Q: How does EMFText aid the model-driven development process?

A: Model-driven development (MDD) is concerned with the definition of models that encode the structure and behavior of the software that is built. Each model has a specific purpose (e.g., it may define the user interface of an application). In contrast to code-centric development, where the majority of an application is specified in the same language (e.g., Java or C++), multiple domain-specific languages are applied in MDD.

EMFText allows to quickly create such domain-specific modeling languages. Based on an Ecore model that defines the concepts of the domain, a textual syntax can be derived and manually adjusted according to specific needs. From these two artifacts, the Ecore models and the syntax specification, EMFText generates Eclipse plug-ins that implement a full-fledged editor for the language. Thus, developers can quickly design new languages according to the requirements of their project.

For small to medium languages the effort to define the Ecore model and the text syntax ranges from a couple of minutes to few hours. This is in contrast to the creation of graphical editors which usually require more effort.

Q: What are the benefits of defining your own textual Domain Specific Languages (DSLs)?

A: Assuming that DSLs are well designed, such languages allow to specify software in a more concise and more abstract fashion. Often, single DSL concepts can be translated to multiple instructions in Java. Thus, ideally less typing is required. Of course, this is not true for all parts of a software project. Some things, for example complex computations, are still best expressed in terms of Java classes and method. But, there are also parts that are very repetitive and that require lots of so-called boilerplate code. In such cases DSLs can actually reduce development costs. Also, DSL models can be translated to multiple target languages, but this is not a very frequent requirement.

A particular benefit of using textual DSLs, instead of graphical ones, is that the former can be created much more quickly. This may change if tools for graphical models get more mature, but from the current perspective textual tools allow for very quick language development. Also, developers are familiar with textual specification and much more comfortable to use a text editor instead of graphical ones. Sophisticated features that are known from the Eclipse JDT (e.g., code completion, syntax highlighting, hyperlinks and so on) are also automatically provided by editors generated with EMFText. The creation of textual models is therefore both easy and efficient.

Q: What's new in the latest, 1.3.4 release of EMFText?

A: Release 1.3.4 of EMFText is mainly a bugfix release, but we've also added some new features to reward our users for performing the update. This time, a new wizard has been added that allows parameterize the creation of a custom syntax using a multitude of options. Previously, the automatic syntax derivation was quite static, we supported HUTN and a Java-like syntax. Now, users can obtain syntax specification that fit their needs more easily, because the amount of manual adjustments have been significantly reduced.

Moreover, we have added two new example languages to the update site. Minimal Ecore is a textual variant of Ecore and can be used as a substitute for the other Ecore editors (e.g., Emfatic). Minimal Ecore aims to create Ecore models as quickly as possible using a very brief syntax and a lot of default behavior. WebTest, the second new language, demonstrates how to design a small scripting language that can perform functional web tests.