WIP: JUnit compatible XML reporting for tcltest

OR: Abusing a refactored tcunit to enable Jenkins statistics on tcltest output

Matthias Kraft

AGENDA

- MOTIVATION
- WHY TCUNIT?
- WHERE TO FIND?
- WHAT'S NEXT?
- THAT'S IT ...

Motivation

- At Software AG we are using Jenkins for continuous integration builds.
- Jenkins can provide quite nice statistics, out of the box e. g. for unit tests → View, Test Results
- A file called **TEST-foo.xml** in an undocumented format is necessary
- To employ these I wrote a script at work, which is "closed source" and more like a quick hack, however

WHY TCLUNIT?

- INITIAL INTENT WAS TO WRITE OWN PACKAGE
- STUMBLED UPON → sf.net WHILE LOOKING IF THE NAME IS ALREADY RESERVED
- FOUND AN EASY AND NON-INTRUSIVE ARCHITECTURE WHICH ONLY NEEDS TO BE REVEALED AND EXTENDED

Where to find?

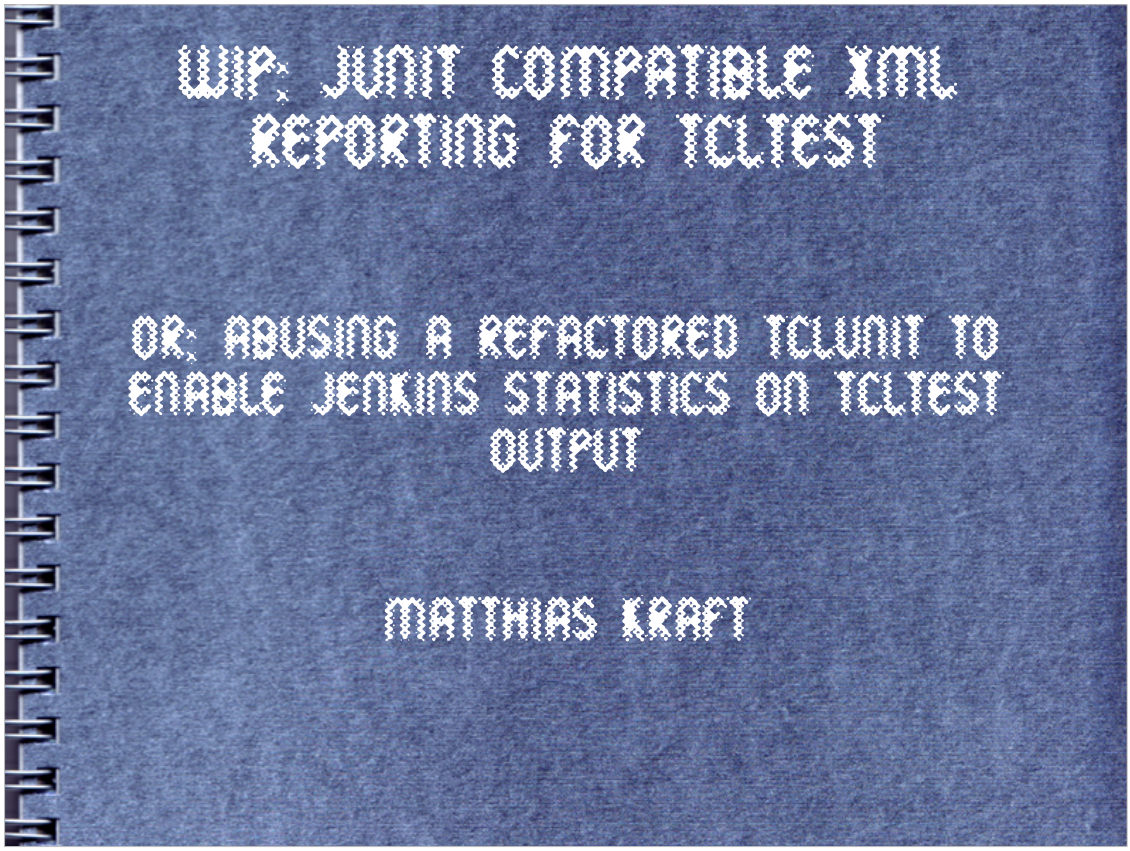
- The old GUI had been extracted and is an extra script now, like the new XML generator
- The tcUnit package is now only an event generator acting on the log messages of tcTest
- It is still work in progress, although it can already be used
- All can be found at → github.com

WHAT'S NEXT?

