aircraft

Tight Rope v0.6

1st December 2015

1 ID Files

1.1 MissionIds

section $MissionIds$ **parents** $scj\_prelude, MissionId$

$$
\begin{array}{|l}
MainMissionID : MissionID \\
TakeOffMissionID : MissionID \\
CruiseMissionID : MissionID \\
LandMissionID : MissionID \\
\hline
distinct\langle nullMissionId, MainMissionID, TakeOffMissionID, \\
CruiseMissionID, LandMissionID\rangle
\end{array}
$$

1.2 SchedulablesIds

section $\mathit{SchedulableIds}$ **parents** $\mathit{scj\_prelude}, \mathit{SchedulableId}$

$$
\begin{array}{|l}
\mathit{MainMissionSequencerID} : \mathit{SchedulableID} \\
\mathit{ACModeChangerID} : \mathit{SchedulableID} \\
\mathit{EnvironmentMonitorID} : \mathit{SchedulableID} \\
\mathit{ControlHandlerID} : \mathit{SchedulableID} \\
\mathit{FlightSensorsMonitorID} : \mathit{SchedulableID} \\
\mathit{CommunicationsHandlerID} : \mathit{SchedulableID} \\
\mathit{AperiodicSimulatorID} : \mathit{SchedulableID} \\
\mathit{LandingGearHandlerTakeOffID} : \mathit{SchedulableID} \\
\mathit{TakeOffMonitorID} : \mathit{SchedulableID} \\
\mathit{TakeOffFailureHandlerID} : \mathit{SchedulableID} \\
\mathit{BeginLandingHandlerID} : \mathit{SchedulableID} \\
\mathit{NavigationMonitorID} : \mathit{SchedulableID} \\
\mathit{GroundDistanceMonitorID} : \mathit{SchedulableID} \\
\mathit{LandingGearHandlerLandID} : \mathit{SchedulableID} \\
\mathit{InstrumentLandingSystemMonitorID} : \mathit{SchedulableID} \\
\mathit{SafeLandingHandlerID} : \mathit{SchedulableID} \\
\hline
\mathit{distinct}\langle \mathit{nullSequencerId}, \mathit{nullSchedulableId}, \mathit{MainMissionSequencerID}, \\
\mathit{ACModeChangerID}, \mathit{EnvironmentMonitorID}, \\
\mathit{ControlHandlerID}, \mathit{FlightSensorsMonitorID}, \\
\mathit{CommunicationsHandlerID}, \mathit{AperiodicSimulatorID}, \\
\mathit{LandingGearHandlerTakeOffID}, \mathit{TakeOffMonitorID}, \\
\mathit{TakeOffFailureHandlerID}, \mathit{BeginLandingHandlerID}, \\
\mathit{NavigationMonitorID}, \mathit{GroundDistanceMonitorID}, \\
\mathit{LandingGearHandlerLandID}, \mathit{InstrumentLandingSystemMonitorID}, \\
\mathit{SafeLandingHandlerID}\rangle
\end{array}
$$

1.3 ThreadIds

section *ThreadIds* **parents** *scj_prelude*, *GlobalTypes*

$SafeLandingHandlerThreadID : ThreadID$
$ACModeChangerThreadID : ThreadID$
$TakeOffFailureHandlerThreadID : ThreadID$
$InstrumentLandingSystemMonitorThreadID : ThreadID$
$FlightSensorsMonitorThreadID : ThreadID$
$TakeOffMonitorThreadID : ThreadID$
$AperiodicSimulatorThreadID : ThreadID$
$LandingGearHandlerLandThreadID : ThreadID$
$LandingGearHandlerTakeOffThreadID : ThreadID$
$GroundDistanceMonitorThreadID : ThreadID$
$ControlHandlerThreadID : ThreadID$
$CommunicationsHandlerThreadID : ThreadID$
$BeginLandingHandlerThreadID : ThreadID$
$NavigationMonitorThreadID : ThreadID$
$EnvironmentMonitorThreadID : ThreadID$

$distinct\langle SafeletThreadId, nullThreadId,$
$SafeLandingHandlerThreadID, ACModeChangerThreadID,$
$TakeOffFailureHandlerThreadID, InstrumentLandingSystemMonitorThreadID,$
$FlightSensorsMonitorThreadID, TakeOffMonitorThreadID,$
$AperiodicSimulatorThreadID, LandingGearHandlerLandThreadID,$
$LandingGearHandlerTakeOffThreadID, GroundDistanceMonitorThreadID,$
$ControlHandlerThreadID, CommunicationsHandlerThreadID,$
$BeginLandingHandlerThreadID, NavigationMonitorThreadID,$
$EnvironmentMonitorThreadID\rangle$

1.4 ObjectIds

section *ObjectIds* **parents** *scj_prelude*, *GlobalTypes*

$$
\begin{array}{|l}
ACSafeletObjectID : ObjectID \\
MainMissionObjectID : ObjectID \\
ACModeChangerObjectID : ObjectID \\
EnvironmentMonitorObjectID : ObjectID \\
ControlHandlerObjectID : ObjectID \\
FlightSensorsMonitorObjectID : ObjectID \\
CommunicationsHandlerObjectID : ObjectID \\
AperiodicSimulatorObjectID : ObjectID \\
TakeOffMissionObjectID : ObjectID \\
LandingGearHandlerTakeOffObjectID : ObjectID \\
TakeOffMonitorObjectID : ObjectID \\
TakeOffFailureHandlerObjectID : ObjectID \\
CruiseMissionObjectID : ObjectID \\
BeginLandingHandlerObjectID : ObjectID \\
NavigationMonitorObjectID : ObjectID \\
LandMissionObjectID : ObjectID \\
GroundDistanceMonitorObjectID : ObjectID \\
LandingGearHandlerLandObjectID : ObjectID \\
InstrumentLandingSystemMonitorObjectID : ObjectID \\
SafeLandingHandlerObjectID : ObjectID \\
\hline
distinct \langle ACSafeletObjectID, MainMissionObjectID, \\
ACModeChangerObjectID, EnvironmentMonitorObjectID, \\
ControlHandlerObjectID, FlightSensorsMonitorObjectID, \\
CommunicationsHandlerObjectID, AperiodicSimulatorObjectID, \\
TakeOffMissionObjectID, LandingGearHandlerTakeOffObjectID, \\
TakeOffMonitorObjectID, TakeOffFailureHandlerObjectID, \\
CruiseMissionObjectID, BeginLandingHandlerObjectID, \\
NavigationMonitorObjectID, LandMissionObjectID, \\
GroundDistanceMonitorObjectID, LandingGearHandlerLandObjectID, \\
InstrumentLandingSystemMonitorObjectID, SafeLandingHandlerObjectID \rangle
\end{array}
$$

2 Network

section $\mathit{NetworkChannels}$ **parents** $\mathit{scj\_prelude}, \mathit{MissionId}, \mathit{MissionIds},$
$\mathit{SchedulableId}, \mathit{SchedulableIds}, \mathit{MissionChan}, \mathit{SchedulableChan}, \mathit{TopLevelMissionSequencerFWChan},$
$\mathit{FrameworkChan}, \mathit{SafeletChan}$

channelset $\mathit{TerminateSync} ==$
$\{\!| \mathit{schedulables\_terminated}, \mathit{schedulables\_stopped}, \mathit{get\_activeSchedulables} |\!\}$

channelset $\mathit{ControlTierSync} ==$
$\{\!| \mathit{start\_toplevel\_sequencer}, \mathit{done\_toplevel\_sequencer}, \mathit{done\_safeletFW} |\!\}$

channelset $\mathit{TierSync} ==$
$\{\!| \mathit{start\_mission}\,.\,, \mathit{done\_mission}\,.\,,$
$\mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer} |\!\}$

channelset $\mathit{MissionSync} ==$
$\{\!| \mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer}, \mathit{register},$
$\mathit{signalTerminationCall}, \mathit{signalTerminationRet}, \mathit{activate\_schedulables}, \mathit{done\_schedulable},$
$\mathit{cleanupSchedulableCall}, \mathit{cleanupSchedulableRet} |\!\}$

channelset $\mathit{SchedulablesSync} ==$
$\{\!| \mathit{activate\_schedulables}, \mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer} |\!\}$

channelset $\mathit{ClusterSync} ==$
$\{\!| \mathit{done\_toplevel\_sequencer}, \mathit{done\_safeletFW} |\!\}$

channelset $\mathit{AppSync} ==$
$\bigcup\{\mathit{SafeltAppSync}, \mathit{MissionSequencerAppSync}, \mathit{MissionAppSync},$
$\mathit{MTAppSync}, \mathit{OSEHSync}, \mathit{APEHSync},$
$\{\!| \mathit{getSequencer}, \mathit{end\_mission\_app}, \mathit{end\_managedThread\_app},$
$\mathit{setCeilingPriority}, \mathit{requestTerminationCall}, \mathit{requestTerminationRet}, \mathit{terminationPendingCall},$
$\mathit{terminationPendingRet}, \mathit{handleAsyncEventCall}, \mathit{handleAsyncEventRet} |\!\}\}$

