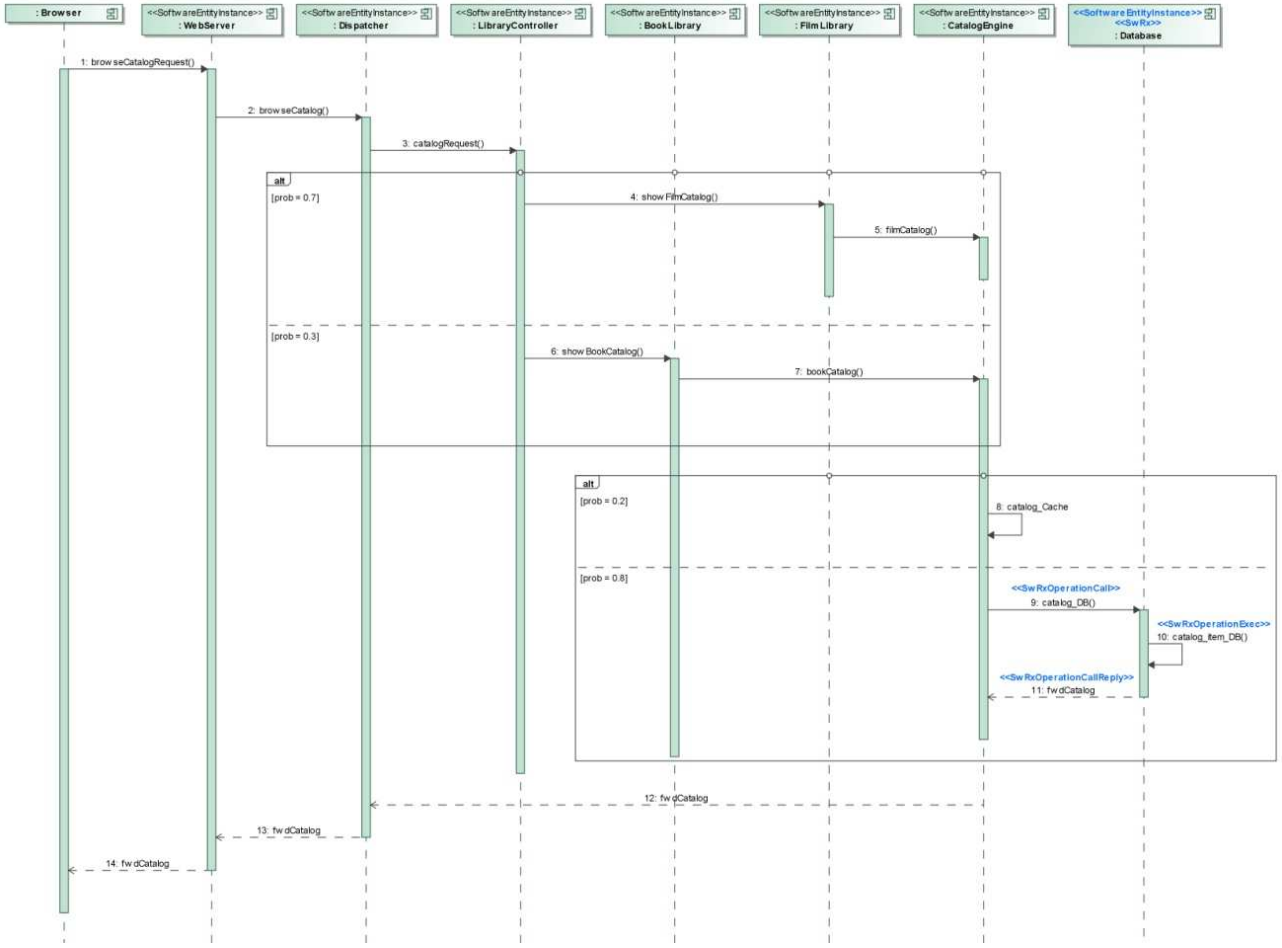

<<ClosedWorkload>>
: Customer
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
<<SwRx>>
: UserController
<<SoftwareEntityInstance>>
<<RemoteFacade>>
: RemoteFacade
<<SoftwareEntityInstance>>
<<LocalFacade>>
: LocalFacade
<<SoftwareEntityInstance>>
<<RemInstance>>
: Database
1: registerCustomerRequest()
2: registerCustomer()
3: registerRequest()
<<Message>>
<<RemFacadeOpCall>>
4: registerLocal()
<<Message>>
<<LocalFacadeOpCall>>
5: registerRemote()
{numMsgs = RemoteMsgsThreshold }
<<Message>>
<<RemInstanceOpCall>>
6: addCustomerProfile_DB()
<<Message>>
<<RemInstanceOpExec>>
7: addCustomerProfile_Item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
8: fwdAddedProfile
<<Message>>
<<RemInstanceOpCall>>
9: addCustomerPreferences_DB()
<<Message>>
<<RemInstanceOpExec>>
10: addCustomerPreferences_Item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
11: fwdAddedPreferences
<<Message>>
<<RemInstanceOpCall>>
12: addCustomerPictures_DB()
<<Message>>
<<RemInstanceOpExec>>
13: addCustomerPictures_Item_DB()
<<Message>>
<<RemInstanceOpCallReply>>
14: fwdAddedPictures
<<Message>>
15: fwdRegister
<<Message>>
<<RemFacadeOpCallReply>>
16: fwdRegister
17: fwdRegister
18: fwdRegisterCustomer
19: fwdRegisterCustomer