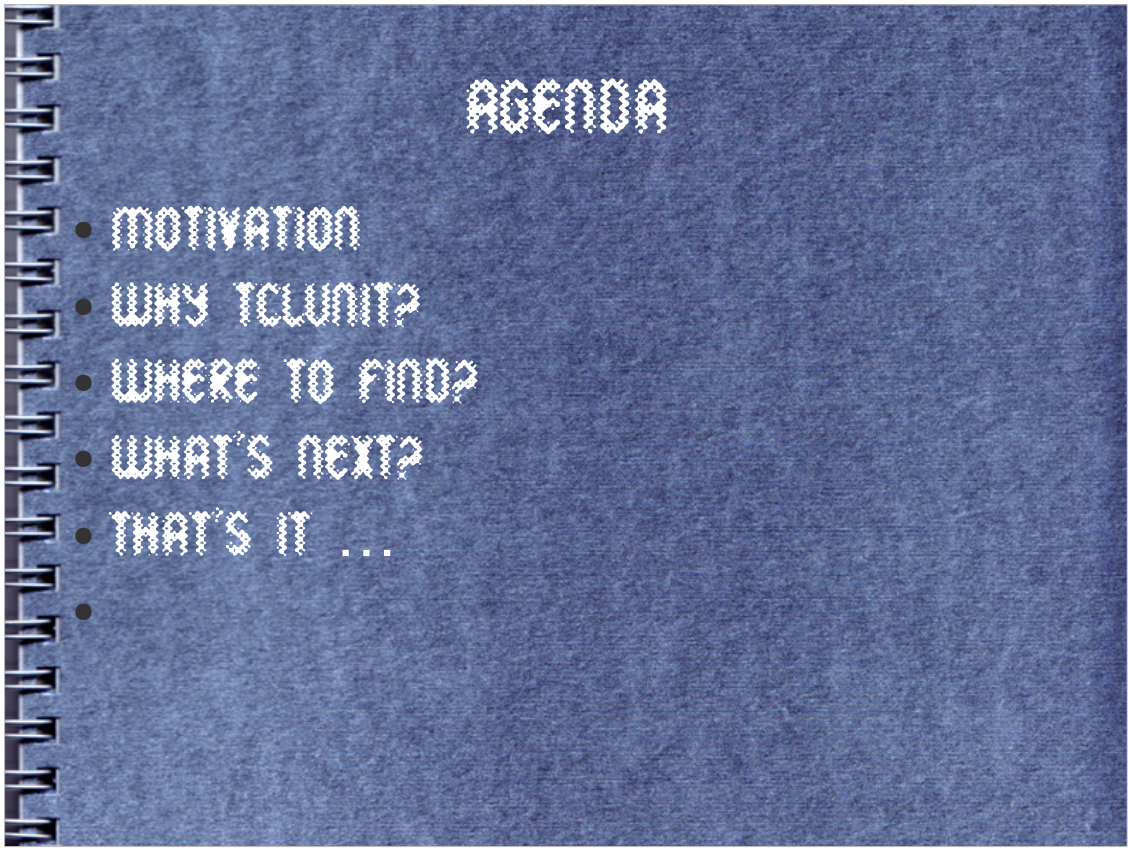
- STILL MISSING ...
  - THE TIMING INFORMATION
  - THE STDOUT/STDERR CAPTURE
  - HANDLING OF ERRORS IN TESTS
- GETTING IT RIGHT, I.E. USE OF TDOM
- MORE FLIERS?

That's it ...

- Thanks for your attention!
- Any suggestions and questions are welcome...



