Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Tímový projekt
RoboCup 3D

Dokumentácia k inžinierskemu dielu

Študijný odbor: Softvérové inžinierstvo, Informačné systémy
Predmet: Tímový projekt
Akademický rok: 2012/2013
Téma: RoboCup 3D
Číslo tímu: 15
Názov tímu: A55 Kickers
Vedúci tímu: Ing. Marián Lekavý, PhD.
Kontakt: tptim15@googlegroups.com

Bc. Matej Červeňák
Bc. Jaroslav Grega
Bc. Martin Gregor
Bc. Michal Chylik
Bc. Gábor Nagy
Bc. Matúš Ondrejkovič
Bc. Filip Sucháč
Bc. Matej Škoda

História zmien[1]

Tabuľka 1.1: História zmien

Zmena	Vykonaná zmena	Dátum zmeny	Vykonal	Platné od verzie
1.	Vytvorená kostra dokumentu	11.11.2012	M. Červeňák	1.0
2.	Pridané časti: Úvod, Plán projektu na zimný semester, Šprinty, Analýza tímu Androids, Analýza tímu rUNSWift, Projekt od konkurenčného tímu, Vytvorenie nového pohybu, Analyzovanie RoboCup wikipedie, Návrh celkovej práce počas projektu, Tvorba dokumentácie	13.11.2012	M. Červeňák	1.1
3.	Pridané časti: Analýza tímu High 5, Analýza tímu beeStambul, Manažérsky softvér	13.11.2012	G. Nagy	1.1
4.	Pridané časti: Analýza tímu Austin Villa, Webová stránka, Server na RoboCup, Spustenie hráča na server, Git, Analýza zdrojových kódov z minulého roka	13.11.2012	M. Gregor	1.1
5.	Pridané časti: Analýza tímu TÍM 17 ŽIJE, Analýza tímu Nexus3D, Analýza robota ASIMO, Návod na písanie kódu	13.11.2012	F. Sucháč	1.1
6.	Pridané časti: Analýza tímu magmaOffenburg, Analýza pohybov	13.11.2012	J. Grega	1.1
7.	Pridané časti: Analýza tímu Karachi Koalas, Analýza pohybov	13.11.2012	M. Ondrejkovič	1.1
8.	Update dokumentu.	13.12.2012	M. Červeňák	2.1
9.	Pridané časti k 3. a 4. šprintu	14.12.2012	M. Červeňák M. Gregor F. Sucháč J. Grega	2.1
10.	Pridaná časť Celkový pohľad	14.12.2012	M. Červeňák M. Gregor J. Grega	2.1
11.	Finalizácia dokumentu	14.12.2012	M. Červeňák	2.1
12.	Pridanie častí: Plán projektu na letný semester, Šprinty 5-8	18.04.2013	M. Červeňák	3.0
13.	Vloženie dokumentácie k úloham špecifikácia videnia sveta, šanca	19.04.2013	M. Gregor	3.1

[1] Tabuľka histórie zmien zahŕňajúca oba semestre

	získania lopty, taktika – som najbližšie k bráne			
14.	Vloženie dokumentácia k úlohám: Špecifikácia a návrh nižších pohybov, Prepojenie low skillov, Kopy do rôznych vzdialeností, Úkroky do strán a po kružnici, Integrovať naše nové funkcie	20.04.2013	G. Nagy	3.2
15.	Vloženie dokumentácia k úlohám: Návrh vylepšenia taktiky, Implementovať diagramy taktiky, Aktualizovať dokumentáciu	20.04.2013	M. Červeňák	3.3
16.	Vloženie dokumentácia k úlohám: Návrh vylepšenia vyšších pohybov Implementovať prihrávanie Optimalizovať kopanie do lopty Príprava materiálov na ITSRC	20.04.2013	J. Grega	3.4
17.	Vloženie dokumentácia k úlohám: Príprava Test Frameworku na turnaj High skill: Drž formáciu Vyladenie zintegrovaného riešenia	21.04.2013	F. Sucháč	3.5
18.	Vloženie dokumentácie k úlohám: Implementovať vedenie na bránu, Vyriešiť problém s plánovačom Opraviť chybu v lokalizovaní lopty	21.04.2013	M. Chylik	3.6
19.	Vloženie dokumentácie k úlohám: Prídenie k lopte zo správnej strany Zabezpečenie technickej stránky prezentácie výsledku	21.04.2013	M. Škoda	3.7
20.	Finálna úprava dokumentu pred kontrolným bodom	21.04.2013	M. Červeňák	3.8