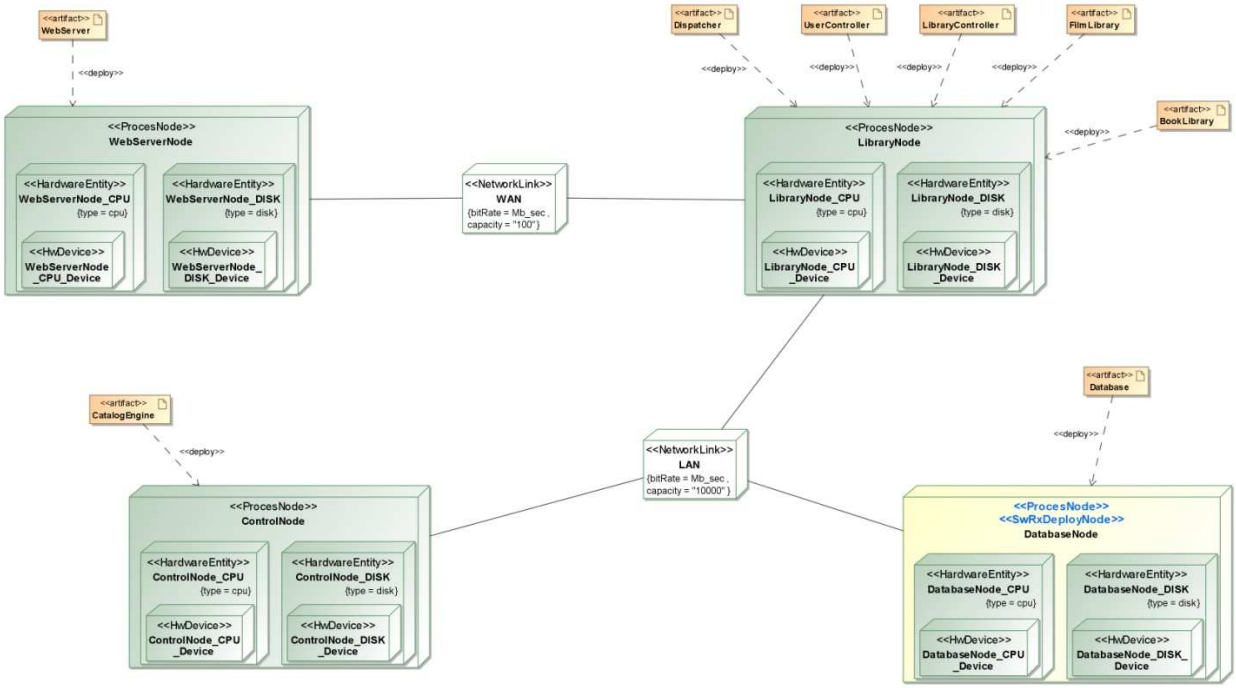

<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode_
CPU_Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100" }
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<deploy>>
<<ProcesNode>>
<<SwRxDeployNode>>
LibraryNode
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU
_Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK
_Device
<<artifact>>
RemoteFacade
<<artifact>>
CatalogEngine
<<deploy>>
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU
_Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK
_Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000" }
<<artifact>>
Database
<<artifact>>
LocalFacade
<<ProcesNode>>
<<RemInstDeployNode>>
DatabaseNode
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU
_Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device

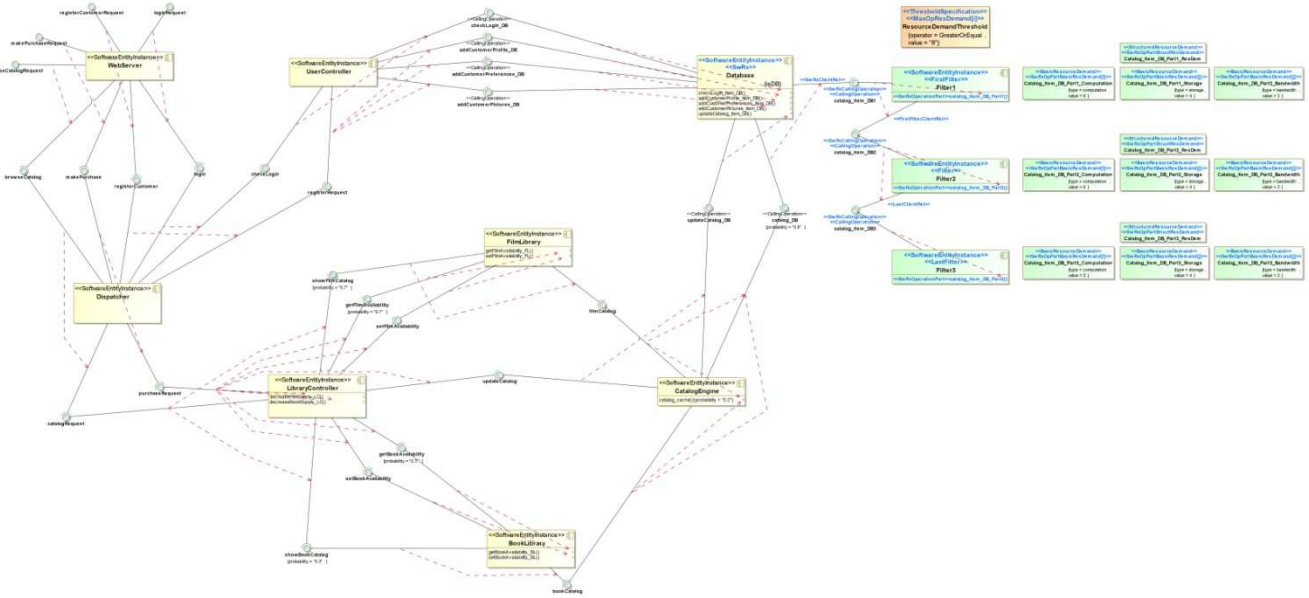
($SRM_1^{P\&F}$, $TRM_1^{P\&F}$)

