Subject: Ideas for the future development
Posted by Volker Knollmann on Tue, 21 Jun 2005 14:47:41 GMT
View Forum Message <> Reply to Message

[To all German readers / An alle deutschen Leser: Sie finden eine deutsche Version dieses Postings unterhalb des englisches Teils!]

Hello friends of railML,

a looong time ago I promised to post some ideas regarding the further development of the infrastructure schema. So here are my proposals. Be warned: this is a long posting, but it's worth reading it! Thanks in advance for your time and your interest!

In this mail, I want to address five major topics:

I) DEVELOPMENT PROCESS AND NEW FEATURES

II) DOCUMENTATION

III) LANGUAGE

IV) MISCELLANEOUS

V) PERSPECTIVE

Let's start with the first one...

I) DEVELOPMENT PROCESS AND NEW FEATURES

In the last postings (sent by Matthias Hengartner, Tobias Bende, Albrecht Achilles and others) we received some hints regarding new features that should be implemented in upcoming versions.

Additionally, we have interfaces e. g. to the interlocking schema which is currently developed at TU Dresden (does anyone have information about the current state of the project) or the existing schemes rolling stock and timetable. Maybe, these interfaces need extensions and new elements in our infrastructure schema.

So there is definitely a need to extend the current schema with new elements and features.

My suggestion is to start with a discussion about new features or elements for our schema; in this discussion we should spend hardly no

time on the implementation. Once we agreed on a set of features (and maybe on the versions they will be implemented in) we can start with the implementation discussion. In short:

1) Collect features

2) Assign features to versions ("roadmap", "feature freeze")

3) Implement; parallel: start again with (1)

This strategy can be found in several successful open-source projects and I think it can be useful for our needs. It has four main advantages:

* People, especially from the industry, can bring in new ideas regardless of their implementation. Now, many readers of the newsgroup don't dare posting new ideas. They are afraid to be asked for concrete XML- or XSD-examples although they are not familiar with XML. But in our discussions we need persons who are familiar with railway systems as well!

* Our development becomes more reliable, predictable and transparent. Users of our schema can estimate which features will be introduced and take it for their development into account.

* The development itself has a clear target; up to now, we only implement what come to our minds. If we have a list of features to be implemented, we can distribute the development easily over a group of developers and we know precisely, when the development is finished. Other things like testing or documentation will be improved, too.

* Due to the early freezing of features and functions, the quality of the resulting schema, especially regarding its structure and data types, will be improved.

That's why I think a more structured development with features freezes and planned versions is suitable for us. And maybe a more transparent development process will motivate more people to contribute and take part in the development. I'd be interested in your opinion about these ideas!

II) DOCUMENTATION

Currently, the documentation does not cover all features of the version

1.0 and is not fully adapted to it (there are some artefacts from prior versions of the schema and the documentation).

To help railML spreading, we should update the documentation. We should provide the following documents

* An Overview about railML-infrastructure (exists, needs update)

* Examples for each element
  (I've created a similar document for internal purposes; I think we can re-use parts of it)

* Simple example net
  (Perhaps not as sophisticated as the current one, but with more element types; again, I can offer to re-use some of my DLR-documents)

In addition, Ulrich Linder suggested to create a kind of "developer handbook", which is generic for all schemes and which introduces to the common types, the structure of a schema, etc.

III) LANGUAGE

Right now, the majority of railML-users and developers are either native German speakers or can at least read German (like many of the Dutch people). We should consider allowing German discussions in the newsgroup together with the option to write an English conclusion of the discussion thread or to translate important parts into English.

I think, many (German) readers don't take part in the discussions because they think that writing English is too time consuming or they are afraid to make themselves not understable enough. Due to this reason, we lose important input for our development.

My opinion is to rather have a German discussion than no discussion at all. Again: we will / we must conclude important threads in English. But in order to "re-activate" the discussion, we should have German postings as well.

