Zadání 1. projektu ATA LS 2022/23

Cílem projektu je praktické cvičení testování založené na datech. Cvičení by mělo pokrýt testování založené na požadavcích dle CEG, kombinační testování při tvorbě dat a automatizaci testů.

Na základě specifikace testované komponenty v přirozeném jazyce navrhněte automatizovanou testovací sadu.

Popis testovaného produktu

Řízení vozíku v robotické továrně

V robotické továrvě je motorizovaný vozík pro převoz materiálu potřebného k výrobě. Vozík má předem určené trasy a více zastávek. Řídicí systém vozíku má na starosti plánování další zastávky a monitorování zátěže a využití kapacity vozíku. Vozík zpracovává požadavky o přesun materiálu. Materiál, resp. požadavek na přesun, je určený svojí zdrojovou a cílovou stanicí, váhou a binární prioritou (přesun je nebo není prioritní). Každý materiál má při převozu alokován jeden slot vozíku. Pokud je materiál pro převoz větší než jeden slot nebo je příliš těžký, bude rozdělen do více slotů. O toto rozdělení se však kontrolní systém vozíku nestará. Uvažujte, že vozík je pužívám v ideálním prostředí (systémy s ním komunikující splňují veškeré předpoklady, data odpovídají reálným stavům). Vozík má pro převoz materiálu celkem 1 až 4 sloty. Vozíky jsou 3 druhů, každý má jinou maximální nosnost: 50 kg, 150 kg a 500 kg. Vozíky s nejmenší nosností mají nejméně 2 sloty, vozíky s největší nosností mají maximálně 2 sloty.

Specifikace chování

Pokud je požadováno přemístění nákladu z jednoho místa do druhého, vozík si materiál vyzvedne do 1 minuty. Pokud se to nestihne, materiálu se nastavuje prioritní vlastnost. Každý prioritní materiál musí být vyzvednutý vozíkem do 1 minuty od nastavení prioritního požadavku. Pokud vozík nakládá prioritní materiál, přepíná se do režimu pouze-vykládka. V tomto režimu zůstává, dokud nevyloží všechen takový materiál. Normálně vozík během své jízdy může nabírat a vykládat další materiály v jiných zastávkách. Na jednom místě může vozík akceptovat nebo vyložit jeden i více materiálů. Pořadí vyzvednutí materiálů nesouvisí s pořadím vytváření požadavků. Vozík neakceptuje materiál, pokud jsou všechny jeho sloty obsazené nebo by jeho převzetím byla překročena maximální nosnost.

Úkoly

1. Upravte specifikaci (ze sekce “Specifikace chování”) dle ambiguity review. Zaměřte se na podmínky, příčiny bez důsledků a neurčité nebo vícenačné výrazy. Pro chybějící části nebo jejich upřesnění si stanovte vlastní

vysvětlení (toto není běžná praxe, je to čistě z důvodu kontroly projektu, tj. bez účelné zpětné vazby zadavatele).

2. Z upravené specifikace vytvořte graf příčin a důsledků (CEG graf). Při stanovení časových závislostí můžete definovat časová pásma, náležitosti příčin a důsledků a jejich souvislosti.
3. Identifikujte vstupní parametry testu. Parametry mohou být přímé (jejich hodnota se přímo objeví v testu) nebo nepřímé (další data z nich budou odvozena), např. "čas vytvoření požadavku č. 3" nebo "maximální váha všech požadavků".
4. Identifikujte charakteristiky parametrů a definujte jejich bloky. Zaměřte se převážně na charakteristiky plánu požadavků a přesunu materiálu (místa, časy požadavků, délky převozu, ale také závislosti mezi těmito aspekty). Specifikujte omezení kombinace bloků různých charakteristik. Navrhněte minimální (suboptimální) kombinace dvojic charakteristik (kritérium Pairwise Coverage).
5. Navrhněte testovací sadu pro *plánovač se zpětnými voláními* (viz soubor `cartctl_test.py`) zahrnující testovací scénáře z rozhodovací tabulky CEG grafu z bodu 2 (měly by být pokryty všechny příčiny a důsledky, obdoba Condition coverage a Decision coverage) a testy odpovídající kombinacím dvojic z bodu 4.

Odevzdání

Odevzdejte archiv proj1.zip obsahující následující strukturu:

```
proj1.zip
  +- specifikace.md     - upravená specifikace (formát Github
  |                       Markdown, formátování textu se pokud
  |                       možno vyhněte),
  +- ceg.txt            - graf příčin a důsledků (formát
  |                       akceptovaný ceg.testos.org)
  +- combine.json       - nastavení pro nástroj combine (volitelné)
  +- testy.md           - dokumentace testů (formát Github Markdown)
  |
  `- cartctl/           - adresář obsahující testovací scénáře vč.
      |                   kódu SUT a jeho okolí
      +- cartctl_test.py  - Hlavní testovací soubor
      |    ...
      `- *test*.py        - volitelné pomocné testovací soubory
```

Poznámky

System Under Test

V informačním systému, v e-Learningu u zadání projektu je v archivu `cartctl.zip` k dispozici příklad implementace vozíku, resp. jeho řízení (control,

ctl). SUT je `cartctl.py`. V archivu je rovněž k dispozici ukázkový testovací soubor:

```
python3 -m unittest cartctl_test.py
```

Úprava specifikace (dle Ambiguity review)