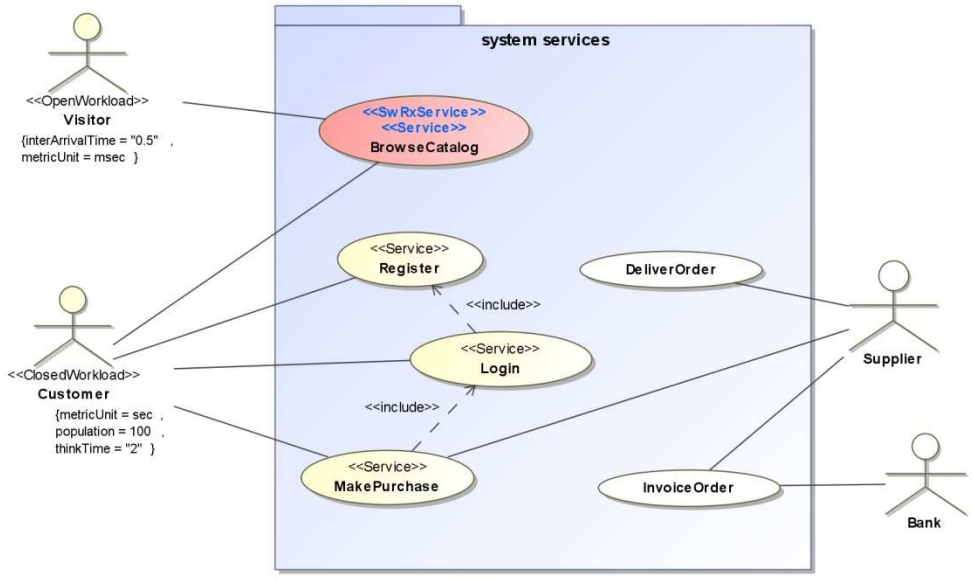


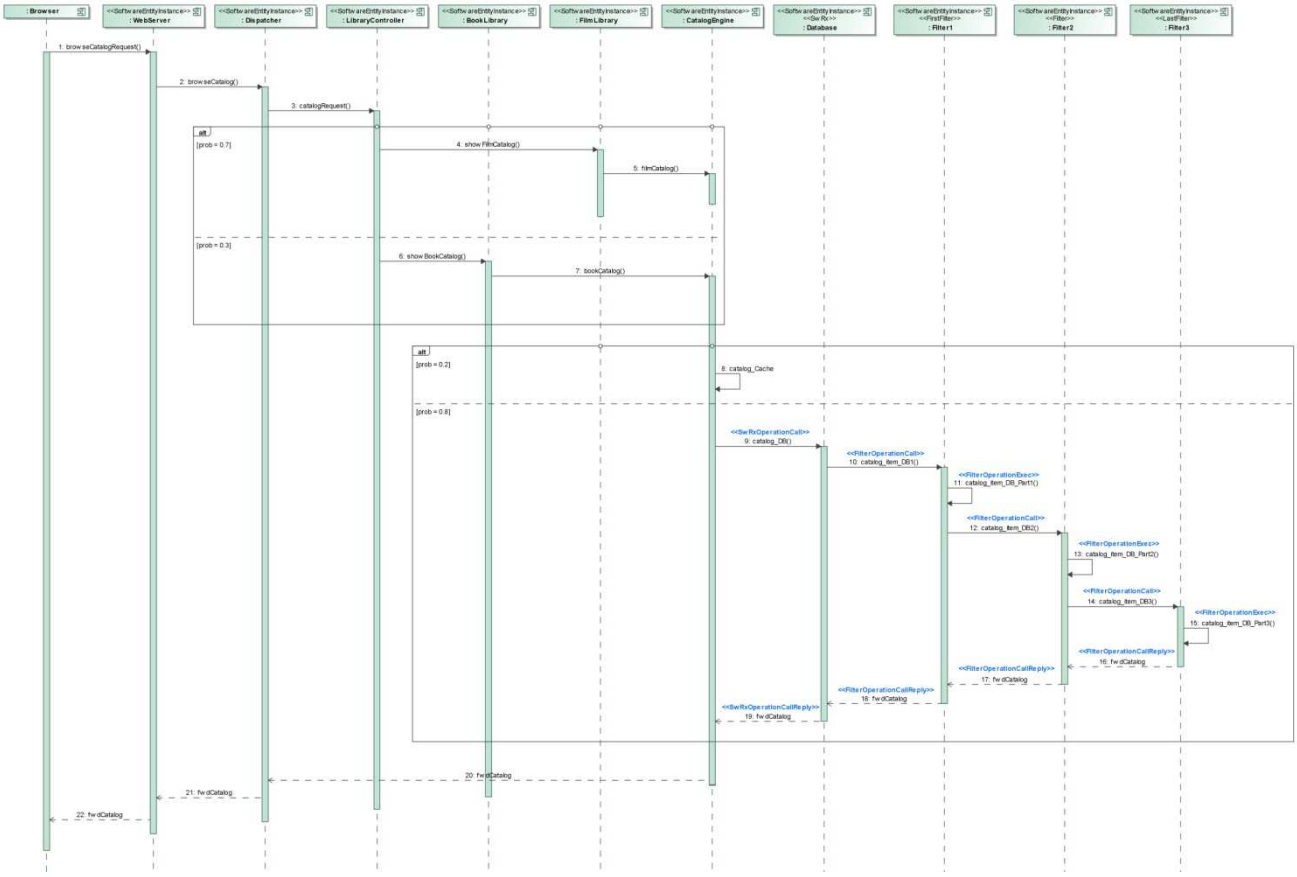


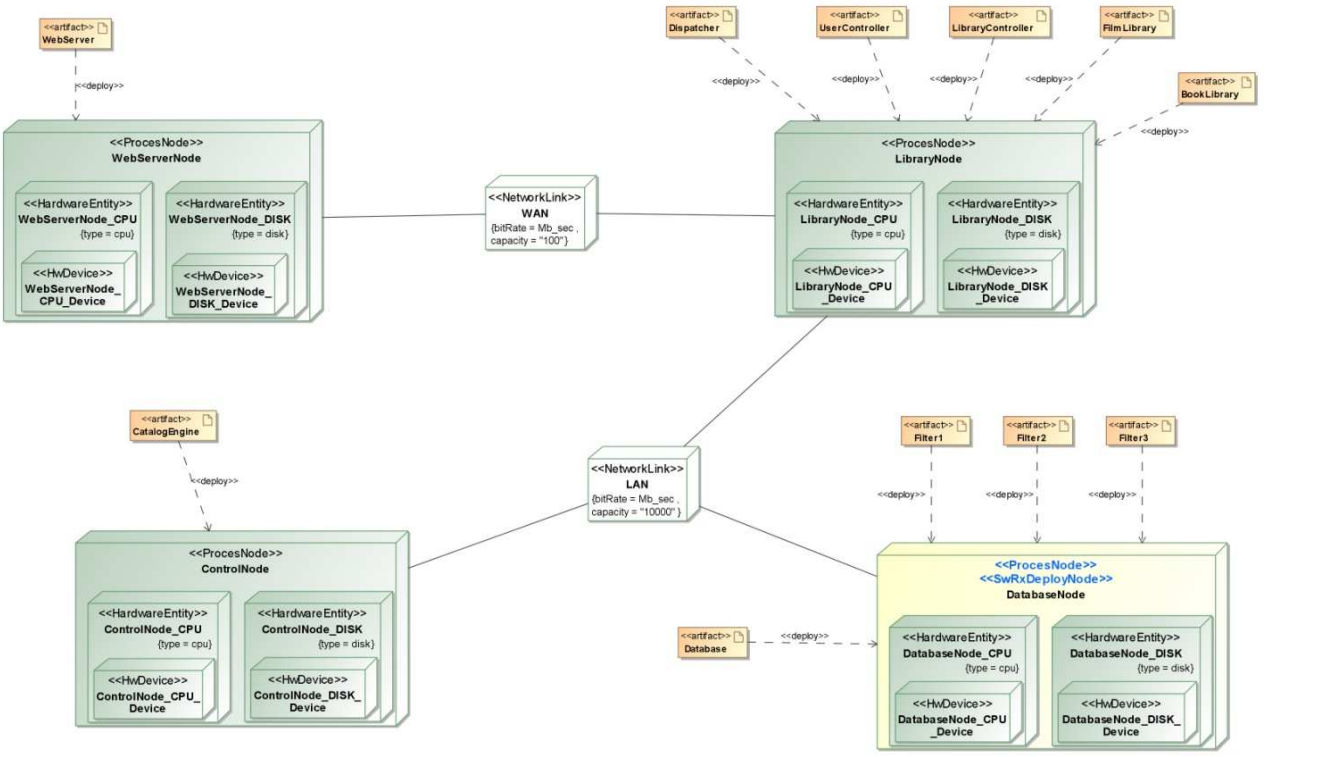

: Browser
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
: LibraryController
<<SoftwareEntityInstance>>
: BookLibrary
<<SoftwareEntityInstance>>
: FilmLibrary
<<SoftwareEntityInstance>>
: CatalogEngine
<<SoftwareEntityInstance>>
<<SwRx>>
: Database
1: browseCatalogRequest()
2: browseCatalog()
3: catalogRequest()
alt
[prob = 0.7]
4: showFilmCatalog()
5: filmCatalog()
[prob = 0.3]
6: showBookCatalog()
7: bookCatalog()
alt
[prob = 0.2]
8: catalog_Cache
[prob = 0.8]
<<SwRxOperationCall>>
9: catalog_DB()
<<SwRxOperationExec>>
10: catalog_item_DB()
<<SwRxOperationCallReply>>
11: fwdCatalog
12: fwdCatalog
13: fwdCatalog
14: fwdCatalog


