AUTOMATIC QUALITY ASSURANCE OF SOFTWARE FOR EMBEDDED DEVICES

Aleš Pernikář

Master Degree Programme (2), FEEC BUT

E-mail: xperni08@stud.feec.vutbr.cz

Supervised by: Peter Honec

E-mail: honecp@feec.vutbr.cz

Abstract: This article concerns automation of software and hardware testing using the robotic testing system called RQA. It describes a design of an interface producing test scenarios using the robotic testing system. Next it elaborates metrics measured during testing used for quality assurance of the system uder test. One of the topics covered is a visualization of test data and measurements for an easy overview of quality.

Keywords: Automation, automated testing, robot, regression tests, quality assurance, data visualization

1 ÚVOD

Vývoj software postupuje vysokým tempem a vznikají nové požadavky na zajišťování jeho kvality. Čistě softwarové testování má své nedostatky, kdy takové testování nemusí obsáhnout všechny problémy a úskalí typická pro spojení se specifickým hardwarem. Tato práce se zabývá otázkou, jak efektivně vytvářet nové testovací scénáře s minimální zátěží pro operátora, ovšem s reálným hardwarem, který navíc může být proměnný [3]. Dalším důležitým aspektem je vizualizace dat testování, aby byly výstupy a závěry testu přehledné a vypovídající pro každého. Výsledky testování je třeba dlouhodobě automaticky vyhodnocovat a zdůraznit trendy ve změně kvality na základě naměřených údajů během testování**.**

2 ROZBOR

Ve firmě YSoft pracuji na projektu Robotického testovací systému pro kontrolu kvality (zkráceně RQA – Robotic quality assurance). Tento byl zpočátku navrhutý pro potřeby testování YSoft SafeQ na multifunkčním zařízení ovládaném přes rozhraní určené primárně pro člověka, tedy dotykový terminál. Jedná se o robotický manipulátor, kameru a řídicí software, systém zpracování obrazu z kamery a další komponenty. Robotický manipulátor provádí akce na dotykovém terminálu[1] (tap, swipe) podle zadaného testovacího scénáře, jak je vidět na *Obrázku 1*.

Celý systém se skládá z několika dílčích aplikací, které mají vymezenou zodpovědnost a komunikují spolu (viz *Obrázek 2*). Aplikace RQA Robot control posílá instrukce robotickému manipulátoru o pohybu na určité pozice na terminálu, robot tedy vykoná tuto akci a terminál zareaguje obvykle změnou obrazovky. Kamera na požadavek zachytí tuto změnu a odešle snímek aplikaci Image processing pro další zpracování. Image processing pošle výsledek zpracování do Robot control, který tak získá zpětnou vazbu o úspěšnosti akce [4].

2.1 TVORBA TESTOVACÍCH SCÉNÁŘŮ

Pro automatizovanou tvorbu testovacích scénářů bylo nutné vytvořit jednotný formát testů, které je robot schopný provést a také uživatelské rozhraní, které je intuitivní a produkuje testovací scénáře v tomto formátu. Do robotického systému RQA jsem implementoval základní akce, jako poklepání na známé tlačítko nebo skupinu tlačítek, ověření požadované obrazovky nebo navigace na určitou obrazovku přes známá tlačítka (dostupná v databázi známých prvků). Spojením těchto základních

akcí dohromady vzniká složená akce, která přináší vyšší funkcionalitu, a může být využita při tvorbě různých scénářů.

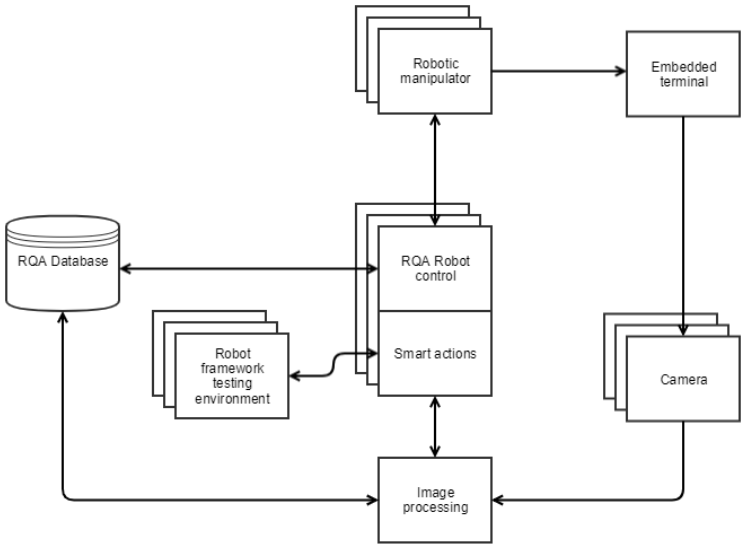
Nejdříve jsem pro tvorbu testů vytvořil open-source framework pro automatizované akceptační a regresní testy jménem Robot Framework využívající syntaxi Gherkin. Ten využívá princip tvorby keywordů, což jsou uživatelsky definované složené akce. Výhodou toho přístupu je znovupoužití keywordů pro různé testované zařízení, testovací scénář se tedy napsal pouze jednou a měl být kompatibilní se všemi testovanými zařízeními. Problémem byly odchylky v nových testovaných zařízeních a následná úprava testovacích scénářů tak, aby se zachovala kompatibilita se současnými zařízeními. Každá úprava byla tedy velmi časově náročná nebo způsobila zanesení nechtěných chyb a vykazování falešně pozitivních nálezů. Další nevýhodou se ukázala poměrně dlouhá doba identifikace přesné příčiny chyby testu. Systém vykazoval velkou míru falešných nálezů (70 až 90 %) a to nejen během ladění testovacích scénářů.