Obsah

História zmien

Tabuľka 6.1: História zmien

Zmena	Vykonaná zmena	Dátum zmeny	Vykonal	Platné od verzie
1.	Vytvorená kostra dokumentu	11.11.2012	M. Červeňák	1.0
2.	Pridané časti: Úvod, Plán projektu na zimný semester, Šprinty, Analýza tímu Androids, Analýza tímu rUNSWift, Projekt od konkurenčného tímu, Vytvorenie nového pohybu, Analyzovanie RoboCup wikipedie, Návrh celkovej práce počas projektu, Tvorba dokumentácie	13.11.2012	M. Červeňák	1.1
3.	Pridané časti: Analýza tímu High 5, Analýza tímu beeStambul, Manažérsky softvér	13.11.2012	G. Nagy	1.1
4.	Pridané časti: Analýza tímu Austin Villa, Webová stránka, Server na RoboCup, Spustenie hráča na server, Git, Analýza zdrojových kódov z minulého roka	13.11.2012	M. Gregor	1.1
5.	Pridané časti: Analýza tímu TÍM 17 ŽIJE, Analýza tímu Nexus3D, Analýza robota ASIMO, Návod na písanie kódu	13.11.2012	F. Sucháč	1.1
6.	Pridané časti: Analýza tímu magmaOffenburg, Analýza pohybov	13.11.2012	J. Grega	1.1
7.	Pridané časti: Analýza tímu Karachi Koalas, Analýza pohybov	13.11.2012	M. Ondrejkovič	1.1
8.	Update dokumentu.	13.12.2012	M. Červeňák	2.1
9.	Pridané časti k 3. a 4. šprintu	14.12.2012	M. Červeňák M. Gregor F. Sucháč J. Grega	2.1
10.	Pridaná časť Celkový pohľad	14.12.2012	M. Červeňák M. Gregor J. Grega	2.1
11.	Finalizácia dokumentu	14.12.2012	M. Červeňák	2.1

Zoznam skratiek

Tabuľka 1.2: Zoznam skratiek

Skratka	Vysvetlenie
pp	Používateľský príbeh
tp	Technický príbeh

1 Úvod

Tento dokument predstavuje dokumentáciu k inžinierskemu dielu tímového projektu *RoboCup – tretí rozmer*, ktorý je vypracovávaný členmi tímu 15 s názvom *A55 Kickers*. Riešenie tímového projektu je obsahom kurzu Tímový projekt na FIIT STU v akademickom roku 2012/2013.

Dokument vytvoril tím 15 a spolu s ostatnými dokumentmi patrí medzi ich internú dokumentáciu. Slúži členom samotného tímu, ako i vedúcemu celého projektu. Takisto ho môžu využiť i ostatní čitatelia ako inšpiráciu pri riešení podobných projektov. Dokument obsahuje tri kapitoly.

V druhej kapitole sa nachádza plán projektu na zimný semester. Plán je rozdelený na jednotlivé týždne resp. šprinty, ku ktorým sa viažu naplánované úlohy.

V tretej kapitole sú spísané jednotlivé šprinty a ich náplne. Ku každému šprintu sú zobrazené používateľské príbehy a úlohy, ktoré sa k príbehom viažu. Obsahom tejto kapitoly sú tiež detailné popisy príbehov a úloh.

V štvrtej kapitole sa nachádza popis celkového pohľadu na projekt.

2 Plán projektu na zimný semester

V tíme vývoj postupuje metódou Scrum v dvojtýždňových intervaloch, kde na konci každého šprintu zhodnotíme čo sme urobili.

Prvé tri týždne slúžili na výber témy projektu a dohodnutie štruktúry tímu.

V zimnom semestri sú naplánované 4 šprinty. Prvé dva budú zamerané na analýzu problémovej časti a následne dva na návrh a prvotnú implementáciu prototypu.

Šprint/Týždeň	Úlohy	Obdobie
1. týždeň	Výber témy tímového projektu Vypracovanie ponúk	24.9.2012 - 30.9.2012
2. týždeň	Pridelenie tém	1.10.2012 - 7.10.2012
1. šprint	Analýza minuloročných tímov Analýza zahraničných tímov Analýza fyzikálnych modelov robotov Analýza wikipedie RoboCupu Manažérsky softvér Web stránka tímu Inštalácia simulačného prostredia Prvé pohyby robotov	10.10.2012 - 31.10.2012
2. šprint	Návod na používanie Git-u Návod na písanie kódu Analýza zdrojových kódov z minulého roka Analýza chôdze hráča Zlúčenie zdrojových kódov tímu 5 a 17 Návrh plánu projektu Vytvorenie dokumentácie	31.10.2012 - 14.10.2012
3. šprint	Vylepšenie low-lewel pohybov (chôdza, kráčanie za loptou, úkroky, kopnutie do lopty...)	14.10.2012 - 28.10.2012
4. šprint	Vytvorenie prototypu Doplnenie chýbajúcej dokumentácie	28.10.2012 - 12.20.2012

3 Šprinty

V tejto časti dokumentu sú opísané jednotlivé šprinty a k nim zodpovedajúce používateľské príbehy, technické príbehy a úlohy.

Keďže v prvých dvoch šprintoch bola našou úlohou analýza stávajúceho systému vytvoreného v predchádzajúcom roku, nebol vytvorený produkt backlog. Ten sa začal napĺňať až počas 2. šprintu. Vykonávanie jednotlivých užívateľských a technických príbehov, obsiahnutých v backlogu, bolo náplňou až ďalších šprintov. No z dôvodu prehľadnejšieho zápisu uvažujeme v prvých dvoch šprintoch jednotlivé úlohy daných šprintov ako používateľské resp. technické príbehy. Číslovanie jednotlivých úloh zostáva zachované.

