Airconditioning demo

This document contains a script of action steps for creating the UML model of the Airconditioning demo with the UML modeling tool MagicDraw. The detailed modeling recipe (refer to http://www.tismtool.com) is used as guideline for the script. Please, look at the completed UML model at that website for filling the missing steps.

N.B. Opening a Specification window of an UML elements is possible (in MagicDraw) by selecting that element, and now:

- press Enter, or
- click RMB (Right Mouse Button), and select Specification, or
- double click, unless it refers to a diagram that will then be shown.

1. **Create involved components :**
2. select the Repository package
3. via RMB, create New Element / Package, named *AircoPackage*
4. select AircoPackage
5. via RMB, create New Element / Component, named *AircoComponent*
6. select AircoComponent, and open its Specification window
7. select the value field of property Applied Stereotype, click …
8. then check the stereotype *HasThread*, click Apply, and Close
9. select the Repository package
10. via RMB, create New Element / Package, named *Terminators*
11. select Terminators
12. via RMB, create New Element / Component, named *UiComponent*
13. create 4 other terminator components: *CoolerComponent*, *FanComponent*, *NvmComponent*, *SensorComponent*

14. **Create FanProtocol:**
15. select Protocols package
16. via RMB, create New Element / Package, named *FanProtocol*
17. select FanProtocol
18. press Enter to open the Specification window of the FanProtocol
19. select the value field of property Applied Stereotype, click …
20. then check the stereotype *TismProtocol*, click Apply, and Close
21. select FanProtocol
22. via RMB, create New Diagram / Class Diagram, named *FanClasses*
23. select FanProtocol
24. via RMB, create New Element / Class, named *FanClient*
25. drag FanClient to the FanClasses diagram
26. select FanProtocol
27. via RMB, create New Element / Class, named *FanServer*
28. drag FanServer to the FanClasses diagram, below FanClient
29. select FanProtocol
30. via RMB, create New Element / Interface, named *IFan*
31. drag IFan to the FanClasses diagram between the 2 classes
32. in diagram FanClasses, select IFan
33. click the small green circle ‘Insert New Operation’
34. change the operation name ‘unnamed1’ into: *Stop*

35. unselect this operation and re-select it to open its Specification window
36. select the value field of property Type, and enter *void*
37. then click Close to close the operation window
38. repeat the above 5 steps to enter the functions: *ToLow()* and *ToHigh()*
39. in diagram FanClasses, select FanClient
40. then from the popup menu, select Port
41. in the requestor 'Select Port Type', browse to Protocols / FanProtocol / FanClient; click OK, and empty the name field (unnamed1) of the port
42. in diagram FanClasses, select the created port of FanClient
43. then from the popup menu, select *Usage*
44. move the mouse to IFan, and when the blue border appears then click the mouse
45. in diagram FanClasses, select FanServer
46. then from the popup menu, select Port
47. in the requestor 'Select Port Type', browse to Protocols / FanProtocol / FanServer; click OK, and empty the name field of the port
48. move the created port to the top of class FanServer
49. in diagram FanClasses, select the port of FanServer
50. then from the popup menu, select *Interface Realization*
51. move the mouse to IFan, and when the blue border appears then click the mouse

52. Create FanServerPsm:

53. select FanServer in the Containment pane
54. via RMB, create New Diagram / State Machine Diagram, named *FanServerPsm* (this creates both the element and the diagram)
55. click Composite State in the toolbar, and click in diagram FanServerPsm
56. enter *Busy* in the name field of the composite state
57. create sub-state *ToLow* inside the composite state Busy
58. create sub-state *ToHigh* inside the composite state Busy
59. create a transition from ToLow to ToHigh
60. open the Specification window of this transition
61. in section Trigger, select the value field of property Event Type, and from the list select *CallEvent*
62. in the Name field (of section Trigger) enter: *ToHigh()*
63. click Close
64. similarly create a transition from ToHigh to ToLow, and enter in the Name field: *ToLow()*
65. create state *Idle* (outside Busy)
66. create a transition from Idle to ToLow with CallEvent named *ToLow()*
67. create a transition from Idle to ToHigh with CallEvent named *ToHigh()*
68. create a transition from Busy to Idle with CallEvent named *Stop()*
69. select the Initial state, click in the diagram above Idle
70. create a triggerless transition from this Initial state to Idle