channelset $\mathit{ObjectSync} ==$
$\{\!| \; |\!\}$

channelset $\mathit{ThreadSync} ==$
$\{\!| \; |\!\}$

channelset $\mathit{LockingSync} ==$
$\{\!| \mathit{lockAcquired}, \mathit{startSyncMeth}, \mathit{endSyncMeth}, \mathit{waitCall}, \mathit{waitRet}, \mathit{notify} |\!\}$

channelset $\mathit{Tier0Sync} ==$
$\{\!| \mathit{done\_toplevel\_sequencer}, \mathit{done\_safeletFW},$
$\mathit{start\_mission}\,.\,, \mathit{done\_mission}\,.\,,$
$\mathit{initializeRet}\,.\,, \mathit{requestTermination}\,.\,.\,,$
$\mathit{start\_mission}\,.\,, \mathit{done\_mission}\,.\,,$
$\mathit{initializeRet}\,.\,, \mathit{requestTermination}\,.\,.\,,$
$\mathit{start\_mission}\,.\,, \mathit{done\_mission}\,.\,,$
$\mathit{initializeRet}\,.\,, \mathit{requestTermination}\,.\,. |\!\}$

section *Program* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MissionFW*,
SafeletFW, *TopLevelMissionSequencerFW*, *NetworkChannels*, *ManagedThreadFW*,
SchedulableMissionSequencerFW, *PeriodicEventHandlerFW*, *OneShotEventHandlerFW*,
AperiodicEventHandlerFW, *ACSafeletApp*, *MainMissionSequencerApp*,
ObjectFW, *ThreadFW*, *MainMissionApp*, *ACModeChangerApp*, *ControlHandlerApp*, *CommunicationsHandlerAp*

process $ControlTier \mathrel{\widehat=}$

$$\left(\begin{array}{l} SafeletFW \\ \quad \llbracket ControlTierSync \rrbracket \\ TopLevelMissionSequencerFW(MainMissionSequencer) \end{array}\right)$$

process $Tier0 \mathrel{\widehat=}$

$$\left(\begin{array}{l} MissionFW(MainMissionID) \\ \quad \llbracket MissionSync \rrbracket \\ \left(\begin{array}{l} SchedulableMissionSequencerFW(ACModeChangerID) \\ \quad \llbracket SchedulablesSync \rrbracket \\ \left(\begin{array}{l} AperiodicEventHandlerFW(ControlHandlerID, (time(10,0), null)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW(CommunicationsHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad \llbracket SchedulablesSync \rrbracket \\ \left(\begin{array}{l} PeriodicEventHandlerFW(EnvironmentMonitorID, (time(10,0), NULL, NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW(FlightSensorsMonitorID, (time(10,0), NULL, NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW(AperiodicSimulatorID, (time(10,0), NULL, NULL, nullSchedulableId)) \end{array}\right) \end{array}\right) \end{array}\right)$$

process $Tier1 \mathrel{\widehat=}$

$$\begin{array}{l} \left(\begin{array}{l} MissionFW(TakeOffMissionID) \\ \quad \llbracket MissionSync \rrbracket \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerTakeOffID, (NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW(TakeOffFailureHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW(TakeOffMonitorID, (time(0,0), time(500,0), NULL, nullSchedulableId)) \end{array}\right) \end{array}\right) \\ \quad \llbracket ClusterSync \rrbracket \\ \left(\begin{array}{l} MissionFW(CruiseMissionID) \\ \quad \llbracket MissionSync \rrbracket \\ \left(\begin{array}{l} AperiodicEventHandlerFW(BeginLandingHandlerID, (NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW(NavigationMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId)) \end{array}\right) \end{array}\right) \\ \quad \llbracket ClusterSync \rrbracket \\ \left(\begin{array}{l} MissionFW(LandMissionID) \\ \quad \llbracket MissionSync \rrbracket \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerLandID, (NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW(SafeLandingHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad \llbracket SchedulablesSync \rrbracket \\ \left(\begin{array}{l} PeriodicEventHandlerFW(GroundDistanceMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId)) \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW(InstrumentLandingSystemMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId) \end{array}\right. \end{array}\right. \end{array}\right. \end{array}$$

process $Framework \mathrel{\widehat=}$

$$\left(\begin{array}{l} ControlTier \\ \quad \llbracket TierSync \rrbracket \\ \left(\begin{array}{l} Tier0 \\ \quad \llbracket Tier0Sync \rrbracket \\ Tier1 \end{array}\right) \end{array}\right)$$

$$
\mathbf{process}\ Application \mathrel{\widehat=}
\left(\begin{array}{l}
ACSafeletApp \\
\interleave \\
MainMissionSequencerApp \\
\interleave \\
MainMissionApp \\
\interleave \\
ACModeChangerApp(MainMissionID) \\
\interleave \\
ControlHandlerApp \\
\interleave \\
CommunicationsHandlerApp \\
\interleave \\
EnvironmentMonitorApp(MainMissionID) \\
\interleave \\
FlightSensorsMonitorApp(MainMissionID) \\
\interleave \\
AperiodicSimulatorApp(controlHandlerID) \\
\interleave \\
TakeOffMissionApp \\
\interleave \\
LandingGearHandlerTakeOffApp(TakeOffMissionID) \\
\interleave \\
TakeOffFailureHandlerApp(TakeOffMissionID, 10.0) \\
\interleave \\
TakeOffMonitorApp(TakeOffMissionID, 10.0, landingGearHandlerID) \\
\interleave \\
CruiseMissionApp \\
\interleave \\
BeginLandingHandlerApp(CruiseMissionID) \\
\interleave \\
NavigationMonitorApp(CruiseMissionID) \\
\interleave \\
LandMissionApp \\
\interleave \\
LandingGearHandlerLandApp(LandMissionID) \\
\interleave \\
SafeLandingHandlerApp(LandMissionID, 10.0) \\
\interleave \\
GroundDistanceMonitorApp(LandMissionID) \\
\interleave \\
InstrumentLandingSystemMonitorApp(LandMissionID)
\end{array}\right)
$$

$$
Threads \mathrel{\widehat=} \\
\left(ThreadFW(Threads,)\right)
$$

$$
\begin{array}{l}
\mathit{Objects} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{ObjectFW}(\mathit{ACSafeletObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{MainMissionObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{ACModeChangerObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{EnvironmentMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{ControlHandlerObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{FlightSensorsMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{CommunicationsHandlerObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{AperiodicSimulatorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{TakeOffMissionObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{LandingGearHandlerTakeOffObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{TakeOffMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{TakeOffFailureHandlerObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{CruiseMissionObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{BeginLandingHandlerObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{NavigationMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{LandMissionObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{GroundDistanceMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{LandingGearHandlerLandObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{InstrumentLandingSystemMonitorObjectID}) \\
\quad \llbracket \mathit{ObjectSync} \rrbracket \\
\mathit{ObjectFW}(\mathit{SafeLandingHandlerObjectID})
\end{array}\right)
\end{array}
$$

$$\mathit{Locking} \mathrel{\widehat=} \mathit{Threads} \interleave \mathit{Objects}$$

$$\mathbf{process}\ \mathit{Program} \mathrel{\widehat=} \left(\mathit{Framework} \llbracket \mathit{AppSync} \rrbracket \mathit{Application}\right) \llbracket \mathit{LockingSync} \rrbracket \mathit{Locking}$$

3 Safelet

section *ACSafeletApp* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*

process *ACSafeletApp* $\widehat{=}$ **begin**

$$
InitializeApplication \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
initializeApplicationCall \longrightarrow \\
initializeApplicationRet \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

$$
GetSequencer \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
getSequencerCall \longrightarrow \\
getSequencerRet\,!\,MainMissionSequencer \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

$$
Methods \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
GetSequencer \\
\Box \\
InitializeApplication
\end{array}\right) \mathbin{;}\ Methods
$$

$$\bullet\ (Methods) \mathbin{\triangle} (end\_safelet\_app \longrightarrow \mathbf{Skip})$$

end

4 Top Level Mission Sequencer

section *MainMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*, *MissionId*, *MissionIds*, *SchedulableId*, *MainMissionSequencerClass*

process *MainMissionSequencerApp* $\widehat{=}$ **begin**

$$\begin{array}{|l}
\textit{State} \\
\hline
\textit{this} : \mathbf{ref}\ \textit{MainMissionSequencerClass} \\
\hline
\end{array}$$

state *State*

$$\begin{array}{|l}
\textit{Init} \\
\hline
\textit{State}\,' \\
\hline
\textit{this}' = \mathbf{new}\ \textit{MainMissionSequencerClass}() \\
\hline
\end{array}$$

$$\begin{array}{l}
\textit{GetNextMission} \mathrel{\widehat{=}} \mathbf{var}\ \textit{ret} : \textit{MissionID} \bullet \\
\left(\begin{array}{l}
\textit{getNextMissionCall} \,.\, \textit{MainMissionSequencer} \longrightarrow \\
\textit{ret} := \textit{this} \,.\, \textit{getNextMission}(); \\
\textit{getNextMissionRet} \,.\, \textit{MainMissionSequencer}\,!\,\textit{ret} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}$$

$$\begin{array}{l}
\textit{Methods} \mathrel{\widehat{=}} \\
\left(\textit{GetNextMission}\right) \;;\; \textit{Methods}
\end{array}$$

$$\bullet\ (\textit{Init} \;;\; \textit{Methods}) \mathbin{\triangle} (\textit{end\_sequencer\_app} \,.\, \textit{MainMissionSequencer} \longrightarrow \mathbf{Skip})$$

end

$\mathbf{class}\ \mathit{MainMissionSequencerClass} \mathrel{\widehat=} \mathbf{begin}$

$$\begin{array}{|l}
\hline
\mathbf{state}\ \mathit{State} \\
\mathit{returnedMission} : \mathbb{B} \\
\hline
\end{array}$$