3.1 Šprint 1 – „alfa“

Tabuľka 3.1: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
1.1	Člen tímu	Vlastniť webovú stránku	Zhromažďovať a zverejňovať všetky informácie o projekte na jednom mieste
1.2	Člen tímu	Funkčný server na RoboCup	Spúšťanie hráča a testovacieho frameworku
1.3	Člen tímu	Získať projekt od konkurenčného tímu z minulého roka	Využiť funkcionalitu obsiahnutú v ich projekte
1.4	Člen tímu	Spustiť hráča na server	Vyvíjať a testovať novú funkcionalitu
1.5	Člen tímu	Vytvoriť nový pohyb hráča	Ľahšie pochopiť nasledujúci vývoj
1.6	Člen tímu	Analyzovať projekty z minulých rokov	Pochopiť a zistiť, čo je alebo nie je implementované
1.7	Člen tímu	Analyzovať wikipediu RoboCupu	Pochopiť a zistiť, čo je alebo nie je implementované
1.8	Člen tímu	Analyzovať zahraničné úspešné tímy	Zistiť možné prístupy a následne ich prípadne navrhnúť a zapracovať
1.9	Člen tímu	Analyzovať fyzikálnych modelov robotov	Zistiť možné prístupy a následne ich prípadne navrhnúť a zapracovať
1.10	Člen tímu	Server na Git	Verziovať zdrojový kód
1.11	Člen tímu	Mať nainštalovaného klienta na Git	Verziovať zdrojový kód
1.12	Člen tímu	Mať manažérsky softvér	Lepšie manažovať riadenie tímu

Tabuľka 3.2: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
1.1	-	1.1	Rozbehať webovú stránku	M. Gregor
1.2	-	1.2	Rozbehať server na RoboCup.	M. Gregor
1.3	-	1.3	Získať projekt od konkurenčného tímu z posledného roka.	M. Červeňák
1.4	-	1.4	Spustiť hráča na server	M. Gregor

1.5	-	1.5	Vytvoriť pohyb alebo upraviť existujúci pohyb hráča v editore.	všetci
1.6	-	1.6	Študovať dokumentáciu a projekty z minulých rokov.	všetci
1.7	-	1.7	Študovať wiki k RoboCup-u.	všetci
1.8	-	1.8	Študovať zahraničné tímy a zápasy.	všetci
1.9	-	1.9	Analyzovať robotov a prečítať si o metódach učenia, stabilizácie...	M. Ondrejkovič J. Grega F. Sucháč
1.10	-	1.10	Rozbehať Git	Martin Gregor
1.11	-	1.11	Nainštalovať Git u seba.	všetci
1.12	-	1.12	Rozbehať manažérsky softvér	Gábor Nagy

3.1.1 Webová stránka (pp 1.1)

3.1.1.1 Špecifikácia

Webové sídlo projektu musí byť dostupné na webe cez adresu http://labss2.fiit.stuba.sk/TeamProject/2012/teamNNSS/, kde `NN` je číslo tímu (pre IS a SI 01 až 15, pre PKSS 01 až 06) a `SS` je skratka študijného programu/programov (`pss` alebo `is-si`). Všetky súčasti webového sídla sa musia nachádzať na našom serveri (t.j. nie presmerovať stránku niekde inde). Stránky musia byť na serveri umiestnené v DocumentRoot, takže sa zobrazia pri prístupe na tento server. Webové sídlo má informovať o postupe prác tímu a treba ho pravidelne (týždenne) aktualizovať. Na konci projektu náš tím vytvorí a odovzdá statický obraz webového sídla a takto ho bude prezentovať na elektronickom médiu a aj na webe.

Minimálny obsah stránky

- názov témy
- riešitelia a základné informácie o nich (napr. z ponuky)
- plán projektu (na semester)
- aktuálny stav plnenia plánu (t.j., úlohy, ktoré vyplynuli zo stretnutí, ich plnenie a vzťah k plánu)
- roly jednotlivých členov tímu (aj dočasné)
- odkazy na doteraz vypracovanú dokumentáciu vrátane záznamov zo stretnutí (záznamy zo stretnutí budú priamo čitateľné na webe, t.j. najlepšie vo formáte pdf, prípadne HTML)
- všetko zaujímavé v súvislosti s projektom a postupom prác na projekte, napr. odkazy na iné zdroje súvisiace s témou projektu

3.1.1.2 Analýza

Vhodným prostredím na rýchle a udržiavateľné informovanie o stave projektu je systém na správu obsahu ako napríklad wordpres, drupal a iné. CMS (Content Management System) systémy ponúkajú množstvo modulov na uľahčenie a podporu tvorby rôznych typov obsahu ako sú texty, zoznamy, triedenie podľa taxonómie. Sú to serverové aplikácie, ktoré pre správnu funkčnosť vyžadujú na serveri nainštalovaný PHP procesor a databázu ako napríklad PostgreSQL alebo MySQL.

