Test report

for

hu.bme.mit.swsv.ris.tsm.impl.SafetyLogicImpl.makeLocalDecision() method

The following test cases were constructed:

1. testNoStatusChangeSwitching: in this test the behavior of the method is tested, when occupancy change is reported, but the occupancy does not change its value. The SafetyLogicImpl should not send out control message (should not call SignalMapper.sendControl() method).

2. testNoDisabledSectionSwitching: in this test case a train goes from the facing side to the divergent, after it passes away the divergent section the turnout changes to straight direction and an another train enters the straight section. All sections should be enabled all the time.

3. testTurnoutSwitching: in this test case the switching of turnout is tested. First all sections are free, they should be enabled. Then facing and straight sections become occupied, they should be disabled, but divergent enabled. Then the turnout changes to divergent direction, so facing section is now enabled (but straight is still disabled). Turnout direction again changes, so again facing and straight sections should be disabled. Finally facing section become free and all three section should be enabled.

4. testOccupiedSectionsLinkage: in this test case the behavior of the local decision is tested, when facing section is occupied, and also one of the other sections is occupied. First the occupied sections are linked with the turnout, so both section should be disabled. Finally, the turnout changes direction and the facing section should be enabled (but the other disabled should remain disabled).

5. testSwitchTrailing: in this test case both straight and divergent sections are occupied and the turnout switches direction from divergent to straight. So first the straight section should be disabled, the others enabled, then the divergent should be disabled, while straight enabled.

After implementing the test cases, 100% code coverage on the makeLocalDecision() method is achieved:

SafetyLogicImpl

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
makeDecision()		93%		88%	2	9	0	12	0	1
static {...}		75%		50%	1	2	0	2	0	1
makeLocalDecision()		100%		92%	2	14	0	19	0	1
getOccupiedFacingDistributedDecision()		100%		100%	0	6	0	9	0	1
neighborStatusChanged(Side, Date, NeighborTSMStatus)		100%		50%	1	2	0	7	0	1
getOccupiedNonFacingDistributedDecision(Side)		100%		100%	0	4	0	6	0	1
combineDecisions(SideTriple, SideTriple)		100%		n/a	0	1	0	4	0	1
sectionOccupancyChanged(Side, SectionOccupancy)		100%		100%	0	2	0	6	0	1
makeDistributedDecision()		100%		n/a	0	1	0	7	0	1
turnoutDirectionChanged(Direction)		100%		100%	0	2	0	5	0	1
directionAndSideEquals(Direction, Side)		100%		100%	0	5	0	1	0	1
SafetyLogicImpl(SideTriple, Direction, SideTriple, LoggerWrapper)		100%		n/a	0	1	0	6	0	1
getNonFacingDistributedDecision(Side)		100%		100%	0	2	0	3	0	1
getFacingDistributedDecision()		100%		100%	0	2	0	3	0	1
join(SectionControl, SectionControl)		100%		100%	0	3	0	3	0	1
setSignalMapper(SignalMapper)		100%		n/a	0	1	0	2	0	1
Total	6 of 413	99%	6 of 82	93%	6	57	0	95	0	16

Created with JaCoCo 0.7.7.201

The fact of "Missed branches" coverage being below 100% is considered a false alarm. These cases are related to else branches, where the previous if condition covers some combinations of the condition expression.