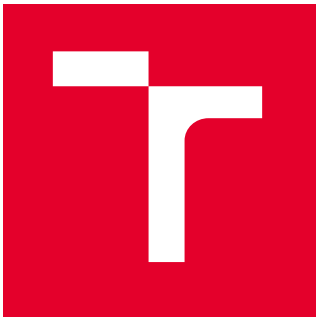
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2023 Vladimír Peňáz



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ROBOCODE - ZABEZPEČENÁ PLATFORMA PRO HODNOCENÍ STUDENTSKÝCH PROJEKTŮ

ROBOCODE - SECURED PLATFORM FOR EVALUATION OF STUDENTS' PROJECTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Vladimír Peňáz**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Radim Burget, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Vladimír Peňáz ***ID:*** 227802

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Robocode - zabezpečená platforma pro hodnocení studentských projektů

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s potřebami projektů předmětu MSC-PDA, které jsou postaveny na hře Robocode. Pro potřeby předmětu navrhněte testovací platformu, s pomocí které bude možné zajistit práva superuživatele a současně předejít riziku útoku či poškození výpočetní stanice či jejich dat, kde dochází k hodnocení prací studentů. V prostředí jazyka JAVA navrhněte platformu, kde se studenti budou moci přihlásit k hernímu serveru a soutěžit mezi sebou. Implementujte jednoduchý příklad v jazyce Python, kde bude možné trénovat chování robota. Pro potřeby studentů vytvořte jednoduchý návod, kde dejte důraz na jednoduchost a přehlednost. Demonstrujte řízení robota ze strany klienta (studenta).

DOPORUČENÁ LITERATURA:

podle pokynů vedoucího práce

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: doc. Ing. Radim Burget, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zaměřuje na návrh a implementaci bezpečné testovací platformy založené na hře Robocode, která slouží k hodnocení studentských projektů v rámci předmětu MSC-PDA. Projekt využívá principy strojového učení a řeší problém třídy složitosti EXPSPACE. Hodnocení kvality výsledků v této třídě složitosti je obtížné a aktuálně neexistuje vhodné prostředí pro tyto účely. Cílem práce je vytvořit bezpečné prostředí, které umožní studentům soutěžit na herním serveru s minimálním rizikem poškození učitelské výpočetní stanice a zajištěním práv superuživatele. Studenti budou své natrénované modely připojovat k hernímu serveru, odkud získají kompletní informace o dění na bitevním poli a podle nich vygenerují instrukce pro svůj tank. Tímto způsobem bude model disponovat stejnými informacemi o bitvě jako člověk hrající manuálně. Na základě konečného skóre bude možné vyhodnotit, který model dosáhl nejlepšího výsledku a označit ho jako nejlepší. Platforma je implementována v jazyce Java a pracuje s modely implementovanými v jazyce Python.

KLÍČOVÁ SLOVA

Robocode, klient-server, Genetické programování, Zpětnovazební učení, TCP/IP, Java, Python, testovací platforma, bezpečnost, hodnocení ML, semestrální projekt

ABSTRACT

This bachelor's thesis focuses on the design and implementation of a secure testing platform based on the game Robocode, which is used for evaluating student projects in the MSC-PDA subject. The project utilizes principles of machine learning and addresses a problem in the complexity class EXPSPACE. Evaluating the quality of results in this complexity class is challenging, and currently, there is no suitable environment available for these purposes. The objective of this thesis is to create a secure environment that allows students to compete on a game server with minimal risk of damaging the teacher's computer and ensures superuser privileges. Students will connect their trained models to the game server, where they will receive complete information about the battlefield, based on which they generate instructions for their tanks. In this way, the model will have the same information about the battle as a manually playing human. Based on the final score, it will be possible to evaluate which model performed the best. The platform is implemented in Java and works with models implemented in Python.

KEYWORDS

Robocode, client-server, Genetic programming, Reinforcement learning, TCP/IP, Java, Python, testing platform, security, ML evaluation, semester project

PEŇÁZ, Vladimír. *ROBOCODE - ZABEZPEČENÁ PLATFORMA PRO HODNOCENÍ STUDENTSKÝCH PROJEKTŮ*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 46 s. Bakalářská práce. Vedoucí práce: doc. Ing. Radim Burget, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Vladimír Peňáz
VUT ID autora:	227802
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	ROBOCODE - ZABEZPEČENÁ PLATFORMA PRO HODNOCENÍ STUDENTSKÝCH PROJEKTŮ

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc.Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

V současné době jsou stále více skloňovány pojmy jako umělá inteligence či strojové učení a není divu, tyto technologie každým dnem dokazují že si tuto míru pozornosti opravdu zaslouží. Se zvýšeným zájmem o problematiku souvisí potřeba praktického vzdělávání nových odborníků v tomto oboru. Problémem ovšem je, že při praktické výuce metod strojového učení je velmi obtížné nalézt zajímavou problematiku, kterou by studenti měli rašit a zároveň je velmi komplikované zhodnotit, jak dobrou práci studenti odvedli. U složitějších problémů totiž nelze jednoznačně říci, zda je nějaké řešení správné či nikoliv, případně porovnat vice řešení zároveň a zhodnotit, které řešení je nejlepší.

Cílem této práce je návrh a následná implementace platformy pro bezpečné využívání hry Robocode při výuce genetického programování (GP) a zpětnovazebního učení (RL). Jedná se o jednoduchou hru, ve které hráč ovládá tank umístěný v bojišti. S tankem se může volně pohybovat a v případě potřeby může vystřelit, kdy za úspěšný zásah získává bodové hodnocení. V základní verzi hra probíhá v módu všichni proti všem, tudíž je cílem každého tanku nasbírat co nejvyšší počet bodů. V případě implementace strojového učení, je změna pouze v tom, že hráč svůj tank neovládá manuálně, ale nechá pracovat algoritmus, aby prováděl tahy automaticky, na základě naučeného modelu. V rámci navrženého cvičení budou studenti během semestru navrhovat a trénovat funkční modely, pomocí RL a GP. Na konci semestru bude uspořádán zápas, kdy se do hry proti sobě nasadí všechny výstupy studentských projektů a na základě získaného bodového skóre se vyhodnotí ty nejlepší implementace.

Ambice k vytvoření takové platformy pramení z potřeby objektivního hodnocení výsledků RL. Problematika, kterou budou studenti v rámci projektu řešit, spadá do třídy složitosti EXPSPACE, u které se velmi obtížně porovnává správnost řešení a aktuálně neexistuje platforma, která by byla schopna vzájemně porovnat hned několik různých modelů najednou. Hra Robocode se již řadu let pro podobné účely využívá a na internetu lze nalézt nemalé množství kvalitně natrénovaných modelů. Mohlo by se zdát, že se jedná o vhodnou platformu pro požadovanou výuku, bohužel to není úplně pravda. Pravdou je, že se jedná o stabilní a plně funkční hru, která vyhovuje všem požadavkům, akorát je relativně komplikované spustit více modelů najednou a bohužel tu je problém s bezpečností. Při využití hry k výuce by bylo nutné, aby byly všechny hodnocené modely spuštěny na hodnotícím zařízení (počítač vyučujícího). V takové situaci studentům nic nebrání v přidání škodlivého kódu k připravenému modelu, čímž mohou poškodit vyučujícího.

Práce řeší tento bezpečnostní problém pomocí modifikace herní platformy, tak aby bylo možné připojit se ke hře vzdáleně ze svého zařízení. S touto úpravou by

stačilo, aby vyučující na svém zařízení pouze spustil hru, studenti by se k ní vzdáleně připojili a pomocí vytvořeného protokolu pouze zadávali instrukce svému tanku. Takto by se vyučující vyhnul nutnosti spouštět cizí kód na svém počítači. Zároveň by využití hry Robocode při výuce mohlo studenty více motivovat k práci. Tato motivace by mohla být rovněž podpořena závěrečným hodnocením, které bude probíhat formou souboje.

Celou platformu se úspěšně podařilo navrhnout a implementovat včetně Python kódu demonstrujícího generovaní instrukcí a následné předání instrukcí platformě. Zároveň byla platforma úspěšně otestována s více připojenými zařízeními, kdy každé mělo svůj vlastní algoritmus generující instrukce.

Přínosem práce je návrh a implementace bezpečného prostředí, ve kterém se mohou studenti prakticky vzdělávat v oblasti strojového učení, aniž by měli přímou možnost poškodit zařízení vyučujícího. Zároveň tato práce přináší unikátní způsob hodnocení RL modelů, který je dobře využitelný ve výuce.

Zbytek práce se skládá z pěti částí. Tou první je teoretický úvod do problematiky RL, GP a hodnocení kvality těchto řešení, dále jsou popsána bezpečnostní rizika spojená s využíváním platformy Robocode. Druhá část se zabývá návrhem cvičení pro účely předmětu MCS-PDA, návrhem platformy a protokolu. Další část je zaměřena na popis implementace zabezpečené platformy. Část čtvrtá obsahuje zhodnocení výsledků s popisem testování a následnou diskusi nad výsledkem práce. Poslední částí je závěr, kde je celá práce celkově shrnuta.

1 Teorie

V současnosti existuje několik platforem, na kterých je možné si vyzkoušet aplikaci metod zpětnovazebního učení a genetického programování. Častým akademickým příkladem může být hra piškvorky, některé arkádové hry z rodiny Atari nebo třeba Robocode. Každá ze zmíněných variant má svá specifika a hodí se pro různý způsob výuky. Hlavní předností hry Robocode je možnost zapojení libovolného množství hráčů, díky čemuž je vyučující schopen zapojit celou třídu tak aby si všichni byli vzájemně rivaly. Tento aspekt by mohl studenty motivovat k vložení většího úsilí do řešení projektu.

Úskalím aktuální implementace Robocode je, že funguje lokálně, což znamená velké bezpečnostní riziko v případě, kdy chce uživatel na své stanici spouštět jiné tanky než své vlastní. V prostředí školy by tato situace nastala v případě, kdy by chtěl vyučující na svém počítači ohodnotit řešení studentů. V takovém případě autoru algoritmu nic nebrání v tom, aby do své implementace vložil část škodlivého kódu, čímž může poškodit zařízení na kterém je robot spuštěn.

1.1 Úvod do zpětnovazebního učení

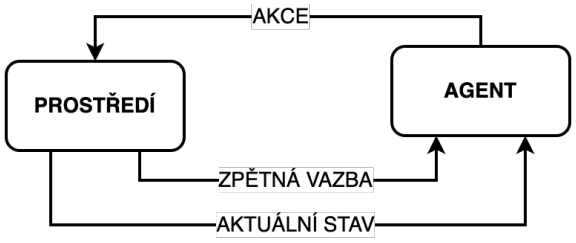
Vedle učení s učitelem či bez učitele je zpětnovazební učení další variantou strojového učení, tento způsob je konkrétně využíván k optimalizaci strategie ve specifickém prostředí. Zpětnovazební učení má tři základní prvky, jedná se o prostředí s agentem, kteří spolu v cyklech pravidelně komunikují, a zpětnou vazbu skrze kterou je hodnoceno chování agenta.

Prostředí je prostor, do kterého je agent umístěn a se kterým má interagovat (například herní pole nebo bludiště), po každé iteraci prostředí ohodnotí výkon agenta zpětnou vazbou, která reflektuje kvalitu řešení.

Zpětná vazba je hodnota, kterou prostředí hodnotí kvalitu agentova řešení, hodnocení závisí na konkrétní implementaci. Jednou možností je binární hodnocení (0 = špatně, 1 = správně). Další variantou je definovaný interval, který při hodnocení umožňuje vyšší variabilitu, opět zde platí pravidlo, že čím vyšší hodnocení, tím lepší implementace.

