Subject: [railML3] Handling changes between minor versions
Posted by Thomas Nygreen on Tue, 16 Jun 2020 08:20:08 GMT
View Forum Message <> Reply to Message

Dear community!

As announced in this month's news article on railML.org the coordinators have been discussing compatibility between minor versions of railML 3.x. On the one hand, full compatibility results in very little flexibility to correct mistakes or improve concepts that have already been modelled in railML. On the other hand, full flexibility to make changes means we cannot guarantee compatibility between minor versions. In railML 2.x the policy has been and will continue to be that we allow new elements and attributes, but keep downward compatibility by deprecating unwanted elements and attributes instead of removing them.

railML 3.x is even more comprehensive than, and has been completely remodeled from, railML 2.x. We can expect the increased practical application of railML 3.x to result in proposals for changes that will further improve the standard. That leaves us with the question of how we should balance flexibility to improve versus the need for stability and compatibility. The coordinators are proposing the following three alternatives:

Follow the path of railML 2.x in railML 3.x: downward compatibility will be guaranteed. Changes may lead to elements or attributes being deprecated, but not removed.
Changes may lead to elements or attributes being deprecated. Elements and attributes that are deprecated in one minor version will be removed in the following minor version. Compatibility is only guaranteed between one minor version and the next.
Changes may lead to elements or attributes being removed in a new minor version, without first being deprecated. No compatibility guaranteed.

Please post your feedback here before August 31st to allow us to include it in the development of railML 3.2, scheduled for beta release around the end of 2020.