71. Create the conjugate state machine at FanClient:

72. select the State Machine FanServerPsm (not the State Machine Diagram FanServerPsm) in the Containment pane
73. click RMB, and select Copy
74. select the class FanClient in the Containment pane
75. click RMB, and select Paste
76. rename the state machine from FanServerPsm into *FanClientPsm*

77. select the transition with trigger Stop(), and open its Specification window
78. in section Trigger, select the value field of property Event Type, and from the list select *<UNSPECIFIED>*
79. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
80. in section Effect, select the value field of property Name, and enter the source code: *Stop();*
81. in section Transition, select the value field of property Applied Stereotype, and click …
82. then check the stereotype *NativeTrigger*, click Apply, and Close
83. repeat the above 6 steps for the other four transitions

84. Create an enumeration for the fan levels:

85. select package TypeDefinitions
86. via RMB, create New Element / Enumeration, named *FanLevel*
87. select FanLevel, and open its Specification window
88. select Enumeration Literals
89. click Create
90. in the Enumeration Literal window, change name ‘unnamed1’ into: *LOW*
91. click Close
92. again click Create
93. in the Enumeration Literal window, change name ‘unnamed1’ into: *HIGH*
94. click Close, click Close

95. Create NvmProtocol:

96. Create (analogous to FanProtocol) the package *NvmProtocol*, the class diagram *NvmClasses*, the classes *NvmClient* and *NvmServer*, their ports, the interface *INvm* and the links to the ports
97. in diagram NvmClasses, select INvm
98. click the small green circle ‘Insert New Operation’
99. change the operation name ‘unnamed1’ into: *GetTarget*
100. unselect this operation and re-select it to open its Specification window
101. select the value field of property Type, and enter *int*
102. click Close
103. again click the small green circle ‘Insert New Operation’
104. change the operation name ‘unnamed1’ into: *SetTarget*
105. unselect this operation and re-select it to open its Specification window
106. select the value field of property Type, and enter *void*
107. in the left pane of the Operation window, select *Parameters*
108. in the right pane, becoming the Parameters pane: click Create
109. in the Parameter window, change name ‘unnamed1’ into: *Temperature*
110. select the value field of property Type, and enter *int*
111. click Close
112. again click the small green circle ‘Insert New Operation’
113. change the operation name ‘unnamed1’ into: *GetFanLevel*
114. unselect this operation and re-select it to open its Specification window
115. select the value field of property Type, and click …
116. select TypeDefinitions / FanLevel, and click OK
117. click Close, click Close
118. again click the small green circle ‘Insert New Operation’

119. change the operation name ‘unnamed1’ into: *SetFanLevel*
120. unselect this operation and re-select it to open its Specification window
121. select the value field of property Type, and enter *void*
122. in the left pane of the Operation window, select *Parameters*
123. the right pane becomes the Parameters pane; click Create
124. in the Parameter window, change name ‘unnamed1’ into: *Level*
125. select the value field of property Type, and click …
126. select TypeDefinitions / FanLevel, and click OK
127. click Close, click Close

128. Create CoolerProtocol (not elaborated)

129. Create SensorProtocol:

130. Create the package *SensorProtocol*, the class diagram *SensorClasses*, the classes *SensorClient* and *SensorServer*, their ports, the *interface* ISensor and the links to the ports
131. At ISensor, create the Operations:
    a. *GetTemperature()* : *double*
    b. *SignalAbove()* : *void*
    c. *SignalBelow()* : *void*
    d. *Stop()* : *void*
132. in diagram SensorClasses, create a second interface named *ICbSensor*
133. in diagram SensorClasses, select ICbSensor
134. then from the popup menu, select *Usage*
135. move the mouse to the port of the SensorServer, and when the blue border appears then click the mouse
136. in diagram SensorClasses, select ICbSensor
137. then from the popup menu, select *Interface Realization*
138. move the mouse to the port of the SensorClient, and when the blue border appears then click the mouse
139. in diagram SensorClasses, select ICbSensor
140. then click the small green circle ‘Insert New Operation’
141. change the operation name ‘unnamed1’ into: *CbAbove*
142. unselect this operation and re-select it to open its Specification window
143. select the value field of property Type, and enter *void*
144. then click Close to close the operation window
145. repeat the above 5 steps to enter the callback functions: *CbBelow()* and *CbStopped()*