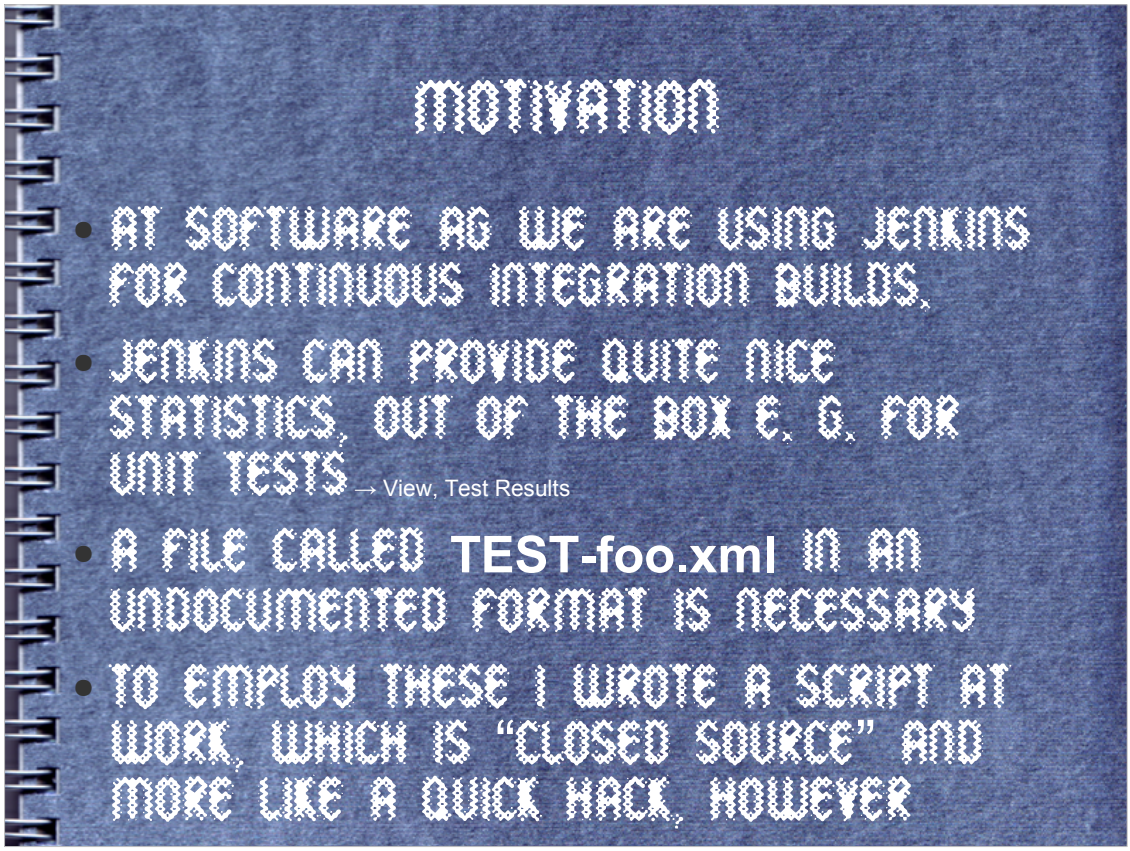
* Welcome to my short Work-in-Progress talk.
* I'll tell you something about abusing a refactored tclunit to enable Jenkins statistics on tcltest output.



* The agenda …
* I use a few terms not everyone might be familiar with...
* Jenkins (which is a fork of Hudson) is a Continuous Integration Build server. It is “making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. The automated, continuous build increases the productivity.”