Zkusil jsem tedy změnit přístup k tvorbě testovacích scénářů a zakomponovat je přímo do robotického systému RQA. Místo vytváření v externím frameworku pomocí textového zápisu jsem zavedl rozhraní pro přenesení znalosti testovacího inženýra (jak vypadá test) přímo do základních akcí robota. Daný návrh se uloží do databáze a dále je k dispozici být kdykoliv automaticky spuštěn při pravidelných testech (každý den nebo častěji podle potřeby).

Obrázek 1: Robotický manipulátor před testovaným terminálem

Tento přístup také přinesl snadnější identifikaci příčiny chyby. Protože testy jsou pod správou samotného testovacího systému, a ne externího frameworku, dojde ke srovnání chybného průběhu se vzorovým, zda se změnila např. pozice definovaného tlačítka, vzhledu obrazovky nebo jiné nastavení. Pokud není zjištěna relevantní změna v interních datech testovacího systému, zvyšuje se pravděpodobnost, že nalezená chyba je skutečně způsobena změnou v testovaném systému.

Pro vizualizaci výsledků jsem vytvořil přehlednou dashboard se shromážděnými výsledky za vybrané období a kategorie testů. Obsahuje procentuální úspěšnost testů, míru pokrytí automatizovanými testy a vývoj úspěšnosti testů na vybraných zařízeních v čase. Tohle se ukázalo jako další zjednodušení práce s robotickým testovacím systémem a ušetření času při analýze výsledků testů.



Obrázek 2: Blokové schéma systému RQA

3 ZÁVĚR

V tomto článku jsem naznačil fungování automatického testovacího systému RQA využívaného zejména pro testování software YSoft SafeQ na multifunkčních zařízeních. Zabýval jsem se způsobem tvoření testovacích scénářů a jejich následném spouštění včetně měření vybraných kritérií kvality. Výsledky testování se dlouhodobě shromažďují a vizualizují se do denního nebo týdenního souhrnu s dalšími užitečnými údaji, jako pokrytí testů, procentuální úspěšnost nebo počet testů ukončených se stejnou chybou. Ve srovnání s manuálním testování má automatizované testování řadu benefitů, zejména ušetřený čas operátora, možnost testování v libovolnou denní dobu, přesné opakování zadaných testů nebo nezávislost na individuálním posuzování operátora. Nevýhody spojené s poměrně vysokým množstvím falešných nálezů chyb jsem zmírnil zvýšením robustnosti samotných akcí, jejich opakováním a změnou přístupu k tvorbě testovacím scénářům. Hlavní přínos se ukázal být v posunu od pouhého nástroje k operování s multifunkčními zařízení na komplexní testovací systém se zodpovědností od tvorby testů přes jejich spuštění až po vizualizaci a vyhodnocení výsledků testů.

REFERENCE

[1] Aleš Pernikář. *Calibration - How robot learns the environment* [online]. 27.10.2015 [cit. 2018-11-19]. Dostupné z: https://www.ysofters.com/2015/10/27/calibration-how-robot-learns-the-environment/

[2] NORVIG, Peter a Stuart RUSSELL. *Artificial Intelligence: A Modern Approach*. 3rd Edition. Pearson, 2016. ISBN 978-1292153964.

[3] LU, Luo. *Software Testing Techniques: Technology Maturation and Research Strategy* [online]. In: . s. 20 [cit. 2018-11-19]. Dostupné z: http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf

[4] KYZLINK, Jiří, Václav NOVOTNÝ, Aleš PERNIKÁŘ, Jakub PAVLÁK a Ondřej KRAJÍČEK. *Universal automated testing of embedded systems*. 2017. United States. US 2018 / 0113774 A1. Uděleno 26.4.2018. Zapsáno 19.10.2017. Dostupné také z: https://patents.google.com/patent/US20180113774A1/