146. Create SensorServerPsm:

147. select SensorServer in the Containment pane
148. via RMB, create New Diagram / State Machine Diagram, named *SensorServerPsm* (this creates both the element and the diagram)
149. from the toolbar, select Composite State / Orthogonal State, click in the diagram, and name this composite state: *Outer*
150. in the lower region, create a State named *Get*
151. select a Choice state from the toolbar, and click in the region below state Get
152. create a transition from state Get to the Choice state with CallEvent named *GetTemperature()*
153. create a transition from the Choice to state Get
154. select this transition, and open its Specification window

155. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
156. in section Effect, select the value field of property Name, and enter the source code: *return 16.0;*
157. select the value field of property Guard, click …
158. then select *Constraint*
159. in the opened Constraint window, clear the Name field
160. select the value field of property Specification
161. enter the text for the informal guard between quotes: *"cold"*
162. click Close, and Close
163. create another transition from the Choice to state Get
164. select this transition, and open its Specification window
165. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
166. in section Effect, select the value field of property Name, and enter the source code: *return 24.0;*
167. select the value field of property Guard, click …
168. then select *Constraint*
169. in the opened Constraint window, clear the Name field
170. select the value field of property Specification
171. enter the text for the informal guard (without quotes): *else*
172. click Close, and Close
173. in the upper region, create state *Stopping* and above it create a (normal) Composite State named *Inner* with 3 sub-states *Idle*, *SenseAbove*, *SenseBelow*
174. create a transition from Idle to SenseAbove with trigger *SignalAbove(Target)*
175. create a transition from Idle to SenseBelow with trigger *SignalBelow(Target)*
176. create a transition from Inner to Stopping with trigger *Stop()*
177. create 4 initial states with a transition to *Outer*, *Inner*, *Idle*, and *Get*
178. create a transition from state SenseAbove to Idle
179. open the Specification window of this transition
180. select the value field of property Applied Stereotype, click …
181. then check the stereotype *NativeTrigger*, click Apply
182. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
183. in section Effect, select the value field of property Name, and enter the source code: *CbAbove ();*
184. click Close
185. create a transition from state SenseBelow to Idle, and repeat the above 6 steps with *CbBelow()*
186. create a transition from state Stopping to Inner, and repeat those above 6 steps with *CbStopped()*

187. Create SensorClientPsm:

188. select the State Machine SensorServerPsm (not the State Machine Diagram SensorServerPsm) in the Containment pane
189. click RMB, and select Copy
190. select the class SensorClient in the Containment pane
191. click RMB, and select Paste
192. rename the state machine from SensorServerPsm into *SensorClientPsm*
193. double click SensorClientPsm to edit the State Machine Diagram SensorClientPsm

194. select the Outer composite state
195. via RMB, select Remove Region / Region 2
196. select the Outer composite state to open its Specification window
197. in the left pane, select *Internal Transitions*, and then in the right pane click Create
198. in section Transition, select the value field of property Applied Stereotype, and click …
199. then check the stereotype *NativeTrigger*, click Apply, and Close
200. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
201. in section Effect, select the value field of property Name, and enter the source code: *GetTemperature();*
202. click Close, and Close
203. change the transitions with the triggers SignalAbove(Target), SignalBelow(Target), and Stop() to their conjugate counterparts
204. change the internal transition with the trigger GetTemperature () to its conjugate counterpart
205. Because of calling the body fragments SignalAbove(Target) and SignalBelow(Target) at verification/simulation, a local variable *Target* must be defined
206. select class SensorClient in the Containment pane, and open its Specification window
207. in the left pane, select Attributes, and then in the right pane click Create
208. in the opened window, enter the Name *Target*, the Type *int*, and the Default Value *21*
209. click Close, and Close
210. select the transition from state SenseAbove to Idle, and open its Specification window
211. select the value field of property Applied Stereotype, click …
212. then uncheck the stereotype *NativeTrigger*, click Apply
213. in section Effect, select the value field of property Behavior Type, and from the list select *<UNSPECIFIED>*
214. in section Trigger, select the value field of property Event Type, and from the list select *SignalEvent*
215. in the Name field (of section Trigger) enter: *CbAbove()*
216. click Close
217. repeat the above 7 steps for the transition from state SenseBelow to Idle resulting in trigger *CbBelow(),* and for the transition from state Stopping to Inner resulting in trigger *CbStopped()*
218. select state Stopping, and from the popup menu, select *Transition to Self*
219. open the Specification window of the self-transition
220. select the value field of property Applied Stereotype, click …
221. then check the stereotype *Ignore*, click Apply
222. in section Trigger, select the value field of property Event Type, and from the list select *SignalEvent*
223. in the Name field (of section Trigger) enter: *CbAbove()*
224. repeat the above 6 steps for the self-transition with trigger *CbBelow()*