Agent většinou představuje algoritmus nebo robota, který je umístěn v prostředí, se kterým je chopen interagovat, dokáže sledovat jeho stav a má definovanou určitou množinu činností, kterými může prostředím manipulovat. V různých implementacích není nutné, aby měl agent informace o celém prostředí a dostává se mu informací pouze o konkrétní části, případně se definují informace, ke kterým by neměl být schopen přistoupit. Na základě získaných informací agent provádí sérii kroků vedoucích k cíli. V každé iteraci od prostředí získá zpětnou vazbu, kterou přičte k celkové

zpětné vazbě. Hlavním cílem agenta je maximalizace celkové zpětné vazby, čemuž uzpůsobuje jednotlivé kroky. Na základě zvolené strategie bere v úvahu různě dlouhý časový horizont, případně je chopen obětovat zpětnou vazbu za aktuální krok na úkor potenciálně vyššího zisku v kroku budoucím. Existují i varianty, kdy je v prostředí agentů více, taková implementace je nazývána multi-agentní [2], [3], [4], [5], [6].



Obr. 1.1: Schéma zpětnovazebního učení

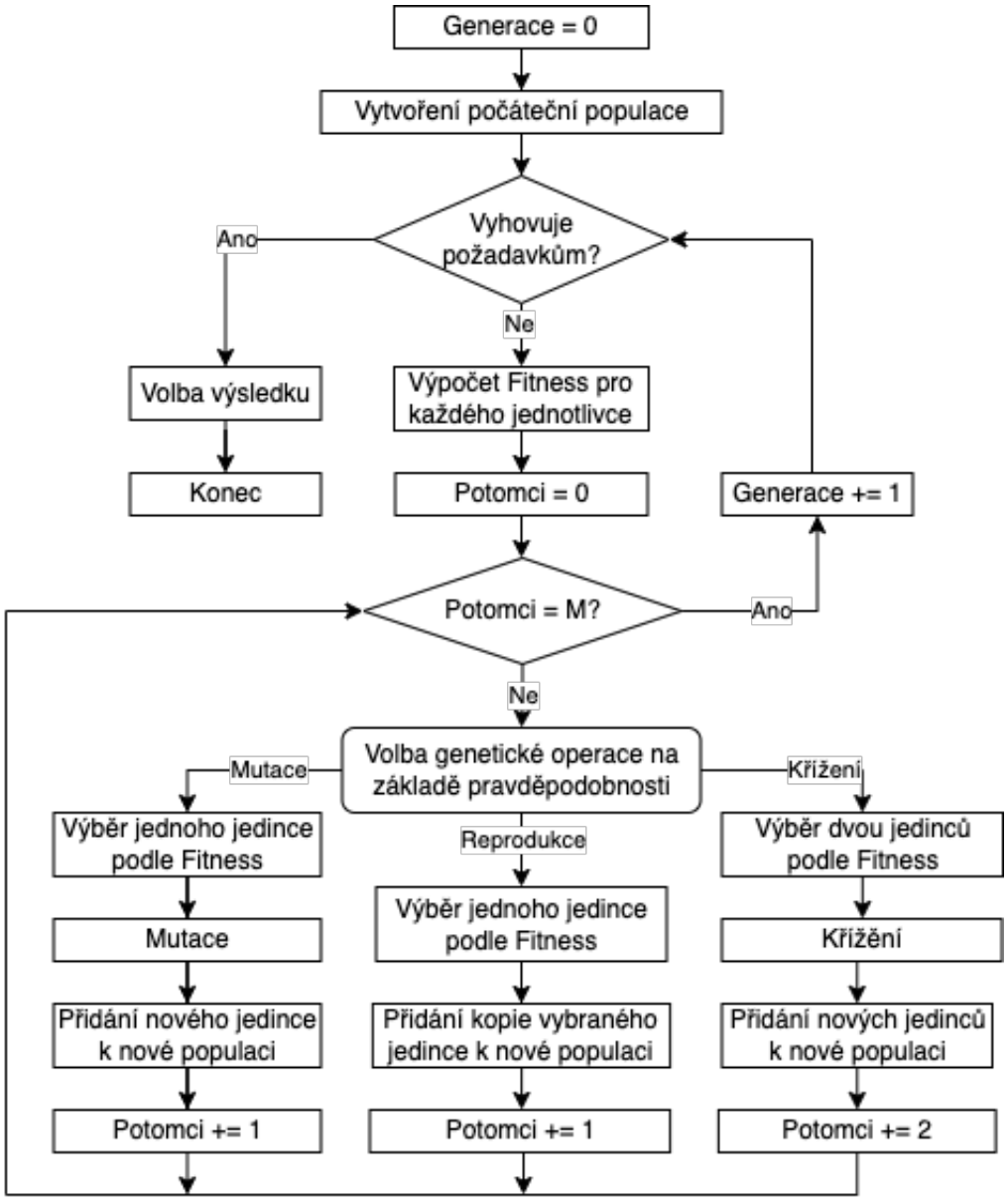
1.1.1 Problémy zpětnovazebního učení

Při implementaci je nutné brát v úvahu i limity, které se zpětnovazebního učení týkají. Častým komplikací může být způsob jakým je stanovována odměna. Pokud je možné stanovit odměnu na základně jedné proměnné (například čas), je jednoduché zpětnou vazbu definovat správně. Problém nastává v situaci, kdy je sledováno více cílů (například čas, získané skóre a počet tahů), tehdy je nutné správně definovat váhu jednotlivých cílů, aby byl model schopen správného učení. Nevhodné nastavení těchto hodnot může způsobit nedostatečnou exploraci prostředí. Problémem je, že ne vždy je správný podíl vah zřejmý a může být velmi obtížné správné nastavení nalézt. Dalším problémem je nedeterministické prostředí nebo prostředí u něhož nemáme jistotu, že v něm existují konečné stavy. Optimalizace chování agenta v takovém prostředí se svojí složitostí může řadit i do kategorie NP-těžkých problémů a takové problémy jsou v současnosti neřešitelné [2], [3].

1.2 Úvod do genetického programování

Genetické programování (GP), je metoda programování založená na principech Evolučních algoritmů (EA). Celá kategorie EA je přímo inspirována biologií, konkrétně biologickou evolucí a jejími procesy, které byly popsány a přeneseny do výpočetního prostředí. GP má ovšem svá specifika, která jej definují a odlišují od ostatních EA. Dvěma hlavními odlišnostmi jsou flexibilní délka výstupu a formát jaký výstup má. Jak již bylo zmíněno, výstup GP má variabilní délku, což jej odlišuje například od Genetických algoritmů, kde je délka fixně definovaná, tato variabilita významně

zvyšuje potenciální komplexnost výstupu, ale může také zapříčinit zvýšení složitosti. Druhým specifikem GP je jeho výstup, v rámci učení se pracuje s podmínkami a jednotlivé operátory do výstupu zanořují nebo v něm modifikují `if-else` výrazy. Výstup je tedy reprezentován jako rozhodovací strom s variabilní délkou [1], [7], [8].



Obr. 1.2: Schéma genetického programování [1]

1.2.1 Princip

Jak již bylo zmíněno, GP je silně inspirováno evolucí a od toho se odvíjí i jednotlivé jeho kroky. V první ředě je nutné správně definovat jedince, respektive jeho DNA.

Každý jedinec má své geny, které v tomto případě představují proměnné parametrů případně uzly stromu, které se skládají do chromozomů, představujících DNA jedince. Dále je nezbytné definovat Fitness funkci aby bylo možné hodnotit vhodnost jedinců. Jakmile jsou tyto podmínky splněny je možné GP spustit [1].

V prvním kroku je vygenerována iniciální populace čítající `M` jedinců. Tato populace je otestována, zda se nejedná o řešení. V případě že ano, je proces ukončen jelikož jsme již v prvním kroku dosáhli cíle. Pokud se ovšem v populaci řešení nenachází, je potřeba v algoritmu pokračovat. Dalším krokem je ohodnocení všech jedinců aktuální populace pomocí Fitness funkce. Dále je inicializovaná nová prázdná populace, do které jsou přidáváni noví jedinci, kteří vznikají z nejlepších jedinců pomocí modifikačních operátorů (reprodukce, mutace a křížení). Takto se děje, dokud nemá nová populace požadované množství jedinců. Jakmile jej dosáhne, přestanou se vytvářet noví jedinci a nová populace je opět otestována, zda neobsahuje požadované řešení. Tento proces se opakuje tak dlouho, dokud není nalezeno vyhovující řešení, případně, u některých implementací, dokud není naplněn limit v počtu vytvořených generací. Celý proces je schematicky znázorněn na obrázku 1.2, [1].

1.2.2 Fitness funkce

Funkce fitness plní v GP zásadní roli, jejím úkolem je ohodnocení kvality/vhodnosti jedinců. Je nezbytné věnovat implementaci této funkce nemalé množství pozornosti, jelikož pokud by fitness funkce nehodnotila jedince správně, je velmi malá šance na úspěšný výstup GP. Na základě nevhodné funkce by byli upřednostňováni nesprávní jedinci, díky čemuž by se evoluce vyvíjela nesprávným směrem.

Fitness funkci lze používat v různých variantách, podle potřeby se zvolí ta nejvhodnější varianta pro konkrétní implementaci. Mezi nejpoužívanější varianty patří nezpracovaná, standardizovaná a normalizovaná fitness. Jejich účel je ve všech případech stejný, liší se pouze tvar výstupu, například u normalizované fitness je cílem snížit rozsah v jakém můžeme získávat její výstup oproti nezpracované [1], [7].

1.2.3 Modifikační operátory

Jedná se o operátory, kteří zajišťují aby při vzniku nové generace došlo k evolučnímu posunu. Pro každého z vybraných jedinců se na základě definované pravděpodobnosti vybere jeden z operátorů, což definuje jak bude nový jedinec vytvořen [1], [7].

Křížení

Operátor křížení reflektuje princip rozmnožovaní u dvojpohlavních organismů. Na základě fitness funkce, případně jiného výběrového kritéria, jsou zvoleni dva jedinci,

kteří jsou zkříženi. Křížení probíhá náhodně, kdy se prohazují části (podstromy) obou jedinců. Výsledkem jsou dva noví jedinci [1], [7].

Reprodukce

Při procesu reprodukce by měli být vybíráni ti jedinci, kteří mají nejlepší hodnoty fitness funkce. U takových jedinců je žádoucí, aby se jejich genom zachoval i do další generace. Proces reprodukce si lze představit jako asexuální rozmnožování, při kterém vznikne kopie jedince která je zařazena do nové populace [1], [7].

Mutace

Podle implementace se liší, zda je mutace považována za primární či sekundární operátor. Záleží, zda je použita samostatně na stejné úrovni jako křížení a reprodukce, pak ji považujeme za primární nebo zda se jedná o krok, který výjimečné nastává v rámci křížení či reprodukce, v takovém případě je sekundární. Cílem mutace je přidat k procesu vytváření nové populace jistou míru náhodnosti. Díky tomu se zvyšuje diverzita nových populací a snižuje závislost na populaci předchozí [1], [7].

1.2.4 Využití

Vzhledem k náročnosti implementace a následně vysokým nárokům na výpočetní kapacitu je vhodné důkladně zvážit, zda je využití GP tou nejvhodnější variantou pro řešení konkrétního problému. Ačkoliv se jedná o mocný nástroj, GP není schopné řešit jakýkoliv problém, ale pouze specifickou množinu. Ve většině případů lze říci, že využití GP je na místě při řešení problémů,

- kde je cílem nalezení velikosti a tvaru konečného řešení,
- kde jsou vazby relevantních proměnných vysoce nelineární,
- kde vazby relevantních proměnných jsou těžko uchopitelné nebo neznámé,
- kde se nedaří získat řešení pomocí konvenční matematické analýzy,
- kde je dostačující částečné řešení,
- kde je oceněno i drobné zlepšení výstupu,
- kde je potřeba zkoumat, klasifikovat a integrovat velké množství dat, ve formě čitelné počítačem,
- kde je pro člověka náročné naprogramovat řešení, ale má konkrétní představu o výstupu,
- kde existují vhodné měřící nástroje pro hodnocení kandidátských řešení [8].

