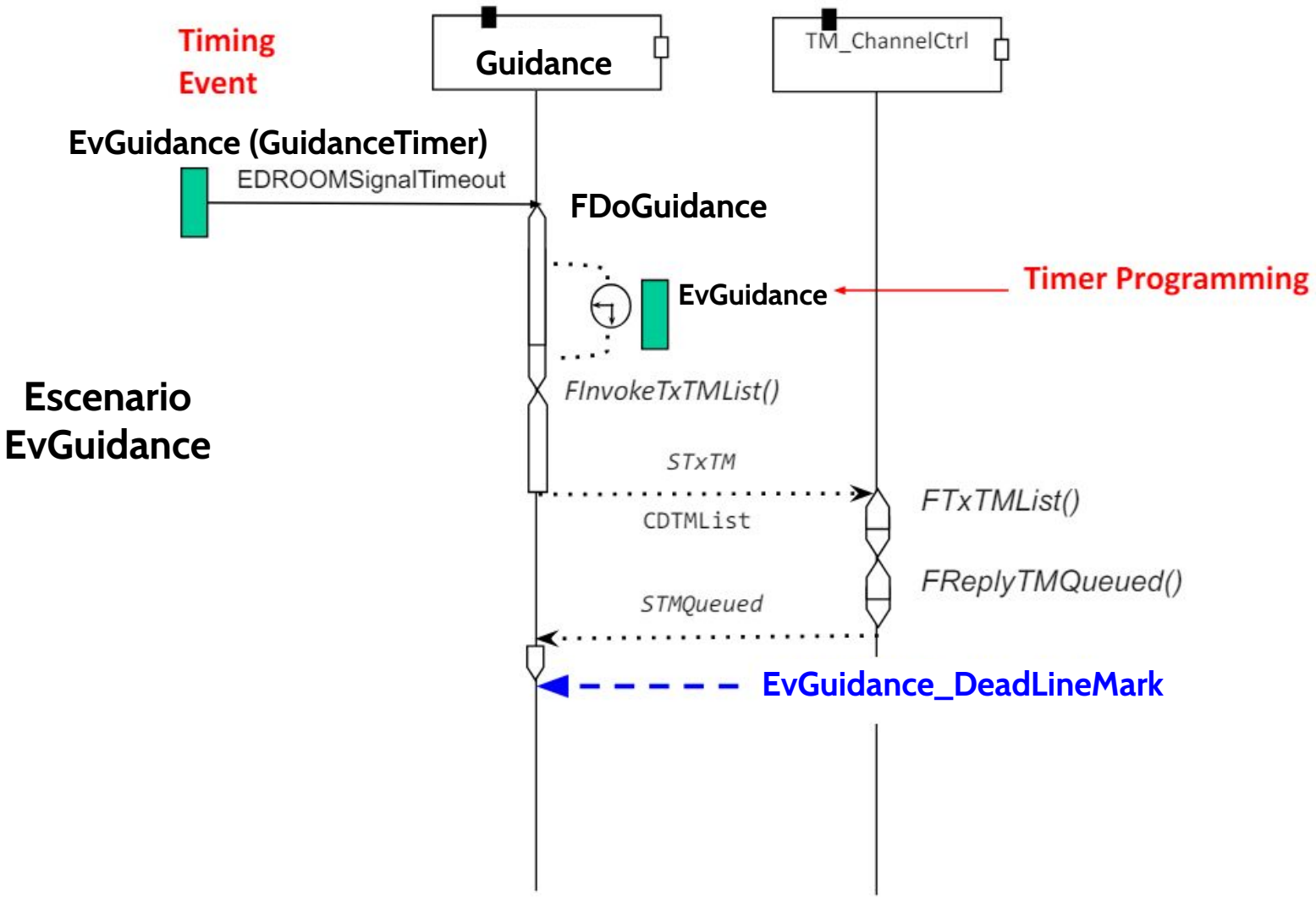
EDROOM UAH EXPLORER

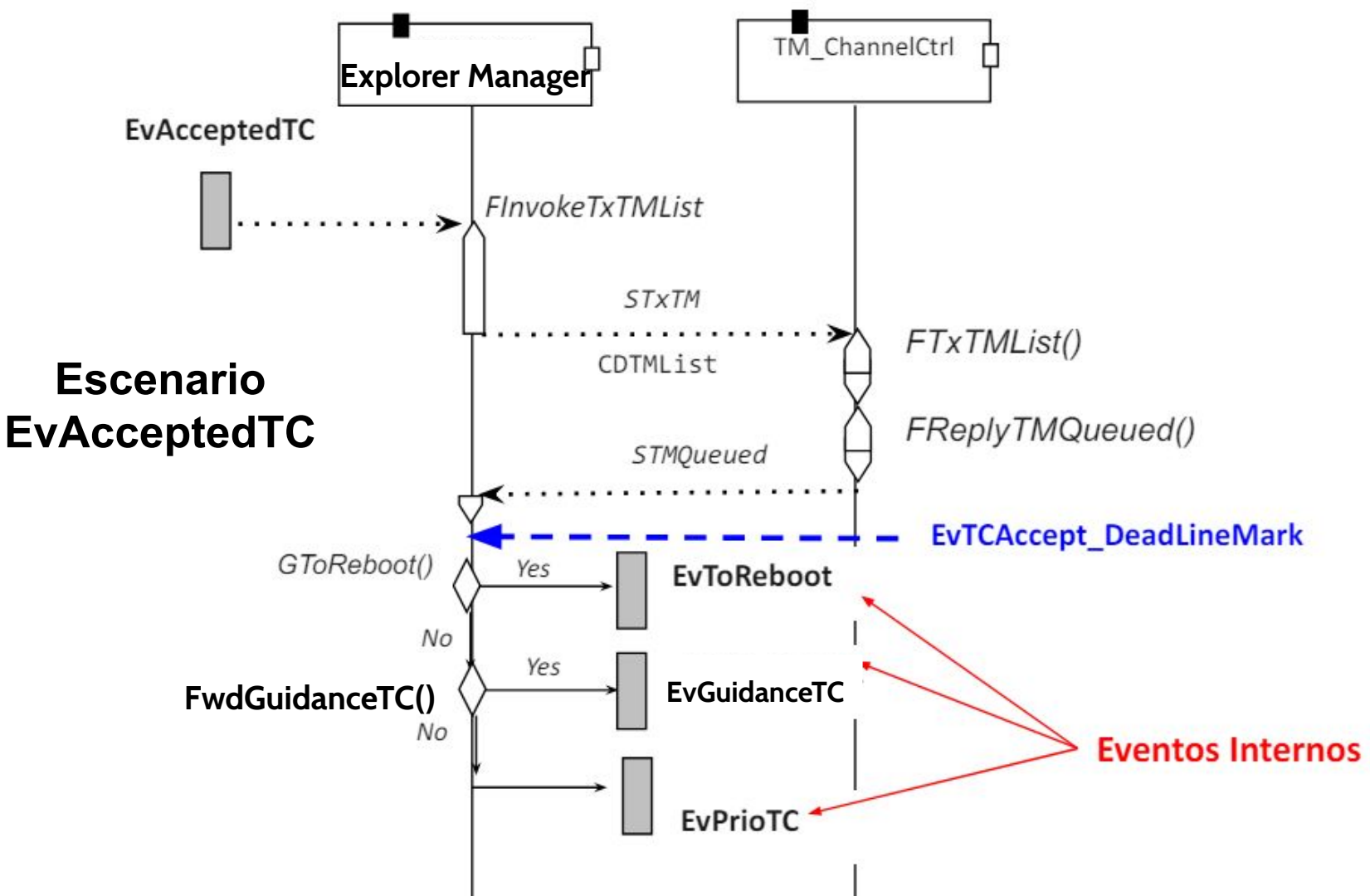
Eduardo Jiménez Cerezuela

Índice

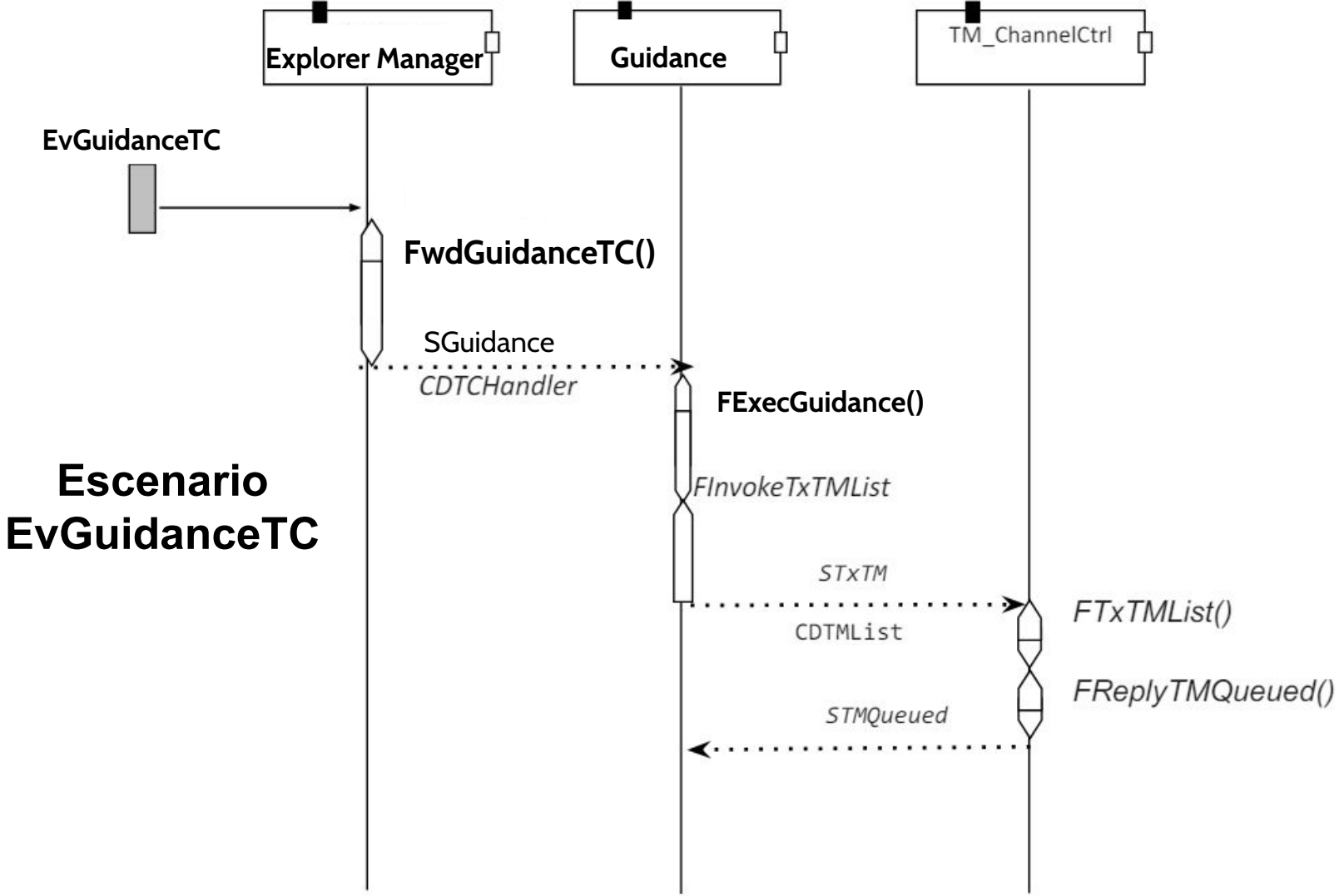
1. Escenarios a añadir al Modelo EDROOM
2. Definición de la clase Procolo a añadir al Modelo EDROOM
3. Diseño de la interfaz de la clase componente CCGuidance
4. Diseño del comportamiento de la clase componente CCGuidance

1. Escenarios a añadir al Modelo EDROOM


Timing
Event
Guidance
TM_ChannelCtrl
EvGuidance (GuidanceTimer)
EDROOMSignalTimeout
FDoGuidance
EvGuidance
Timer Programming
Escenario
EvGuidance
FInvokeTxTMList()
STxTM
CDTMList
FTxTMList()
FReplyTMQueued()
STMQueued
EvGuidance_DeadLineMark