225. Create AircoProtocol:

226. Create the package AircoProtocol, the class diagram AircoClasses, the classes AircoClient and AircoServer, their ports, the interface IAirco and the links to the ports

227. At IAirco, create the Operations:
    - *On()* : *FanLevel*
    - *Off()* : *void*
    - *GetLevel() : FanLevel*
    - *GetTarget()* : *int*
    - *SetTarget(Target : int)* : *void*
    - *Start()* : *void*
    - *Stop()* : *void*
    - *FanLow()* : *void*
    - *FanHigh()* : *void*
228. select class AircoClient in the Containment pane, and open its Specification window
229. in the left pane, select Attributes, and then in the right pane click Create
230. in the opened window, enter the Name *Level* and the Type *FanLevel*
231. click Close
232. again click Create
233. in the opened window, enter the Name *Target*, the Type *int*, and the Default Value *20*
234. click Close, and Close
235. select class AircoServer in the Containment pane, and open its Specification window
236. in the left pane, select Attributes, and then in the right pane click Create
237. in the opened window, enter the Name *MyTarget* , the Type *int*, and the Default Value *21*
238. click Close, and Close

239. Create *AircoServerPsm* (described partly):
240. select AircoServer in the Containment pane
241. via RMB, create New Diagram / State Machine Diagram, named *AircoServerPsm* (this creates both the element and the diagram)
242. from the toolbar, select Composite State / Orthogonal State, and click in the diagram
243. open the Specification window of this state, and name it: Active
244. in the left pane, select Inner Elements, expand it by clicking +
245. select the upper element, and in the right pane, name this region: *Cooler*
246. also, select the lower element, and in the right pane, name this region: *Fan*
247. click Close
248. the region names become visible in Active, after clicking RMB and selecting option: *Show Region Name*
249. select a Choice state from the toolbar, and click in region Fan
250. create state *Low* in region Fan
251. create a transition from the Choice state to state Low
252. select this transition, and open its Specification window
253. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
254. in section Effect, select the value field of property Name, and enter the source code: *return FanLevel.LOW;*
255. select the value field of property *Guard*, click …
256. then select *Constraint*
257. in the opened Constraint window, clear the Name field
258. select the value field of property *Specification*
259. enter the text for the informal guard between quotes: *"low level"*
260. click Close, and Close

261. create state *High* in region Fan
262. create a transition from the Choice state to state High
263. select this transition, and open its Specification window
264. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
265. in section Effect, select the value field of property Name, and enter the source code: *return FanLevel.HIGH;*
266. select the value field of property *Guard*, click …
267. then select *Constraint*
268. in the opened Constraint window, clear the Name field
269. select the value field of property *Specification*
270. enter the text for the guard: *else*
271. click Close, and Close

