Subject: [railML3] Strategy of documentation at wiki.railML.org
Posted by Ferri Leberl on Tue, 06 Sep 2016 11:08:20 GMT
View Forum Message <> Reply to Message

Dear Everybody,

As the changes between versions 2.x and 3.x are considered to be big, DI Kolmorgen asked me to present some ideas on how we should deal with these differences in the documentation.

Essentially, there are two poles -- explaining the different versions within the same, existing wiki versus writing a new wiki from scratch -- and there might be some compromise by starting a new wiki, but importing certain existing content.

Besides this question, we ask you for ideas on how to standardize the documentation, and how to integrate UML or XML definitions into wiki pages.

I have enclosed a draft of the alternatives and their pros and cons. It is written in German. If it is required we will make up an English translation.

Thank you in advance for your interest and your ideas.

Yours,
Mag. Ferri Leberl

File Attachments
1) impuls.pdf, downloaded 276 times

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by on Thu, 15 Sep 2016 11:17:25 GMT
View Forum Message <> Reply to Message

Dear Ferri Leberl,

thank you for your Diskussionsanstoß f•or documentation of railML 3
from my side. I can follow your problem description and suggestions.

I personally would prefer solution 3 "Recycling" but I am not sure
whether this may mean an effort still too high. I possibly would follow
this solution if there is a possibility to "recycle step by step",
leaving a current topic on-line until it is recycled. All existing
information should always stay available, better in an old contents
(railML 2.x) than not at all.

Concerning "Schemeneinbindung", I understand the problem but do not have
enough experience to suggest anything. I can only offer some possible
help by programming a kind of tool which parses and converts xsd files
into a file format which can possibly better be handled by existing

tools. The question would be whether this solution would have any advantages over "Einbindung von HTML" - probably not. Anyway, at least it could be a solution for "Entfernung des Kopfes und Umwandlung von Links". Please do not hesitate to contact me if you feel that I can help in this direction.

With best regards,
Dirk Bräuer.

P.S.: Please do not MIME-encode attachments in forum posts. There are news readers (as mine) which do not decode them and present them in-line. Better provide a download link.

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by Bob Janssen on Thu, 15 Sep 2016 13:59:48 GMT
View Forum Message <> Reply to Message

Dear all,

Up to Present, the wiki contains information generated by the XSD editing tool XMLspy - great.

Now, we're using Enterprise Architect. This tool lets us write UML which is much richer than XSD schemata. In fact, UML registers documentation, both graphically and textually. The information is far richer than what XMLspy can capture.
It stands to reason that we make Enterprise Architect generate the HTML-information that we publish on the site. EA allows us to tailor the HTML-output and this is where Herr Leberl comes in: I suggest that he do the tailoring so that EA produces appealing HTML.

Please refer to http:// www.sparxsystems.com/enterprise_architect_user_guide/12.1/report_generation/htmlreport.html for more information about the templating that EA can do. I would suggest that he create some templates.

Yours,
Bob Janssen

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by Ferri Leberl on Tue, 22 Nov 2016 13:42:54 GMT
View Forum Message <> Reply to Message

Dear Readers,

Let me summarize the current state of work for the automatic wiki page generation. The process shall include the following steps:

Extract essential information on each element (element name, element required/optional, element

documentation, subschema, parent element, children, attributes, attributes required/optional, attributes documentation) from the XSD and collect it in a tableEmploy a script to build a wiki page for every line of the table (every element) as a starting point. These pages should be developed further manually and contain ..... templates to include the essentials of the element. These template pages are generated with another, similar script, and they are not supposed to be changed and developed manually, but actualized with every change of the schema automatically.

I have meanwhile written scripts that solve the latter part of the path: employing the table to generate wiki pages for either purpose.

I ask you for help to solve the first part of the puzzle: reading the railML schema files and deriving a table presenting the structure of the XSD. These are the requirements for the table:

The table should have one line for each element.The table should have, as mentioned above, a column for each element property we consider (element name, element required/optional, element documentation, subschema, parent element, children, attributes, attributes required/optional, attributes documentation).It should be a tab separated list, and where there may be several entries within a column (an element may have several children and several attributes), these entries should be separated by a semicolon.

Are there any obstacles we did not see?
Do you need any further information from our side?
Is there anybody in this forum who has the capacity to solve this part?

Thank you in advance for your opinions, your contributions and your help.

Yours,
Mag. Ferri Leberl

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by Ferri Leberl on Fri, 23 Jun 2017 10:07:09 GMT
View Forum Message <> Reply to Message

Dear Bob Janssen,

Could you please give us a sample of what the EA-exports look like?

Could it be a copyright issue to publish EA-output?

Thank you in advance for your answer,
Yours, Ferri Leberl

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by Ferri Leberl on Tue, 27 Jun 2017 09:44:22 GMT
View Forum Message <> Reply to Message

Dear All,

As the release of railML 3 approaches, I would like to revive the discussion about its documentation. Common understanding for railML.org will be a well documented first scheme version railML 3.1 with less room for interpretation than in the previous versions.

As mentioned in the last conferences railML 3 is going to be specified in UML.
Which changes will this bring for developers and users respectively?
Could this be a reason to shift the paradigm of the documentation? At the time being, the documentation is organized along elements. Should the element be the basic brick of the documentation of railML 3, too?

As I mentioned in my Impuls in #msg_1415 from September 6, 2016, templates help to standardize content but are difficult to edit. A compromise might be to use wiki templates to describe an element but to expand them as soon as the basics are laid down, as to realize both a high degree of standardization and an easy way to edit. The drawback would be that it would be difficult to change the general structure later on.
What is your opinion towards this tradeoff?

Is there a desire to switch the content management system (CMS)? I assume, that we will rely on MediaWiki also in future because of its low entry hurdle, but there should be room to discuss alternatives if you suggest any.

Thank you in advance for your ideas.

Yours, Ferri Leberl

Subject: Re: [railML3] Strategy of documentation at wiki.railML.org
Posted by Ferri Leberl on Thu, 20 Jul 2017 14:32:05 GMT
View Forum Message <> Reply to Message

Dear all,
As I have mentioned earlier, the workflow should be first to extract an element list from the schema, and then, at second, to transform the element list into wiki code.
DI Wunsch provided us an XSLT-script that transforms a railML-document into code suitable for the MediaWiki-extension "Tree and Menu". Principally, it should be possible to change this script such, that, if you feed it with a "complete" railML-file, containing all elements, with all attributes, exactly once, it builds the element list needed for step two. This would be a second best alternative for step one. It would not deliver all required information (particularly the documentation tags would be lost), but it would provide a base.
Is there an efficient way to produce such examples in XMLSpy or Oxygen?
Thank you in advance for hints, heuristics, and examples.
Yours,
Mag. Ferri Leberl