$\mathbf{state}\ \mathit{State}$

$$\begin{array}{|l}
\hline
\mathbf{initial}\ \mathit{Init} \\
\mathit{State}' \\
\hline
\end{array}$$

$$\begin{array}{l}
\mathbf{protected\ sync}\ \mathit{getNextMission} \mathrel{\widehat=} \mathbf{var}\ \mathit{ret} : \mathit{MissionID} \bullet \\
\left(\begin{array}{l}
\mathbf{if}\ (\lnot\ \mathit{returnedMission} = \mathbf{True}) \longrightarrow \\
\qquad \left(\begin{array}{l} \mathit{this} \mathbin{.} \mathit{returnedMission} := \mathit{true}; \\ \mathit{ret} := \mathit{MainMission} \end{array}\right) \\
[\!]\ \lnot\ (\lnot\ \mathit{returnedMission} = \mathbf{True}) \longrightarrow \\
\qquad \left(\mathit{ret} := \mathit{nullMissionId}\right) \\
\mathbf{fi}
\end{array}\right)
\end{array}$$

$\bullet\ \mathbf{Skip}$

$\mathbf{end}$

section $MainMissionSequencerMethChan$ **parents** $scj\_prelude, GlobalTypes, MissionId, SchedulableId$

channel $getNextMissionCall : SchedulableID$
channel $getNextMissionRet : SchedulableID \times MissionID$

5 Missions

5.1 MainMission

section *MainMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MainMissionClass* , *MainMissionMethChan*

process *MainMissionApp* $\widehat{=}$
$test : \mathbb{Z} \bullet$ **begin**

$$\begin{array}{|l}
\textit{State} \\ \hline
this : \mathbf{ref}\ MainMissionClass \\ \hline
\end{array}$$

state *State*

$$\begin{array}{|l}
\textit{Init} \\ \hline
State\,' \\ \hline
this' = \mathbf{new}\ MainMissionClass() \\ \hline
\end{array}$$

$$InitializePhase \mathrel{\widehat=} \left(\begin{array}{l}
initializeCall\,.\,MainMission \longrightarrow \\
register\,!\,ACModeChanger\,!\,MainMission \longrightarrow \\
register\,!\,EnvironmentMonitor\,!\,MainMission \longrightarrow \\
register\,!\,ControlHandler\,!\,MainMission \longrightarrow \\
register\,!\,FlightSensorsMonitor\,!\,MainMission \longrightarrow \\
register\,!\,CommunicationsHandler\,!\,MainMission \longrightarrow \\
register\,!\,AperiodicSimulator\,!\,MainMission \longrightarrow \\
initializeRet\,.\,MainMission \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

$$CleanupPhase \mathrel{\widehat=} \left(\begin{array}{l}
cleanupMissionCall\,.\,MainMission \longrightarrow \\
cleanupMissionRet\,.\,MainMission\,!\,\mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

$$getAirSpeedSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \left(\begin{array}{l}
getAirSpeedCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getAirSpeed(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getAirSpeedRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)$$

$$getAltitudeSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \left(\begin{array}{l}
getAltitudeCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getAltitude(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getAltitudeRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)$$

$$
\begin{array}{l}
getCabinPressureSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \\
\left(\begin{array}{l}
getCabinPressureCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getCabinPressure(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getCabinPressureRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
getEmergencyOxygenSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \\
\left(\begin{array}{l}
getEmergencyOxygenCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getEmergencyOxygen(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getEmergencyOxygenRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
getFuelRemainingSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \\
\left(\begin{array}{l}
getFuelRemainingCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getFuelRemaining(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getFuelRemainingRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
getHeadingSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \\
\left(\begin{array}{l}
getHeadingCall\,.\,MainMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getHeading(); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
getHeadingRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
setAirSpeedSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
setAirSpeedCall\,.\,MainMission\,?\,thread\,?\,airSpeed \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
this\,.\,setAirSpeed(airSpeed); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
setAirSpeedRet\,.\,MainMission\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
setAltitudeSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
setAltitudeCall\,.\,MainMission\,?\,thread\,?\,altitude \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\
this\,.\,setAltitude(altitude); \\
endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\
setAltitudeRet\,.\,MainMission\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{setCabinPressureSyncMeth} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{setCabinPressureCall} \,.\, \mathit{MainMission} \,?\, \mathit{thread} \,?\, \mathit{cabinPressure} \longrightarrow \\
\left(\begin{array}{l}
\mathit{startSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{lockAcquired} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{this} \,.\, \mathit{setCabinPressure}(\mathit{cabinPressure}); \\
\mathit{endSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{setCabinPressureRet} \,.\, \mathit{MainMission} \,.\, \mathit{thread} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{setEmergencyOxygenSyncMeth} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{setEmergencyOxygenCall} \,.\, \mathit{MainMission} \,?\, \mathit{thread} \,?\, \mathit{emergencyOxygen} \longrightarrow \\
\left(\begin{array}{l}
\mathit{startSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{lockAcquired} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{this} \,.\, \mathit{setEmergencyOxygen}(\mathit{emergencyOxygen}); \\
\mathit{endSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{setEmergencyOxygenRet} \,.\, \mathit{MainMission} \,.\, \mathit{thread} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{setFuelRemainingSyncMeth} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{setFuelRemainingCall} \,.\, \mathit{MainMission} \,?\, \mathit{thread} \,?\, \mathit{fuelRemaining} \longrightarrow \\
\left(\begin{array}{l}
\mathit{startSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{lockAcquired} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{this} \,.\, \mathit{setFuelRemaining}(\mathit{fuelRemaining}); \\
\mathit{endSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{setFuelRemainingRet} \,.\, \mathit{MainMission} \,.\, \mathit{thread} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{setHeadingSyncMeth} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{setHeadingCall} \,.\, \mathit{MainMission} \,?\, \mathit{thread} \,?\, \mathit{heading} \longrightarrow \\
\left(\begin{array}{l}
\mathit{startSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{lockAcquired} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{this} \,.\, \mathit{setHeading}(\mathit{heading}); \\
\mathit{endSyncMeth} \,.\, \mathit{MainMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{setHeadingRet} \,.\, \mathit{MainMission} \,.\, \mathit{thread} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$Methods \mathrel{\widehat=} \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getAirSpeedSyncMeth \\ \Box \\ getAltitudeSyncMeth \\ \Box \\ getCabinPressureSyncMeth \\ \Box \\ getEmergencyOxygenSyncMeth \\ \Box \\ getFuelRemainingSyncMeth \\ \Box \\ getHeadingSyncMeth \\ \Box \\ setAirSpeedSyncMeth \\ \Box \\ setAltitudeSyncMeth \\ \Box \\ setCabinPressureSyncMeth \\ \Box \\ setEmergencyOxygenSyncMeth \\ \Box \\ setFuelRemainingSyncMeth \\ \Box \\ setHeadingSyncMeth \end{array}\right) ;~ Methods$$