I'd be interested in the opinion especially of the not German speaking readers regarding this point. What do you think?

IV) MISCELLANEOUS

In this short section, I want to refer to two technical aspects we should consider during the future development.

The first one is the harmonization between the three railML-schemes. For example, we should use identical attribute names for the same data (e. g. element IDs) and we should consider the use of namespaces. The harmonization especially targets on the interfaces between the schemes.

The second one is the broader usage of complex types. Right now, complex types are used in some parts of the schema; in other parts, types are directly "hard coded" in the branch of the schema. If we used complex types more often, we can re-use types easier and can gain a harmonization with other schemes much simpler.

V) Perspective

Finally, I want to give you an example how the development of the next versions can look like if we use the above described development process.

Roadmap:

Version 1.01: Bugfixes, updated documentation
Version 1.02: Introduction of complex types, harmonization
Version 1.1x: First introduction of new features or elements (development versions)
Version 1.20: First introduction of new features or elements (stable version)

Tasks:
---
* Report and collect bugs

* Collect ideas and contents for the documentation

* Start bug fixing and updating the docs. During this, we collect ideas on complex types and harmonization (for the docs, we have to review the complete schema in detail, so this is a good chance to think about types)

* Release V 1.01

* Collect, implement, release, ...

FAMOUS LAST WORDS

Congratulations if you worked all your way through this lengthy posting! Please don't hesitate to comment, argue, discuss, damn or support my ideas!

Contribute to the development! The basic idea is that many people do less work and not that a few people do much work! With a more transparent development process, we hopefully can gain more active supporters of the railML-schema!

I'm looking forward to your comments!
(Even if this posting is already some days or weeks old when you read it; it will not expire that fast...)

Best reagards from Braunschweig,
Volker Knollmann

========== GERMAN TRANSLATION ==========

Hallo railML-Interessenten,

vor laaaanger Zeit versprach ich, einige Ideen hinsichtlich der weiteren Entwicklung des Infrastruktur-Schemas hier zu posten. Hier sind nun endlich meine Vorschläge. Bitte seien Sie gewarnt: der Beitrag ist ziemlich lang, aber er ist es dennoch wert, gelesen zu werden! Danke im Voraus für Ihr Interesse und Ihre Zeit!

In diesem Beitrag möchte ich die folgenden fünf Punkte ansprechen:

I) ENTWICKLUNGSPROZESS UND EINARBEITUNG NEUER FEATURES

II) DOKUMENTATION

III) SPRACHE

IV) VERSCHIEDENES

V) AUSBLICK

I) ENTWICKLUNGSPROZESS UND EINARBEITUNG NEUER FEATURES

In den lezten Beiträgen (u. a. von M. Hengartner, T. Bende und A. Achilles) waren bereits einige Wünsche nach neuen Features für künftige railML-Versionen enthalten.

Darüber hinaus haben wir Berührpunkte mit anderen Schemas, beispielsweise mit dem neuen Stellwerks-Schema, das gerade an der TU-Dresden entwickelt wird (kennt jemand den aktuellen Stand des Projektes?) oder den beiden anderen existierenden railML-Schemen. Wahrscheinlich benötigen wir Erweiterungen und neue Elemente in unserem Schema, um diese Berührpunkte und Schnittstellen noch besser zu gestalten.

Wir müssen also auf jeden Fall die Entwicklung des Infrastruktur-Schemas fortsetzen und neue Elemente und neue Features ergänzen.

Ich schlage vor, dass wir dazu zunächst die neuen Features und Elemente, die wir künftig unterstützen wollen, zusammentragen. Im Rahmen der damit verbundenen Diskussion sollten wir noch nicht über Implementierungsdetails diskutieren.