<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode
_CPU_Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<ProcesNode>>
LibraryNode
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU
_Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK
_Device
<<artifact>>
CatalogEngine
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000"}
<<artifact>>
Database
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU
_Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK
_Device
<<ProcesNode>>
<<SwRxDeployNode>>
DatabaseNode
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU
_Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device

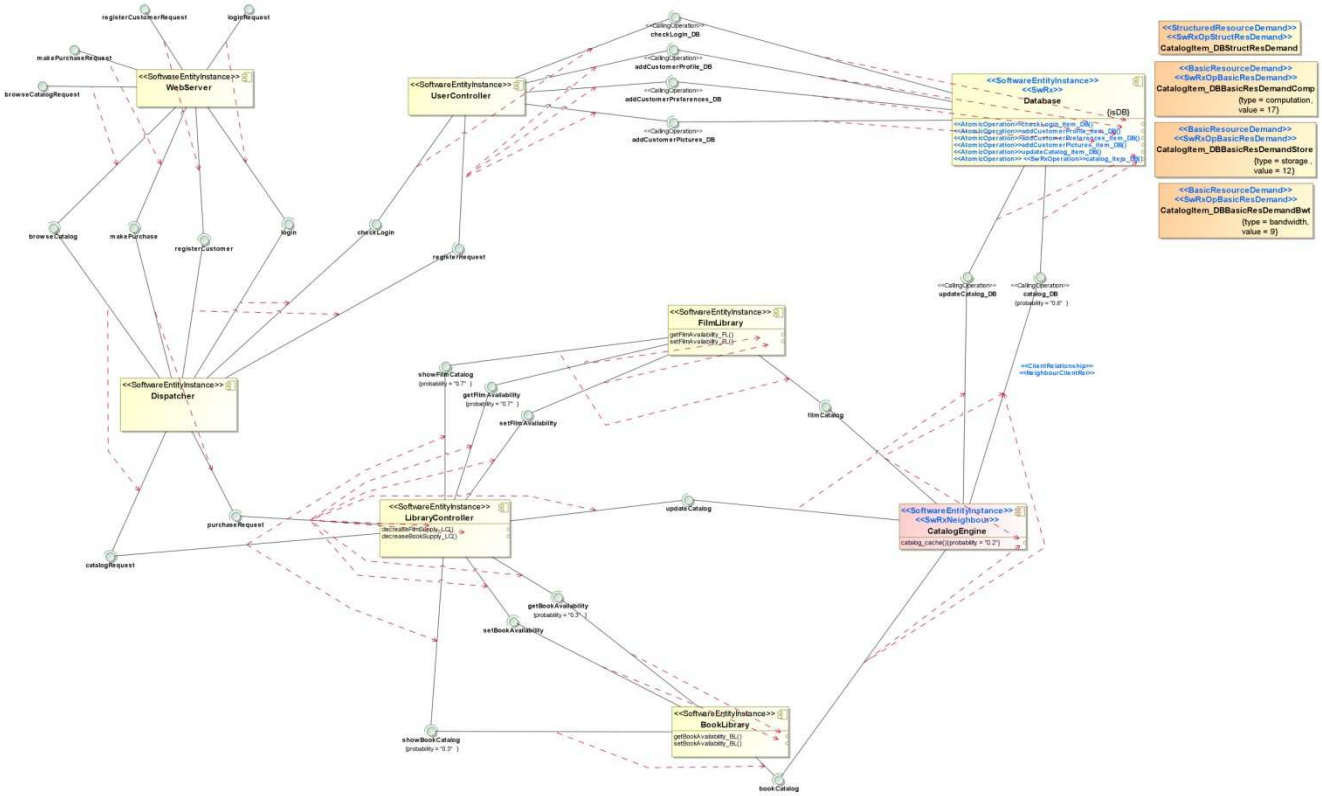


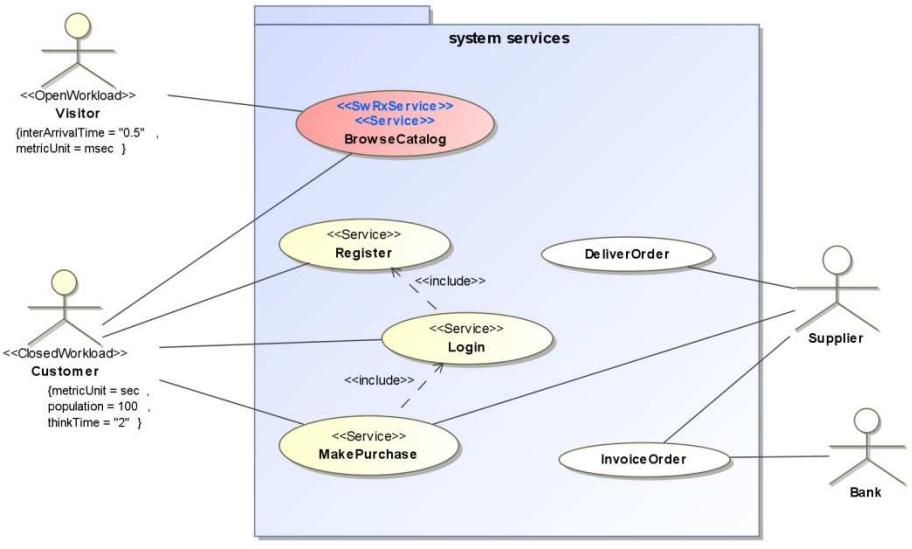

system services
<<OpenWorkload>>
Visitor
{interArrivalTime = "0.5" ,
metricUnit = msec }
<<SwRxService>>
<<Service>>
BrowseCatalog
<<Service>>
Register
DeliverOrder
<<include>>
<<ClosedWorkload>>
Customer
{metricUnit = sec ,
population = 100 ,
thinkTime = "2" }
<<Service>>
Login
Supplier
<<include>>
<<Service>>
MakePurchase
InvoiceOrder
Bank