$\bullet\ (Init ;~ Methods) \vartriangle (end\_mission\_app\,.\,MainMission \longrightarrow \mathbf{Skip})$

end

class *MainMissionClass* $\widehat{=}$ **begin**

state *State*

$ALTITUDE\_READING\_ON\_GROUND : double$
$test : \mathbb{Z}$
$cabinPressure : double$
$emergencyOxygen : double$
$fuelRemaining : double$
$altitude : double$
$airSpeed : double$
$heading : double$

state *State*

initial *Init*

$State\,'$

$ALTITUDE\_READING\_ON\_GROUND' = 0.0$
$test' = 0$

public sync $getAirSpeed \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := airSpeed\right)$

public sync $getAltitude \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := altitude\right)$

public sync $getCabinPressure \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := cabinPressure\right)$

public sync $getEmergencyOxygen \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := emergencyOxygen\right)$

public sync $getFuelRemaining \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := fuelRemaining\right)$

public sync $getHeading \mathrel{\widehat{=}} \mathbf{var}\ ret : double \bullet$
$\left(ret := heading\right)$

public sync $setAirSpeed \mathrel{\widehat{=}}$
$\left(this\ .\ this.airSpeed := airSpeed\right)$

public sync $setAltitude \mathrel{\widehat{=}}$
$\left(this\ .\ this.altitude := altitude\right)$

public sync $setCabinPressure \mathrel{\widehat{=}}$
$\left(this\ .\ this.cabinPressure := cabinPressure\right)$

public sync $setEmergencyOxygen \mathrel{\widehat{=}}$
$\left(this\ .\ this.emergencyOxygen := emergencyOxygen\right)$

public sync $setFuelRemaining \mathrel{\widehat=}$
$\big(this \,.\, this.fuelRemaining := fuelRemaining\big)$

public sync $setHeading \mathrel{\widehat=}$
$\big(this \,.\, this.heading := heading\big)$

• **Skip**

end

section *MainMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel $getAirSpeedCall : MissionID \times ThreadID$
channel $getAirSpeedRet : MissionID \times ThreadID \times double$

channel $getAltitudeCall : MissionID \times ThreadID$
channel $getAltitudeRet : MissionID \times ThreadID \times double$

channel $getCabinPressureCall : MissionID \times ThreadID$
channel $getCabinPressureRet : MissionID \times ThreadID \times double$

channel $getEmergencyOxygenCall : MissionID \times ThreadID$
channel $getEmergencyOxygenRet : MissionID \times ThreadID \times double$

channel $getFuelRemainingCall : MissionID \times ThreadID$
channel $getFuelRemainingRet : MissionID \times ThreadID \times double$

channel $getHeadingCall : MissionID \times ThreadID$
channel $getHeadingRet : MissionID \times ThreadID \times double$

channel $setAirSpeedCall : MissionID \times ThreadID \times double$
channel $setAirSpeedRet : MissionID \times ThreadID$

channel $setAltitudeCall : MissionID \times ThreadID \times double$
channel $setAltitudeRet : MissionID \times ThreadID$

channel $setCabinPressureCall : MissionID \times ThreadID \times double$
channel $setCabinPressureRet : MissionID \times ThreadID$

channel $setEmergencyOxygenCall : MissionID \times ThreadID \times double$
channel $setEmergencyOxygenRet : MissionID \times ThreadID$

channel $setFuelRemainingCall : MissionID \times ThreadID \times double$
channel $setFuelRemainingRet : MissionID \times ThreadID$

channel $setHeadingCall : MissionID \times ThreadID \times double$
channel $setHeadingRet : MissionID \times ThreadID$

5.2 Schedulables of MainMission

section *ACModeChangerApp* **parents** *TopLevelMissionSequencerChan*, *MissionId*, *MissionIds*, *SchedulableId*, *ACModeChangerClass*

process *ACModeChangerApp* $\widehat{=}$
controllingMission : *MissionID* $\bullet$ **begin**

$$GetNextMission \mathrel{\widehat{=}} \mathbf{var}\ ret : MissionID \bullet \left(\begin{array}{l} getNextMissionCall \,.\, ACModeChanger \longrightarrow \\ ret := this \,.\, getNextMission(); \\ getNextMissionRet \,.\, ACModeChanger\, !\, ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$advanceModeMeth \mathrel{\widehat{=}} \left(\begin{array}{l} advanceModeCall \,.\, ACModeChanger \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \mathbf{if}\ (modesLeft = 3) \longrightarrow \\ \qquad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(launchMode) \end{array}\right) \\ [\!]\ \neg\ (modesLeft = 3) \longrightarrow \\ \qquad \mathbf{if}\ (modesLeft = 2) \longrightarrow \\ \qquad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(cruiseMode) \end{array}\right) \\ [\!]\ \neg\ (modesLeft = 2) \longrightarrow \\ \qquad \mathbf{if}\ (modesLeft = 1) \longrightarrow \\ \qquad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(landMode) \end{array}\right) \\ [\!]\ \neg\ (modesLeft = 1) \longrightarrow \\ \qquad \left(changeTo(\mathbf{null})\right) \\ \mathbf{fi} \\ \mathbf{fi} \\ \mathbf{fi} \end{array}\right); \\ advanceModeRet \,.\, ACModeChanger \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$changeToSyncMeth \mathrel{\widehat{=}} \left(\begin{array}{l} changeToCall \,.\, ACModeChanger\, ?\, thread\, ?\, newMode \longrightarrow \\ \left(\begin{array}{l} startSyncMeth \,.\, ACModeChangerObject \,.\, thread \longrightarrow \\ lockAcquired \,.\, ACModeChangerObject \,.\, thread \longrightarrow \\ \left(this \,.\, currentMode := newMode\right); \\ endSyncMeth \,.\, ACModeChangerObject \,.\, thread \longrightarrow \\ changeToRet \,.\, ACModeChanger \,.\, thread \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$$

$$Methods \mathrel{\widehat{=}} \left(\begin{array}{l} GetNextMission \\ \Box \\ advanceModeMeth \\ \Box \\ changeToSyncMeth \end{array}\right); \ Methods$$

$\bullet\ (Methods) \mathbin{\triangle} (end\_sequencer\_app \,.\, ACModeChanger \longrightarrow \mathbf{Skip})$

```
end
```

class $ACModeChangerClass \mathrel{\widehat=}$ **begin**

state $State$

$modesLeft : \mathbb{Z}$
ref $currentModeClass : ModeClass$
ref $launchModeClass : ModeClass$
ref $cruiseModeClass : ModeClass$
ref $landModeClass : ModeClass$

state $State$

initial $Init$

$State'$

$modesLeft' = 3$
ref $currentModeClass' =$ **new** $ModeClass()$
ref $launchModeClass' =$ **new** $ModeClass()$
ref $cruiseModeClass' =$ **new** $ModeClass()$
ref $landModeClass' =$ **new** $ModeClass()$

protected sync $getNextMission \mathrel{\widehat=}$ **var** $ret : MissionID \bullet$

$$\left(\begin{array}{l}
\mathbf{if}\ (modesLeft = 3) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := TakeOffMission \end{array}\right) \\
[\!]\ \neg\ (modesLeft = 3) \longrightarrow \\
\quad \mathbf{if}\ (modesLeft = 2) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := CruiseMission \end{array}\right) \\
[\!]\ \neg\ (modesLeft = 2) \longrightarrow \\
\quad \mathbf{if}\ (modesLeft = 1) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := LandMission \end{array}\right) \\
[\!]\ \neg\ (modesLeft = 1) \longrightarrow \\
\quad \left(ret := nullMissionId\right) \\
\mathbf{fi} \\
\mathbf{fi} \\
\mathbf{fi}
\end{array}\right)$$

$\bullet$ **Skip**

end

section $ACModeChangerMethChan$ **parents** $scj\_prelude, GlobalTypes, MissionId, SchedulableId$

channel $advanceModeCall : SchedulableID$
channel $advanceModeRet : SchedulableID$

channel $changeToCall : SchedulableID \times ThreadID \times Mode$
channel $changeToRet : SchedulableID \times ThreadID$

section *ControlHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *ControlHandlerApp* $\widehat{=}$ **begin**

$$handlerAsyncEvent \mathrel{\widehat{=}} \left(\begin{array}{l} handleAsyncEventCall\,.\,ControlHandler \longrightarrow \\ \left(\mathbf{Skip}\right)\,; \\ handleAsyncEventRet\,.\,ControlHandler \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$Methods \mathrel{\widehat{=}} \left(handlerAsyncEvent\right)\,;\; Methods$$

$$\bullet\ (Methods) \mathbin{\triangle} (end\_\_app\,.\,ControlHandler \longrightarrow \mathbf{Skip})$$

end

class $\mathit{ControlHandlerClass} \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section *ControlHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *CommunicationsHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *CommunicationsHandlerApp* $\widehat{=}$ **begin**

$$
handlerAsyncEvent \mathrel{\widehat=} \\
\left(\begin{array}{l}
handleAsyncEventCall \,.\, CommunicationsHandler \longrightarrow \\
\left(\mathbf{Skip}\right) ; \\
handleAsyncEventRet \,.\, CommunicationsHandler \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

$$
Methods \mathrel{\widehat=} \\
\left(handlerAsyncEvent\right) ;\ Methods
$$

$$\bullet\ (Methods) \mathbin{\triangle} (end\_\_app \,.\, CommunicationsHandler \longrightarrow \mathbf{Skip})$$

end

class $CommunicationsHandlerClass \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section *CommunicationsHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *EnvironmentMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
MainMissionMethChan

process *EnvironmentMonitorApp* $\widehat{=}$
    mainMission : *MissionID* ● **begin**

$$
\begin{array}{l}
\mathit{handlerAsyncEvent} \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
\mathit{handleAsyncEventCall} \,.\, \mathit{EnvironmentMonitor} \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
\mathit{setCabinPressureCall} \,.\, \mathit{controllingMission} \,!\, 0 \longrightarrow \\
\mathit{setCabinPressureRet} \,.\, \mathit{controllingMission} \longrightarrow \\
\mathbf{Skip}; \\
\mathit{setEmergencyOxygenCall} \,.\, \mathit{controllingMission} \,!\, 0 \longrightarrow \\
\mathit{setEmergencyOxygenRet} \,.\, \mathit{controllingMission} \longrightarrow \\
\mathbf{Skip}; \\
\mathit{setFuelRemainingCall} \,.\, \mathit{controllingMission} \,!\, 0 \longrightarrow \\
\mathit{setFuelRemainingRet} \,.\, \mathit{controllingMission} \longrightarrow \\
\mathbf{Skip}
\end{array}\right); \\
\mathit{handleAsyncEventRet} \,.\, \mathit{EnvironmentMonitor} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Methods} \mathrel{\widehat{=}} \\
\left(\mathit{handlerAsyncEvent}\right) \; ; \;\; \mathit{Methods}
\end{array}
$$