272. Create AircoClientPsm:

273. Copy the AircoServerPsm to a *AircoClientPsm* in the same way as was done for SensorClientPsm
274. select state Outer, and open its Specification window
275. in the left pane, select *Internal Transitions*, and then in the *General* section of the right pane click the icon of the (unnamed) top row
276. now edit the transition (add stereotype *NativeTrigger*, delete *CallEvent*, and add *OpaqueBehavior*)
277. close the Specification window
278. select the transition from the Choice state to state Low, and open its Specification window
279. in section Effect, select the value field of property *Behavior Type*, and from the list select *<UNSPECIFIED>*
280. in the left pane, select Inner Elements, and then in the *Constraints* section of the right pane click the icon of the top row
281. select the value field of property *Specification*, and change the guard into: *Level==FanLevel.LOW*
282. click Close, and Close
283. also, select the transition from the Choice state to state High, and open its Specification window
284. in section Effect, select the value field of property *Behavior Type*, and from the list select *<UNSPECIFIED>*
285. click Close

286. **Create the Context :**
287. select the Architecture package
288. via RMB,create New Element / Execution Environment, named *ExecEnv*
289. select the ExecEnv Execution Environment
290. via RMB, create New Diagram / Composite Structure Diagram, named *Context*
291. select the AircoComponent in the Repository, drag it to the Context diagram, and name this instance: *Airco*
292. select the UiComponent in the Repository, drag it to the Context diagram above Airco, and name this instance: *Ui*
293. also drag the 4 other terminator components to the Context diagram below Airco, and name them: *Cooler*, *Fan*, *Nvm*, *Sensor*
294. select Airco

295. then from the popup menu, select *Port*
296. in the requestor ‘Select Port Type’, browse to Protocols / AircoProtocol / AircoServer; click OK, and rename the provided port to: *d_pp_airco*
297. Repeat the above 3 steps for the other 4 protocols, selecting the classes with the client role, to create required ports: *d_rp_cooler*, *d_rp_fan*, *d_rp_nvm* and *d_rp_sensor*
298. select Ui
299. then from the popup menu, select *Port*
300. in the requestor ‘Select Port Type’, browse to Protocols / AircoProtocol / AircoClient; click OK, and rename the port to: *rp_airco*
301. select port *rp_airco*
302. then from the popup menu, select *Connector*
303. move the mouse to the port *d_pp_airco* of Airco, and click when *d_pp_airco* becomes blue
304. Repeat the above 6 steps for the component instances: Cooler, Fan, Nvm and Sensor to create and connect ports *pp_cooler*, *pp_fan*, *pp_nvm* and *pp_sensor*
305. select Ui, and open its Specification window
306. select the value field of property Applied Stereotype, click …
307. then check the stereotype *Mock*, click Apply, and Close
308. Repeat the above 3 steps for the component instances: Cooler, Fan, Nvm and Sensor to create mocks

309. Create the Airco structure:

310. select AircoComponent
311. via RMB, create New Element / Class, named *AircoClass*
312. select AircoComponent
313. via RMB, create New Diagram / Composite Structure Diagram, named *AircoStructure*
314. select AircoClass, drag it inside the AircoStructure diagram, and name it: *AircoInstance*
315. in diagram AircoStructure, select AircoInstance
316. then from the popup menu, select *Port*
317. in the requestor ‘Select Port Type’, browse to Protocols / AircoProtocol / AircoServer; click OK, and rename the port to: *pp_airco*
318. select port *pp_airco*, and from the popup menu, select *Connector*
319. then move the mouse to the port *d_pp_airco*, and click when it becomes blue
320. Repeat the above 5 steps for the ports: *rp_cooler* of type CoolerClient, *rp_fan* of type FanClient, *rp_nvm* of type NvmClient, *rp_sensor* of type SensorClient

321. Create sequence diagrams:

322. select Architecture / Sequence Diagrams
323. via RMB, create New Diagram / Sequence Diagram, named *Start*
324. select Architecture / ExecEnv / Ui, and drag it to the sequence diagram
325. select Ui, and via RMB, disable *Show Classifier* to reduce the box size
326. repeat this for Airco, Cooler, Fan, Nvm, Sensor
327. from the toolbar, select *Call Message*
328. click the lifeline of Ui, and then the lifeline of Airco
329. open the Specification window of the message
330. select the value field of property Operation, and then in the dropdown box select *On()*

