TestFramework

## Implementácia v Jim hrácovi

Implementované boli dve nové triedy TestFrameworkCommunication a Message. Prvá z týchto tried je spustená hneď pri spustení hráča a jej úlohou je pripojiť sa na TCP server testovacieho frameworku, udržiavať spojenie a posielať správy. Určenie adresy samotného testovacieho frameworku ako aj možnosť zapnutia samotnej spätnej väzby je vykonané prostredníctvom konfiguračného súboru premennými „TestFramework\_monitor\_enable„“ a „TestFramework\_monitor\_address“. Trieda Message následne umožňuje vytváranie správ na odoslanie a zabezpečuje správny formát týchto správ. Vytvorenie konkrétnej správy je umožnené rôznymi statickými metódami. Pri spustení triedy s nedostatočnými informáciami sú doplnené údaje z konfiguračného súboru (napríklad tftp port pre vykonávanie príkazov)

## Implementácia v TestFrameworku

Vytvorená bola nová trieda AgentMonitor, ktorý zohráva úlohu TCP servera pre prichádzajúce nové spojenia. Po príchode nového spojenia je vytvorená nová inštancia triedy AgentMonitorThread, ktorá vykonáva čítanie správ z daného spojenia a následne ich posiela na spracovanie triede AgentMonitorMessage. Trieda AgentMonitorThread vytvára ešte nový typ správy TYPE\_DESTROY v prípade zrušenia spojenia s daným hráčom. Po úspešnom spracovaní správy sú notifikované všetky registrované triedy implementujúce rozhranie IAgentMonitorListener na počúvanie (návrhový vzor observer). Pri registrovaní na počúvanie je možné určiť presne pre ktoré typy správy chce byť daná trieda notifikovaná a to určením čísla hráča, jeho tímu alebo typu správy.

**Vytvorenie testu**

Pre vytvorenie nového testu na skúšanie hráča je vytvoriť triedu ktorá dedí od „sk.fiit.testframerotk.trainer.testsuite.TestCase“. Poskytuje následovné metódy ktoré je možné „overridovať“:

• init()

• isStopCriterionMet(SimulationState)

• evaluate(SimulationState)

• destroy()

Metóda init je spúštaná tesne pred začat' vykonávania testu. Vhodné je teda vykonať akúkoľvek inicializáciu (pripojiť hráčov, nastaviť ich pozíciu, pozíciu lopty a iné). Metóda má návratovú hodnotu boolean. Pri vrátenej hodnote „false“ je inicializácia a teda aj celý test požadovaný za neúspešný a test končí.

Metóda „isSopCriterionMet“ je volaná v častým intervaloch. Jej návratovou hodnotou je boolean jej hlavnou úlohou je zisťit či má test skončiť. Pri hodnote „true“ test končí.

Po ukončení behu testu je spustená metóda „evaluate“. Jej úlohou ve určit úspešnosť testu. Návratová hodnota testu je objekt „TextCaseResult“ do ktorého je možné uložit výsledok testovania. Tento výsledok dostanú všetky listenery na dianie testov.

Posledná metóda je „destroy“ v ktorej je možné dealokovať všetky nepotrebné objekty (aj hráčov).

Okrem týchto metód trieda „TestCase“ poskytuje aj rôzne premenné. Poskytujú prístup k bežným objektom a premenným testovacieho frameworku.

* RobocupServerAddress address
* RobocupServer server
* RobocupMonitor monitor
* SimulationState simulationState
* AgentMonitor agentServer

Spusťit testCase je následovne možné pridať do rady na vykonanie pomocou inštancie aktúalnej „Implementácie“ aplikácie. Tent krok je ale väčšinou vykonávaný inou častou testovacieho frameworku (GUI, sama implementácia pomocou argumentov na spustenie, …). Pri spúštaní testu je možné zaregistrovať objekt ako listener na prijatie výsledku daného testu.

Implementation impl = ImplementationFactory.getImplementationInstance();

impl.enqueueTestCase(testcase, listener);

**Spustenie a vyhodnotenie testov**

Každý test dedí od triedy „TestCase“. Ich spúštanie je umožnené pomocou konkrétnej Implementácie rozhrania „Implementation“, ktorá riadi celý beh aplikácie. Pridanie testu na vykonanie v danej implementácii je umožnené pomocou metódy:

enqueueTestCase(TestCase testCase, ITestCaseObserver observer)

Druhý argument tejto metódy je listener (môže byť null), ktorý je následne upozornení na výsledok testu (TestCaseResult objekt ktorý daná konkrétna TestCase implementácia vráti). Spúštanie viacerých testov naraz a následne porovnávanie výsledok je umožnené napríklad nádledovným Runnable objektom:

public class StandUp implements Runnable, ITestCaseObserver {

// … inicializacia objektu

public void run() {

Implementation impl = ImplementationFactory.getImplementationInstance();

impl.enqueueTestCase(new StandUpTest(), this);

impl.enqueueTestCase(new StandUpTest(), this);

impl.enqueueTestCase(new StandUpTest(), this);

impl.enqueueTestCase(new StandUpTest(), this);

impl.enqueueTestCase(new StandUpTest(), this);

}

// notifikacia o ukonceny testu – teda ulozenie vysledku

public void testFinished(TestCaseResult result) {

testResults.add(result);

if (testResults.size() == 5) {

evaluateFastTestResults(testResults);

testResults.clear();

}

}

// vyhodnotenie celkoveho vysledku

public void evaluateFastTestResults(List<TestCaseResult> results) {

double result = 0;

for (TestCaseResult testCaseResult : results) {

result += testCaseResult.getFitness();

}

}

}

V opisovanom príklade je spustení 5-krát test „StandUpTest“. Pri ukončení jedného z testov je vždy spustená metóda „testFinished“, ktorá výsledok testu uloží. Pri ukončení všetkých testov (výsledkov je 5) sa celkové časy spočítaju a je dostupný finálny výsledok testovania.

## Anotácie

Na základe uvedených požiadaviek sme sa rozhodli ukladať anotácie k pohybom v samostatných súboroch typu XML. Takto dostupné anotácie je možné pomerne jednoducho spracovávať a meniť tak človekom ako i strojom. V prípade potreby je možné takúto reprezentáciu anotácie vložiť priamo do XML súboru pohybu. V tabuľke 1 sú popísané všetky podstatné atribúty anotácie.

**Viacero anotácií pre jeden pohyb**

Pri vytváraní XML súboru s anotáciou sa program pozerá do používateľom zadaného výstupnéhho priečinka a postupuje nasledovne: 1. nájde všetky anotácie patriace tomu pohybu, ktorý bol práve testovaný, 2. nájde súbor s najvyšším poradovým číslom, 3. vytvorí súbor s poradovým číslo o jeden vyšším. Všetky tieto operácie prebiehajú na úrovni mien súborov, preto si na ne treba dávať pozor a udržiavať ich v nasledovnom formáte: *meno\_pohybu-poradové\_číslo\_anotácie.xml*.