1.3 Možnosti testování výsledků výuky RL a GP

Při výuce by bylo vhodné využít co možná nejzajímavější platformu, která by přilákala pozornost studentů a přiměla je věnovat projektu více pozornosti. V nejideálnějším případě by měla práce studenty na projektu bavit a motivovat k nejlepšímu výkonu. Zároveň by mělo být z výsledku zřejmé, o jak kvalitní řešení studentů se jedná. V neposlední řadě je podstatné aby byla platforma určena pro větší množství uživatelů, jelikož je cílem ohodnocení prací celé třídy zároveň. Z požadavků nepřímo vyplývá, že nejlepším kandidátem by byla jednoduchá hra pro více hráčů, ve které se nějakým způsobem počítá skóre, které by mohlo alespoň přibližně definovat, jak dobrá jsou jednotlivá řešení [9].

Platforem na výuku programování existuje na internetu nepřeberné množství a mnoho z nich lze lehce upravit tak aby na nich bylo možné aplikovat i metody strojového učení. Bohužel je většina těchto her určena pro jednoho hráče, tudíž nejsou vhodné pro využití v rámci výuky. Je zde ale pár variant, které jsou pro dané účely vyhovující. Mezi nejznámější platformy tohoto charakteru patří hra Robocode a arkádové hry společnosti Atari [9].

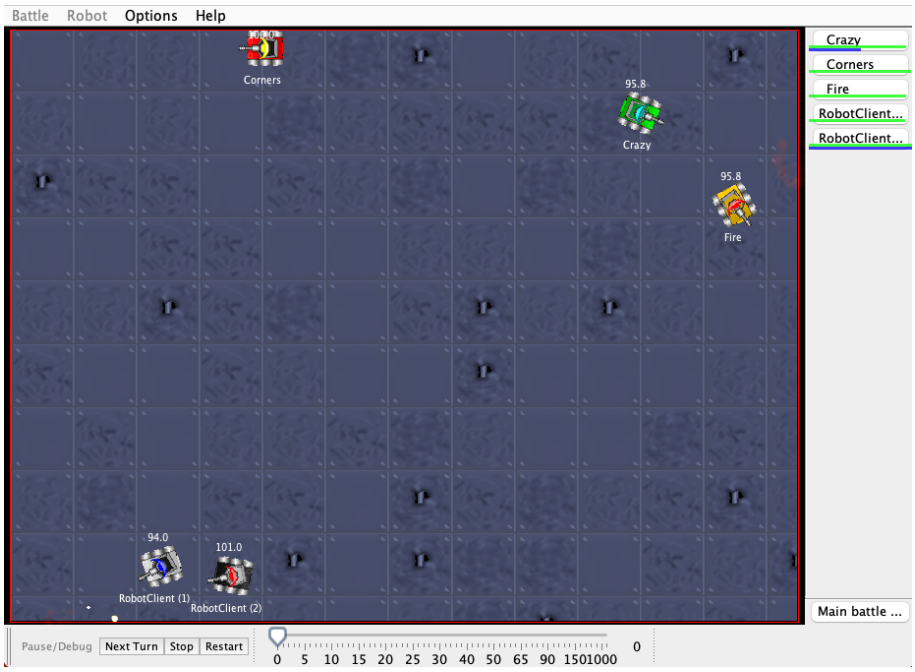
1.3.1 Robocode

Robocode je hra vytvořena v laboratořích IBM, která vyšla v roce 2001. Cílem tohoto projektu byl alternativní způsob výuky programování. Úkolem hráče je vytvoření rozhodovacího algoritmu pro jeho tank, tento tank je následně umístěn do arény s roboty, kde probíhá zápas, na jehož konci hráč získá bodové hodnocení svého výkonu. V současnosti našla hra nové využití, tím je výuka strojového učení, konkrétně je vhodná pro hodnocení u optimalizačních metod. Zásadní výhodou této hry je možnost zapojení více tanků najednou, což umožňuje účast všech studentských návrhů v jednom zápasu [10], [11].

Princip a pravidla

Hráč má omezené množství akcí, konkrétně se jedná o tři (pohyb tanku po celém bojišti, otáčení věže a výstřel). Hrací plocha je dvourozměrná a pohyb je možný dopředu, dozadu, případně lze tankem zatáčet (nelze se pohybovat bokem). Věž tanku se otáčí o 360° a je schopna střelby. Dále se na věži nachází radar, který detekuje nepřátelské tanky v okruhu 120 pixelů. Při střelbě je hráč schopen ovlivnit sílu výstřelu, čím je výstřel silnější, tím více poškození způsobuje, ale zároveň stojí tank který vystřelil více energie [11].

Hra probíhá v několika kolech. Hráči začínají kolo se 100 body energie, tyto body znázorňují jejich život, jakmile energie klesne na 0 je tank zničen. Energii tank



Obr. 1.3: GUI hry Robocode

ztrácí pokaždé, když je zasažen a když vystřelí aniž by zasáhl protivníka. Získat energii lze primárně při úspěšném zásahu nepřítele nebo případně pokud je některý z nepřátel zničen. Cílem hry není přežít jako poslední, ale získat co nejvyšší bodové skóre v rámci všech kol [11].

1.3.2 Atari

V případě Atari se nejedná o jednu konkrétní hru, ale spíše celý soubor. Jelikož jde o hry s relativně jednoduchým ovládáním, je taktéž možné aplikovat metody strojového učení. Dalo by se říci, že se jedná o zajímavou alternativu použitelnou při výuce, bohužel tomu tak není. Na tyto hry sice lze aplikovat metody GP a RL, což by nárokům vyhovovalo, ale u žádně z her zde není možnost zapojení většího množství hráčů než čtyř. To není pro účely výuky dostačující, vzhledem k množství studentů [12].

1.3.3 Robot Battle

Jedná se o alternativu hry Robocode, ale je zde několik rozdílů. Hlavním rozdílem je programovací jazyk na implementaci tanku. Nejedná se o žádný ze standardní sady jazyků, ale o vlastní jazyk přímo vyvinutý pro tuto platformu, tudíž je pro práci nutná jeho znalost. Zajímavým rozdílem je i rozšíření hry o předměty, které lze v rámci zápasů sbírat. Tyto předměty poskytují jejich držiteli výhodu v boji.

Přestože je herní rozšíření zajímavé, je tato hra těžko využitelná vzhledem k jejímu vlastnímu programovacímu jazyku [13].

1.3.4 Tank Battle

Tato hra je taktéž v principu velmi podobná jako Robocode, hráč si zde modifikuje svůj tank, který je následně nasazen do boje. Tato platforma vznikla v rámci bakalářské práce na Karlově univerzitě v Praze. Ačkoliv se jedná o zajímavou alternativu, jsou zde velmi omezené možnosti modifikace chování tanku, které nevyhovují požadavkům [13].

1.4 Potenciální bezpečnostní hrozby

Jak již bylo zmíněno problémem platformy Robocode je zásadní bezpečnostní riziko v případě, kdy je na stanici spouštěn cizí a neprověřený robot. Tento problém není nijak ojedinělý a toto riziko podstupuje majitel stanice kdykoliv, když na svém zařízení spustí neprověřený kód. Takový program v může obsahovat část, jejíž cílem je poškodit zařízení nebo jeho majitele. Může se jednat o program, který způsobuje viditelné škody na zařízení nebo o kód jehož cílem je nepozorované infiltrace zařízení (nahrání Malware).

Nejznámějším příkladem škodlivého kódu, který má přímo poškodit zařízení, je použití příkazu `rm -rf`, který na zařízení vymaže domovský adresář. Existují i situace, kdy útočník nemá zájem disk mazat ale chce jej pouze zašifrovat, aby mohl po poškozeném požadovat výkupné za dešifrování.

Infiltrované zařízení v běžném uživateli nevzbudí žádné podezření a Malware v něm může běžet klidně několik měsíců, v závislosti na jeho účelu. Některý má za úkol získat přihlašovací údaje nebo data uživatele, jiný umožní útočníkovi vzdálený přístup do počítače, dalším příkladem je využívání výkonu zařízení pro DDoS útoky. Přítomnost takového programu v zařízení se špatně odhaluje a často se stává, že pokud k jejich odhalení dojde bývá již pozdě.

1.4.1 Možná prevence

Řešením zmíněného bezpečnostního rizika je úprava platformy tak aby nebylo nutné spouštět program tanku na učitelské stanici. Zároveň je nutné umožnit hru více hráčům zároveň, pro účely hodnocení. Pokud uživatel nemůže důvěřovat vytvořeným tankům natolik aby je spustil na své stanici a zároveň je zde potřeba hry více tanků, řešením je implementace klient–server platformy, kam by se studenti mohli se svými roboty připojit a pouze předávat příkazy pro ovládání jejich tanku na hracím poli.

V praxi by souboj mohl vypadat tak, že lektor na svém zařízení spustí hru Robocode i s vizuální reprezentací a všichni studenti se do hry vzdáleně připojí ze svých počítačů na kterých spustí svého robota, který bude odesílat instrukce učitelskému zařízení. V rámci celého souboje tedy nenastane situace, kdy by musel vyučující na svém zařízení kompilovat programy studentů, čímž se vyvaruje riziku poškození jeho stanice.

1.5 Využitelné nástroje

Tato kapitola se zabývá výběrem správného programovacího prostředí pro vývoj zabezpečené platformy a také vyhovujícího komunikačního protokolu, pro předávání informací mezi klientem a serverem.

1.5.1 Jazyk

Při tvorbě platformy je nezbytné správně zvolit programovací prostředí. Vzhledem k vlastnostem by v úvahu přicházely jazyky Java a Python. V obou případech se jedná o bezpečné a hojně využívané jazyky, každý z nich má ale svá specifika. V případě Pythonu je hlavní předností jeho jednoduchost, díky tomu je tento funkcionální jazyk hojně využíván začínajícími programátory. Od náročnosti se odvíjí i časová náročnost programování, kdy vytvoření obdobného kódu v Pythonu je ve většině případů podstatně rychlejší než v jazyce Java. V případě druhého kandidáta, kterým je objektově orientovaný jazyk Java, je třeba vyzdvihnout vyšší rychlost, lepší škálovatelnost a o něco vyšší bezpečnost. Na rozdíl od Pythonu, se v Javě kód kompiluje před během programu, tudíž se zde nachází prostor pro bezpečnostní kontrolu kódu [14].

1.5.2 TCP/IP

Jedná se o model definující princip komunikace po síti, k tomuto modelu patří skupina protokolů, vhodných ke komunikaci na různých vrstvách. Na rozdíl od ISO/OSI modelu nemá sedm vrstev ale pouze čtyři. To neznamená, že by byla omezena jeho funkcionalita, pouze došlo ke sloučení některých vrstev. Vrstvy jsou Aplikační, Transportní, Síťová a Vrstva síťového rozhraní. Každá z nich má specifický účel a slouží ke komunikaci na jiné úrovni. [15], [16]

Vrstva síťového rozhraní

Jedná se o nejnižší z vrstev, jejím úkolem je zajištění přístupu k přenosovému médiu a řízení datového spoje.

Síťová vrstva

Tato vrstva je využívána k adresaci v síti. U této vrstvy je kladen velký důraz na rychlost, což se odráží na spolehlivosti, jelikož většina sítových protokolů není spojovaná a předpokládá se, že případnou ztrátu informace řeší vyšší vrstva. Protokoly využívanými na síťové vrstvě jsou například IP verze 4 nebo 6, ICMP nebo třeba ARP protokol.