Školský server pristupuje k našim serverom cez reverznú proxy. CMS systémy majú možnosť v konfigurácii nastaviť prístup prostredníctvom reverznej proxy.

3.1.1.3 Návrh

Vytvorili sme stránku na CMS systéme Drupal. Stránka má funkcionalitu pridávania obsahu aktualít, kde sa dá nastaviť dátum, ku ktorému sa aktualita viaže. Ďalej na stránke sa dá vytvoriť typ obsahu súbor, ktorý sa dá zaradiť do kategórie pomocou priradenia výrazu z taxonomického slovníka. Dá sa vytvoriť aj obsah typu webový dokument, kde môžeme vytvárať webové tabuľky, číslované/nečíslované zoznamy, pridávať odkazy a aj obyčajný alebo rôzne formátovaný text. V stránke existuje špeciálny výpis zoznamu aktualít zoradených podľa dátumu, ku ktorému sa viažu a zoznam súborov utriedených do kategórií z taxonomického slovníka. Z obsahových častí obsahuje stránka ešte úvodnú stránku, výpis informácií o členoch tímu a tiež stránku s užitočnými odkazmi použitými počas vývoja produktu. Prístup k stránkam je v prehľadnom menu. Pridávanie obsahu môže len prihlásený používateľ, ktorého overenie a bezpečnosť pri práci zabezpečuje CMS systém Drupal.

3.1.1.4 Implementácia

CMS systém Drupal 7 sme postavili na technológii hypertextového preprocesora PHP 5.3 a databázového servera PostgreSQL 9.1. Všetky technologické komponenty boli nainštalované na webovom serveri Apache 2.2 na operačnom systéme Fedora 17. Aby správne Drupal fungoval potrebovali sme pridať do súboru /sites/default/settings.php nastavenie na akceptovanie cookies súborov cez reverzný proxy server. Do CMS systému sme nainštalovali moduly Views, Taxonomy, Autopath, DatePicker, CKEditor, IMCE. Inštalácia modulov prebehla krokmi:

1. Stiahnutie požadovaných modulov z Drupal portálu.
2. Rozbalenie zazipovaných súborov a ich nakopírovanie do priečinku od koreňa webu /sites/all/modules/
3. Vyčistenie Cache pamäte Drupalu a povolenie nakopírovaných modulov cez administračné rozhranie drupalu.
4. Konfigurácia modulov

Následne sme v systéme vytvorili typy obsahu aktuality s poliami datum, title, body a typ obsahu Dokumenty s poliami title, súbor, kategória. Typ obsahu stránka (webový dokument) je defaultný typ obsahu v CMS systéme Drupal. Vytvorili sme slovník Typy dokumentov, ktorý obsahuje slová (kategórie) Technická dokumentácia, Zápisy stretnutí, Šablóny, Metodiky, Produkt na stiahnutie, Ostatné. Pole kategória v type obsahu Dokumenty berie kategórie z vytvoreného slovníka Typy dokumentov.

3.1.2 Server na RoboCup (pp 1.2)

3.1.2.1 Analýza

Server na RoboCup je Simspark server. Defaultná verzia na stiahnutie je určená pre 32 bitové operačné systém. Pre operačné systémy s 64 bitovou architektúrou je odporúčané si simspark server skompilovať na danom operačnom systéme. Návod je kompiláciu je dostupný Wiki stránke projektu RoboCup. Simspark server vyžaduje nainštalované knižnice ako ruby 1.9 a vyššie, openGL

a iné, ktorých zoznam nájdete v manuáli inštalácie Simspark servera. Simspark server ponúka tri nástroje a to rcssserver, rcssmonitor a rcssagent. Nástroj rcssserver spúšťa server prostredia simulovaného robotického futbalu. Nástroj rcssmonitor graficku vizualizuje svet robotického futbalu. A nakoniec rcssagent ponúka testovacieho agenta, s pustením ktorého si skontrolujeme funkčnosť servera.

3.1.2.2 Návrh

Pri inštalácii na 32 bitovej architektúre operačného systému stačí stiahnuť skompilovaný inštalačný súbor a otestovať spustenie servera a monitora simulovaného robotického futbalu Simspark. Ak nastane chyba a bude obsahovať infomáciu o chýbajúcej knižnici, tak knižnicu doinštalujeme. Na 64 bitových architektúrach treba stiahnuť zdrojové súbory Simsparku a skompilovať ich na danom stroji.

3.1.2.3 Implementácia

Všetci členovia tímu používajú 32 bitové architektúry operačných systémov, čiže pri inštalácii len stiahli skompilované súbory simsparku a server robotického futbalu naištalovali. Z chýbajúcich knižníc hráčom chýbala iba ruby knižnica, ktorú si doinštalovali. Server úspešne spustil každý člen tímu.