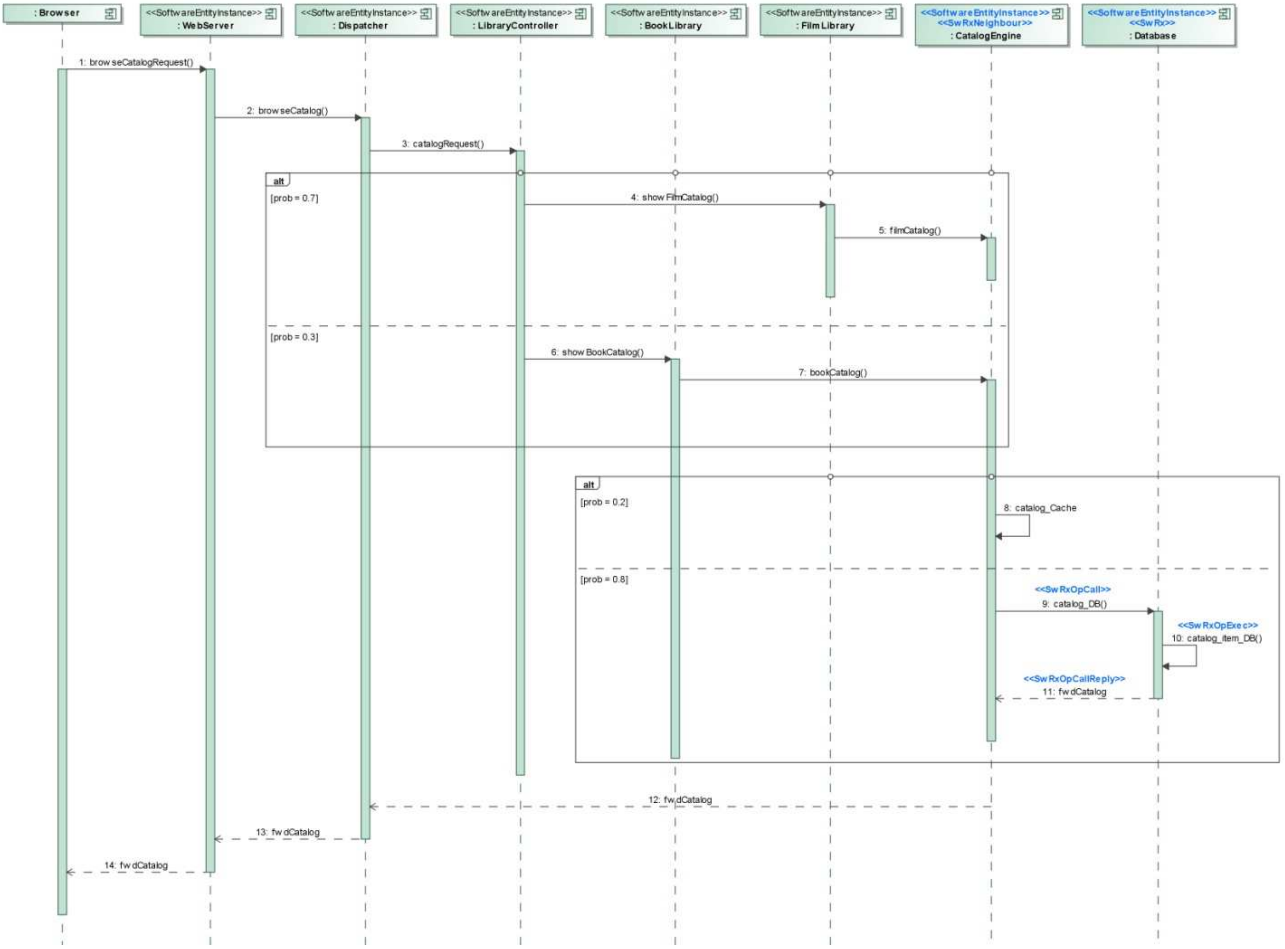

: Browser
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
: LibraryController
<<SoftwareEntityInstance>>
: BookLibrary
<<SoftwareEntityInstance>>
: FilmLibrary
<<SoftwareEntityInstance>>
: CatalogEngine
<<SoftwareEntityInstance>>
<<SwRx>>
: Database
<<SoftwareEntityInstance>>
<<FirstFilter>>
: Filter1
<<SoftwareEntityInstance>>
<<Filter>>
: Filter2
<<SoftwareEntityInstance>>
<<LastFilter>>
: Filter3
1: browseCatalogRequest()
2: browseCatalog()
3: catalogRequest()
alt
[prob = 0.7]
4: showFilmCatalog()
5: filmCatalog()
[prob = 0.3]
6: showBookCatalog()
7: bookCatalog()
alt
[prob = 0.2]
8: catalog_Cache
[prob = 0.8]
<<SwRxOperationCall>>
9: catalog_DB()
<<FilterOperationCall>>
10: catalog_Item_DB1()
<<FilterOperationExec>>
11: catalog_Item_DB_Part1()
<<FilterOperationCall>>
12: catalog_Item_DB2()
<<FilterOperationExec>>
13: catalog_Item_DB_Part2()
<<FilterOperationCall>>
14: catalog_Item_DB3()
<<FilterOperationExec>>
15: catalog_Item_DB_Part3()
<<FilterOperationCallReply>>
16: fwdCatalog
<<FilterOperationCallReply>>
17: fwdCatalog
<<FilterOperationCallReply>>
18: fwdCatalog
<<SwRxOperationCallReply>>
19: fwdCatalog
20: fwdCatalog
21: fwdCatalog
22: fwdCatalog