Transportní vrstva

Transportní vrstva má za úkol zajišťovat celistvost přenášených dat, to se může projevit na rychlosti přenosu, proto jsou na této vrstvě nejčastěji využívány dva protokoly. Prvním je TCP, který je spojovaný a kontroluje zda nedošlo ke ztrátám, což ale snižuje jeho rychlost. Druhým je UDP, který není spojovaný, ale je podstatně rychlejší.

Aplikační vrstva

Jde o nejvyšší vrstvu TCP/IP modelu, která zajišťuje přenos konkrétních dat. V TCP/IP modelu se jedná o jednu vrstvu ale plní stejnou funkci jako nejvyšší tři vrstvy ISO/OSI modelu. Protokoly aplikační vrstvy jsou například SSH, DNS, HTTP, FTP,...

1.5.3 Protokoly

Pro účely komunikace mezi klientem a serverem je třeba zvolit vhodný transportní protokol pro přenos dat, směrování bude zajištěno IPv4 protokolem. Nejpoužívanějšími protokoly pro přenos dat jsou UDP a TCP. Hlavním rozdílem mezi těmito protokoly je spolehlivost a rychlost. UDP protokol totiž nezaručuje doručení paketu, jelikož u něj neprobíhá kontrola. Díky tomu je komunikace pomocí UDP rychlejší a má menší režii. To z něj dělá ideální volbu například pro streaming či oblast telefonie. Protokol TCP je spojovaný, to znamená, že dochází ke kontrole doručení paketu a v případe problému je odeslán opakovaně. Tento přenos je oproti UDP pomalejší a je zde větší režie, ale je zaručeno doručení všech paketů. Proto je tento protokol vhodný pro přenos dat [17].

2 Návrh semestrálního cvičení a platformy

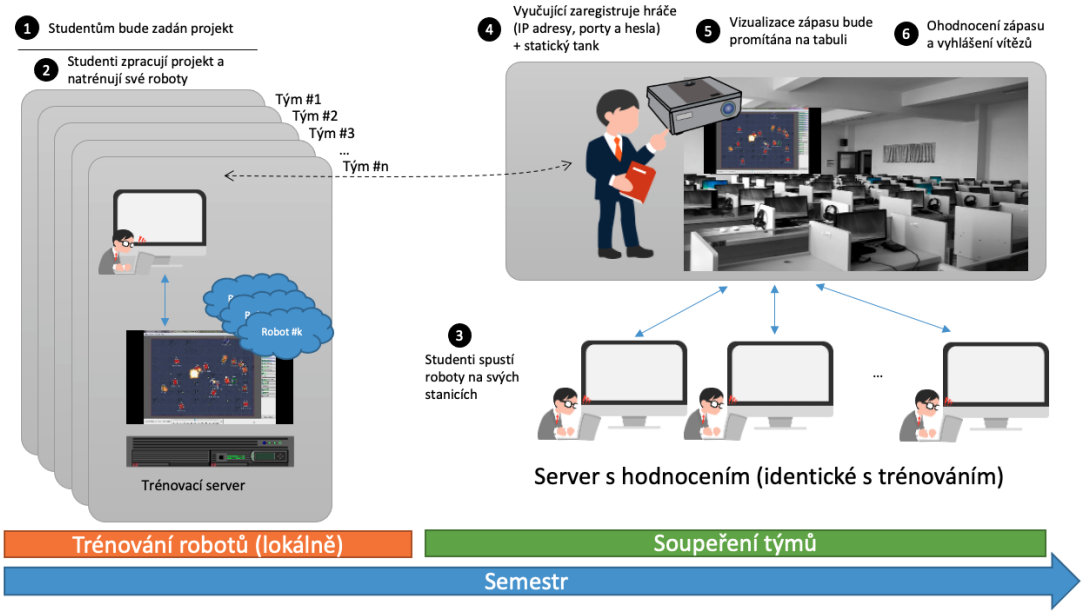
Praktická část této práce se zabývá návrhem a následnou implementací zabezpečené platformy vhodné pro hodnocení studentských projektů v rámci výuky genetického programování a zpětnovazebního učení. V rámci platformy je třeba, zajistit bezpečí hodnotící stanice, kde se studentské projekty spouští s právy superuživatele. Zároveň je třeba využít takový hodnotící systém, kde lze ohodnotit celou třídy zároveň. Pro tyto účely se jako nejvhodnější varianta jeví hra Robocode, kde si hráči programují své tanky, kteří následně bojují v aréně. Bude ale nutná její modifikace, aby bylo hodnocení pro učitelské zařízení bezpečné. Místo toho, aby vyučující spouštěl celou hru i se studentskými tanky lokálně na svém zařízení, bude hra probíhat po síti, čímž se minimalizuje riziko zanesení škodlivého kódu do zařízení.

2.1 Návrh cvičení

Cvičení bude založeno na upravené verzi hry Robocode. Tato varianta byla zvolena na základě splnění všech potřebných požadavků jako je možnost hry více hráčů, možnost aplikace strojového učení a také vytvořený hodnotící systém (počítání skóre), který nám umožní jednoznačně určit nejlepší studentské řešení. Slabou stránkou této platformy je bezpečnost, která ale bude ošetřena v rámci zmíněné upravené verze. Tato úprava spočívá v modifikaci platformy tak, aby umožňovala připojení hráčů po síti. Tím se znemožní spouštění škodlivého kódu od studentů, tudíž bude zajištěna větší bezpečnost pro zařízení vyučujícího.

2.1.1 Princip

Server (robot) a klient (hra) spolu navážou TCP/IP spojení, přes které budou komunikovat, pomocí vlastního protokolu. Během hry bude stanice s klientem pravidelně odesílat dotazy s informacemi o stavu bitevního pole na server. V dotazu se nachází informace o pozicích protivníků, pozicích střel, aktuálním skóre případně rozměrech hrací plochy. Server tyto informace zpracuje a vyhodnotí, která akce bude nejvhodnější. Zvolenou akci server odešle zpátky klientovi formou odpovědi, ta obsahuje informace o pohybu, směru a střelbě. Klient provede požadovanou akci. Tento proces probíhá celou dobu až do konce zápasu. V případě trénování robotů student nebude komunikovat s učitelskou stanicí, ale připojí se na klienta, kterého si sám lokálně spustí.



Obr. 2.1: Návrh hodnocení cvičení

2.1.2 Práce studentů

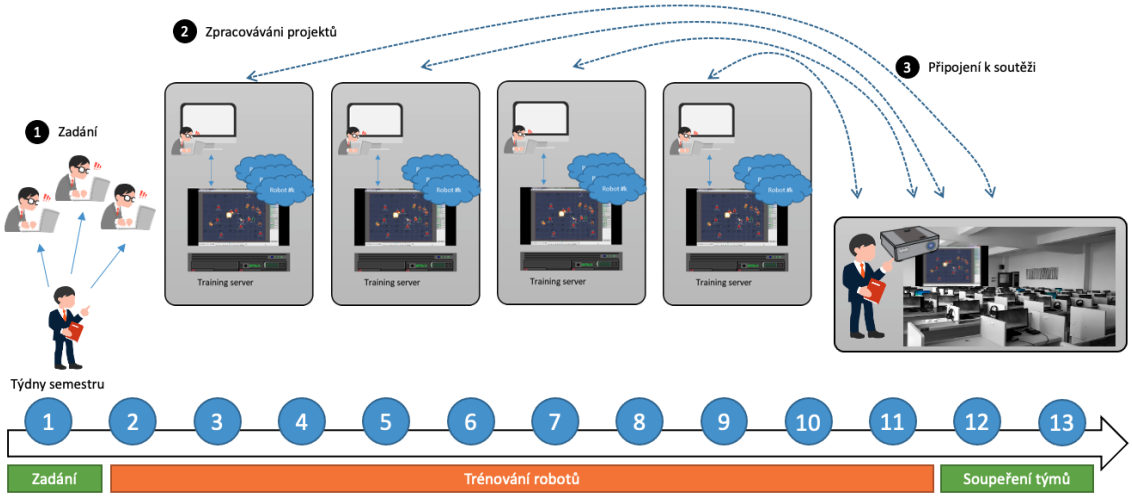
Úkolem studentů bude vytvoření svého tanku do hry Robocode pomocí metod GP a RL. Konkrétně se jedná o vytvoření modelu, který bude celý semestr učen a zdokonalován na trénovacím serveru. Výstup by měl být schopen autonomního fungování ve hře, kde bude nasazen do boje.

Jelikož se nejedná o jednoduchou práci, je vhodné, aby studenti nepracovali samostatně, ale v týmech. Tak aby se adekvátně rozložila pracovní zátěž mezi členy týmů. Náročnost projektu tkví v nutnosti práce s GP a RL. Při aplikaci GP je nutné správně navrhnout strukturu chromozomu tanku případně vhodně definovat modifikační operátory, tak aby bylo dosaženo dostatečné variability. V případě RL je, mimo nastudování látky, nezbytné správně definovat princip odměn. V neposlední řadě je nutné vytvořit algoritmus pro zpracování získávaných informací.

2.1.3 Časový plán

Během semestru budou v rámci předmětu studenti detailněji seznamováni s problematikou GP a RL, tak aby byli schopni jejich využití v projektu. V prvním týdnu budou studenti seznámeni se zadáním a náležitostmi semestrálního projektu. Další týdny by měly být využity ke studiu problematiky a vytváření prvních trénovacích modelů. Přibližně v osmém týdnu by studenti měli mít vytvořený obstojný model. Zbylé týdny by měla probíhat práce na optimalizaci tanku. V posledních dvou týdnech semestru proběhne obhajoba a hodnocení týmových řešení. Součástí hodnocení

bude i soutěž, při které dojde k nasazení všech tanků do boje. Na základě výsledného skóre z turnaje je určí nejlepší týmy, které lze odměnit bonusovým bodovým ziskem.

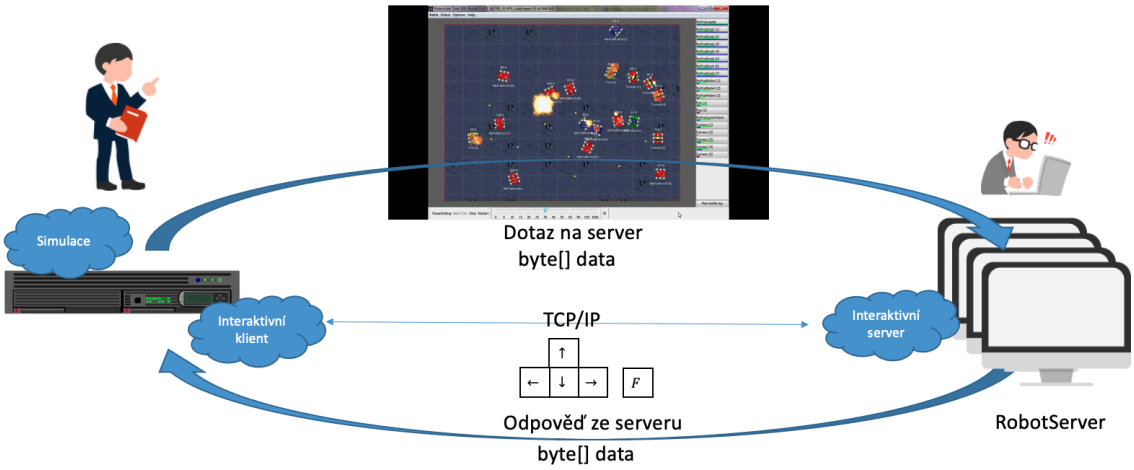


Obr. 2.2: Časový plán semestrálního projektu

2.2 Design zabezpečené platformy

Platforma byla vytvořena tak, aby se z procesu hodnocení vytratila nutnost spouštění studentského kódu. Toho bylo docíleno doplněním původní platformy Robocode o architekturu klient-server, která umožňuje bezpečné hodnocení. Klienta představuje učitelská stanice, na které v případě hodnocení probíhá zápas. Studentské počítače jsou servery, ke kterým se klient připojí a zažádá o instrukce pro studentův tank. Před začátkem hodnocení musí lektor zadat IP adresy všech studentských zařízení do konfiguračního souboru, odkud budou přečteny a dojde k vytvoření spojení. Pro účely trénování robotů mohou studenti využívat komunikaci na lokální adrese. V takovém případě na jednom počítači spustí klienta i server zároveň a místo adresy počítače zadá 127.0.0.1 nebo localhost, tím bude zajištěna komunikace v rámci jednoho zařízení.