3.1.3 Projekt od konkurenčného tímu (pp 1.3)

Keďže sme začali pracovať na projekte, ktorý na škole pôsobí už istý čas, pokračujeme vo vývoji z minulého roka. V minulom roku boli na predmete Tímový projekt dva projekty RoboCup. Teda dva tímy nezávisle od seba vyvíjali rovnaký systém. My sme začali pracovať na jednom z nich. Druhý sme ale k dispozícii nemali, preto bolo potrebné ho získať od minuloročného projektu, aby sme mohli pracovať na zlúčení funkcionality oboch hráčov.

3.1.4 Spustenie hráča na server (pp 1.4)

3.1.4.1 Analýza

Zámerom úlohy bolo spustenie hocijakého hráča na Simspark severi a tým aj otestovanie fuknčnosti servera Simspark. Možnosti ako spustiť hráča sú:

1. Spustiť hráča z nástroja rcssagent
2. Spustiť hráča zo školského projektu Jim
3. Spustiť hráča zo školského projektu TestFramework

3.1.4.2 Návrh

Najrýchlejším riešením je spustenie hráča pomocou nástroja rcssagent. Ale tiež bolo potrebé aj vyskúšať fuknčnosť školských projektov.

3.1.4.3 Implementácia

Každý člen tímu na bežiacom serveri a monitore simulovaného robotického futbalu spustil hráča zo školského projektu Jim a z nástroja rcssagent.

3.1.5 Vytvorenie nového pohybu (pp 1.5)

3.1.5.1 Analýza

Zámerom úlohy bolo vytvorenie nového pohybu alebo upravenie existujúceho pohybu v editore pohybov. Editor pohybov má vypracovaný návod na použitie.

3.1.5.2 Návrh

Najlepším spôsobom ako vytvoriť pohyb je spustenie editoru pohybov a postupovať pomocou návodu na použitie.

3.1.5.3 Implementácia

Každý člen tímu vytvoril nový pohyb hráča, ktorý si overil na bežiacom serveri a monitore simulovaného robotického futbalu.

3.1.6 Analyzovanie dokumentácie z minulých rokov (pp 1.6)

3.1.6.1 Analýza tímu Androids

3.1.6.1.1 Pohyby agenta

Pohyb agenta pozostáva z fáz. Medzi jednoduché pohyby patrí napríklad kopnutie a medzi zložitejšie resp. cyklické napríklad chôdza.

Tím Androids v špecifikácii uvádzal viacero možností, ktoré by radi implementovali. V následujúcej tabuľke sú uvedené všetky. Vypísané sú pohyby, ktoré boli implementované a do kategórie „Návrh do budúcna“ sú zaradené tie, ktoré neboli z časového hľadiska implementované.

Tabuľka 3.3: Pohyby agenta

Pohyb	Implementované	Návrh do budúcna
Vstávanie	vstávanie z brucha vstávanie z chrbta vsávanie zo sedu rozkročmo	vstávanie bez využitia prostredia (posúvaním po ihrisku)
Chôdza	chôdza dopredu chôdza dozadu chôdza do strán (úkroky)	rozdelenie chôdze na rýchlu - nestabilnú a pomalú – stabilnú
Otáčanie	vylepšovanie už použitých pohybov implementovaných v XML vlastné pohyby (90°, 180°, vojenské otočenie)	
Kopanie	vytvorenie XML pre kopanie	

	kopanie dopredu špičkou a stranou porovnanie kopanie celým telom vs. Kopanie kĺbov spodnej končatiny	
Bránenie	bránenie pádom brankára s vystretými rukami do strán bránenie sadnutím na zem a rozkročenie nôh	

XML súbory slúžia na vytváranie zostáv pohybov, ktoré pozostávajú z fáz.

3.1.6.1.2 Vyššia logika

Po analýze agenta JIM predchádzajúceho tímu zistili, že agent nedisponuje vyššími schopnosťami. Tie teda implementovali. Vyššie schopnosti agenta umožňujú abstrahovať od detailov nižších schopností agenta a tak zjednodušiť ich výber a postupnosť. Definovaním pohybov vyššej logiky je možné kombinovať pohyby vyššej úrovne, ktoré by sa mohli využiť pri plánovaní stratégie.

- **Chôdza** – predstavuje schopnosť dostať sa na určené miesto. Budú využité nižšie schopnosti agenta ako pohyb dopredu, či otočenie sa.
- **Vstávanie** – predstavuje schopnosť vstať z polohy ležmo. Budú použité nižšie schopnosti vstávania z chrbta a vstávania z brucha.
- **Kop do lopty** - predstavuje schopnosť kopnúť do lopty na stanovený cieľ. Týmto cieľom môže byť bránka, spoluagent, prípadne akýkoľvek bod na ihrisku. Použijeme nižšie schopnosti rôznych typov kopania do lopty a otáčania.
- **Vedenie lopty** - prestavuje schopnosť vedenia lopty jedným agentom. Bude pozostávať zo schopností kopu do lopty a chôdze.
- **Zorientovanie sa** – predstavuje schopnosť zistiť svoju vlastnú polohu na základe polohy bránok a polohu lopty. Táto schopnosť bude pozostávať zo schopnosti otáčania sa.
- **Blokovanie strely** – predstavuje schopnosť zablokovať strelu správnym postavením agenta a následným zablokovaním strely nižšími schopnosťami ako pádom, či rozkročením. Túto schopnosť bude využívať predovšetkým brankár.

