denx
software
engineering

tbot
Automatic Test Framework

Was bisher geschah:

Softwareentwickler sind nicht gerade Fans von Test Tools. Sie benutzen sie, wenn das Projektmanagement es verlangt, aber sie sind nicht wirklich begeistert. Warum ist das so?

Softwareprojekte werden oft in die Phasen Spezifikation, Design, Implementierung, Test, Dokumentation und Wartung unterteilt. Die meisten Entwickler finden vor allem die ersten drei Phasen kreativ und interessant, die letzten drei eher langweilig. Gute Werkzeuge erfüllen einen speziellen Zweck. Test Tools sind zum Testen gemacht. Testen ist langweilig. Obwohl also Test Tools hilfreich sein können (indem sie z. B. die zum Testen benötigte Zeit reduzieren), werden sie nicht mit den interessanteren Teilen des Entwicklungsprozesses assoziiert. Test Tools werden vor allem für Manager gemacht - Manager lieben sie, Entwickler eher nicht.

Auftritt tbot:

In der täglichen Arbeit gibt es oft keine harte Grenze zwischen Implementierung und Test - selbst ein Compilerlauf stellt letztlich einen Test dar (z. B. dafür, ob Syntaxfehler vorhanden sind). Gängige Test Tools helfen hier nicht viel, denn sie sind für die Testphase gemacht, nicht für die Implementierung. Die meisten jedenfalls. Nicht so tbot: Das wurde von Entwicklern für Entwickler geschrieben. Sein Hauptzweck ist nicht das Erzeugen schöner Testgrafiken für die Manager (das macht es aber natürlich auch), sondern das Automatisieren der Routinearbeiten des Entwicklers. tbot hilft damit vor allem bei Aufgaben, die ein Entwickler häufig wiederholt ausführen muß, und eliminiert so notwendige, aber wenig kreative Teile der täglichen Arbeit. Während die meisten anderen Test Tools zusätzliche Arbeit verursachen, reduziert tbot die Arbeitsmenge.

Im Ergebnis hat der Entwickler mehr Zeit für die kreativen, herausfordernden Teile seiner Arbeit und der Manager hat Zugriff auf eine Menge statistischer Daten aus den durchgeführten Tests. Und sie lebten glücklich und zufrieden bis ans Ende ihrer Tage.

Ein hübsches Märchen? Keinesfalls! Probiert es selbst:

The Story So Far:

Software Engineers are not really fond of test tools. They use them, if the project management requires so, but usually they don't love them. Why is this so?

Software development projects are often subdivided in phases Specification, Design, Implementation, Testing, Documentation, and Maintenance. Most software engineers consider the first three phases creative and interesting, while the last three are more or less boring. Good tools are made for a specific purpose. Test tools are made for testing. Testing is boring. So while test tools can help (for example, by reducing the time needed to run the tests), they are not associated with the sexy parts of the development process. Test tools are primarily made for managers, so managers love them, developers don't.

Enter tbot:

In reality, there is often not such a hard cut between implementation and testing - even a compiler run is some kind of test (for syntax errors, for example). But test tools don't help here, as they are made for testing, not for the implementation phase. Well, most of them. Except tbot. tbot was written by developers, for developers. It's main purpose is not to generate test charts for the management (it does that, too, of course), but to automate boring parts of the developer's work. So tbot is focussed on the tasks that developers have to perform routinely, to eliminate necessary, but non-creative parts of the daily work. While most other test tools add to the work a developer has to perform, tbot actually reduces the amount of work.

In the end, the developer has more time to focus on the creative, mentally challenging, sexy parts of his job, and the manager can access a lot of statistical data about performed tests. And they all lived happily ever after...

A nice fairy tale? No! Try it out yourself:

tbot | *https://www.tbot.tools/*
https://github.com/hsdenx/tbot

Die wichtigsten Fakten:

- Die Tests laufen auf realer Hardware ab.
- Auch komplexe Szenarien werden uneingeschränkt unterstützt, wie Ein- und Ausschalten, Tests für Bootloader (z. B. U-Boot), Betriebssysteme (z. B. Linux) und Gerätetreiber, Applikationen, Kommunikation mit anderen Systemen usw.
- tbot ist in Python implementiert. Erweiterungen und Testfälle werden ebenfalls in Python geschrieben und sind damit sehr einfach möglich.
- tbot ist modular und leicht konfigurierbar. Es kann einfach an unterschiedliche Laboreinrichtungen angepasst werden (z. B. computer-gesteuertes Ein- und Ausschalten, „Stecken“ und „Ziehen“ von SDcards oder USB Sticks usw.).
- Testfälle können andere Testfälle aufrufen, so dass sich sehr leicht Testgruppen und -hierarchien bilden oder vorhandene Testfälle für andere Projekte wiederverwenden lassen.
- Durch ein eventgesteuertes Postprocessing können die Ergebnisse beliebig aufbereitet und z. B. Statistiken und Grafiken erzeugt werden. Eine andere interessante Anwendung ist das automatische Generieren von Dokumentation.
- tbot kann leicht in andere CI- und Testsysteme wie Buildbot oder Jenkins integriert werden.
- tbot ist vollständig Open Source (GPLv2).

Important Facts:

- *Tests are performed on real hardware.*
- *Even complex scenarios are fully supported (for example power cycling, tests for boot loader (e.g. U-Boot), operating system (e.g. Linux) and device drivers, applications, communication with other systems etc.*
- *tbot is implemented in Python. Extensions and test cases are also written in Python and thus very easy.*
- *tbot is modular and easy to configure. It can be flexibly adapted to different lab equipment (for example computerized power switching, plugging and removing of SDCards or USB sticks, etc.)*
- *Test cases can run other test cases, so that it is very easy to construct test groups and hierarchies, or to re-use existing test cases for other projects.*
- *An event-driven postprocessing allows to generate for example statistical data or graphic representations of the test results. Another interesting use case is the automatic generation of documentation.*
- *tbot can easily be integrated into other CI and Test Systems like Buildbot or Jenkins.*
- *tbot is completely Open Source (GPLv2).*

denx software engineering GmbH | Kirchenstraße 5 | 82194 Gröbenzell | Germany
Tel.: +49 (0) 81 42 6 69 89 - 0 | Fax: +49 (0) 81 42 6 69 89 - 80 | Email: info@denx.de | Web: www.denx.de