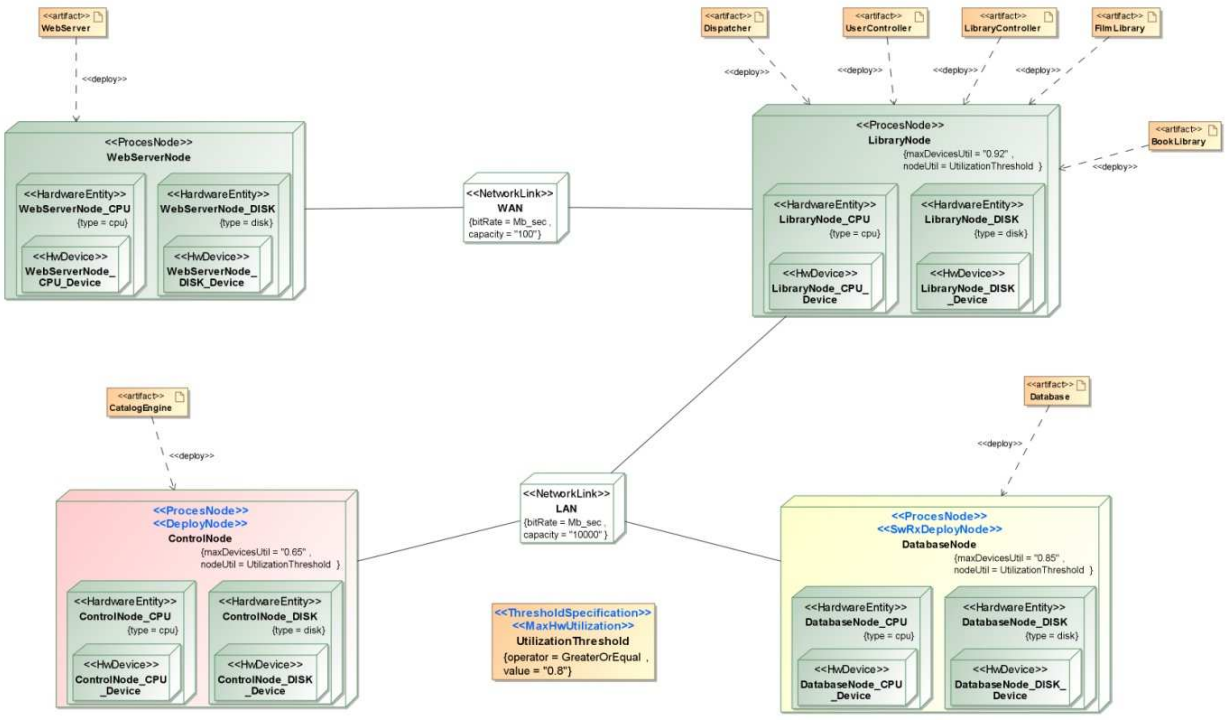

<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode_
CPU_Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<artifact>>
BookLibrary
<<ProcesNode>>
LibraryNode
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU
_Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK
_Device
<<artifact>>
CatalogEngine
<<ProcesNode>>
ControlNode
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU_
Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK_
Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000"}
<<artifact>>
Filter1
<<artifact>>
Filter2
<<artifact>>
Filter3
<<ProcesNode>>
<<SwRxDeployNode>>
DatabaseNode
<<artifact>>
Database
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU
_Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device

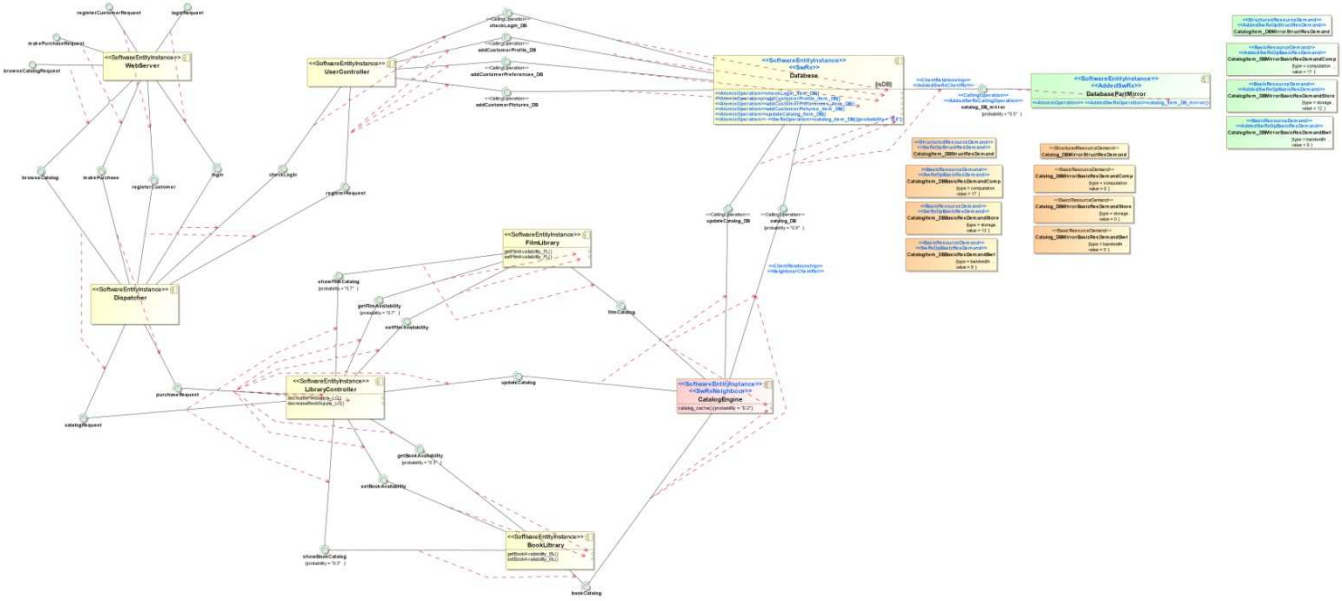
($SRM_2^{P\&F}$, $TRM_2^{P\&F}$)