3.1.6.1.3 Editor správania

V editore pohybov implementovali:

- Pridanie panela, ktorý slúži používateľovi na výber, či sa má kĺb ohýbať samostatne alebo spolu s jeho „bratom“ (symetrickým párovým kĺbom) a možnosť voľby, či sa má pohyb vykonať symetricky alebo po zrkadlovej trajektórii.
- Plug-in „Editor správania“, implementovaný v jazyku Java.

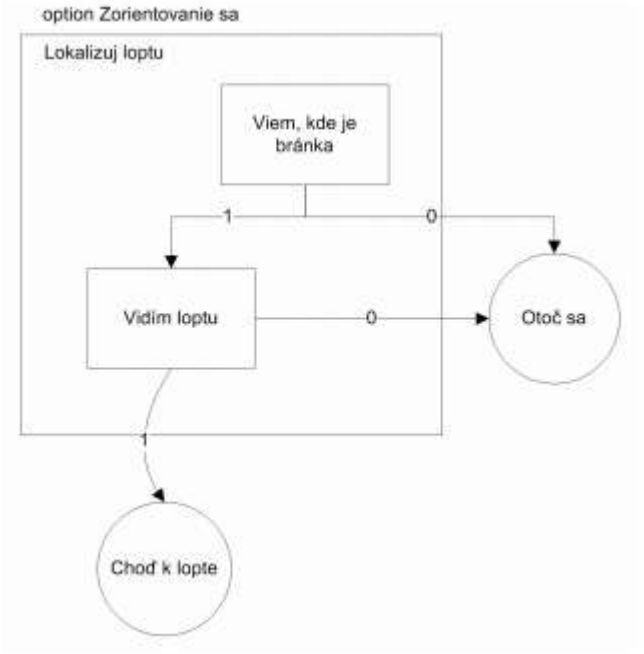
3.1.6.1.4 Definovanie vyššej logiky pomocou XABSL

Ako alternatíva produkčného systému plánovali použiť jazyk na definovanie vyššej logiky XABSL (Extensible Agent Behavior Specification Language), ktorý je vhodný na opísanie vyššej logiky hráča. K dispozícii je aj Java XABSL library. Opis správanie pomocou XABSL môže byť výhodné najmä, ak komplexnosť správania začne rásť a implementovanie v jazyku Java by začalo byť neefektívne.

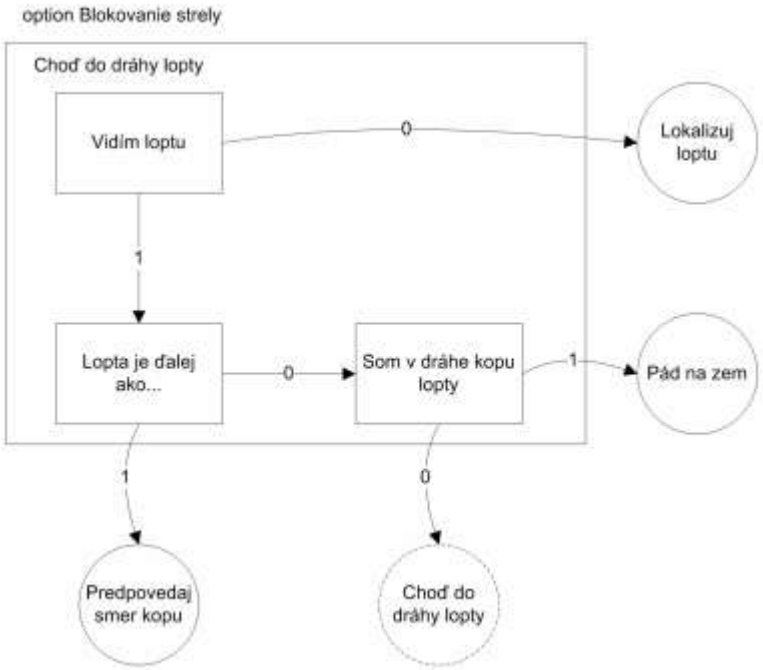
Agent v XABSL pozostáva z menších celkov – tzv. options, čo sú vlastne moduly správania, z ktorých každý je konečným automatom. V našom prípade pôjde o:

- **zorientovanie sa**
- **blokovanie strely**

Tieto sú zoskupené v strome, ktorého koncové uzly tvoria prvky nižšieho správania, ktoré vyvolajú požadovanú akciu. Prechody stavov realizujeme ako rozhodovacie stromy.