Escenario EvAcceptedTC

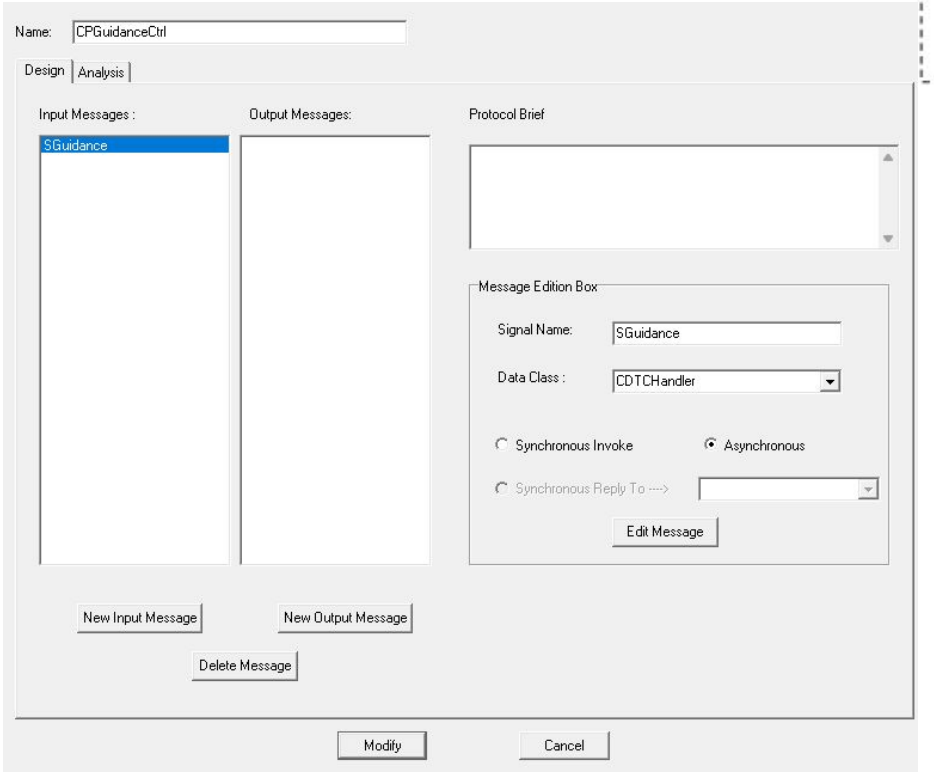

Explorer Manager
Guidance
TM_ChannelCtrl
EvGuidanceTC
FwdGuidanceTC()
SGuidance
CDTCHandler
FExecGuidance()
FInvokeTxTMList
STxTM
CDTMList
FTxTMList()
FReplyTMQueued()
STMQueued
Escenario
EvGuidanceTC