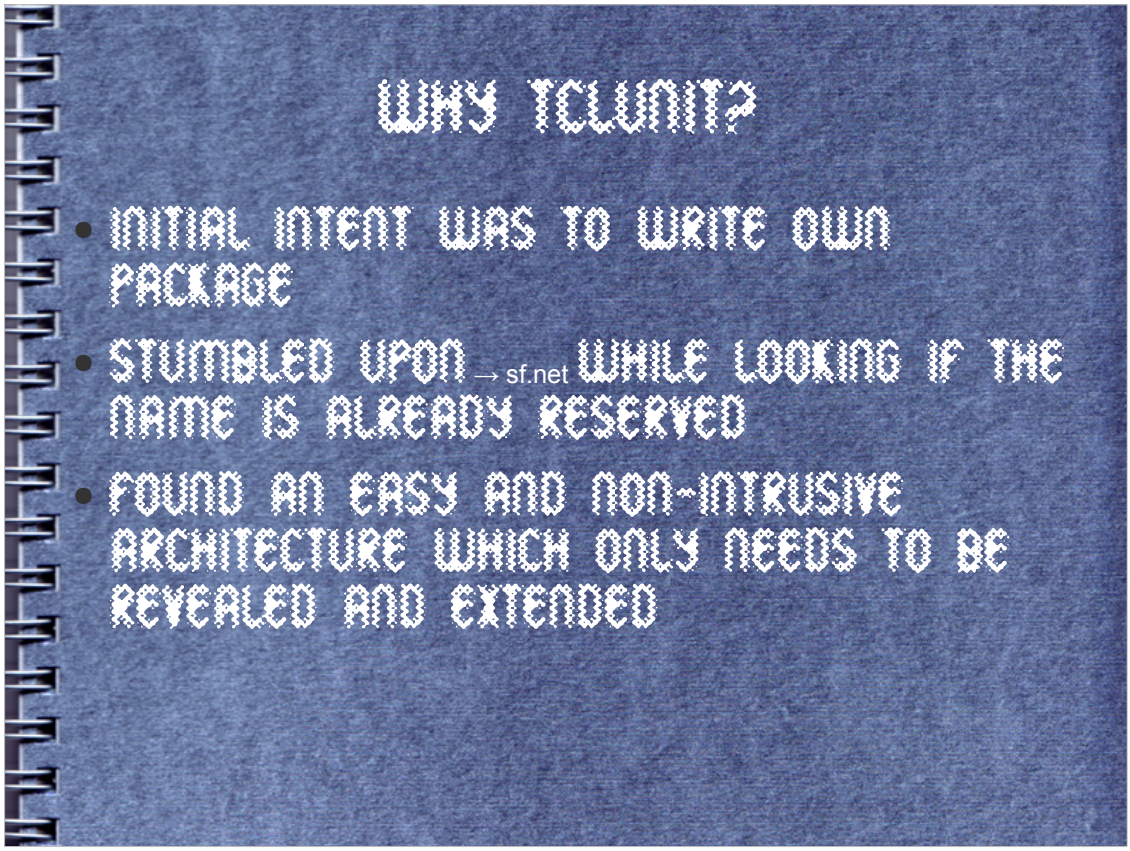
→ jenkins-ci.org

* Continuous Integration is an agile development method. Together with Unit tests, e. g. written during Test Driven Development, it provides a possibility to deliver small pieces of software in short cycles with a high quality.



→ softwareag.com

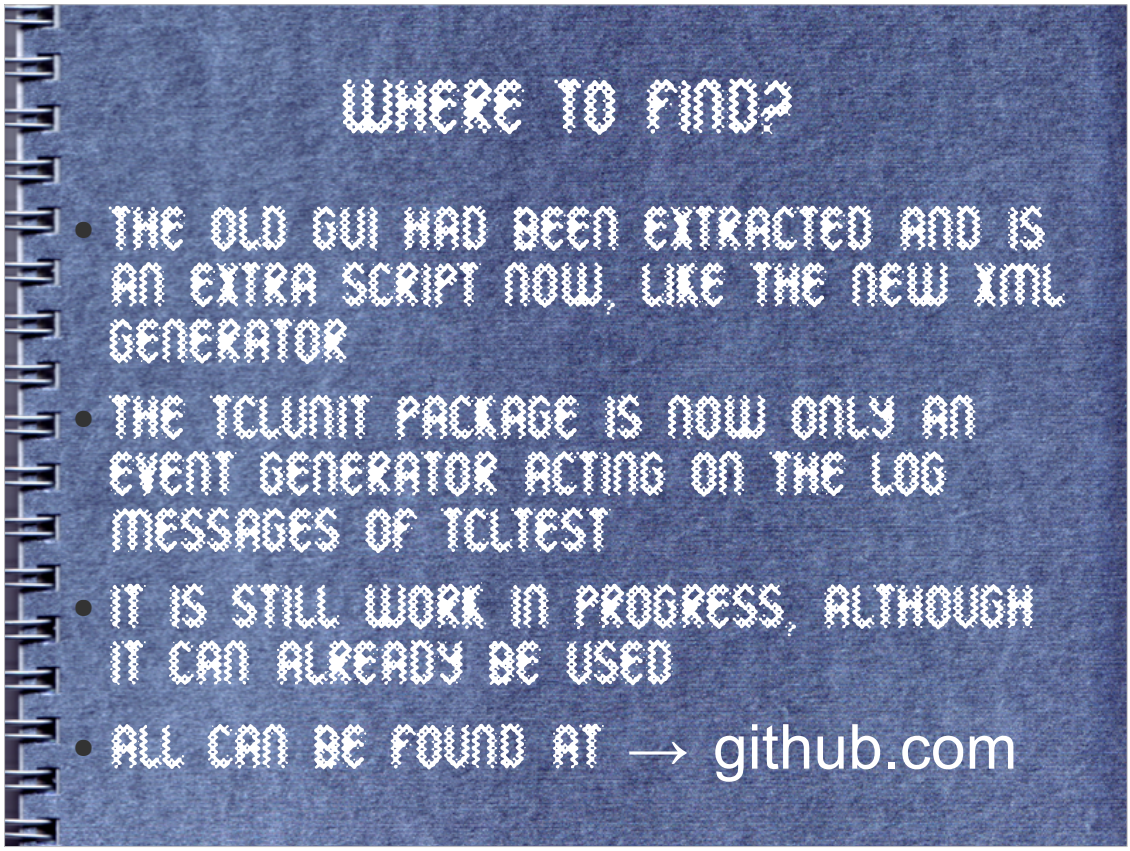
* "View" is a link to my local Jenkins showing two projects with their current test abstract and history
* "Test Results" is also a link into my Jenkins, but deep into a project. It shows a table of the test scripts run and how many of their tests passed, were skipped or failed. It also reports a failed test case. … and by clicking on the links there more details can be displayed or history information could be retrieved, etc.
* the test logs have to come in a special XML format
* it however is **not** documented by a DTD or a Schema, only by the implementation within the ant-junit task, or Jenkins (directory core/src/main/java/hudson/tasks/junit/)