Obrázok 3.1: Pohyb vyššia logiky



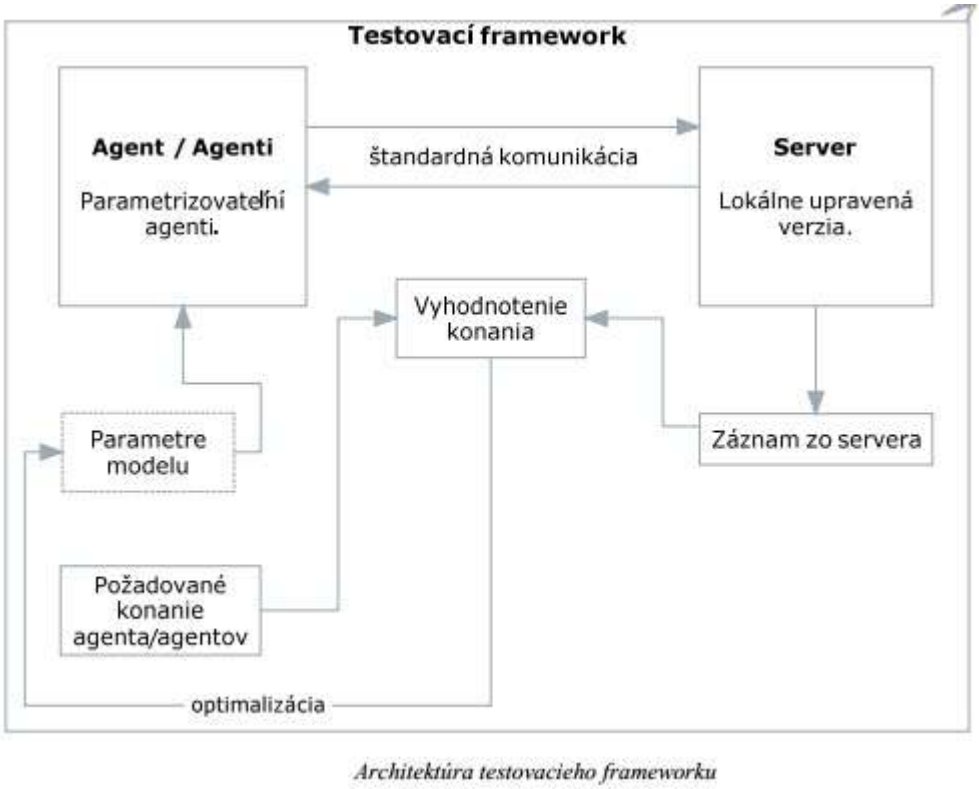
Obrázok 3.2: Pohyb vyššia logiky

3.1.6.1.5 Testovací framework

Z dôvodu zrýchlenia simulácie zasiahli do testovacieho frameworku a tak docielili efektívnejšie testovanie agenta.

Architektúra testovacieho frameworku

Nakoľko požadovaná funkcionalita pracuje na úrovni nad samotným agentom, testovací framework musí umožňovať vloženie agenta do systému. To platí v prípade testovania schopností nižšej úrovne. V prípade testovania vyššej logiky, musí framework umožňovať vloženie viacerých agentov do systému. Testovanie agenta vyžaduje špecifikovanie očakávaného výsledku. Následne po uskutočnení simulácie sa pomocou záznamu o zápase vyhodnotí, či výsledok dosiahol požadovanú úroveň. Ak nie, framework umožní technikami umelej inteligencie hľadať lepšie nastavenie parametrov. Vzhľadom na povahu problému bude vhodné použiť genetické algoritmy. Na obrázku je architektúra systému znázornená diagramom.



Obrázok 3.3: Architektektúra testovacieho frameworku

V nasledujúcej časti sú opísané netriviálne bloky, ktoré sa nachádzajú na obrázku:

- **Parametre modelu**. Blok špecifikuje jednoducho meniteľné parametre agenta, resp. agentov, ktoré budú môcť byť ovplyvňované v rámci optimalizácie. Parametre modelu predstavujú v skutočnosti komunikačný protokol pre použité genetické algoritmy a agenta.
- **Záznam zo servera**. Záznam zo servera slúži na získanie informácií o simulácii. Informácie, ktoré sa týmto spôsobom získajú, je následne potrebné pretransformovať do opisu, ktorý bude kompatibilný s opisom v bloku Požadované konanie agenta/agentov.
- **Požadované konanie agenta/agentov**. Požadované konanie je blok, do ktorého zadáva informácie používateľ. Tieto informácie sú opisom očakávaných udalostí, ktoré sa v bloku Vyhodnotenie konania budú porovnávať s informáciami získanými zoZáznamu zo servera.
- **Vyhodnotenie konania**. V predchádzajúcich opisoch bol uvedený účel blokuVyhodnotenie konania. Je potrebné uvedomiť si, že jeho zložitosťzávisí od miery rozdielu medzi informáciami získanými od používateľa a informáciami, ktoré získava blok zo Záznamu zo servera. Nakoľko informácie zo servera nie sú rýdzo deterministické,je potrebné vytvoriť opis, ktorý bude vždy môcť uvažovať s vopred stanovenou mierou nepresnosti.