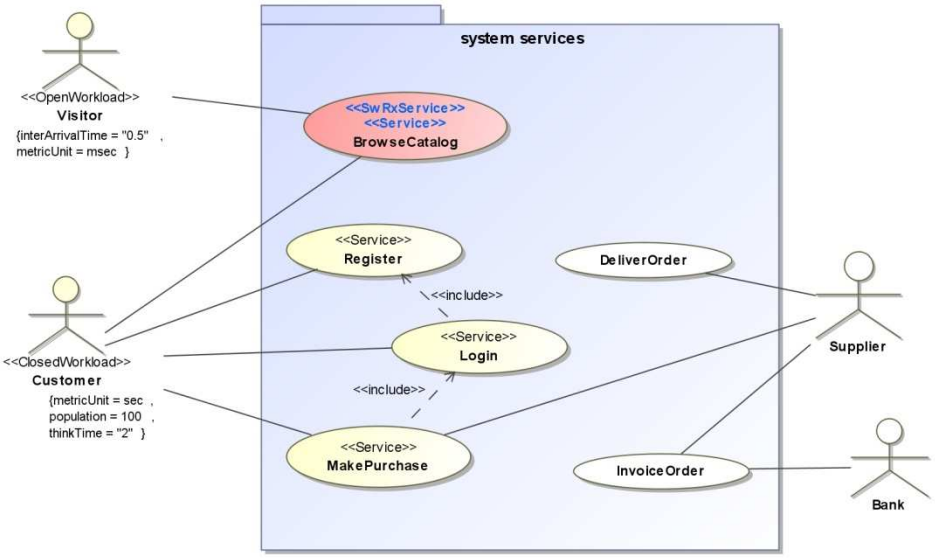


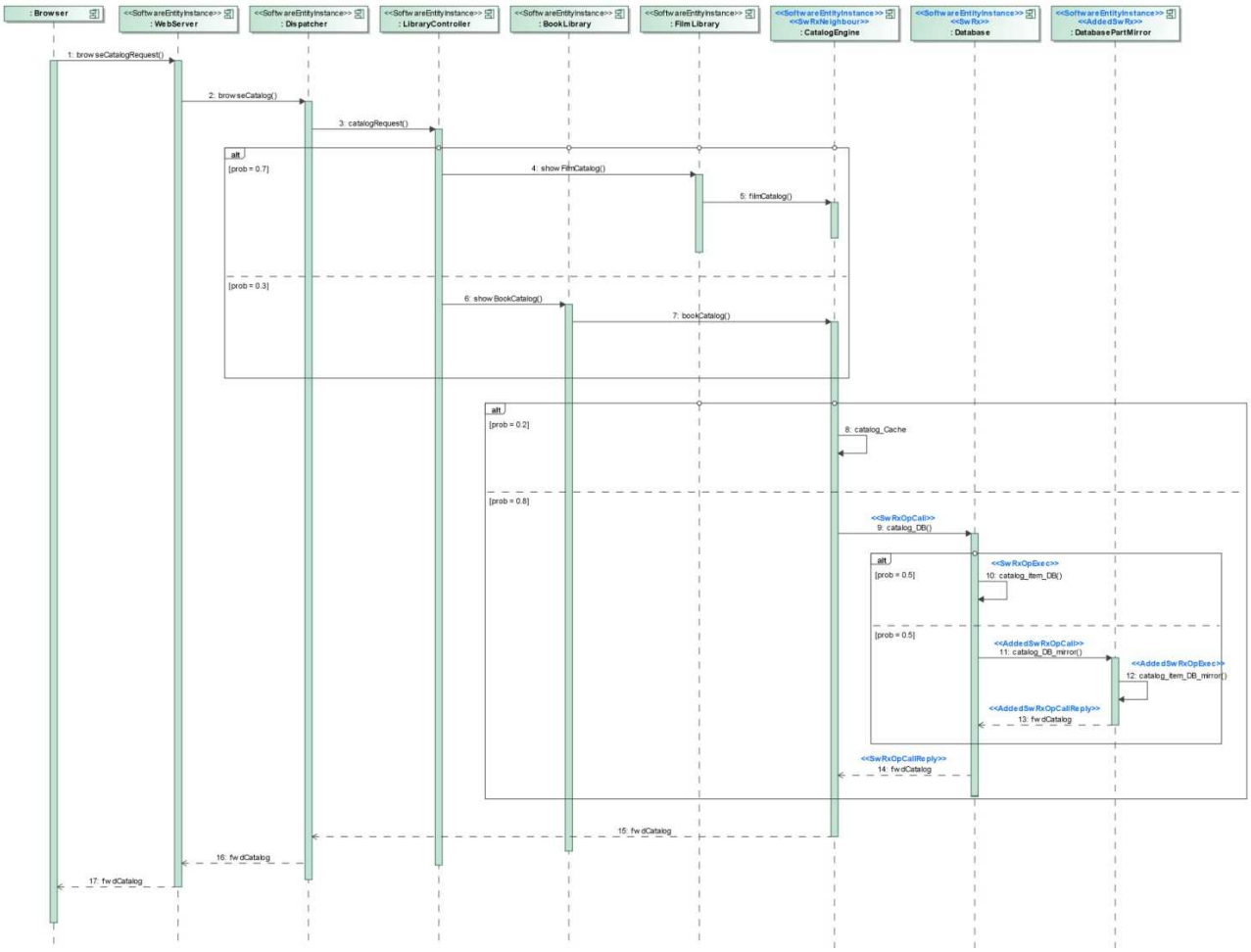


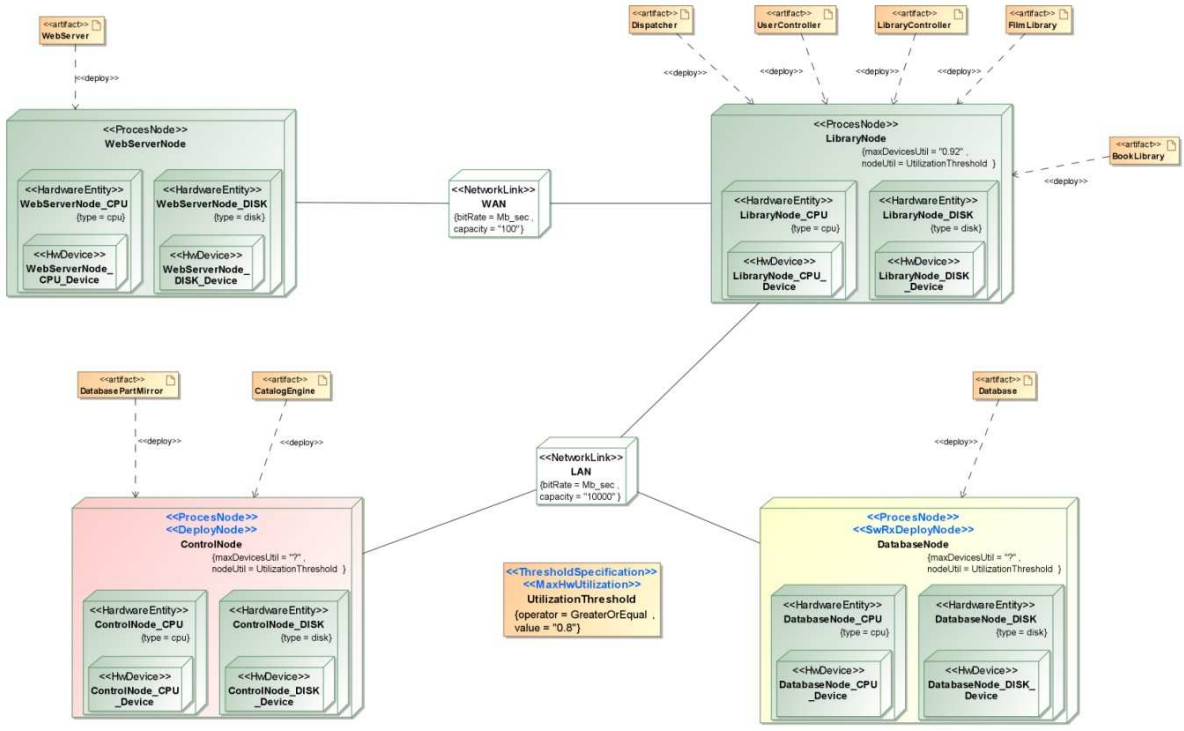

: Browser
<<SoftwareEntityInstance>>
: WebServer
<<SoftwareEntityInstance>>
: Dispatcher
<<SoftwareEntityInstance>>
: LibraryController
<<SoftwareEntityInstance>>
: BookLibrary
<<SoftwareEntityInstance>>
: FilmLibrary
<<SoftwareEntityInstance>>
<<SwRxNeighbour>>
: CatalogEngine
<<SoftwareEntityInstance>>
<<SwRx>>
: Database
1: browseCatalogRequest()
2: browseCatalog()
3: catalogRequest()
alt
[prob = 0.7]
4: showFilmCatalog()
5: filmCatalog()
[prob = 0.3]
6: showBookCatalog()
7: bookCatalog()
alt
[prob = 0.2]
8: catalog_Cache
[prob = 0.8]
<<SwRxOpCall>>
9: catalog_DB()
<<SwRxOpExec>>
10: catalog_item_DB()
<<SwRxOpCallReply>>
11: fwdCatalog
12: fwdCatalog
13: fwdCatalog
14: fwdCatalog