Při hodnocení se naváže TCP/IP spojení mezi klientem a servery, které spolu budou komunikovat pomocí vytvořeného protokolu. Klient bude serverům nepřetržitě odesílat dotazy s informacemi o stavu bitevního pole. Server přijme dotaz, zpracuje informace o probíhající bitvě a na základě získaných dat rozhodne o následujícím kroku. Tento krok bude následně zapsán do odpovědi, ta se odešle klientovi, který provede přijaté instrukce na studentově tanku.



Obr. 2.3: Schéma návrhu komunikace klient-server

2.3 Návrh protokolu

Pro účely komunikace v rámci upravené platformy Robocode byl vytvořen nový protokol umožnující přenos informací z bojiště a následně instrukcí od hráče. Protokol je založen na TCP protokolu, který je pro tyto účely vhodný, jelikož na rozdíl od UDP zaručuje doručení vysílaných paketů. Maximální objem přenášených dat v rámci jednoho paketu je omezen na 65536 bajtů. Ze zmíněného datového objemu je nutné odečíst velikosti hlaviček ostatních protokolů, tudíž se jedná spíše o teoretickou maximální hodnotu a ve skutečnosti bude kapacita o několik set bajtů menší. Ačkoliv se jedná o velké množství dat, v případě Robocode – dotazu byla kapacita překročena. V rámci původního návrhu byl zájem o odesílání celé bitevní plochy, jelikož jeden pixel odpovídá jednomu bajtu a standardní velikost plochy je 800x800 (640 000 bajtů), nebyl tento přenos reálný. Proto došlo ke změnám a aktuální paket přenáší pozice všech objektů formou souřadnic. Takto upravený dotaz klade podstatně nižší nároky na prostor a je velmi nepravděpodobné, že by došlo vyčerpání prostoru v rámci hry. Co se týče formátu odpovědi, tento paket má fixní velikost a s velikostí prostoru nehrozí.

2.3.1 Návrh paketu typu dotaz

Pomocí tohoto paketu je server vyzván klientem k reakci na aktuální situaci na bojišti. Součástí paketu jsou záznamy nesoucí informace o stavu hry. Jedná se o výšku a šířku bitevní plochy, dále jsou přiloženy souřadnice patřící studentovu tanku, které uživateli usnadní identifikaci svého tanku mezi ostatními. Důležitou informací je pro server stav ostatních tanků a střel na herní ploše, proto je součástí paketu seznam všech aktivních střel a konkurenčních tanků. U střel jsou obsažena data o jejich

aktuální pozici ve formě souřadnic směru, kterým střela letí a v neposlední řadě je zde informace o síle střely. K jednotlivým tankům jsou zahrnuty informace o energii tanku, aktuální rychlosti, aktuální poloze (opět formou souřadnic) směru, kterým se pohybuje, směru kam míří dělo, teplota děla a aktuální stav.

	Šířka obrazovky	Výška obrazovky	Souřadnice X	Souřadnice X
Střely	Směr střely #1	Souřadnice X střely #1	Souřadnice Y střely #1	Síla střely #1
	...	...	...	...
	Směr střely #n	Souřadnice X střely #n	Souřadnice Y střely #n	Síla střely #n
Tanky	Energie tanku #1	Rychlost tanku #1	Souřadnice X tanku #1	Souřadnice Y tanku #1
	Směr tanku #1	Směr děla tanku #1	Teplota děla tanku #1	Stav tanku #1
	...	...	...	...
	Energie tanku #n	Rychlost tanku #n	Souřadnice X tanku #n	Souřadnice Y tanku #n
	Směr tanku #n	Směr děla tanku #n	Teplota děla tanku #n	Stav tanku #n

Obr. 2.4: Návrh paketu typu dotaz

2.3.2 Návrh paketu typu odpověď

Jelikož se jedná o paket nesoucí pouze instrukce je podstatně menšího rozsahu než dotaz. Žádné informace o stavu serveru klienta nezajímají, proto stačí, když jsou přenášeny pouze instrukce agenta. Jedná se o sedm číselných záznamů. První dva nesou informaci o tom, zda se má agent hýbat dopředu, couvat nebo zda setrvat na místě. Druhá dvojice instrukcí definuje směr, kterým se tank bude pohybovat. Další dvojice slouží k ovládání otáčení věže. Posledním záznamem je rozhodnutí o výstřelu, zda se má vystřelit, případně s jakou silou.

Pohyb vpřed	Pohyb vzad	Pohyb doprava	Pohyb doleva	Dělo doprava	Dělo doleva	Výstřel
Vzdálenost v pixelech		Otočení ve stupních		Otočení ve stupních		0... nestřílet 1-5... výstřel

Obr. 2.5: Návrh paketu typu odpověď

2.4 Návrh komunikace

Jak již bylo zmíněno v předchozích kapitolách, cílem práce je fungování cvičení po síti. Pro tyto účely byla zvolena architektura typu klient-server. V rámci upravené platformy jsou počítače studentů servery a učitelský počítač, na kterém běží samotná hra, je klientem. Díky takovému rozdělení je snížen objem dat, které je nutné přenášet po sítí a zároveň je takový model snazší na implementaci, jelikož není nutné řešit synchronizaci všech zařízení. V praxi si studenti před začátkem boje spustí servery na svých zařízeních, poté bude spuštěna hra na učitelské stanici, která naváže spojení s každým ze serverů a v průběhu hry se bude dotazovat pomocí paketů `Robocode - dotaz` na které budou servery odpovídat `Robocode - odpověď` paketem.

Server je napojen na ovládací jednotku tanku (agenta). Při každém dotazu od klienta, ve kterém jsou obsaženy informace o aktuálním dění na bojišti, agent vygeneruje sadu instrukcí pro tank, které jsou serverem odeslány zpět klientovi. Tato komunikace mezi klientem a jednotlivými servery probíhá po celou dobu zápasu. V případě, že je tank zničen, klient jeho směrem přestává odesílat pakety a čeká do začátku dalšího kola, kdy se se serverem opět spojí a komunikace probíhá znovu. Pro navázání spojení mezi klientem a serverem je využíván IPv4 protokol. Tudíž je nutné zadání správné IP adresy serveru (studentova zařízení) a volného portu na kterém bude komunikace probíhat.

3 Implementace zabezpečené platformy

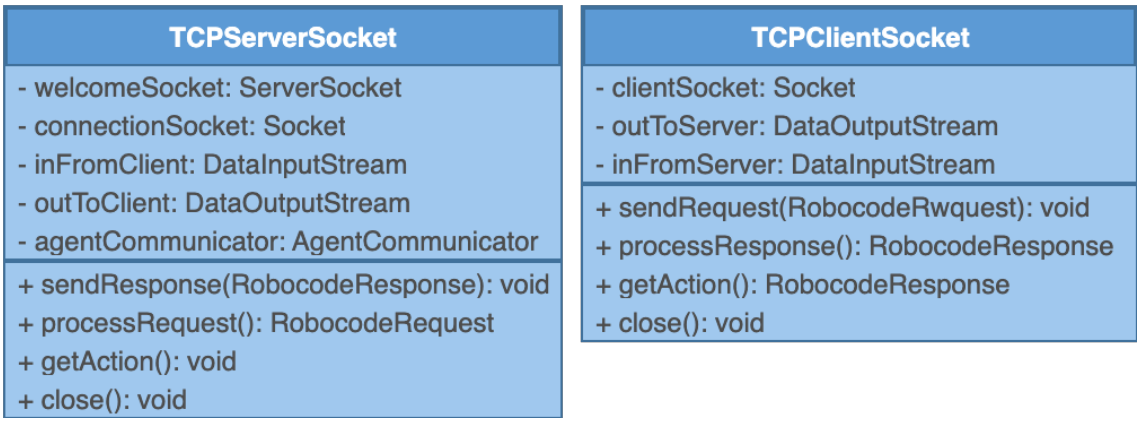
V této kapitole je popsána implementace navržené platformy. Zabývá se popisem konkrétních tříd vytvořených pro potřeby komunikace mezi klientem a serverem, definicí komunikačních protokolů a implementací speciálního hráčského tanku. Dále je zde popis samotného třídy spouštějící hru, popis platformy komunikující s agentem i příkladu samotného agenta. V závěru kapitoly jsou stručně popsány procesy probíhající během celého běhu programu.

Při implementaci platformy byla využita Java ve verzi 11 a její základní knihovny, pro práci s protokoly byla využita externí knihovna `it.rmarcello` a pro implementaci cvičného agenta byl využit programovací jazyk Python.

3.1 Implementace komunikačního kanálu

Pro navázání spojení a následnou podporu komunikace mezi klientem a serverem byly implementovány třídy `TCPServerSocket` a `TCPClientSocket`. Obě třídy mají mnoho společných prvků, ale každá má svoji specifickou funkci.

Hlavní funkcí třídy `TCPClientSocket` je, že přebírá informace z herní platformy, tato data převede na `RobocodeRequet` paket, který odesílá na server a čeká na jeho odpověď ve formě `RobocodeResponse`. Na rozdíl od toho, třída `TCPServerSocket` čeká dokud neobdrží `RobocodeRequet` paket, ze kterého přečte informace, předá je agentovi a na základě jeho odpovědi vytvoří `RobocodeResponse` paket, který je odeslán zpět na klienta.



Obr. 3.1: UML diagram tříd TCPServerSocket a TCPClientSocket

3.1.1 Implementace klienta

Klient obsahuje tři neměnné atributy (označené `final`), první z těchto atributů slouží k inicializaci spojení a zbylé jsou ukazateli na příchozí a odchozí datový tok. Při vytvoření objektu této třídy konstruktor naváže spojení se serverem na zadané adrese a portu a inicializuje proměnné pro příchozí a odchozí datový tok (`outToServer` a `inFromServer`).

Metody této třídy zajišťují správný chod komunikace včetně zpracování odchozích a příchozích dat. Konkrétně se jedná o čtyři metody. První je `sendRequest()`, která jako parametr přijímá objekt typu `RoobcodeRequest`, ten odesílá na server a vyprazdňuje odchozí stream. Metoda `processResponse()` čeká na příchozí datový tok ze serveru, jakmile přijde odpověď přijde načte se do bufferu a je parsována z pole bajtů zpět na objekt typu `RobocodeResponse` který vrací. Ve třídě se dále nachází metoda `close` ukončující spojení se serverem. Hlavní metodou je `getAction`, která je využívána nejvíce. Tato metoda spojuje celý proces výměny dat se serverem. Při zavolání na vstupu přijímá objekt typu `RobocodeRequest`, ten je následně odeslán na server zavoláním metody `sendRequest()` a na zpracování odpovědi vrácené serverem je aplikována metoda `processResponse()`. Metoda na konci vrací objekt typu `RoobcodeResponse`.