2. Definición de la clase Procolo a añadir al Modelo EDROOM

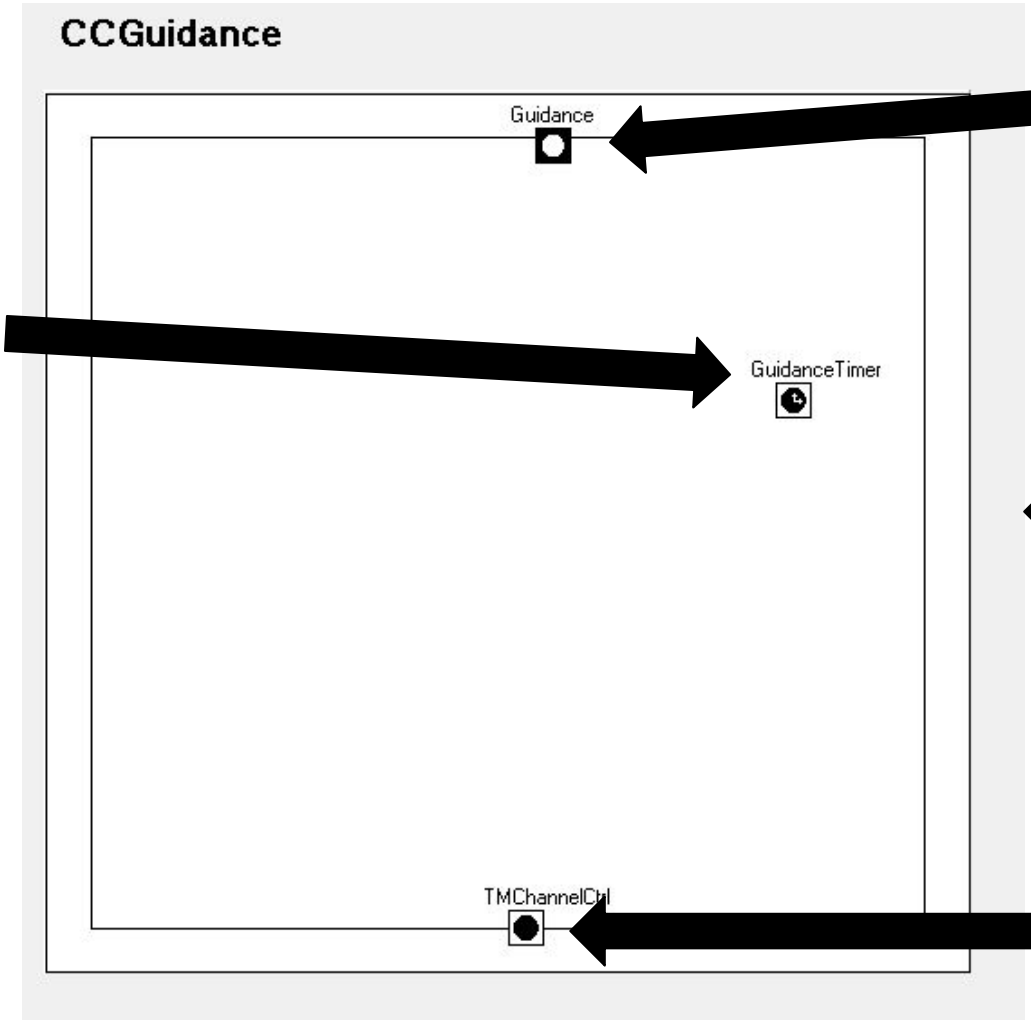
CPGuidanceCtrl

Msg

In/Out	Signal	Data
In	SGuidance	*CDTCHandler*

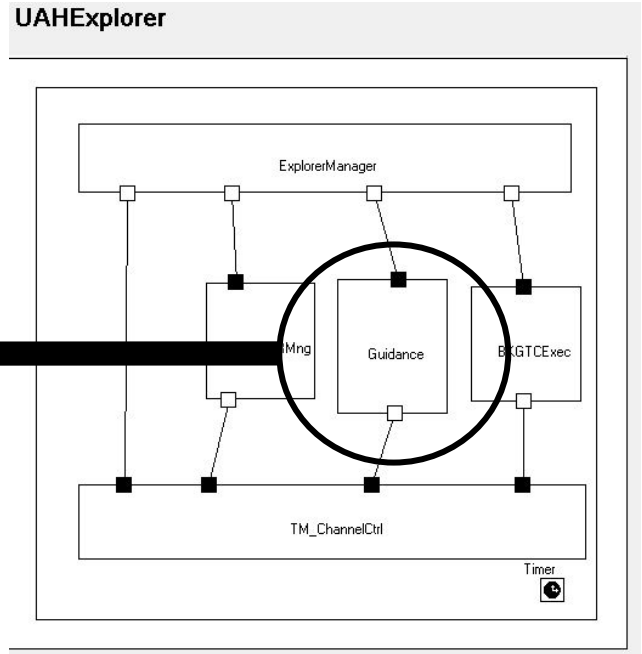


3. Diseño de la interfaz de la clase componente CCGuidance



Puerto nominal

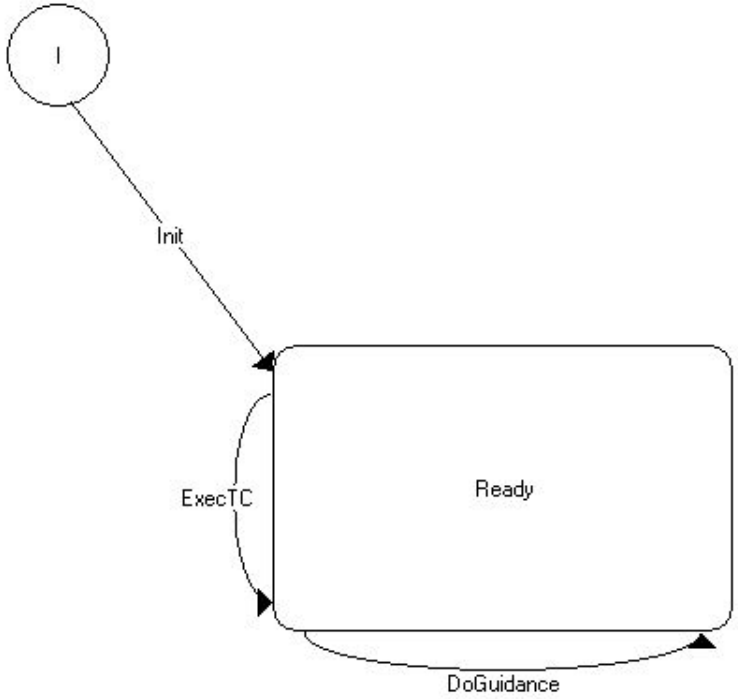
Puerto Timer Interno



Puerto conjugado

