# Coverage Report

**Created: 2025-10-02 17:52**

/repo/SafetySystms/src/SafetyRules/include/SafetyRules/SafetyRules.h

| Line | Count | Source (jump to first uncovered line) |
|---|---|---|
| 1 | | #pragma once |
| 2 | | #include "SafetyRules/ISafetyRules.h" |
| 3 | | #include <cassert> |
| 4 | | |
| 5 | | namespace safety |
| 6 | | { |
| 7 | | |
| 8 | | using Event = ISafetyRules::Event; |
| 9 | | using TopState = ISafetyRules::State; |
| 10 | | using LoaderSubstate = ISafetyRules::LoaderSub; |
| 11 | | |
| 12 | | class SafetyRules final : public ISafetyRules |
| 13 | | { |
| 14 | | public: |
| 15 | | // ----- Construction |
| 16 | | SafetyRules() |
| 17 | 46 | { |
| 18 | 46 | reset(); |
| 19 | 46 | } |
| 20 | | |
| 21 | | // ----- ISafetyRules (control) |
| 22 | | void reset() override |
| 23 | 92 | { |
| 24 | 92 | current = State::Idle; |
| 25 | 92 | loader = LoaderSub::None; |
| 26 | | |
| 27 | 92 | if (onEnterIdle) |
| | | Branch (27:19): [True: 45, False: 47] |
| 28 | 45 | { |
| 29 | 45 | onEnterIdle(); |
| 30 | 45 | } |
| 31 | 92 | } |
| 32 | | |
| 33 | | void dispatch(Event ev) override |
| 34 | 133 | { |
| 35 | 133 | switch (current) |
| | | Branch (35:23): [True: 0, False: 133] |
| 36 | 133 | { |
| 37 | 50 | case State::Idle: |
| | | Branch (37:19): [True: 50, False: 83] |
| 38 | 50 | { |
| 39 | 50 | if (ev == Event::evPowerOn) |
| | | Branch (39:27): [True: 41, False: 9] |
| 40 | 41 | { |
| 41 | 41 | transitionTo(State::Active); |
| 42 | 41 | } |

```
43
44  50                  break;
45   0              }
46
47  20              case State::Active:
        Branch (47:19): [True: 20, False: 113]
48  20              {
49  20                  if (ev == Event::evPowerOff)
        Branch (49:27): [True: 5, False: 15]
50   5                  {
51   5                      transitionTo(State::Idle);
52   5                  }
53
54  15                  else if (ev == Event::evFault)
        Branch (54:32): [True: 8, False: 7]
55   8                  {
56   8                      transitionTo(State::Faulted);
57   8                  }
58
59  20                  break;
60   0              }
61
62  10              case State::Faulted:
        Branch (62:19): [True: 10, False: 123]
63  10              {
64  10                  if (ev == Event::evPowerOn)
        Branch (64:27): [True: 2, False: 8]
65   2                  {
66   2                      transitionTo(State::Active);
67   2                  }
68
69  10                  break;
70   0              }
71
72  53              case State::BuildPlateLoader:
        Branch (72:19): [True: 53, False: 80]
73  53              {
74                      // Fault escape from any substate
75  53                  if (ev == Event::evFault)
        Branch (75:27): [True: 3, False: 50]
76   3                  {
77   3                      exitLoaderSubmachine();
78   3                      transitionTo(State::Faulted);
79   3                      break;
80   3                  }
81
82  50                  switch (loader)
83  50                  {
84  21                      case LoaderSub::OpenDoor:
        Branch (84:27): [True: 21, False: 29]
85  21                      {
```

```
86  21                          if (ev == Event::evDoorOpened)
        Branch (86:35): [True: 16, False: 5]
87  16                          {
88  16                              enterLoaderSub(LoaderSub::DoorOpened);
89  16                          }
90
91  21                          break;
92   0                      }
93
94  16                      case LoaderSub::DoorOpened:
        Branch (94:27): [True: 16, False: 34]
95  16                      {
96  16                          if (ev == Event::evBuildPlateLoaded)
        Branch (96:35): [True: 13, False: 3]
97  13                          {
98  13                              enterLoaderSub(LoaderSub::BuildPlateLoaded);
99  13                          }
100
101 16                          break;
102  0                      }
103
104 13                      case LoaderSub::BuildPlateLoaded:
        Branch (104:27): [True: 13, False: 37]
105 13                      {
106 13                          if (ev == Event::evDoorClosed)