3.1.2 Implementace serveru

Server obsahuje pět neměnných atributů (označené `final`), první dva z těchto atributů slouží k inicializaci spojení další dva jsou ukazateli na příchozí a odchozí datový tok, poslední atribut drží ukazatel na objekt typu `AgentCommunicator`, kterého se dotazuje na instrukce. Při spuštění serveru konstruktor vyčkává na zařízení se zájmem o komunikaci, jakmile je dotázán jsou inicializovány všechny zmíněné atributy a server je připraven přijímat a odesílat informace. Server soket funguje podobně jako `TCPClientSocket()` akorát se liší typ paketu se kterým metody pracují.

V rámci serveru je implementováno několik metod, které umožňují jeho ovládání. Mimo konstruktor je ze metoda `close()`, která umožňuje ukončení spojení. Metoda `processRequest()` zpracuje přijatý paket a převede jej na snadno čitelný formát `RoobcodeRequest`. Úkolem metody `sendResponse()` je odeslání paketu typu `RobocodeResponse` zpět na klienta.

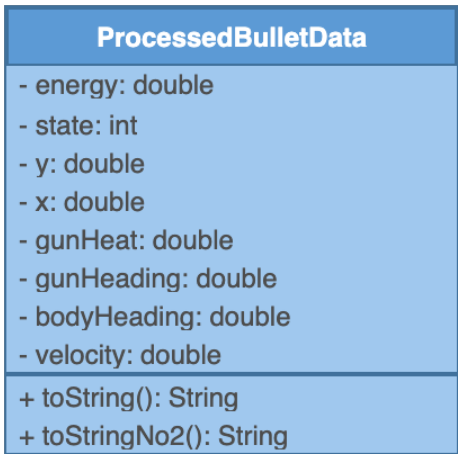
Hlavní metodou je opět `getAction()`, ta v konfiguraci serveru nic nevrací, pouze organizuje činnost ostatních metod. Prvně volá metodu `processRequest()`, která čeká dokud neobdrží data od klienta, ta zpracuje a formátuje na objekt `RobocodeRequest`. Následně zapracovaný paket předá na vstup metodě pro generování instrukcí a tyto instrukce odešle klientovi zavoláním metody `sendResponse()`.

3.2 Zpracování dat pro přenos

V rámci vytvořené platformy je třeba přenášet data o aktuálním dění na bojišti, jak již bylo zmíněno v kapitole zabývající se návrhem protokolu, požadovaná data se týkají všech tanků na bojišti a všech aktivních střel. Proto bylo třeba vytvořit třídy `ProcessedTankData` a `ProcessedBulletData`, jejichž atributy odpovídají záznamům které o všech tancích a střelách chceme přenášet. Dále byla vytvořena třída. pro ukládání všech objektů předchozích tříd s názvem `ProcessedBattleData`, tento objekt je přímo předáván do protokolu ve formě pole bajtů. Všechny tři třídy implementují rozhraní `Serializable` aby byla možná jejich konverze na pole bajtů a zpět.

3.2.1 Zpracování informací o tanku

Třída `ProcessedTankData` se využívá při zpracovávání dat o jednotlivých tancích ve hře. Pro každý z tanků ve hře je při dotazu na server vytvořena instance této třídy. Její atributy odpovídají informacím z navrženého paketu pro dotaz, je ze informace o energii tanku, rychlosti a směru pohybu, stavu děla (směr a teplota), souřadnicích tanku a jeho aktuálním stavu.



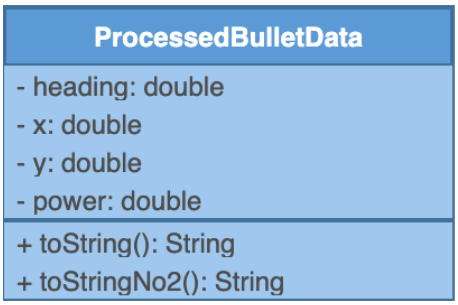
Obr. 3.2: UML diagram třídy ProcessedTankData

Téměř všechny atributy mají definovaný datový typ `double` jelikož se jedná o desetinné číselné hodnoty, v případě informací o směru jsou předávány radiány místo stupňů. Jediný záznam informující o stavu tanku má datový typ `int`, jelikož se jedná o výčtový prvek se čtyřmi definovanými hodnotami. Pokud je hodnota atributu `state` rovna [0] znamená to, že je tank aktivní, v případě hodnoty [1] tank narazil do zdi, hodnota [2] značí srážku s jiným tankem a v případě zničení tanku je hodnota nastavena na [3]. Součástí třídy jsou dvě metody převádějící objekt na

text, jedna slouží pro klasický a snadno čitelný přepis na text. Druhá vrací textovou reprezentaci atributů ve formě seznamu, tato metoda se využívá pro předávání dat Python skriptu s modelem.

3.2.2 Zpracování informací o střele

Tato třída je využívána pro ukládání požadovaných informací o každé aktivní střele na ploše. Podobně jako v předchozím případě je pro každou střelu vytvořena nová instance, která je poté uložena pro přenos. Třída obsahuje pouze 4 atributy všechny typu `double` nesoucí informace o směru kterým se střela pohybuje, v radiánech, aktuální pozici střely formou souřadnic a síle střely. Stejně jako u předchozí třídy se zde nachází dvě metody pro přepis atributů na formát `String`, jedna slouží pro obyčejný výpis a druhá se využívá k předávaní dat.



Obr. 3.3: UML diagram třídy ProcessedBulletData

3.2.3 Agregace zpracovaných dat

Jelikož množství tanků a střel ve hře není nijak konkrétně definováno a může se v každé hře lišit, bylo třeba vytvořit flexibilní způsob, jak tyto objekty ukládat nezávisle na jejich množství. Proto byla vytvořena třída `ProcessedBattleData`, která slouží k ukládání všech objektů typu `ProcessedTankData` a `ProcessedBulletData`. Třída obsahuje dva atributy typu `ArrayList<>`, jeden pro každý typ záznamů. Pro střely slouží `listOfProcessedBullets` a `listOfProcessedTanks` je pro tanky.

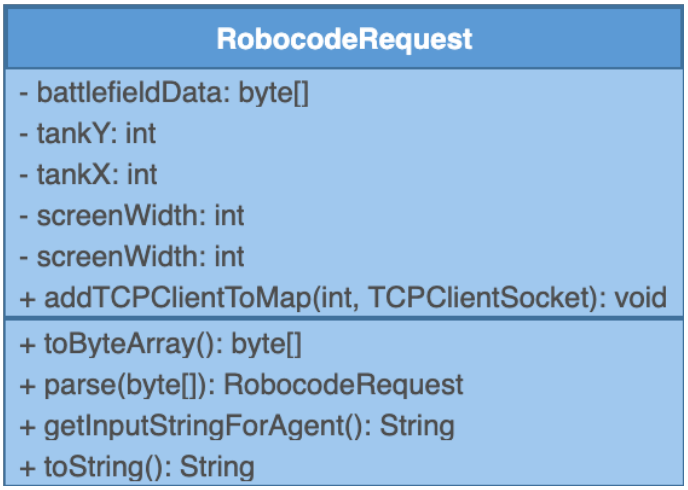
Díky této třídě lze snadno odesílat data ze hry, aniž by bylo třeba řešit konkrétní počet prvků ve hře (tanků/střel), jelikož se data pouze přidají do listů a celý objekt typu `ProcessedBattleData` je poté převeden na pole bajtů, které již lze snadno přenést pomocí vytvořeného protokolu.

3.3 Implementace protokolu

Pro komunikaci mezi klientem a serverem byla využita knihovna `it.rmarcello`, díky které je možné upravovat podobu jednotlivých paketů, tudíž lze definovat pakety pro dotaz – `RobocodeRequest` a pro odpověď – `RobocodeResponse` podle potřeby. Limitací této knihovny je podpora pouze základních celočíselných typů, binárního operátoru `Bolean` nebo bajtů. Tudíž musí být všechna desetinná čísla převedena zaokrouhlena a převedena na typ `int`, což v tomto případě není problém, jelikož data jsou i po zaokrouhlení dostatečně přesná. Obě dvě třídy implementují rozhraní `RobocodePacket`, které jim zakládá povinnost vytvoření funkce pro převod na pole bajtů přenositelné mezi klientem a serverem.

3.3.1 Implementace paketu typu dotaz

Tato třída definuje podobu paketu, který je odesílán jako dotaz klienta na server. Měl by obsahovat všechny potřebné informace, na jejichž základě potom na serveru probíhá rozhodování agenta o následujících krocích.



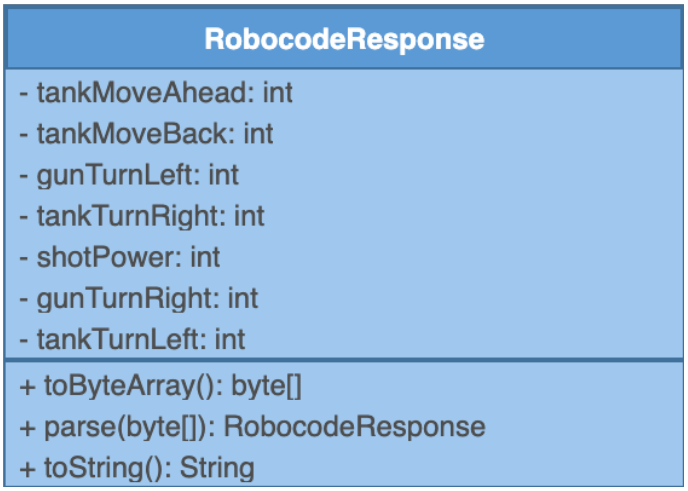
Obr. 3.4: UML diagram třídy RobocodeRequest

První z informací se týkají výšky a šířky bitevního pole, jedná se o hodnoty v pixelech a jejich datový typ je `int`. Další dvojicí záznamů jsou souřadnice hráčova tanku, ačkoliv se jedná o duplicitní informaci, je zde zahrnuta kvůli identifikaci tanku hráče v seznamu mezi ostatními tanky, jinak by hráč velmi těžko věděl, na kterém místě se nachází a kam by bylo vhodné se pohybovat nebo střílet. Posledním a nejrozsáhlejším atributem této třídy je `battlefieldData`, jehož datový typ je pole bajtů. Ve skutečnosti se o objekt typu `ProcessedBattleData`, obsahující informace o všech tancích a střelách ve hře, který byl převeden na bajty, jelikož použitá knihovna nedokáže zpracovat složitější datové typy. Třída má také definovány dvě metody

`toByteArray` a `parse`, které umožňují převod objektu na pole bajtů a zase zpět na objekt typu `RobocodeRequest`. Další metodou je `getInputStringForAgent()`, která vrací textovou reprezentaci rozměrů herní plochy tak, aby byl agent schopen ji zpracovat.

3.3.2 Implementace paketu typu odpověď

Druhým typem paketu je `RobocodeResponse`, ten je využíván k přenosu dat ze serveru ke klientovi. Obsahuje instrukce pro hráčův tank, které vygeneroval agent na základě informací, které získal z paketu `RobocodeRequest`.



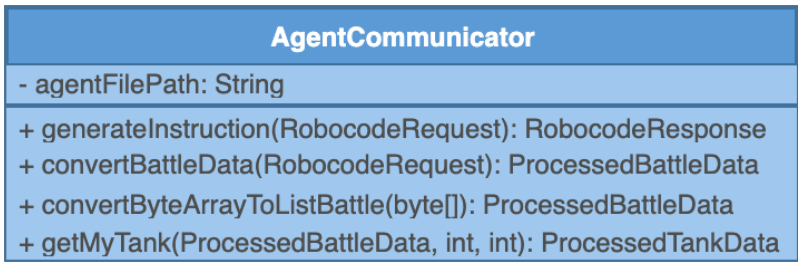
Obr. 3.5: UML diagram třídy RobocodeResponse

Třída obsahuje atributy definující délku pohybu tanku vpřed a vzad v pixelech, dále míru zatáčení tanku a otáčení děla o určité množství stupňů a sílu výstřelu, případně zda vůbec střílet. Všechny atributy jsou definovány jako datovým typem `int` a většina z nich je zdvojená, jelikož přenos záporných čísel byl problematický. Jinak by stačilo pohyb doleva definovat jako záporný pohyb vpravo a stejně je to i s jízdou vpřed. Takto duplicitní hodnoty jsou pouze v rámci paketu, když je pohyb definován agentem, případně čten u klienta, gettery a settery těchto atributů hodnoty slučují do jedné.