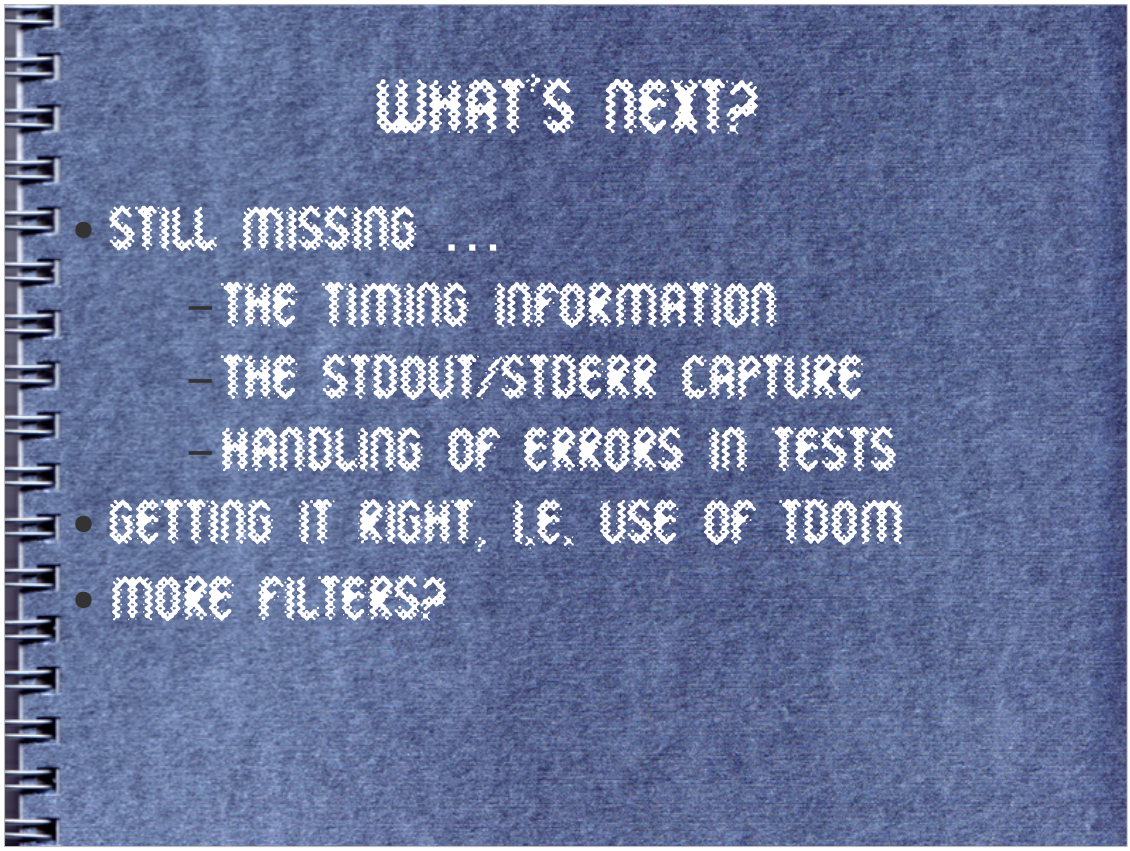
→ “sf.net”:
tcllib.cvs.sf.net/viewvc/tcllib/tclapps/apps/tclunit/