Váš report z úpravy (`specifikace.md`) bude mít následující formát:

```
Jedna věta, která byla zkopírována z původní specifikace.

  - CATEGORY, krátké vysvětlení.
  - CATEGORY, další typ chyby včetně krátkého popisu.

*Upravená věta opravující odhalené chyby. Doplňující věta, pokud
je třeba informaci přidat.*

Další věta z původní specifikace.

  - CATEGORY, další chyba další větě...
```

Kde `CATEGORY` je kategorie chyby Ambiguity review. Použijte následující identifikátory (ze seznamu níže použijte pouze identifikátory, krátké vysvětlení napište specifické k dané větě):

- DANGLING_ELSE - missing else or potential else misinterpretation
- UNSPECIFIED_SUBJECT - wrongly specified or unspecified subject
- AMB_SUBJECT - ambiguous subject
- AMB_REFERENCE - ambiguity of reference
- OMISSION - missing effects or causes
- AMB_LOGIC - ambiguous logical operators (or, and, implicit connectors, compound operators, ...)
- AMB_STATEMENT - ambiguous statements (verab, adjectives, variables, aliases)
- RANDOM_STATEMENT - random organization, mixed causes, random case sequence (logical/chronological sequence)
- IMPLICIT - implicit, silent, built-in cases or assumptions
- AMB_TEMPORAL - temporal ambiguity
- AMB_BOUNDARY - boundary ambiguity
- OTHER - category for the rest of ambiguity review problems

Dokumentace testů

V souboru `testy.md` musí být dokumentace testů. Skládá se z několika částí:

1. CEG graf (obrázek) a výsledná rozhodovací tabulka (může být screenshot tabulky z ceg.testos.org). Dále nechť je test z rohodovací tabulky označován "testovací scénář".

2. Identifikace vstupních parametrů, které budou tvořit test, ve formě tabulky dvou sloupců (název/identifikátor parametru, stručný popis).
3. Popište charakteristiky parametrů ve formě tabulky (https://docs.github.com/en/github/writing-on-github/organizing-information-with-tables) vč. omezení, příklad níže.
4. Identifikujte požadavky (testy) na kombinace všech dvojic bloků (může být screenshot tabulky z nástroje combine).
5. Charakteristiky a omezení uložte do souboru `combine.json`. Obsah tohoto souboru by měl být použitelný pro nástroj combine.testos.org. Získat ho můžete odchycením HTTP požadavku z webového prohlížeče (Např. Chrome / Developer tools / Network / endpoint `generate` po stisku tlačítka Generate / Headers, část Request payload). Webový nástroj není výkonnostně připraven ani chráněn, prosím opatrně. Nástroj combine (https://pajda.fit.vutbr.cz/testos/combine) můžete rozjet i lokálně.

Příklad formátu popisu charakteristiky:

Cx	Popis charakteristiky Cx
1	popis prvního bloku
2	popis druhého bloku
...	...
n	popis bloku n

- Cx.1 -> !Cy.3

Hodnocení

K dobrému hodnocení přispěje:

- správné použití nástrojů (ceg.testos.org a combine.testos.org),
- použití omezení v CEG grafu,
- správná definice charakteristik,
- použití omezení u definice charakteristik a bloků.

Poznámka z fóra ATA 2020/21

- Hlavním cílem studentského projektu je naučit se pracovat s vybranými technologiemi a vyzkoušet si vybrané metody. Zde se jedná o Ambiguity review, CEG, modelování vstupní domény, kombinační testování a implementace automatických testů.
- Podrobnější ambiguity review povede automaticky ke složitějšímu CEG a to povede k větším požadavkům na testy. Pokud budete mít příliš složitý CEG, ovšem efektivní tj. bez zbytečných ekvivalencí, nemusíte programovat úplně všechny testy. Rozumný počet automatických testů je 5-15.

- Každý test zdokumentujte. Mělo by být trasovatelné, který požadavek na test daný test pokrývá (kterou kombinaci z rozhodovací tabulky a/nebo kterou kombinaci bloků parametrů). Pro tuto vazbu zvažte vyjádření pomocí tabulky (TestId, TestovacíScénář, KteréKombinaceBloků), případně jiné tabulky, pokud budete mít rozumně malý počet charakteristik a bloků: (TestId, TestovacíScénář, C1.Blok1, C1.Blok2, ..., C2.Blok1, ..., CN.BlokM). Tabulka bývá přehledná, ale není nutná. Jen pro informaci: pro úplnou dokumentaci se používají tabulky oboustranné (pokud jsou příliš velké): první tabulky vyjadřující, který požadavek je pokryt kterým testem, a další tabulky pro vyjádření, který test pokrývá které požadavky.
- Kritérium pokrytí CEG a vstupní metody (combine) spolu nesouvisí. Smyslem projektu je vyzkoušet splnit dvě nesouvisející kritériia pokrytí. Ovšem můžete vytvořit testy, které se mapují na požadavky na testy ze dvou kritérií (tj. jednou ranou zabijete mouchy dvě).
- Chyby v implementaci mohou být dvojího druhu: zamýšlené a nezamýšlené. Zamýšlené chyby jsou dány volbou i implementací plánovacího algoritmu - ten je velmi trivální, který nedokáže uspokojit všechny rozumné požadavky. Nezmýšlené jsou dány moji nedůkladnou kontrolou při implementaci - tedy nevím o nich. Pokud se vám podaří chybu opravit, prosím zařaďte do archivu také patch soubor a dodatečný popis opravy. Aktuální verze, kterou máte k dispozici, je dána výběrem některých patchů vašich předchůdců.