$\bullet \; (\mathit{Methods}) \mathbin{\triangle} (\mathit{end\_periodic\_app} \,.\, \mathit{EnvironmentMonitor} \longrightarrow \mathbf{Skip})$

end

class $\mathit{EnvironmentMonitorClass} \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section *FlightSensorsMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
MainMissionMethChan

process *FlightSensorsMonitorApp* $\widehat{=}$
$\quad mainMission : MissionID$ ● **begin**

$$handlerAsyncEvent \mathrel{\widehat=} \\ \left(\begin{array}{l} handleAsyncEventCall \,.\, FlightSensorsMonitor \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ setAirSpeedCall \,.\, controllingMission \,!\, 0 \longrightarrow \\ setAirSpeedRet \,.\, controllingMission \longrightarrow \\ \mathbf{Skip}; \\ setAltitudeCall \,.\, controllingMission \,!\, 0 \longrightarrow \\ setAltitudeRet \,.\, controllingMission \longrightarrow \\ \mathbf{Skip}; \\ setHeadingCall \,.\, controllingMission \,!\, 0 \longrightarrow \\ setHeadingRet \,.\, controllingMission \longrightarrow \\ \mathbf{Skip} \end{array}\right); \\ handleAsyncEventRet \,.\, FlightSensorsMonitor \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$Methods \mathrel{\widehat=} \\ \big(handlerAsyncEvent\big) \; ; \; Methods$$

$$\bullet \; (Methods) \vartriangle (end\_periodic\_app \,.\, FlightSensorsMonitor \longrightarrow \mathbf{Skip})$$

end

class *FlightSensorsMonitorClass* $\widehat{=}$ **begin**

$\bullet$ **Skip**

end

section $AperiodicSimulatorApp$ **parents** $PeriodicEventHandlerChan, SchedulableId, SchedulableIds$

process $AperiodicSimulatorApp \mathrel{\widehat=}$
$\quad aperiodicEvent : SchedulableID \bullet$ **begin**

$$handlerAsyncEvent \mathrel{\widehat=}$$
$$\left(\begin{array}{l} handleAsyncEventCall \,.\, AperiodicSimulator \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ releaseCall \,.\, event \longrightarrow \\ releaseRet \,.\, event \,?\, release \longrightarrow \\ \mathbf{Skip} \end{array}\right); \\ handleAsyncEventRet \,.\, AperiodicSimulator \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$Methods \mathrel{\widehat=}$$
$$\left(handlerAsyncEvent\right) ; \ Methods$$

$\bullet\ (Methods) \mathbin{\triangle} (end\_periodic\_app \,.\, AperiodicSimulator \longrightarrow \mathbf{Skip})$

end

class $\mathit{AperiodicSimulatorClass} \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

5.3 TakeOffMission

section *TakeOffMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *TakeOffMissionClass* , *TakeOffMissionMethChan*

process *TakeOffMissionApp* $\widehat{=}$
$controllingMission : MissionID \bullet$ **begin**

$$\begin{array}{|l}
\hline
\textit{State} \\
this : \mathbf{ref}\ TakeOffMissionClass \\
\hline
\end{array}$$

state *State*

$$\begin{array}{|l}
\hline
\textit{Init} \\
State' \\
\hline
this' = \mathbf{new}\ TakeOffMissionClass() \\
\hline
\end{array}$$

$$
\begin{array}{l}
InitializePhase \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
initializeCall\,.\,TakeOffMission \longrightarrow \\
register\,!\,LandingGearHandlerTakeOff\,!\,TakeOffMission \longrightarrow \\
register\,!\,TakeOffMonitor\,!\,TakeOffMission \longrightarrow \\
register\,!\,TakeOffFailureHandler\,!\,TakeOffMission \longrightarrow \\
initializeRet\,.\,TakeOffMission \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
CleanupPhase \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
cleanupMissionCall\,.\,TakeOffMission \longrightarrow \\
cleanupMissionRet\,.\,TakeOffMission\,!\,\mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
deployLandingGearMeth \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
deployLandingGearCall\,.\,TakeOffMission \longrightarrow \\
\left(this\,.\,landingGearDeployed := true\right); \\
deployLandingGearRet\,.\,TakeOffMission \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
abortSyncMeth \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
abortCall\,.\,TakeOffMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
this\,.\,abort(); \\
endSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
abortRet\,.\,TakeOffMission\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
getControllingMissionSyncMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : MissionID \bullet \\
\left(\begin{array}{l}
getControllingMissionCall\,.\,TakeOffMission\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getControllingMission(); \\
endSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\
getControllingMissionRet\,.\,TakeOffMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
setControllingMissionSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
setControllingMissionCall \,.\, TakeOffMission \,?\, thread \,?\, controllingMission \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
this \,.\, setControllingMission(controllingMission); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
setControllingMissionRet \,.\, TakeOffMission \,.\, thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
cleanUpSyncMeth \mathrel{\widehat=} \mathbf{var}\; ret : \mathbb{B} \bullet \\
\left(\begin{array}{l}
cleanUpCall \,.\, TakeOffMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
ret := this \,.\, cleanUp(); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
cleanUpRet \,.\, TakeOffMission \,!\, thread \,!\, ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
stowLandingGearSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
stowLandingGearCall \,.\, TakeOffMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
this \,.\, stowLandingGear(); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
stowLandingGearRet \,.\, TakeOffMission \,.\, thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
isLandingGearDeployedSyncMeth \mathrel{\widehat=} \mathbf{var}\; ret : \mathbb{B} \bullet \\
\left(\begin{array}{l}
isLandingGearDeployedCall \,.\, TakeOffMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
ret := this \,.\, isLandingGearDeployed(); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
isLandingGearDeployedRet \,.\, TakeOffMission \,!\, thread \,!\, ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
Methods \mathrel{\widehat=} \left(\begin{array}{l}
InitializePhase \\
\Box \\
CleanupPhase \\
\Box \\
deployLandingGearMeth \\
\Box \\
abortSyncMeth \\
\Box \\
getControllingMissionSyncMeth \\
\Box \\
setControllingMissionSyncMeth \\
\Box \\
cleanUpSyncMeth \\
\Box \\
stowLandingGearSyncMeth \\
\Box \\
isLandingGearDeployedSyncMeth
\end{array}\right) ; \; Methods
$$

$$
\bullet \; (Init \;;\; Methods) \mathbin{\triangle} (end\_mission\_app \,.\, TakeOffMission \longrightarrow \mathbf{Skip})
$$

end

class *TakeOffMissionClass* $\widehat{=}$ **begin**

$$
\begin{array}{|l}
\textbf{state}\ State \\
\hline
SAFE\_AIRSPEED\_THRESHOLD : double \\
TAKEOFF\_ALTITUDE : double \\
abort : \mathbb{B} \\
landingGearDeployed : \mathbb{B} \\
\hline
\end{array}
$$

state *State*

$$
\begin{array}{|l}
\textbf{initial}\ Init \\
\hline
State' \\
\hline
SAFE\_AIRSPEED\_THRESHOLD' = 10.0 \\
TAKEOFF\_ALTITUDE' = 10.0 \\
abort' = false \\
\hline
\end{array}
$$

public sync $abort \ \widehat{=}$
$\left(this\ .\ abort := true \right)$

public sync $getControllingMission \ \widehat{=}$ **var** $ret : MissionID$ ●
$\left(ret := controllingMission \right)$

public sync $setControllingMission \ \widehat{=}$
$\left(this\ .\ this.controllingMission := controllingMission \right)$

public sync $cleanUp \ \widehat{=}$ **var** $ret : \mathbb{B}$ ●
$$\left(\begin{array}{l} \textbf{Skip}; \\ ret := (\neg\ abort = \textbf{True}) \end{array} \right)$$

public sync $stowLandingGear \ \widehat{=}$
$\left(this\ .\ landingGearDeployed := false \right)$

public sync $isLandingGearDeployed \ \widehat{=}$ **var** $ret : \mathbb{B}$ ●
$\left(ret := landingGearDeployed = \textbf{True} \right)$