* my initial idea was to take tcltest and make it emit the XML test log, then call it tclunit
* but tclunit was already taken by a tclapp written a couple of years ago by Bob Techentin
* although his tclapp was just a GUI for running a test script (or all test scripts within a directory) he wrote in a very clean style
* he basically starts a remote interpreter process running the tests and captures its output acting on the log message
* btw another considered name was already taken, too

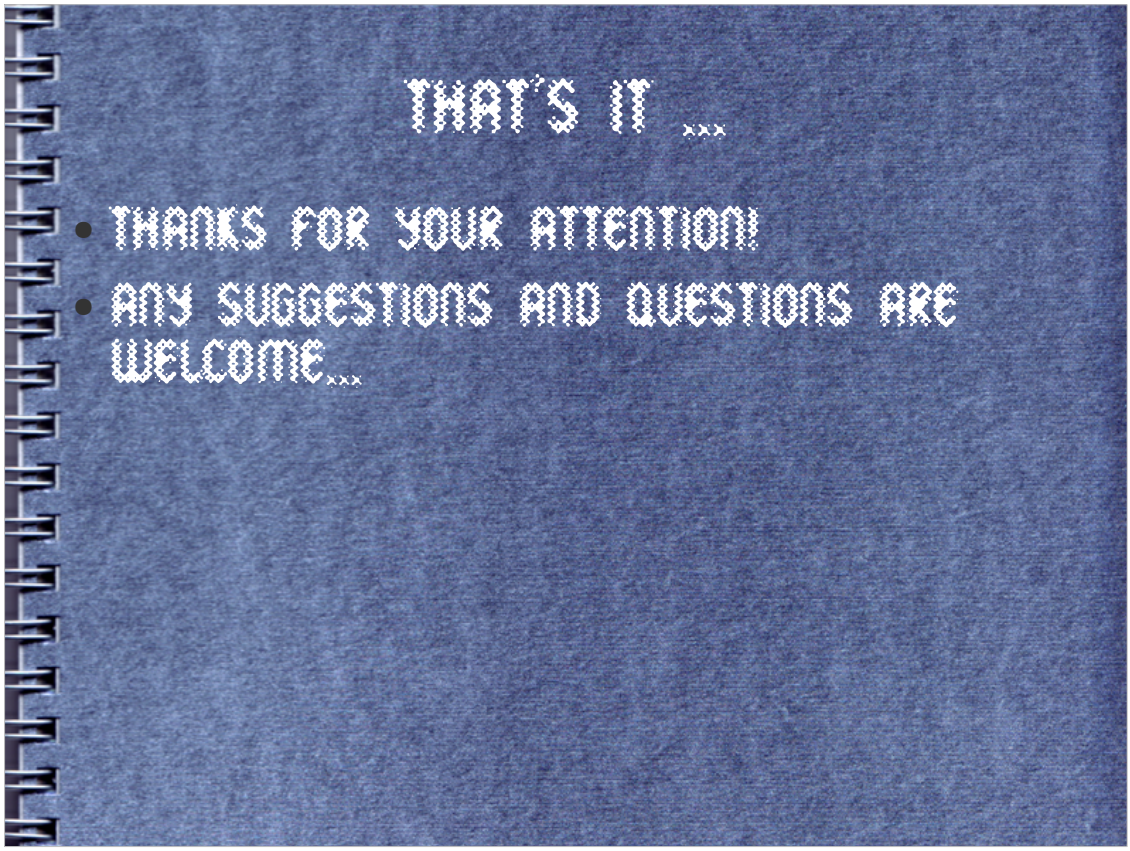
→ docs.tinyos.net/tinywiki/index.php/TUnit



→ “github.com”: https://github.com/makr/tclunit
* so all I had to do was separating GUI and functionality and make a proper tcl package from it
* then writing the XML generator
* although a couple of things are still missing it can already be used, as seen in the project page, where the “make test” output of Tcl is parsed by the script



* the test cases as well as the complete test run can have timing information
** from tcltest currently only the complete test run timing information is available which would need to be stored as attribute in the testsuite header
** due to my as-easy-as-possible approach this is not possible
* stdout/stderr can be captured and stored in the XML file, too
* errors in test cases are currently completely ignored
* to add timing and output the current serial plain text handling has to make place for a proper DOM tree build up
*no more todos from my side, but if anyone has other ideas, feel free to fork on github and I will accept pull requests then



* now its time to wake up and ask questions or tell me that I did something completely stupid in my spare time :-)