<<artifact>>
WebServer
<<deploy>>
<<ProcesNode>>
WebServerNode
<<HardwareEntity>>
WebServerNode_CPU
{type = cpu}
<<HwDevice>>
WebServerNode_
CPU_Device
<<HardwareEntity>>
WebServerNode_DISK
{type = disk}
<<HwDevice>>
WebServerNode_
DISK_Device
<<NetworkLink>>
WAN
{bitRate = Mb_sec ,
capacity = "100"}
<<artifact>>
Dispatcher
<<artifact>>
UserController
<<artifact>>
LibraryController
<<artifact>>
FilmLibrary
<<ProcesNode>>
LibraryNode
{maxDevicesUtil = "0.92" ,
nodeUtil = UtilizationThreshold }
<<artifact>>
BookLibrary
<<HardwareEntity>>
LibraryNode_CPU
{type = cpu}
<<HwDevice>>
LibraryNode_CPU_
Device
<<HardwareEntity>>
LibraryNode_DISK
{type = disk}
<<HwDevice>>
LibraryNode_DISK
_Device
<<artifact>>
CatalogEngine
<<ProcesNode>>
<<DeployNode>>
ControlNode
{maxDevicesUtil = "0.65" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
ControlNode_CPU
{type = cpu}
<<HwDevice>>
ControlNode_CPU
_Device
<<HardwareEntity>>
ControlNode_DISK
{type = disk}
<<HwDevice>>
ControlNode_DISK
_Device
<<NetworkLink>>
LAN
{bitRate = Mb_sec ,
capacity = "10000" }
<<artifact>>
Database
<<ProcesNode>>
<<SwRxDeployNode>>
DatabaseNode
{maxDevicesUtil = "0.85" ,
nodeUtil = UtilizationThreshold }
<<HardwareEntity>>
DatabaseNode_CPU
{type = cpu}
<<HwDevice>>
DatabaseNode_CPU
_Device
<<HardwareEntity>>
DatabaseNode_DISK
{type = disk}
<<HwDevice>>
DatabaseNode_DISK_
Device
<<ThresholdSpecification>>
<<MaxHwUtilization>>
UtilizationThreshold
{operator = GreaterOrEqual ,
value = "0.8"}


WebServer
UserController
Database
Database(PartMirror
Dispatcher
FilmLibrary
LibraryController
CatalogEngine
BookLibrary


system services
<<OpenWorkload>>
Visitor
{interArrivalTime = "0.5" ,
metricUnit = msec }
<<SwRxService>>
<<Service>>
BrowseCatalog
<<Service>>
Register
DeliverOrder
<<include>>
<<ClosedWorkload>>
Customer
{metricUnit = sec ,
population = 100 ,
thinkTime = "2" }
<<Service>>
Login
<<include>>
<<Service>>
MakePurchase
InvoiceOrder
Supplier
Bank


: Browser
<<SoftwareEntityInstance>> : WebServer
<<SoftwareEntityInstance>> : Dispatcher
<<SoftwareEntityInstance>> : LibraryController
<<SoftwareEntityInstance>> : BookLibrary
<<SoftwareEntityInstance>> : FilmLibrary
<<SoftwareEntityInstance>> <<SwRxNeighbour>> : CatalogEngine
<<SoftwareEntityInstance>> <<SwRx>> : Database
<<SoftwareEntityInstance>> <<AddedSwRx>> : DatabasePartMirror
1: browseCatalogRequest()
2: browseCatalog()
3: catalogRequest()
alt
[prob = 0.7]
4: showFilmCatalog()
5: filmCatalog()
[prob = 0.3]
6: showBookCatalog()
7: bookCatalog()
alt
[prob = 0.2]
8: catalog_Cache
[prob = 0.8]
<<SwRxOpCall>>
9: catalog_DB()
alt
[prob = 0.5]
<<SwRxOpExec>>
10: catalog_item_DB()
[prob = 0.5]
<<AddedSwRxOpCall>>
11: catalog_DB_mirror()
<<AddedSwRxOpExec>>
12: catalog_item_DB_mirror()
<<AddedSwRxOpCallReply>>
13: fwdCatalog
<<SwRxOpCallReply>>
14: fwdCatalog
15: fwdCatalog
16: fwdCatalog
17: fwdCatalog


<<artifact>> WebServer
<<deploy>>
<<ProcesNode>> WebServerNode
<<HardwareEntity>> WebServerNode_CPU {type = cpu}
<<HwDevice>> WebServerNode_CPU_Device
<<HardwareEntity>> WebServerNode_DISK {type = disk}
<<HwDevice>> WebServerNode_DISK_Device
<<NetworkLink>> WAN {bitRate = Mb_sec , capacity = "100"}
<<artifact>> Dispatcher
<<artifact>> UserController
<<artifact>> LibraryController
<<artifact>> FilmLibrary
<<ProcesNode>> LibraryNode {maxDevicesUtil = "0.92" , nodeUtil = UtilizationThreshold }
<<HardwareEntity>> LibraryNode_CPU {type = cpu}
<<HwDevice>> LibraryNode_CPU_Device
<<HardwareEntity>> LibraryNode_DISK {type = disk}
<<HwDevice>> LibraryNode_DISK_Device
<<artifact>> BookLibrary
<<artifact>> DatabasePartMirror
<<artifact>> CatalogEngine
<<artifact>> Database
<<NetworkLink>> LAN {bitRate = Mb_sec , capacity = "10000" }
<<ProcesNode>> <<DeployNode>> ControlNode {maxDevicesUtil = "?" , nodeUtil = UtilizationThreshold }
<<HardwareEntity>> ControlNode_CPU {type = cpu}
<<HwDevice>> ControlNode_CPU_Device
<<HardwareEntity>> ControlNode_DISK {type = disk}
<<HwDevice>> ControlNode_DISK_Device
<<ThresholdSpecification>> <<MaxHwUtilization>> UtilizationThreshold {operator = GreaterOrEqual , value = "0.8"}
<<ProcesNode>> <<SwRxDeployNode>> DatabaseNode {maxDevicesUtil = "?" , nodeUtil = UtilizationThreshold }
<<HardwareEntity>> DatabaseNode_CPU {type = cpu}
<<HwDevice>> DatabaseNode_CPU_Device
<<HardwareEntity>> DatabaseNode_DISK {type = disk}
<<HwDevice>> DatabaseNode_DISK_Device

[Smith-Williams-2003] Smith, C. U., and Williams, L. G. “*More New Software Antipatterns: Even MoreWays to Shoot Yourself in the Foot*”. In International Computer Measurement Group Conference (2003), pp. 717–725.

[Arcelli-2011] Davide Arcelli, Master Thesis [in italian] “Refactoring di modelli software orientato alla soluzione di performance antipatterns”, 2011, available at http://www.di.univaq.it/cortelle/docs/Arcelli-MasterThesis.pdf