4. Diseño del comportamiento de la clase componente CCGuidance

Máquina de estados para CCGuidance

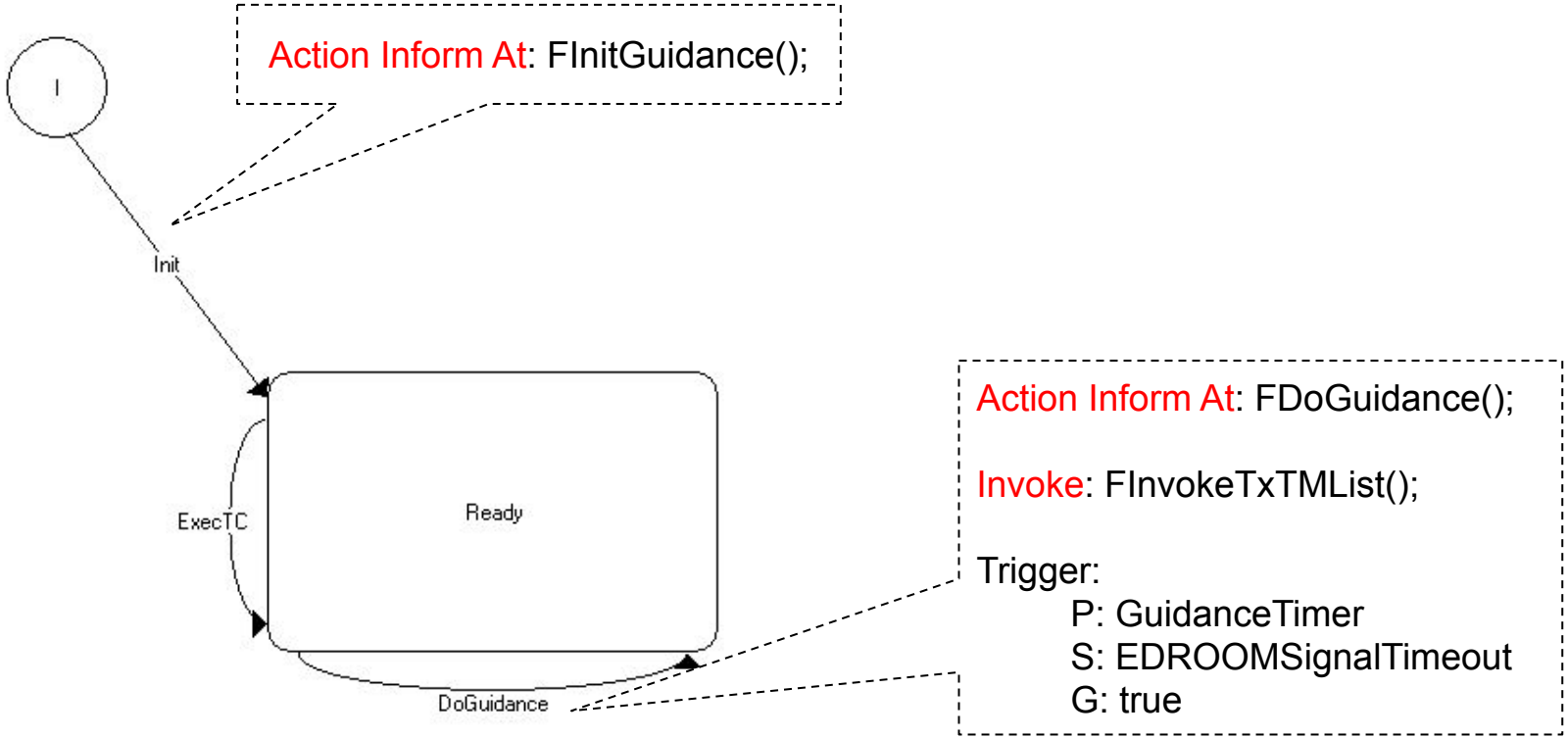


Vars:

Pr_time VNextTimeout;

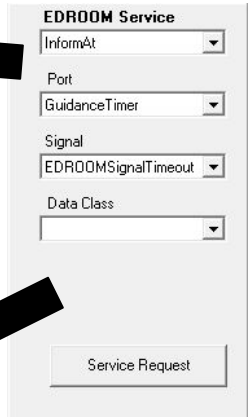
CDTMList VCurrentTMList;

Máquina de estados para CCGuidance

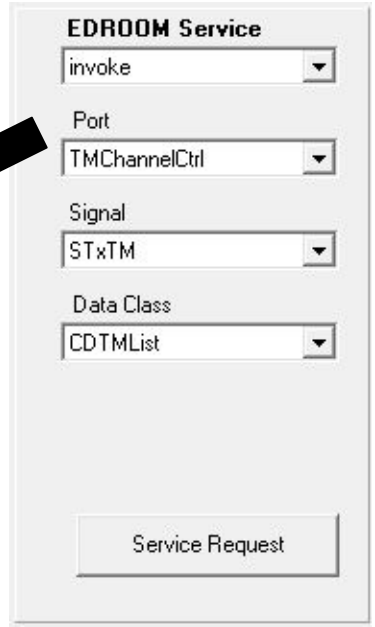


Funciones para la máquina de estados en CCGuidance

```
FInitGuidance(){
Pr_Time time;
time.GetTime(); // Get current monotonic time
time+=Pr_Time(1,0); // Add X sec + Y microsec
VNextTimeout=time;
GuidanceTimer.InformAt(time);
}
```




```
FDoGuidance(){
Pr_Time time;
VNextTimeout+= Pr_Time(1,0); // Add X sec + Y microsec
time=VNextTimeout;
PUSService129::GuidanceControl();
GuidanceTimer.InformAt(time);
}
```