Nachdem wir uns auf eine Reihe von Features geeinigt haben (und auf die Version, in der sie erscheinen sollen) wird die eigentlich Umsetzung in Angriff genommen. Kurz gesagt:

1) Vorschläge hinsichtlich Funktionalität und Elementen sammeln

2) Die Vorschläge Versionen zuordnen ("Roadmap", "Feature Freeze")

3) Umsetzung beginnen; gleichzeitig: neue Vorschläge sammeln ==> (1)

Dieses Vorgehen wird auch in mehreren erfolgreichen Open-Source-Projekten verwendet und ich denke, dass es nützlich für uns sein könnte. Die vier wichtigsten Vorteile sind:

* Auch Personen, die nicht Experte in Sachen XML / XSD sind, können Wünsche und Vorschläge einbringen. Momentan lassen sich viele Leser der Newsgroup von der Angst abschrecken, bei Vorschlägen und Diskussionsbeiträgen sofort nach konkreten XML-Beispielen befragt zu werden. Diese Angst ist natürlich unbegründet. Wir brauchen sowohl die Beiträge von XML- als auch von Eisenbahnexperten!

* Der Entwicklungsvorgang wird zuverlässiger, vorhersagbarer und transparenter. Anwender unseres Schemas können abschätzen, was die künftigen Schemas bringen und ihre eigene Entwicklung entsprechend anpassen.

* Die Entwicklung selbst ist zielgerichteter; bis jetzt haben wir

immer das implementiert, was uns gerade notwendig erschien. Wenn wir aber anhand einer Liste von Features vorgehen, können wir den Entwicklungsaufwand leichter auf mehrere Personen verteilen und wir können auch genau sagen, wenn die Entwicklung einer neuen Version abgeschlossen ist. Darüber hinaus werden auch andere Dinge wie Testen und Dokumentieren unserer Entwicklung erleichtert.

* Durch die frühzeitige Festlegung der zu implementierenden Funktionen wird die Qualität der Entwicklung, insbesondere der Schemastruktur und der Datentypen, erhöht.

Aus den genannten Gründen glaube ich, dass uns ein sauber strukturierter Entwicklungsprozess mit "Feature Freezes" und "Versionsroadmap" Vorteile bringt. Und vielleicht motiviert eine transparentere Entwicklung mehr Personen, uns bei der Umsetzung und der Entwicklung der Ideen zu helfen!

Mich würde Ihre Meinung zu diesen Ideen sehr interessieren!

II) DOKUMENTATION

Die aktuelle Version der Dokumentation umfasst leider nicht alle Elemente und alle Attribute der Version 1.0 und ist auch nicht gänzlich an sie angepaßt (es gibt immer noch Abschnitte, die sich auf alte Teile der Schemas beziehen, die so nicht mehr gültig sind).

Um die weitere Verbreitung von railML zu unterstützen, sollten wir die Dokumentation dringend updaten. Dazu sollten wir folgende Dokumente erstellen:

* Einen Überblick über das Schema (existiert bereits, muß aktualisiert werden)

* Beispiele für jedes XML-Element
(Ich habe ein ähnliches Dokument für interne Zwecke angelegt; ich denke, dass wir Teile davon wiederverwenden können)

* Einfaches Beispielnetz
(Möglichst nicht so komplex und ausgefeilt wie das jetzige, aber mit mehr Elementtypen; auch hierzu kann ich anbieten, Teile von DLR-Dokumenten wiederzuverwenden)

Darüber hinaus hat Ulrich Linder vorgeschlagen, eine Art "Entwicklerhandbuch" zu erstellen, dass generisch für alle Schemas gültig ist und in dem gemeinsame Datentypen, die Schemastruktur und ähnliches eingeführt werden.

III) SPRACHE

Momentan ist die Mehrheit der railML-Interessenten entweder direkt deutschsprachig oder in der Lage, deutsche Texte zu verstehen (wie beispielsweise einige Niederländer). Wir sollten daher überlegen, in der Newsgroup auch deutsche Diskussion zu führen, mit der Einschränkung, dass wir am Ende einer Diskussion (oder an entscheidenen Stellen) eine englische Zusammenfassung erstellen bzw. eine englische Übersetzung bereitstellen.