        Branch (106:35): [True: 9, False: 4]
107  9                          {
108                             // Completion of submachine -> Active
109  9                          exitLoaderSubmachine();
110  9                          transitionTo(State::Active);
111  9                          }
112
113 13                          break;
114  0                      }
115
116  0                      case LoaderSub::None:
        Branch (116:27): [True: 0, False: 50]
117  0                      default:
        Branch (117:27): [True: 0, False: 50]
118  0                      {
119  0                          assert(false && "Invalid loader substate in BuildPlateLoader");
120  0                          break;
121  0                      }
122 50                  }
123
124 50                  break;
125 50              }
126 133         }
127 133     }
128
129         void startLoader() override
130 26      {
```

```
131  26             if (current != State::Active)
     Branch (131:19): [True: 3, False: 23]
132   3             {
133   3                 return; // ignore unless in Active
134   3             }
135
136  23             transitionTo(State::BuildPlateLoader);
137
138                 // Submachine initial: [*] -> OpenDoor
139  23             enterLoaderSub(LoaderSub::OpenDoor);
140  23         }
141
142             // ----- ISafetyRules (observability)
143             State getState() const override
144  78         {
145  78             return current;
146  78         }
147
148             LoaderSub getLoaderSubstate() const override
149  43         {
150  43             return loader;
151  43         }
152
153             // ----- ISafetyRules (callback setters)
154  52         void setOnEnterIdle(VoidFn cb) override                { onEnterIdle = std::move(cb); }
155  51         void setOnExitIdle(VoidFn cb) override                 { onExitIdle = std::move(cb); }
156
157  52         void setOnEnterActive(VoidFn cb) override              { onEnterActive = std::move(cb); }
158  55         void setOnExitActive(VoidFn cb) override               { onExitActive = std::move(cb); }
159
160  54         void setOnEnterFaulted(VoidFn cb) override             { onEnterFaulted = std::move(cb); }
161  51         void setOnExitFaulted(VoidFn cb) override              { onExitFaulted = std::move(cb); }
162
163  54         void setOnEnterBuildPlateLoader(VoidFn cb) override    { onEnterBuildPlateLoader = std::move(cb); }
164  54         void setOnExitBuildPlateLoader(VoidFn cb) override     { onExitBuildPlateLoader = std::move(cb); }
165
166  53         void setOnRequestDoorOpen(VoidFn cb) override          { onRequestDoorOpen = std::move(cb); }
167  53         void setOnRequestLoadBuildPlate(VoidFn cb) override    { onRequestLoadBuildPlate = std::move(cb); }
168  53         void setOnRequestDoorClose(VoidFn cb) override         { onRequestDoorClose = std::move(cb); }
169
170         private:
171             // ----- Top-level transitions with entry/exit hooks
172             void transitionTo(State next)
173  91         {
174  91             if (next == current)
     Branch (174:19): [True: 0, False: 91]
175   0             {
176   0                 return;
177   0             }
178
179                 // Exit old top-level state
180  91             switch (current)
     Branch (180:23): [True: 0, False: 91]
181  91             {
```

```
182  41                 case State::Idle:
     Branch (182:19): [True: 41, False: 50]
183  41                 {
184  41                     if (onExitIdle) onExitIdle();
     Branch (184:27): [True: 37, False: 4]
185  41                     break;
186   0                 }
187
188  36                 case State::Active:
     Branch (188:19): [True: 36, False: 55]
189  36                 {
190  36                     if (onExitActive) onExitActive();
     Branch (190:27): [True: 28, False: 8]
191  36                     break;
192   0                 }
193
194   2                 case State::Faulted:
     Branch (194:19): [True: 2, False: 89]
195   2                 {
196   2                     if (onExitFaulted) onExitFaulted();
     Branch (196:27): [True: 1, False: 1]