● **Skip**

end

section *TakeOffMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel $deployLandingGearCall : MissionID$
channel $deployLandingGearRet : MissionID$

channel $abortCall : MissionID \times ThreadID$
channel $abortRet : MissionID \times ThreadID$

channel $getControllingMissionCall : MissionID \times ThreadID$
channel $getControllingMissionRet : MissionID \times ThreadID \times MissionID$

channel $setControllingMissionCall : MissionID \times ThreadID \times MissionID$
channel $setControllingMissionRet : MissionID \times ThreadID$

channel $cleanUpCall : MissionID \times ThreadID$
channel $cleanUpRet : MissionID \times ThreadID \times \mathbb{B}$

channel $stowLandingGearCall : MissionID \times ThreadID$
channel $stowLandingGearRet : MissionID \times ThreadID$

channel $isLandingGearDeployedCall : MissionID \times ThreadID$
channel $isLandingGearDeployedRet : MissionID \times ThreadID \times \mathbb{B}$

5.4 Schedulables of TakeOffMission

section *LandingGearHandlerTakeOffApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* , *TakeOffMissionMethChan*, *ObjectIds*, *ThreadIds*

process *LandingGearHandlerTakeOffApp* $\widehat{=}$
    $mission : MissionID \bullet$ **begin**

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
handleAsyncEventCall\,.\,LandingGearHandlerTakeOff \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
isLandingGearDeployedCall\,.\,mission \longrightarrow \\
isLandingGearDeployedRet\,.\,mission\,?\,isLandingGearDeployed \longrightarrow \\
\\
\mathbf{var}\ landingGearIsDeployed : \mathbb{B} \bullet landingGearIsDeployed := isLandingGearDeployed \\
\mathbf{if}\ landingGearIsDeployed = \mathbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l}
stowLandingGearCall\,.\,mission \longrightarrow \\
stowLandingGearRet\,.\,mission \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
[\!]\ \neg\ landingGearIsDeployed = \mathbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l}
deployLandingGearCall\,.\,mission\,.\,LandingGearHandlerTakeOffThread \longrightarrow \\
deployLandingGearRet\,.\,mission\,.\,LandingGearHandlerTakeOffThread \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
\mathbf{fi}
\end{array}\right); \\
handleAsyncEventRet\,.\,LandingGearHandlerTakeOff \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\big(handlerAsyncEvent\big)\ ;\ \ Methods
\end{array}
$$

$\bullet\ (Methods) \vartriangle (end\_\_app\,.\,LandingGearHandlerTakeOff \longrightarrow \mathbf{Skip})$

end

class $LandingGearHandlerTakeOffClass \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section *LandingGearHandlerTakeOffMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *TakeOffFailureHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* , *TakeOffMissionMethChan*

process *TakeOffFailureHandlerApp* $\widehat{=}$
    takeoffMission : *MissionID*,
    threshold : *Double* ● **begin**

$$
\begin{array}{l}
\mathit{handlerAsyncEvent} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{handleAsyncEventCall} \,.\, \mathit{TakeOffFailureHandler} \longrightarrow \\
\left(\begin{array}{l}
\mathit{getControllingMissionCall} \,.\, \mathit{takeoffMission}.\mathit{getControllingMission}() \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{takeoffMission}.\mathit{getControllingMission}() \,?\, \mathit{getControllingMission} \longrightarrow \\
\\
\mathbf{var}\ \mathit{currentSpeed} : \mathit{double} \bullet \mathit{currentSpeed} := \mathit{getAirSpeed} \\
\mathbf{if}\ (\mathit{currentSpeed} < \mathit{threshold}) \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathbf{Skip}; \\
\mathit{abortCall} \,.\, \mathit{takeoffMission} \longrightarrow \\
\mathit{abortRet} \,.\, \mathit{takeoffMission} \longrightarrow \\
\mathbf{Skip}; \\
\mathit{requestTerminationCall} \,.\, \mathit{takeoffMission} \longrightarrow \\
\mathit{requestTerminationRet} \,.\, \mathit{takeoffMission} \,?\, \mathit{requestTermination} \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
\square\ \neg\ (\mathit{currentSpeed} < \mathit{threshold}) \longrightarrow \\
\qquad (\mathbf{Skip}) \\
\mathbf{fi}\ \mathbf{Skip}
\end{array}\right); \\
\mathit{handleAsyncEventRet} \,.\, \mathit{TakeOffFailureHandler} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Methods} \mathrel{\widehat=} \\
(\mathit{handlerAsyncEvent}) \;;\; \mathit{Methods}
\end{array}
$$

$\bullet\ (\mathit{Methods}) \bigtriangleup (\mathit{end\_\_app} \,.\, \mathit{TakeOffFailureHandler} \longrightarrow \mathbf{Skip})$

end

class $\mathit{TakeOffFailureHandlerClass} \mathrel{\widehat=}$ **begin**

state *State*

$\mathit{threshold} : \mathit{double}$

state *State*

initial *Init*

State'

$\bullet$ **Skip**

end

section $\mathit{TakeOffFailureHandlerMethChan}$ **parents** $\mathit{scj\_prelude}, \mathit{GlobalTypes}, \mathit{MissionId}, \mathit{SchedulableId}$

section *TakeOffMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
TakeOffMissionMethChan

process *TakeOffMonitorApp* $\widehat{=}$
    takeoffMission : *MissionID*,
    takeOffAltitude : *double*,
    landingGearHandler : *SchedulableID* ● **begin**

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat=} \\
\left(\begin{array}{l}
handleAsyncEventCall\,.\,TakeOffMonitor \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
getControllingMissionCall\,.\,takeoffMission.getControllingMission() \longrightarrow \\
getControllingMissionRet\,.\,takeoffMission.getControllingMission()\,?\,getControllingMission \longrightarrow \\
\\
\mathbf{var}\ altitude : double \bullet altitude := getAltitude \\
\mathbf{if}\ (altitude > takeOffAltitude) \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathbf{Skip}; \\
releaseCall\,.\,landingGearHandler \longrightarrow \\
releaseRet\,.\,landingGearHandler\,?\,release \longrightarrow \\
requestTerminationCall\,.\,takeoffMission \longrightarrow \\
requestTerminationRet\,.\,takeoffMission\,?\,requestTermination \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
[\!]\ \neg\,(altitude > takeOffAltitude) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}\,; \\
\mathbf{Skip}
\end{array}\right); \\
handleAsyncEventRet\,.\,TakeOffMonitor \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\big(handlerAsyncEvent\big)\ ;\ Methods
\end{array}
$$

$\bullet\ (Methods) \vartriangle (end\_periodic\_app\,.\,TakeOffMonitor \longrightarrow \mathbf{Skip})$

end

class *TakeOffMonitorClass* $\widehat{=}$ **begin**

state *State*

$takeOffAltitude : double$

state *State*

initial *Init*

$State'$

• **Skip**

end

5.5 CruiseMission

$\mathbf{section}\ \mathit{CruiseMissionApp}\ \mathbf{parents}\ \mathit{scj\_prelude}, \mathit{MissionId}, \mathit{MissionIds},$
$\quad \mathit{SchedulableId}, \mathit{SchedulableIds}, \mathit{MissionChan}, \mathit{SchedulableMethChan}, \mathit{CruiseMissionClass}$
$\quad , \mathit{CruiseMissionMethChan}$

$\mathbf{process}\ \mathit{CruiseMissionApp} \mathrel{\widehat=}$
$\quad \mathit{controllingMission} : \mathit{MissionID} \bullet \mathbf{begin}$

$$\begin{array}{|l}
\textit{State} \\ \hline
\mathit{this} : \mathbf{ref}\ \mathit{CruiseMissionClass} \\ \hline
\end{array}$$