Stejně jako v `RobocodeRequest` paketu jsou součástí třídy metody `toByteArray` a `parse`, které zprostředkovávají konverzi objektu na pole bajtů a následně pak pole bajtů zpět na objekt typu `RobocodeResponse`.

3.4 Získávání instrukcí z modelu

Jelikož je upravená platforma hry Robocode implementována v jazyce Java a natrénované studentské modely budou implementovány v jazyce Python, je třeba vytvořit způsob, kterým si budou vzájemně předávat informace. Python model potřebuje získávat informace o dění na bojišti na základě, kterých potom vygeneruje instrukce, které zase musí vrátit platformě. A právě pro tyto potřeby byla vytvořena třída `AgentCommunicator`.



Obr. 3.6: UML diagram třídy AgentCommunicator

Objekt této třídy je volán serverem, při požadavku na získání instrukcí pro tank. Třída má pouze jeden atribut, kterým je `agentFilePath`, držící cestu k Python skriptu s natrénovaným modelem a čtyři metody. Dvě z nich slouží k dodatečnému zpracování paketu typu `RobocodeRequest` konkrétně atributu `battleFieldData`, který je ve formě pole bajtů. Dále se ve třídě nachází metoda `getMytank()`, která ze zpracovaných dat vyextrahuje hráčův tank podle jeho aktuálních souřadnic a odstraní jej ze seznamu.

V tomto případě je hlavní metodou `generateInstructions()`, která vše organizuje. V první řadě volá metody převádějící pole bajtů na čitelná data o bojišti a následně je vyextrahován hráčův tank. Jakmile je vše zpracováno, je spuštěn agent, který na vstup dostává všechna data formou textové reprezentace seznamu hodnot. Agent na základě získaných informací vygeneruje instrukce pro tank, které vrací výpisem ve formátu `"RESPONSE:20,-10,1,30"`. Tento formát odpovědi se zpracuje a je vytvořena nová odpověď typu `RobocodeResponse`, kterou metoda vrátí. Tento objekt je předán serveru a odeslán klientovi, kde jsou instrukce provedeny.

3.5 Platforma

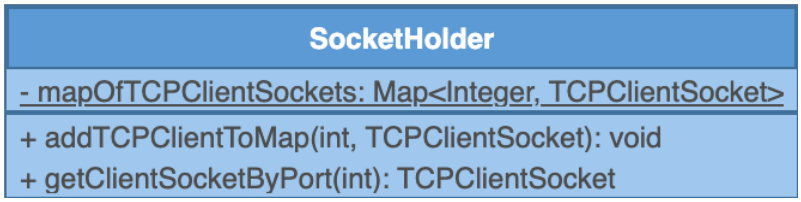
Tato sekce se zabývá implementací těch tříd, které jsou přímo napojeny na původní platformu Robocode a využívají jejich knihovny. Lze říci, že tyto třídy slouží ke spuštění samotné hry a nastavení prostředí tak, aby vůbec mohl zápas proběhnout.

Do této kategorie patří třída `SocketsHolder` sloužící k udržení navázaných TCP spojení se serverem po celou dobu zápasu. Dále také třída `RobotClient`, která ve hře reprezentuje předpis tanku, akorát je upravena tak aby se místo předdefinovaných kroků dotazovala serveru na instrukce. Třída `BattleObserver` umožňuje sledování průběhu boje a vypisuje důležité události na konzoli, aby bylo možné sledovat hru i mimo grafické rozhraní. Poslední třídou je `RobocodeRunner`, která jak název napovídá, je zodpovědná za spuštění samotné hry.

3.5.1 Agregace vytvořených TCP klientů

Objekty typu `RobotClient` jsou na konci každého kola smazány a v dalším kole vytvořeny nové, což způsobovalo problémy ve spojení se serverem. Server má stále otevřený komunikační kanál s předchozím klientem, tím pádem se na něj nemůže připojit nový.

Na tento problém byla navržena dvě možná řešení. První pracovalo s příznakem, který by se odesílal jako součást paketu `RobocodeRequest` a oznamoval by serveru, že je čas na uzavření aktuálního spojení a otevření nového. Tato možnost se neukázala jako vhodná, jelikož celý proces spuštění a navázání nového spojení byl časově náročný a mnohdy se jej nepodařilo dokončit včas, tím pádem se ve stávalo, že ve hře byly tanky bez připojení na sever a tím pádem byly neaktivní.

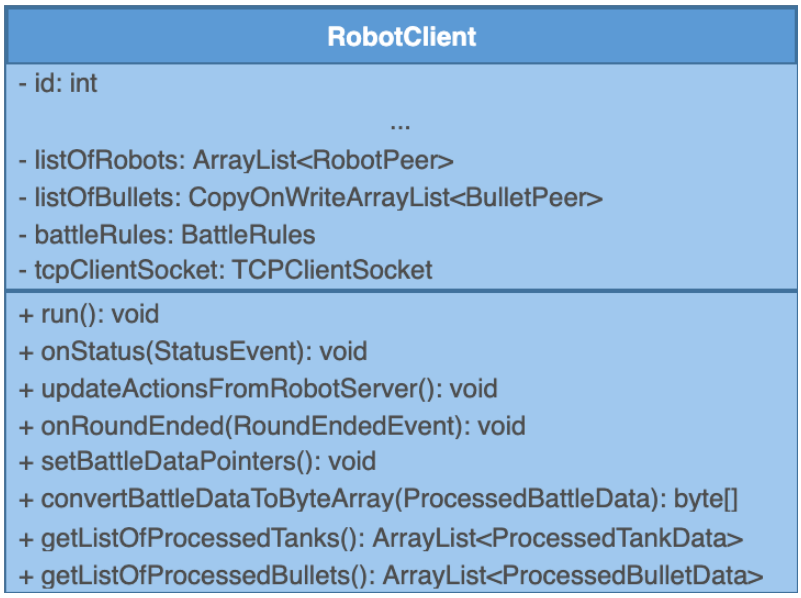


Obr. 3.7: UML diagram třídy SocketsHolder

Druhou možností, která se nakonec ukázala jako funkční, bylo vytvoření objektu třídy `SocketsHolder`. Tato třída je navržena tak, aby ukládala všechna aktivní spojení se servery a klientský tank si z ní při inicializaci vezme ukazatel na navázané spojení, takže se nemusí vytvářet nové. Spojení se ukládají do slovníkové struktury typu `HashMap<>`, ve které je klíčem záznamu port, na kterém spojení komunikuje, a hodnotou je objekt typu `TCPClientSocket` který je spojen s patřičným serverem. Slovník `mapOfTCPClientSockets` udržující potřebná spojení je statický. To znamená že je pro všechny instance třídy `SocketsHolder` stejný a pokud v jeho libovolné instanci dojede ke změně, propíše se do všech. Ke změnám ve slovníku jsou definovány dvě metody na přidávání a odebírání prvků ze slovníku.

3.5.2 Implementace klientského tanku

Ve standardní hře Robocode jsou využívány tanky s předdefinovanou logikou běžící lokálně na zařízení. Ovšem v této variantě je nezbytné, aby byl tank schopen přijímat instrukce ze serveru, nikoliv lokálně. Proto bylo třeba vytvořit třídu `RobotClient`, která se pro základní platformu Robocode tváří jako standardní tank, který platforma dokáže zařadit do hry, ale ve skutečnosti bude tank komunikovat se serverem.



Obr. 3.8: UML diagram třídy RobotClient

Jak již bylo zmíněno, tato třída funguje jinak než ostatní tanky. Při inicializaci je na základě čísla portu třídě přiděleno spojení se serverem. Toto spojení se vybírá z instance třídy `SocketsHolder`, kde jsou uložena všechna spojení.

Třída obsahuje několik podstatných metod ovlivňujících chování tanku ve hře. Jednou z nich je metoda `onStatus`, která se volá při každé změně statusu tanku a v tu chvíli přenastaví lokální proměnné představující aktuální souřadnice používané pro identifikaci a zároveň, pokud tank stále nemá nastavené ukazatele volá metodu `setBattleDataPointers()`. Tato metoda nastavuje ukazatele na seznam tanků ve hře a seznam aktivních střel. Jelikož jsou oba seznamy zapouzdřeny v privátních objektech, bylo nutné využití reflexe na několika úrovních. Díky nastavení ukazatelů je nutné využívání reflexe pouze jednou na začátku kola, což podstatně zvyšuje efektivitu programu. V metodě `run()` na počátku dochází k nastavení vlastností tanku, v tomto případě pouze barev a potom následuje nekonečná smyčka ovládající tank. Na počátku každé iterace je zavolána metoda `updateActionsFromRobotServer()`, která zpracuje všechna požadovaná data a vytvoří nový paket typu `RobocodeRequest`. S vytvořeným paketem na vstupu volá metodu `getAction` z třídy `TCPClientSocket`

a podle následné odpovědi nastaví ovládací atributy. Jakmile jsou instrukce aktualizovány dochází k jejich provedení a cyklus se opakuje.

3.5.3 Inicializace serveru

Pro potřeby jednoduššího ovládání ze strany studentů byla vytvořena spustitelná třída `StudentServerRunner()`, sloužící ke spuštění serveru. Jedná se o jednoduchou třídu se třemi atributy. Je zde číselná hodnota reprezentující číslo portu na hlavní stanici, na kterém bude probíhat komunikace. Další atribut je typu `String` a umožňuje zadat jméno skriptu s agentem který má být při dotazování využíván. Agent musí být umístěn ve složce `src/cz/vutbr/feec/robocode/studentRobot/` a název je zadáván ve formátu `NÁZEV.py`. Posledním atributem je samotný server čekající na spojení. Po navázání spojení je v nekonečné smyčce volána jeho metoda `getAction()`.

V implementaci se ještě nachází třída `ServerRunnerTestSecond`. tato třída je s předchozí totožná a slouží pro účely lokálního testování dvou modelů. Pokud by měl student zájem otestovat své modely ještě před závěrečným hodnocením, lze pro to využít tuto testovací třídu.

3.5.4 Inicializace herního prostředí

Tato třída se využívá ke spuštění hry Robocode s konfigurací, která je schopna komunikovat se serverem. Na počátku je nahrána konfigurace studentských robotů ze souboru `config/game.properties`, ve kterém je definováno kolik robotů je potřeba připojit k serveru a jejich specifikace jako IP adresa, port, heslo a barva tanku. V dalším kroku lze definovat automatické roboty, kteří budou součástí hry, celá nabídka těchto robotů je ve složce `sample`, lze využít všechny až na `RobotClient`, který se vyhrazen pro komunikaci se serverem a samostatně nefunguje. Jedná se předdefinované roboty, které lze využít například pro trénink. Poté se vytvoří objekt třídy `SocketsHolder` do kterého se uloží komunikační kanály pro všechny studentské tanky, tyto sokety lze namapovat pomocí čísla portu. Jakmile jsou vytvořena všechna spojení, volá se funkce `runRobocode()`.

