Common Development Principles

Both T3Q and T3D are build upon the TRex Core infrastructure. This means that both tools share various common concepts which are derived from the basic mechanisms for ?standalone TRex Core usage. The basic framework consists of roughly three parts:

- the analyzer (lexing, parsing, symbol table creation, etc.), which results in a AST/parse tree
- the visitor, which traverses the AST/parse tree and executes designated actions upon certain types of nodes in the tree; the actions are generally delegated to the processor
- the processor (guideline checking, documentation generation, etc.), which is called by the visitor and defines the designated actions

It is possible to add any number of visitors for separate tasks, which are performed sequentially, over multiple traversals of the tree (multi-pass), each starts from the beginning after the previous visitor terminates. It is also possible to add any number of processors to a visitor, to perform different tasks simultaneously, during a single traversal of the tree (single-pass). Which approach is better suited for a particular set of tasks is largely dependent on the specific tasks - guideline checking and naming conventions checking are performed simultaneously, while documentation generation and local dependency generation are performed sequentially.

To make a tool more flexible and useful, a number of further infrastructure components are necessary. T3Q and T3D share a significant part of these infrastructure components, such as input and output interfaces, configuration handlers, etc, which are delegated to a separate T3Tools Common Components package. The IO interfaces define the CLI and file system interfaces - what input parameters are available and how they are interpreted, including reading from and writing to files. These are in part utilized by the configuration handlers to access configuration files. The configuration handlers also manage the profiles and settings, which affect mostly the above mentioned processors, but also the main tool components (e.g. execution parameters such as recursive processing, logging

options, etc.).

The use of the analyzer is pretty schematic: parse the input files, perform post processing. provide input trees for the visitors.

The visitors are based on an abstract visitor which is automatically generated from the currently available TTCN-3 grammar (available in the required TRex Core package). The exact location of the grammar will have to be specified in the visitor generation tool. A concrete visitor is then derived from the abstract visitor by only including the methods for the desired node types.

The processors should be preferably structured in a systematic way as well. The guideline checking processor, for example, contains methods for all the available guidelines which are called from the appropriate node types in the visitor. Additionally, there are helper methods, which may be further delegated to a separate helper class. The methods also generally follow a consistent structure, although in some cases exceptions are necessary due to the nature of a particular guideline.