331. etc

332. Create AircoFsm (described partly):
333. select AircoServerPsm in the Containment pane
334. click RMB, and select Copy
335. select the class AircoClass in the Containment pane
336. click RMB, and select Paste
337. rename the pasted state machine of AircoClass into: *AircoFsm*
338. double click AircoFsm to start editing its diagram
339. All function calls acting as trigger must be preceded by the name of the provided port implementing the AircoProtocol. For all involved transitions, select the value field of property *Name* in section *Trigger*, and insert: *pp_airco->*
340. For most triggers the body must be filled in with activities for the controlled components.
341. Since Airco has a thread, all call events must be concluded with a transition having stereotype *SyncReturn*.
342. select the transition from state ReportLevel to the Choice state, and open its Specification window
343. as already mentioned in general at step 339:in section *Trigger*, select the value field of property *Name*, and change the trigger into: *pp_airco->GetLevel()*
344. in section Effect, select the value field of property Behavior Type, and from the list select *OpaqueBehavior*
345. in section Effect, select the value field of property Name, and enter the source code: *Level=rp_nvm->GetFanLevel();*
346. select the transition from the Choice state to state Low, and open its Specification window
347. select the value field of property Applied Stereotype, click …
348. then check the stereotype *SyncReturn*, click Apply
349. in the left pane, select Inner Elements, and then in the *Constraints* section of the right pane click the icon of the top row
350. select the value field of property *Specification*, and change the guard into: *Level==FanLevel.LOW*
351. click Close, and Close
352. add local variable *Level* of type *FanLevel* to the AircoClass
353. add local variable *MyTarget* of type *int* to the AircoClass
354. add local variable *MyTemp* of type *double* to the AircoClass
355. select AircoFsm in the Containment pane
356. via RMB, create New Element / State Machine, named *CoolingSubFsm*
357. select CoolingSubFsm
358. via RMB, create New Diagram / State Machine Diagram, named *CoolingSubFsm*
359. double click CoolingSubFsm to edit it
360. create a Composite State named *Composite*
361. add Entry action: *MyTarget=rp_nvm->GetTarget();*
362. add Exit action: *rp_nvm->SetTarget(MyTarget);*
363. create an initial state, a junction state, a choice state, and 2 Composite States named *Sensing* and *Cooling*
364. at the transition between the junction state and the choice state add Effect: *MyTemp=rp_sensor->GetTemperature();*
365. select AircoFsm in the Containment pane
366. via RMB, create New Element / State Machine, named *StoppingSubFsm*

367. select StoppingSubFsm
368. via RMB, create New Diagram / State Machine Diagram, named *StoppingSubFsm*
369. edit the StoppingSubFsm (not elaborated)
370. activate the CoolingSubFsm diagram
371. from the toolbar, select Submachine state, and click below state Composite
372. in the requestor 'Select Submachine', browse to Repository / AircoComponent / AircoClass / AircoFsm / StoppingSubFsm; click OK, and rename the state to: *OffStopping*
373. repeat the above 2 steps to create a SubMachine named *Stopping*
374. from the toolbar, select *Exit Point*
375. click the mouse inside the CoolingSubFsm diagram below OffStopping, and name the exitpoint : *ExitOff*
376. create a transition from OffStopping to ExitOff
377. from the toolbar, select *Exit Point*
378. click the mouse inside the CoolingSubFsm diagram below Stopping, and name the exitpoint : *ExitStop*
379. create a transition from Stopping to ExitStop
380. activate the AircoFsm diagram
381. from the toolbar, select Submachine state, and click in region Active / Cooler
382. in the requestor 'Select Submachine', browse to Repository / AircoComponent / AircoClass / AircoFsm / CoolingSubFsm; click OK, and rename the state to: *Cooling*
383. from the toolbar, select Connection Point Reference, and click in state *Cooling*
384. in the requestor 'Select Entry/Exit Point', select *ExitStop*, and click OK
385. create a triggerless transition from this Exit Point Reference *ExitStop* to state Idle
386. create a Final State in region Active / Cooler
387. from the toolbar, select Connection Point Reference, and click in state *Cooling*
388. in the requestor 'Select Entry/Exit Point', select *ExitOff*, and click OK
389. create a triggerless transition from this Exit Point Reference *ExitOff* to the final state
390. etc.