Metoda `runRobocode()` vytváří kopii třídy `RobotClient` ve složce, kde je spustitelná ve hře. Následně jsou všechna jména definovaných robotů, jak studentských tak automatů, sepsána do jedné proměnné `finalListOfTanks` typu `String`, kde jsou odděleny čárkou, tento formát je později zpracován a jsou s jeho pomocí nahrány tanky do hry. V dalších krocích je vytvořen a nastaven engine hry, takže mu je vytvořen a přidělen správný Listener. Viditelnost je nastavena na hodnotu `true`, což znamená, že se zobrazí grafické rozhraní s bitvou. Nakonec je vytvořena instance

třídy `BattleSpecification`, skrze kterou jsou definovány náležitosti bitvy, jako počet kol, velikost herní plochy v pixelech a vybrané tanky. Po tomto kroku se se spustí hra se všemi nastavenými parametry. Když je celá bitva u konce je zavolán výpis výsledků a vše se ukončí.

3.5.5 Výpis informací o průběhu hry

Třída `BattleObserver` přímo dědí funkcionalitu třídy `BattleAdaptor` a díky tomu má přístup k funkcím které reagují na aktivitu a události na pozadí. To umožňuje sledovat průběh hry a výskyt definovaných událostí na terminálu. V tomto případě se jedná o začátek či konec zápasu a kola, dále jsou vypisovány zprávy prostředí a errory. Podstatnou funkcí této třídy je její schopnost získat skóre, což umožňuje hodnocení studentů nebo v případě tréninku kvalitu aktuálního řešení.

4 Výsledky a diskuze

V první části této kapitoly jsou rozebrány výsledky práce, je zhodnocena implementace a je zde popsán způsob jakým probíhala validace funkčnosti. Druhá část je diskuze, ve které je popsán úspěch, kterého se podařil docílit, problémy doprovázející implementaci a případný prostor pro inovace.

4.1 Výsledky

V rámci práce byl vypracován návrh bezpečného semestrálního cvičení založeného na hře Robocode, vhodného pro potřeby předmětu MSC-PDA, zabývajícího se výukou ML. Díky využití této platformy se jedná o unikátní možnost, jak objektivně ohodnotit kvalitu výstup RL. Tímto způsobem lze porovnat hned několik řešení mezi sebou a vybrat z nich to nejlepší. Pro zajištění bezpečnosti byla navržena architektura klient-server, kde je učitelská stanice klientem a dotazuje se serverů, které spouštějí studenti se svým natrénovaným modelem. Komunikace mezi klientem a servery by probíhá pomocí navržených paketů pro dotaz a odpověď. Paket typu dotaz přenáší kompletní informace z bojiště, díky tomu má model stejné množství informací jako by měl manuálně hrající člověk. Po zpracování přijatých informací by měl agent definovat kroky, které má jeho tank provést a skrze paket typu odpověď je odeslat zpět na platformu.

Implementace upravené platformy odpovídá předchozímu návrhu, byly implementovány úpravy základní platformy umožňující spuštění s námi definovanými tanky. K tomu patří i implementace speciálního klientského tanku, který je upraven tak, aby byl schopen přijímat instrukce ze vzdáleného serveru. Zároveň byla implementována platforma serveru, která je schopna přijímat a zpracovávat dotazy od klienta. Jakmile jsou získána data z paketu, mohou být předána natrénovanému modelu, který je implementován v jazyce Python. Model podle informací navrhne kroky, které by měl tank provést, tyto instrukce jsou následně odeslány klientovi, pomocí paketu typu odpověď, kde jsou provedeny. Součástí implementace je i příklad v jazyce Python demonstrující komunikaci s platformou a návod pro studenty popisující postup při spouštění platformy.

Funkčnost implementace byla několikanásobně testována s jedním tankem ovládaným vzdáleně a několika lokálními tanky. Tyto testy neobjevily žádný problém v implementaci. Následně byl proveden test s více vzdáleně ovládanými zařízeními a různými modely. Ani tyto testy neodhalily žádný zásadní problém v implementaci.

4.2 Diskuze

Hlavním přínosem této práce je vytvoření platformy umožňující objektivní hodnocení a porovnání výstupů strojového učení u problémů spadající do třídy složitosti EXPSPACE. Tyto problémy se vyznačují vysokými nároky na paměťový prostor, a proto je velmi obtížné objektivně zhodnotit, který model je nejlepší. V současnosti existují způsoby, jak odhadnout kvalitu jednoho řešení, ale není zde žádná možnost, jak vzájemně porovnat řešení více. A tento problém řeší právě tato práce, díky úpravám provedeným ve hře Robocode, je nyní možné do bitvy nasadit hned několik modelů a na základě výsledného skóre rozhodnout, který je nejlepší.

Dalším přínosem je návrh unikátního cvičení pro účely výuky GP a RL, které by mohlo být zajímavé pro studenty. Díky prvku závěrečného souboje, kde bojuje každý proti každému, by se mohla zvýšit motivace studentů na projektu pracovat.

Ačkoliv je platforma plně funkční je zde stále prostor pro inovace. Především se jedná o informovanost agenta nebo styl hry. Současná implementace je vytvořena tak, že množství přenášených informací odpovídá informacím, které má manuálně hrající člověk. To znamená, že vidí celou herní plochu, všechny tanky, střelu a skóre ostatních hráčů. Ve standardní hře, kde hrají pouze automatické tanky, jsou informace omezené. Každý tank má v této variantě svůj radar, který detekuje objekty pouze ve směru, kterým míří a pouze na určitou vzdálenost. Při hře s omezenou informovaností by se zde mohl naskytnout další zajímavý prvek, a to taktika prohledávání bojiště, případně snaha minimalizaci situací, kdy bude tank detekován. Další možností rozšíření je herní mód. Aktuálně hra probíhá v režimu, kdy každý hráč hraje sám za sebe a mohlo by být zajímavé rozšířit platformu pro hru v týmech.

Závěr

Práce se zabývá návrhem semestrálního cvičení vhodného pro výuku metod strojového učení. Dále řeší problematiku hodnocení takových projektů využitím modifikované platformy Robocode. V neposlední řadě se zabývá návrhem úpravy platformy a následnou implementací.

Hlavním problémem při hodnocení těchto projektů je jejich komplexita, jelikož se jedná o problém spadající do třídy složitosti EXPSPACE, je velmi obtížné vyhodnotit, zda je konkrétní řešení dobré. Ohodnocení jednoho modelu samo o sobě komplikované a hodnocení několika modelů najednou je ještě složitější. Aktuálně totiž neexistuje žádná vhodná platforma, která by umožňovala takto vzájemně porovnat více naučených modelů.

V rámci této práce byl zpracován návrh bezpečné platformy založené na hře Robocode, která byla upravena tak aby bylo možné ovládat tanku nejen lokálně, ale i ze vzdáleného serveru. Zároveň byl vypracován návrh komunikačního protokolu, sloužícího k výměně informací se serverem. Navržená platforma byla následné implementována v jazyce Java a úspěšně otestována. Dále byl vytvořen ukázkový kód v jazyce Python demonstrující zpracování dat z platformy a následné předání vygenerovaných instrukcí zpět platformě. A nakonec byl k platformě zpracován stručný návod pro ovládání platformy.

Hlavním přínosem této práce je vytvoření bezpečné platformy schopné porovnávat několik natrénovaných modelů zároveň. Takové hodnocení, pro modely řešící problémy z třídy EXPSPACE, doposud nebylo možné. Dalším přínosem je vytvoření neobvyklého cvičení pro studenty v rámci, kterého si mohou procvičit metody strojového učení.

V případě úspěchu tohoto cvičení zde stále je prostor pro inovace, především se jedná o princip získávání dat z bojiště. Aktuální implementace odesílá serveru informace o všech tancích a střelách na bitevní ploše, což odpovídá pohledu, který má hráč ovládající hru manuálně. V případě autonomních tanků ovšem získávání dat z bojiště funguje na jiném principu. Například je zde využita funkce radaru, který detekuje nepřátele pouze na určitou vzdálenost ve směru, kterým míří. Podobný princip funguje i s detekcí střel. Toto rozšíření by ke hře přidalo další zajímavý aspekt, a to strategii prohledávání prostoru. Případně by mohla být platforma rozšířena o možnost hry v týmech.

Literatura

[1] John R. Koza. *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, 1994.

[2] Vojtěch Havránek. Umělé neuronové sítě a zpětnovazebné učení. 2008.

[3] Martin Pilát. Zpětnovazební učení. URL: `https://martinpilat.com/cs/prirodou-inspirovane-algoritmy/zpetnovazebni-uceni`.

[4] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[5] Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[7] Petr Babor. Využití genetického programování v evoluci robotů. 2014.

[8] John R Koza. Human-competitive results produced by genetic programming. *Genetic programming and evolvable machines*, 11(3):251–284, 2010.

[9] Yehonatan Shichel, Eran Ziserman, and Moshe Sipper. Gp-robocode: Using genetic programming to evolve robocode players. In *European Conference on Genetic Programming*, pages 143–154. Springer, 2005.

[10] CPA van Run. Reinforcement learning: Developing a battle tank for the robocode battle simulator. B.S. thesis, 2011.

[11] M Gade, M Knudsen, R Kjær, T Christensen, C Larsen, M Pedersen, and J Andersen. Applying machine learning to robocode. *JSK, Aalborg*, 2003.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[13] Pavel Nohejl. Boj robotů ve virtuálním prostředí. 2007.

[14] Sean Anderson. Python vs java: A deep dive comparison, Aug 2022. URL: `https://streamsets.com/blog/python-vs-java-comparison/`.

[15] Behrouz A Forouzan. *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002.

[16] Mohammed M Alani. Tcp/ip model. In *Guide to OSI and TCP/IP models*, pages 19–50. Springer, 2014.

[17] D Madhuri and P Chenna Reddy. Performance comparison of tcp, udp and sctp in a wired network. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–6. IEEE, 2016.

Seznam symbolů a zkratek

GP Genetické Programování

RL Zpětnovazební učení (Reinforcement learning)

ML Strojové učení (Machine Learning)

EA Evoluční algoritmy

DDoS Distribuované odepření služby (Distributed Denial of Service)

DNA Deoxyribonukleová kyselina

GUI Grafické uživatelské rozhraní (Graphical user interface)

TCP Protokol pro řízení přenosu (Transmission Control Protocol)

UDP Protokol pro přenos datagramů (User Datagram Protocol)

IP Protokol síťového připojení (Internet protocol)

A Obsah elektronické přílohy

Elektronická příloha obsahuje projekt v jazyce Java. Jedná se o kompletní implementaci platformy vypracované v rámci této bakalářské práce. V příloze lze nalézt, implementaci klienta i serveru. Zároveň se zde nachází skript v jazyce Python demonstrující funkci modely (zpracování dat na vstupu a generování instrukcí). V neposlední řadě je přiložen návod k ovládání platformy.

```
RoboCode ..... adresář projektu
├── config ..... konfigurační soubory
│   ├── game.properties
│   └── ...
├── libraries ..... záloha knihoven
│   └── ...
├── manualImages ..... adresář s obrázky pro manuál
│   └── ...
├── src
│   └── cz/vutbr/feec/robocode. ..... balíček s kódem
│       ├── battle. ..... spuštění hry
│       │   ├── BattleObserver
│       │   ├── RobocodeRunner
│       │   ├── ServerRunnertestSecond
│       │   └── StudentServerRunner
│       ├── data. ..... zpracování dat
│       │   ├── ProcessedBattleData
│       │   ├── ProcessedBulletData
│       │   └── ProcessedTankData
│       ├── protocol. ..... definice paketů
│       │   ├── RobocodePacket
│       │   ├── RobocodeRequest
│       │   └── RobocodeResponse
│       ├── socket ..... implementace serveru a klienta
│       │   ├── AgentCommunicator
│       │   ├── SocketsHolder
│       │   ├── TCPClientSocket
│       │   └── TCPServerSocket
│       └── studentRobot. ..... adresář pro studentské modely
│           ├── default.py
│           └── practice.py
│   └── sample ..... adresář s autonomními roboty
│       ├── RobotClient ..... klientský tank
│       └── ...
└── README ..... uživatelský manuál
```