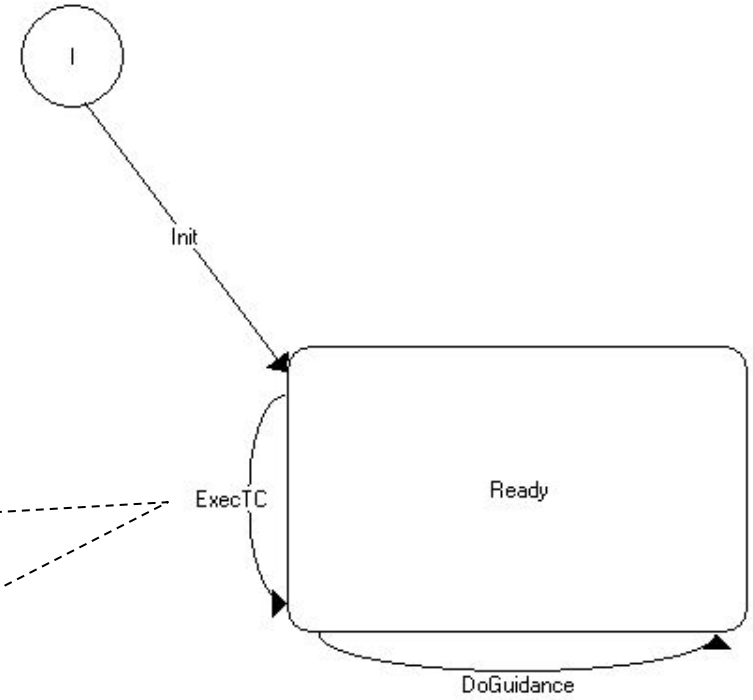

```
void FInvokeTxTMList(){
CDTMList * pSTxTM_Data = EDROOMPoolCDTMList.AllocData();
*pSTxTM_Data=VCurrentTMList;
VCurrentTMList.Clear();
MsgBack=TMChannelCtrl.invoke(STxTM,pSTxTM_Data, &EDROOMPoolCDTMList);
}
```

Máquina de estados para CCGuidance

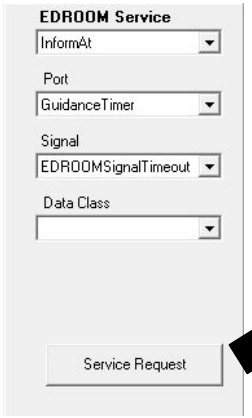
Msg-> Data H: FExecGuidance();

Invoke: FInvokeTxTMList();

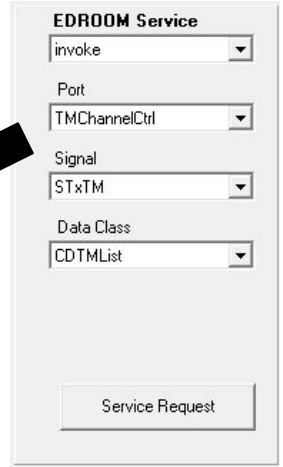
Trigger:
P: Guidance
S: SGuidance
G: true



Funciones para la máquina de estados en CCGuidance




```
FExecGuidance(){
CDTCHandler & varSGuidance = *(CDTCHandler *) Msg->data;
CDEventList TCExecEventList;
PUS_GuidanceTCExecutor::ExecTC(varSGuidance,VCurrentTMList,TCExecEventList);
}
```




```
void FInvokeTxTMList(){
CDTMList * pSTxTM_Data = EDROOMPoolCDTMList.AllocData();
*pSTxTM_Data=VCurrentTMList;
VCurrentTMList.Clear();
MsgBack=TMChannelCtrl.invoke(STxTM,pSTxTM_Data, &EDROOMPoolCDTMList);
}
```

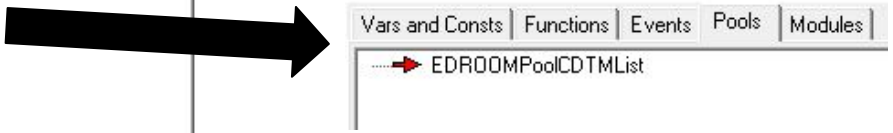
Máquina de estados para CCGuidance

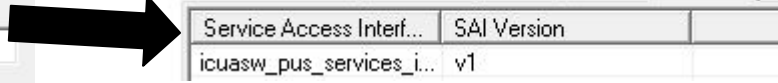
POOL

Name: EDROOMPoolCDTMList:

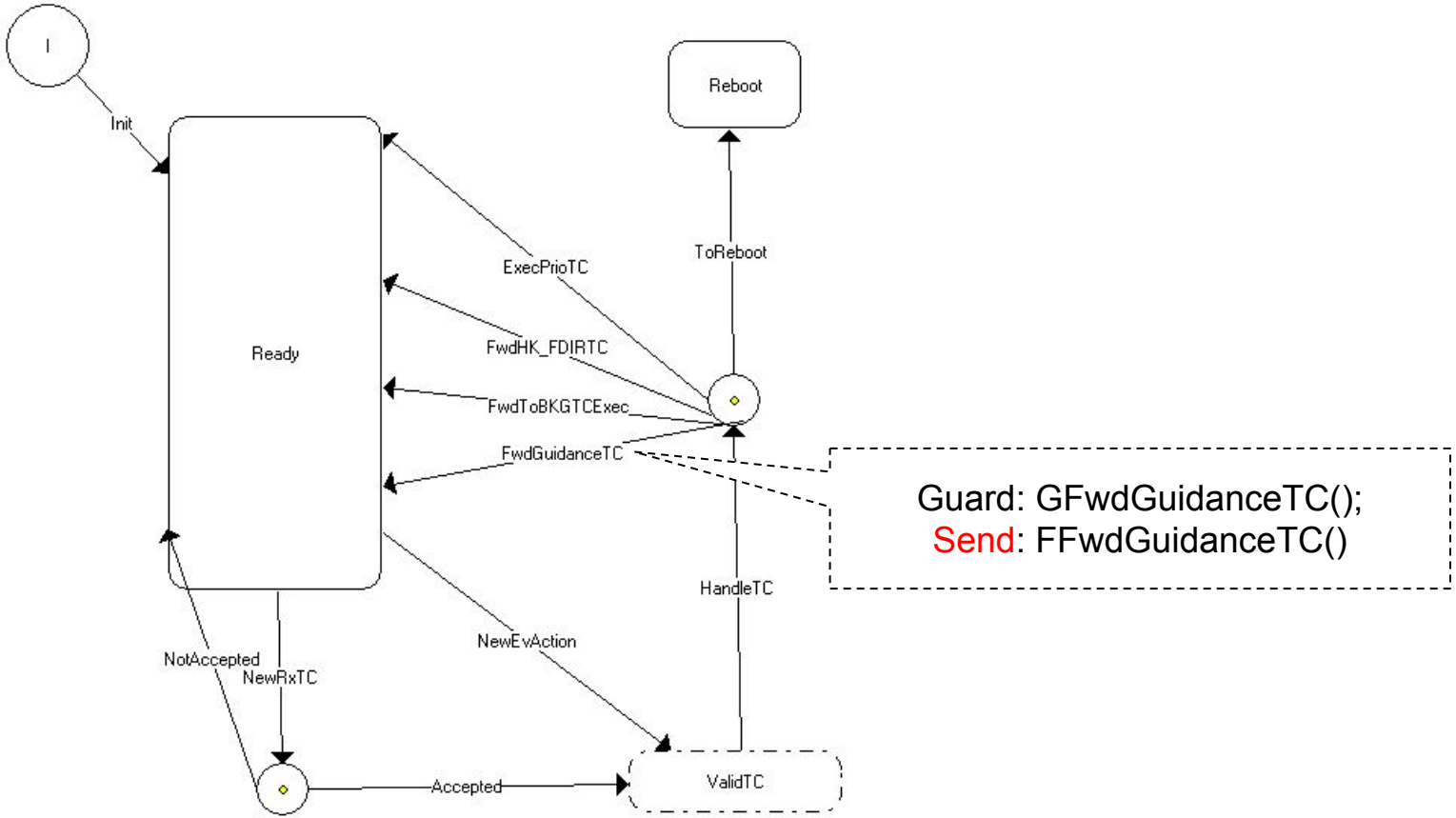
DataClass: CDTMList

Element Number=10



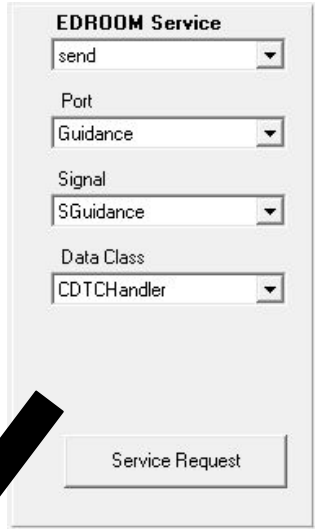


Máquina de estados para CCExplorerManager



Funciones para la máquina de estados en CCExplorerManager

```
bool GFwdGuidanceTC(){
return VCurrentTC.IsGuidanceTC();
}
```




```
void FFwdGuidanceTC(){
CDTCHandler * pSGuidance_Data = EDROOMPoolCDTCHandler.AllocData();
*pSGuidance_Data=VCurrentTC;
Guidance.send(SGuidance, pSGuidance_Data, &EDROOMPoolCDTCHandler);
}
```