197   2                     break;
198   0                 }
199
200  12                 case State::BuildPlateLoader:
     Branch (200:19): [True: 12, False: 79]
201  12                 {
202  12                     if (onExitBuildPlateLoader) onExitBuildPlateLoader();
     Branch (202:27): [True: 8, False: 4]
203  12                     break;
204   0                 }
205  91             }
206
207  91             current = next;
208
209                 // Enter new top-level state
210  91             switch (current)
     Branch (210:23): [True: 0, False: 91]
211  91             {
212   5                 case State::Idle:
     Branch (212:19): [True: 5, False: 86]
213   5                 {
214   5                     if (onEnterIdle) onEnterIdle();
     Branch (214:27): [True: 1, False: 4]
215   5                     break;
216   0                 }
217
218  52                 case State::Active:
     Branch (218:19): [True: 52, False: 39]
219  52                 {
```

```
220  52                      if (onEnterActive) onEnterActive();
     Branch (220:27): [True: 47, False: 5]
221  52                      break;
222   0                  }
223
224  11                  case State::Faulted:
     Branch (224:19): [True: 11, False: 80]
225  11                  {
226  11                      if (onEnterFaulted) onEnterFaulted();
     Branch (226:27): [True: 7, False: 4]
227  11                      break;
228   0                  }
229
230  23                  case State::BuildPlateLoader:
     Branch (230:19): [True: 23, False: 68]
231  23                  {
232  23                      if (onEnterBuildPlateLoader) onEnterBuildPlateLoader();
     Branch (232:27): [True: 18, False: 5]
233  23                      break;
234   0                  }
235  91              }
236  91          }
237
238              // ----- Submachine helpers
239              void enterLoaderSub(LoaderSub sub)
240  52          {
241  52              loader = sub;
242
243  52              switch (loader)
244  52              {
245  23                  case LoaderSub::OpenDoor:
     Branch (245:19): [True: 23, False: 29]
246  23                  {
247  23                      if (onRequestDoorOpen) onRequestDoorOpen(); // entry action
     Branch (247:27): [True: 16, False: 7]
248  23                      break;
249   0                  }
250
251  16                  case LoaderSub::DoorOpened:
     Branch (251:19): [True: 16, False: 36]
252  16                  {
253  16                      if (onRequestLoadBuildPlate) onRequestLoadBuildPlate(); // entry action
     Branch (253:27): [True: 9, False: 7]
254  16                      break;
255   0                  }
256
257  13                  case LoaderSub::BuildPlateLoaded:
     Branch (257:19): [True: 13, False: 39]
258  13                  {
259  13                      if (onRequestDoorClose) onRequestDoorClose(); // entry action
     Branch (259:27): [True: 7, False: 6]
260  13                      break;
```

```
261   0                 }
262
263                     // Dead Code
264   0                 case LoaderSub::None:
      Branch (264:19): [True: 0, False: 52]
265   0                 default:
      Branch (265:19): [True: 0, False: 52]
266   0                     break;
267  52             }
268  52         }
269
270             void exitLoaderSubmachine()
271  12         {
272  12             loader = LoaderSub::None;
273  12         }
274
275         private:
276             // ----- Data
277             State     current { State::Idle };
278             LoaderSub loader  { LoaderSub::None };
279
280             // Entry/exit hooks
281             VoidFn onEnterIdle;
282             VoidFn onExitIdle;
283
284             VoidFn onEnterActive;
285             VoidFn onExitActive;
286
287             VoidFn onEnterFaulted;
288             VoidFn onExitFaulted;
289
290             VoidFn onEnterBuildPlateLoader;
291             VoidFn onExitBuildPlateLoader;
292
293             // Substate entry actions
294             VoidFn onRequestDoorOpen;
295             VoidFn onRequestLoadBuildPlate;
296             VoidFn onRequestDoorClose;
297     };
298
299 } // namespace safety
```