$\mathbf{state}\ \mathit{State}$

$$\begin{array}{|l}
\textit{Init} \\ \hline
\mathit{State}\,' \\ \hline
\mathit{this}' = \mathbf{new}\ \mathit{CruiseMissionClass}() \\ \hline
\end{array}$$

$$\mathit{InitializePhase} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{initializeCall} \,.\, \mathit{CruiseMission} \longrightarrow \\
\mathit{register} \,!\, \mathit{BeginLandingHandler} \,!\, \mathit{CruiseMission} \longrightarrow \\
\mathit{register} \,!\, \mathit{NavigationMonitor} \,!\, \mathit{CruiseMission} \longrightarrow \\
\mathit{initializeRet} \,.\, \mathit{CruiseMission} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

$$\mathit{CleanupPhase} \mathrel{\widehat=} \\
\left(\begin{array}{l}
\mathit{cleanupMissionCall} \,.\, \mathit{CruiseMission} \longrightarrow \\
\mathit{cleanupMissionRet} \,.\, \mathit{CruiseMission} \,!\, \mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

$$\mathit{getControllingMissionSyncMeth} \mathrel{\widehat=} \mathbf{var}\ \mathit{ret} : \mathit{MissionID} \bullet \\
\left(\begin{array}{l}
\mathit{getControllingMissionCall} \,.\, \mathit{CruiseMission} \,?\, \mathit{thread} \longrightarrow \\
\left(\begin{array}{l}
\mathit{startSyncMeth} \,.\, \mathit{CruiseMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{lockAcquired} \,.\, \mathit{CruiseMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{ret} := \mathit{this} \,.\, \mathit{getControllingMission}(); \\
\mathit{endSyncMeth} \,.\, \mathit{CruiseMissionObject} \,.\, \mathit{thread} \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{CruiseMission} \,!\, \mathit{thread} \,!\, \mathit{ret} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)$$

$$\mathit{Methods} \mathrel{\widehat=} \left(\begin{array}{l}
\mathit{InitializePhase} \\
\Box \\
\mathit{CleanupPhase} \\
\Box \\
\mathit{getControllingMissionSyncMeth}
\end{array}\right) ; \ \mathit{Methods}$$

$\bullet\ (\mathit{Init}\ ;\ \ \mathit{Methods}) \mathbin{\triangle} (\mathit{end\_mission\_app} \,.\, \mathit{CruiseMission} \longrightarrow \mathbf{Skip})$

$\mathbf{end}$

class $\mathit{CruiseMissionClass} \mathrel{\widehat=}$ **begin**

public sync $\mathit{getControllingMission} \mathrel{\widehat=}$ **var** $\mathit{ret} : \mathit{MissionID} \bullet$
$\left(\mathit{ret} := \mathit{controllingMission}\right)$

$\bullet$ **Skip**

end

section *CruiseMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel $getControllingMissionCall : MissionID \times ThreadID$
channel $getControllingMissionRet : MissionID \times ThreadID \times MissionID$

5.6 Schedulables of CruiseMission

section *BeginLandingHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *BeginLandingHandlerApp* $\widehat{=}$
  controllingMission : *MissionID* ● **begin**

$$
handlerAsyncEvent \mathrel{\widehat=}
\left(\begin{array}{l}
handleAsyncEventCall \,.\, BeginLandingHandler \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
requestTerminationCall \,.\, controllingMission \longrightarrow \\
requestTerminationRet \,.\, controllingMission \,?\, requestTermination \longrightarrow \\
\mathbf{Skip}
\end{array}\right); \\
handleAsyncEventRet \,.\, BeginLandingHandler \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

$$
Methods \mathrel{\widehat=} \\
\big(handlerAsyncEvent\big) \;;\; Methods
$$

$\bullet\ (Methods) \mathbin{\triangle} (end\_\_app \,.\, BeginLandingHandler \longrightarrow \mathbf{Skip})$

end

class $BeginLandingHandlerClass \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section $BeginLandingHandlerMethChan$ **parents** $scj\_prelude, GlobalTypes, MissionId, SchedulableId$

section $\mathit{NavigationMonitorApp}$ **parents** $\mathit{PeriodicEventHandlerChan}, \mathit{SchedulableId}, \mathit{SchedulableIds}$,
$\mathit{CruiseMissionMethChan}$

process $\mathit{NavigationMonitorApp} \mathrel{\widehat=}$
$\quad \mathit{mission} : \mathit{MissionID} \bullet$ **begin**

$\mathit{handlerAsyncEvent} \mathrel{\widehat=}$

$$\left(\begin{array}{l}
\mathit{handleAsyncEventCall} \,.\, \mathit{NavigationMonitor} \longrightarrow \\
\left(\begin{array}{l}
\mathit{getControllingMissionCall} \,.\, \mathit{mission}.\mathit{getControllingMission}() \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{mission}.\mathit{getControllingMission}() \,?\, \mathit{getControllingMission} \longrightarrow \\
\\
\mathbf{var}\ \mathit{heading} : \mathit{double} \bullet \mathit{heading} := \ \mathit{getHeading} \\
\mathit{getControllingMissionCall} \,.\, \mathit{mission}.\mathit{getControllingMission}() \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{mission}.\mathit{getControllingMission}() \,?\, \mathit{getControllingMission} \longrightarrow \\
\\
\mathbf{var}\ \mathit{airSpeed} : \mathit{double} \bullet \mathit{airSpeed} := \ \mathit{getAirSpeed} \\
\mathit{getControllingMissionCall} \,.\, \mathit{mission}.\mathit{getControllingMission}() \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{mission}.\mathit{getControllingMission}() \,?\, \mathit{getControllingMission} \longrightarrow \\
\\
\mathbf{var}\ \mathit{altitude} : \mathit{double} \bullet \mathit{altitude} := \ \mathit{getAltitude} \\
\mathbf{Skip}
\end{array}\right) ; \\
\mathit{handleAsyncEventRet} \,.\, \mathit{NavigationMonitor} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

$\mathit{Methods} \mathrel{\widehat=}$
$\left(\mathit{handlerAsyncEvent}\right) ; \ \mathit{Methods}$

$\bullet \ (\mathit{Methods}) \mathrel{\triangle} (\mathit{end\_periodic\_app} \,.\, \mathit{NavigationMonitor} \longrightarrow \mathbf{Skip})$

end

class *NavigationMonitorClass* $\widehat{=}$ **begin**

• **Skip**

end

5.7 LandMission

section *LandMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *LandMissionClass* , *LandMissionMethChan*

$$\textbf{process}\ LandMissionApp \mathrel{\widehat=} controllingMission : MissionID \bullet \textbf{begin}$$

$$\begin{array}{|l} \hline State \\ \hline this : \textbf{ref}\ LandMissionClass \\ \hline \end{array}$$

state *State*

$$\begin{array}{|l} \hline Init \\ \hline State' \\ \hline this' = \textbf{new}\ LandMissionClass() \\ \hline \end{array}$$

$$InitializePhase \mathrel{\widehat=} \left(\begin{array}{l} initializeCall\,.\,LandMission \longrightarrow \\ register\,!\,GroundDistanceMonitor\,!\,LandMission \longrightarrow \\ register\,!\,LandingGearHandlerLand\,!\,LandMission \longrightarrow \\ register\,!\,InstrumentLandingSystemMonitor\,!\,LandMission \longrightarrow \\ register\,!\,SafeLandingHandler\,!\,LandMission \longrightarrow \\ initializeRet\,.\,LandMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$CleanupPhase \mathrel{\widehat=} \left(\begin{array}{l} cleanupMissionCall\,.\,LandMission \longrightarrow \\ cleanupMissionRet\,.\,LandMission\,!\,\textbf{True} \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$deployLandingGearMeth \mathrel{\widehat=} \left(\begin{array}{l} deployLandingGearCall\,.\,LandMission \longrightarrow \\ \left(this\,.\,landingGearDeployed := true\right); \\ deployLandingGearRet\,.\,LandMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$stowLandingGearSyncMeth \mathrel{\widehat=} \left(\begin{array}{l} stowLandingGearCall\,.\,LandMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\ this\,.\,stowLandingGear(); \\ endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ stowLandingGearRet\,.\,LandMission\,.\,thread \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$$isLandingGearDeployedSyncMeth \mathrel{\widehat=} \textbf{var}\ ret : \mathbb{B} \bullet \left(\begin{array}{l} isLandingGearDeployedCall\,.\,LandMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,isLandingGearDeployed(); \\ endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ isLandingGearDeployedRet\,.\,LandMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$$
\begin{array}{l}
getControllingMissionSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : MissionID \bullet \\
\left(\begin{array}{l}
getControllingMissionCall \,.\, LandMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, LandMissionObject \,.\, thread \longrightarrow \\
ret := this \,.\, getControllingMission(); \\
endSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
getControllingMissionRet \,.\, LandMission \,!\, thread \,!\, ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
abortSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
abortCall \,.\, LandMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, LandMissionObject \,.\, thread \longrightarrow \\
this \,.\, abort(); \\
endSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
abortRet \,.\, LandMission \,.\, thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
cleanUpSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : \mathbb{B} \bullet \\
\left(\begin{array}{l}
cleanUpCall \,.\, LandMission \,?\, thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, LandMissionObject \,.\, thread \longrightarrow \\
ret := this \,.\, cleanUp(); \\
endSyncMeth \,.\, LandMissionObject \,.\, thread \longrightarrow \\
cleanUpRet \,.\, LandMission \,!\, thread \,!\, ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
Methods \mathrel{\widehat=} \left(\begin{array}{l}
InitializePhase \\
\Box \\
CleanupPhase \\
\Box \\
deployLandingGearMeth \\
\Box \\
stowLandingGearSyncMeth \\
\Box \\
isLandingGearDeployedSyncMeth \\
\Box \\
getControllingMissionSyncMeth \\
\Box \\
abortSyncMeth \\
\Box \\
cleanUpSyncMeth
\end{array}\right) ; \ Methods
$$