Ich glaube, viele (deutsche) Leser der Newsgroup lassen sich von eigenen Beiträgen abhalten, weil sie befürchten, dass das Verfassen englischer Texte zuviel Zeit in Anspruch nimmt oder dass sie sich nicht ausreichend verständlich machen können. Dadurch verlieren wir wichtige Anregungen für unsere Arbeit.

Meine persönliche Ansicht ist, dass wir lieber eine deutschsprachige Diskussion haben sollten als überhaupt keine. Aber nochmals: wir müssen bzw. werden wichtige Diskussionen auf englisch zusammenfassen. Um aber die Diskussion überhaupt wieder aufleben zu lassen, sollten wir auch deutsche Postings zulassen.

Zu diesen Thema würde mich vor allem die Meinung der nicht-deutschsprachigen Leser interessieren! Was denken Sie?

IV) VERSCHIEDENES

In diesem kurzen Abschnitt möchte ich zwei eher technische Aspekte anbringen, die wir bei künftigen Entwicklungen berücksichtigen sollten.

Der erste Punkt ist die Harmonisierung zwischen den verschiedenen railML-Schemas. Beispielsweise sollten wir identische Attributnamen für die gleichen Daten (z. B. Element IDs) verwenden. Darüber hinaus wäre der Einsatz von Namespaces zu überlegen. Die Harmonisierung zielt vor allem auf die Berührpunkte zwischen den Schemas ab.

Der zweite Punkt ist der verstärkte Einsatz komplexer Datentypen. Bis jetzt werden komplexe Datentypen nur an einigen Stellen des Schemas eingesetzt; an anderen Stellen sind Datentypen "hart" in den entsprechenden Zweig des Schemas codiert. Würden wir komplexe Datentypen häufiger verwenden, könnten wir diese Typen auch leichter wiederverwenden und so auch eine leichtere Harmonisierung mit den

anderen Schemas erreichen.

V) AUSBLICK

Abschließend möchte ich ein kleines Beispiel präsentieren, wie der Entwicklung der nächsten Versionen ablaufen kann, wenn man den oben beschriebenen Entwicklungsprozess anwendet:

Roadmap:

Version 1.01: Bugfixes, aktualisierte Version
Version 1.02: Einführung komplexer Datentypen, Harmonisierung
Version 1.1x: Erste Einführung neuer Features oder Elemente
(Entwicklerversionen)
Version 1.20: Erste Einführung neuer features oder Elemente
("Stabile" Version)

Aufgaben:
---
* Fehler melden und sammeln

* Ideen und Inhalte für die Dokumentation sammeln

* Mit der Fehlerbehebung und dem Aktualisieren der Doku anfangen. Im Rahmen dieser Arbeit müssen wir sowieso das gesamte Schema im Detail untersuchen, so dass wir gleichzeitig Ideen für komplexe Datentypen oder die Harmonisierung sammeln können.

* Veröffentlichung V 1.01

* Beiträge sammeln, implementieren, veröffentlichen, usw.

LETZTE WORTE

Herzlichen Glückwunsch! Sie haben sich endlich zum Ende dieses langen Beitrags vorgearbeitet! Zögern Sie bitte nicht, meine hier vorgebrachten Ideen zur kommentieren, sie zu diskutieren, zu verfluchen oder zu unterstützen!

Tragen Sie zur weiteren Entwicklung bei! Die grundsätzliche Idee ist, dass viele Leute ein wenig beitragen und nicht wenige Leute viel! Mit

einer transparenteren Entwicklung können wir hoffentlich mehr aktive Unterstützung für railML erreichen!

Ich freue mich auf Ihre Kommentare!
(Auch wenn dieser Beitrag schon einige Tage oder Wochen alt ist, wenn Sie ihn lesen; er wird so schnell nicht veralten...)

Grüße aus Braunschweig,
Volker Knollmann