$\bullet\ (Init \,;\ Methods) \mathbin{\triangle} (end\_mission\_app \,.\, LandMission \longrightarrow \mathbf{Skip})$

end

class *LandMissionClass* $\widehat{=}$ **begin**

state *State*

$SAFE\_LANDING\_ALTITUDE : double$
$abort : \mathbb{B}$
$landingGearDeployed : \mathbb{B}$

state *State*

initial *Init*

$State~'$

$SAFE\_LANDING\_ALTITUDE' = 10.0$
$abort' = false$

public sync *stowLandingGear* $\widehat{=}$
$\left(this\ .\ landingGearDeployed := false\right)$

public sync *isLandingGearDeployed* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$
$\left(ret := landingGearDeployed = \mathbf{True}\right)$

public sync *getControllingMission* $\widehat{=}$ **var** $ret : MissionID \bullet$
$\left(ret := controllingMission\right)$

public sync *abort* $\widehat{=}$
$\left(this\ .\ abort := true\right)$

public sync *cleanUp* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$
$$\begin{pmatrix}\mathbf{Skip}; \\ ret := (\neg\ abort = \mathbf{True})\end{pmatrix}$$

$\bullet$ **Skip**

end

section *LandMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel $deployLandingGearCall : MissionID$
channel $deployLandingGearRet : MissionID$

channel $stowLandingGearCall : MissionID \times ThreadID$
channel $stowLandingGearRet : MissionID \times ThreadID$

channel $isLandingGearDeployedCall : MissionID \times ThreadID$
channel $isLandingGearDeployedRet : MissionID \times ThreadID \times \mathbb{B}$

channel $getControllingMissionCall : MissionID \times ThreadID$
channel $getControllingMissionRet : MissionID \times ThreadID \times MissionID$

channel $abortCall : MissionID \times ThreadID$
channel $abortRet : MissionID \times ThreadID$

channel $cleanUpCall : MissionID \times ThreadID$
channel $cleanUpRet : MissionID \times ThreadID \times \mathbb{B}$

5.8 Schedulables of LandMission

section *LandingGearHandlerLandApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
LandMissionMethChan, *ObjectIds*, *ThreadIds*

process *LandingGearHandlerLandApp* $\widehat{=}$
  mission : *MissionID* ● **begin**

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
handleAsyncEventCall \,.\, LandingGearHandlerLand \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
isLandingGearDeployedCall \,.\, mission \longrightarrow \\
isLandingGearDeployedRet \,.\, mission \,?\, isLandingGearDeployed \longrightarrow \\
\\
\mathbf{var}\ landingGearIsDeployed : \mathbb{B} \bullet landingGearIsDeployed := isLandingGearDeployed \\
\mathbf{if}\ landingGearIsDeployed = \mathbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l}
stowLandingGearCall \,.\, mission \longrightarrow \\
stowLandingGearRet \,.\, mission \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
[\!] \neg\ landingGearIsDeployed = \mathbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l}
deployLandingGearCall \,.\, mission \,.\, LandingGearHandlerLandThread \longrightarrow \\
deployLandingGearRet \,.\, mission \,.\, LandingGearHandlerLandThread \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
\mathbf{fi}
\end{array}\right) ; \\
handleAsyncEventRet \,.\, LandingGearHandlerLand \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\big(handlerAsyncEvent\big) \ ; \ Methods
\end{array}
$$

$\bullet\ (Methods) \mathbin{\triangle} (end\_\_app \,.\, LandingGearHandlerLand \longrightarrow \mathbf{Skip})$

end

class $LandingGearHandlerLandClass \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end

section *LandingGearHandlerLandMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *SafeLandingHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* , *LandMissionMethChan*

process *SafeLandingHandlerApp* $\widehat{=}$
    landMission : *MissionID*,
    threshold : *Double* $\bullet$ **begin**

$$
\mathit{handlerAsyncEvent} \mathrel{\widehat=}
\left(\begin{array}{l}
\mathit{handleAsyncEventCall} \,.\, \mathit{SafeLandingHandler} \longrightarrow \\
\left(\begin{array}{l}
\mathit{getControllingMissionCall} \,.\, \mathit{landMission}.\mathit{getControllingMission}() \longrightarrow \\
\mathit{getControllingMissionRet} \,.\, \mathit{landMission}.\mathit{getControllingMission}() \,?\, \mathit{getControllingMission} \longrightarrow \\
\\
\mathbf{var}\ \mathit{altitude} : \mathit{double} \bullet \mathit{altitude} := \ \mathit{getAltitude} \\
\mathbf{if}\ (\mathit{altitude} < \mathit{threshold}) \longrightarrow \\
\qquad \left(\mathbf{Skip}\right) \\
[\!]\ \neg\ (\mathit{altitude} < \mathit{threshold}) \longrightarrow \\
\qquad \left(\mathbf{Skip}\right) \\
\mathbf{fi}
\end{array}\right) ; \\
\mathit{handleAsyncEventRet} \,.\, \mathit{SafeLandingHandler} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

$$
\mathit{Methods} \mathrel{\widehat=} \\
\left(\mathit{handlerAsyncEvent}\right) \ ; \ \mathit{Methods}
$$

$\bullet\ (\mathit{Methods}) \bigtriangleup (\mathit{end\_\_app} \,.\, \mathit{SafeLandingHandler} \longrightarrow \mathbf{Skip})$

end

class *SafeLandingHandlerClass* $\widehat{=}$ **begin**

$$\begin{array}{|l}
\textbf{state}\ State \\ \hline
threshold : double \\ \hline
\end{array}$$

state *State*

$$\begin{array}{|l}
\textbf{initial}\ Init \\ \hline
State' \\ \hline
\end{array}$$

● **Skip**

end

section *SafeLandingHandlerMethChan* **parents** $scj\_prelude, GlobalTypes, MissionId, SchedulableId$

section *GroundDistanceMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
LandMissionMethChan

process *GroundDistanceMonitorApp* $\widehat{=}$
    landMission : *MissionID* ● **begin**

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
handleAsyncEventCall \,.\, GroundDistanceMonitor \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow \\
getControllingMissionRet \,.\, mission.getControllingMission() \,?\, getControllingMission \longrightarrow \\
\\
\mathbf{var}\ distance : double \bullet distance := \ getAltitude \\
\mathbf{if}\ (distance = readingOnGround) \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathbf{Skip}; \\
requestTerminationCall \,.\, mission \longrightarrow \\
requestTerminationRet \,.\, mission \,?\, requestTermination \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
[\!]\ \neg\ (distance = readingOnGround) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}\,; \\
\mathbf{Skip}
\end{array}\right); \\
handleAsyncEventRet \,.\, GroundDistanceMonitor \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\big(handlerAsyncEvent\big)\ ;\ \ Methods
\end{array}
$$

$\bullet\ (Methods) \vartriangle (end\_periodic\_app \,.\, GroundDistanceMonitor \longrightarrow \mathbf{Skip})$

end

class *GroundDistanceMonitorClass* $\widehat{=}$ **begin**

```
── state State ──────────────────────────
│ readingOnGround : double
└────────────────────────────────────────
```

state *State*

```
── initial Init ─────────────────────────
│ State′
├──────────
└────────────────────────────────────────
```

• **Skip**

end

section $\mathit{InstrumentLandingSystemMonitorApp}$ **parents** $\mathit{PeriodicEventHandlerChan}, \mathit{SchedulableId}, \mathit{SchedulableIds}$

process $\mathit{InstrumentLandingSystemMonitorApp} \mathrel{\widehat=}$
$\quad \mathit{mission} : \mathit{MissionID} \bullet$ **begin**

$$\mathit{handlerAsyncEvent} \mathrel{\widehat=}$$
$$\left(\begin{array}{l}\mathit{handleAsyncEventCall} \,.\, \mathit{InstrumentLandingSystemMonitor} \longrightarrow \\ \left(\mathbf{Skip}\right) ; \\ \mathit{handleAsyncEventRet} \,.\, \mathit{InstrumentLandingSystemMonitor} \longrightarrow \\ \mathbf{Skip}\end{array}\right)$$

$$\mathit{Methods} \mathrel{\widehat=}$$
$$\left(\mathit{handlerAsyncEvent}\right) ; \ \mathit{Methods}$$

$$\bullet\ (\mathit{Methods}) \mathbin{\triangle} (\mathit{end\_periodic\_app} \,.\, \mathit{InstrumentLandingSystemMonitor} \longrightarrow \mathbf{Skip})$$

end

class $InstrumentLandingSystemMonitorClass \mathrel{\widehat=}$ **begin**

$\bullet$ **Skip**

end