3.1.6.1.6 Prototyp

Do prototypu boli zahrnuté nasledujúce funkcie:

- Implementácia testovacieho frameworku
- Pohyby agenta
- Úpravy v editore pohybov

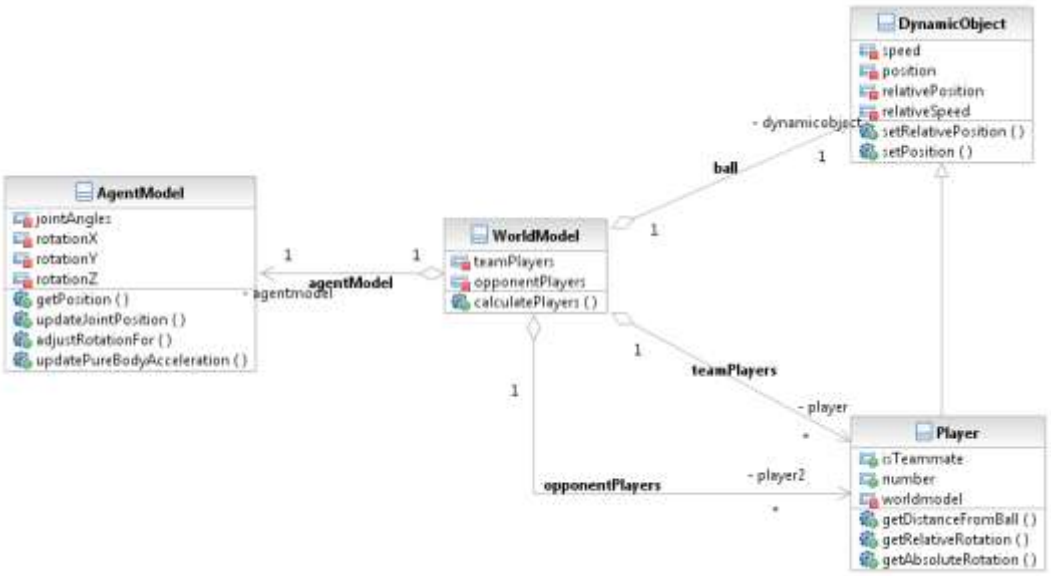
3.1.6.2 Analýza tímu High 5

3.1.6.2.1 Poloha ostatných hráčov

Tím High5 upravil parsovanie “see správ”, ktorých zdrojom je see receptor za účelom parsovania informácií o polohách ostatných hráčov na ihrisku.

Dáta o hráčoch sa ukladali do objektu PlayerData triedy Player. Trieda Player bola rozšírená aby uchovávala aj informácie o jednotlivých častí hráčov. Takto riešili získanie a uchovávanie informácií o polohe hráčov na ihrisku a o ich natočení.

Počítali aj vzdialenosť hráčov od lopty. Z absolútnej polohy hráčov (hlava) a lopty. Ďalej natočenie hráča voči lopte.



Obrázok49: Časť diagramu tried WorldModelu s dôrazom na hráčov

Obrázok 3.4: Časť diagramu tried WorldModelu s dôrazom na hráčov

Ako je vidno na obrázku tak upravená trieda Player už má implementované metódy:

- getDistanceFromBall() – ktorá používa metódu na určenie absolútnej polohy hráča na ihrisku
- getRelativeRotation()
- getAbsoluteRotation()

3.1.6.2.2 Testovací framework

Tím High5 vykonal dôkladný refaktoring testovacieho frameworku.

Rozdelili moduly na:

- Moduly vykonávajúce činnosti
- Implementačné moduly

Vyriešili spúšťanie pomocou jednej inicializačnej metódy, upravili logovanie aby bolo viac konzistentné a pribudol aj konfiguračný súbor.

Vďaka vyriešení obojsmernej komunikácie zo serverom je server informovaný o plánovaní hráčov (rozšírenie triedy AgentMonitor).

Tím vytvoril aj automatické spúšťanie hráča a servera, čo napomáha aj testovaniu. Vytvorené boli testy pohybov, ktoré dokážu ohodnotiť ich úspešnosť. Ďalej automatické vytváranie anotácií vytvorené v testovacom frameworku.

3.1.6.2.3 Pohyby

Optimalizácia kopu priniesla viacero úrovní sily kopu špičkou. Po analýze kopnutia špičkou po tíme Androids nebolo treba veľa zmeniť. Jediný problém bol s otáčaním hráča okolo svojej osi pri opakovanom vykonávaní pohybu. Problém bol odstránený. Tím vytvoril aj ďalšie 2 pohyby pre každú nohu, ktoré vychádzajú z kopnutia špičkou, líšia sa ale vo veľkosti napriahnutia a sile kopnutia.

Kopnutie bokom vylepšili väčším zohnutím kolena a členku pri kopaní, takto sa hráč nakloní viac dopredu a už s oveľa väčšou úspešnosťou triafa loptu.

Prerobili aj pohyb sadnutia, t.j. blokovania kopu sadnutím. Stabilitu pri tomto pohybe zlepšili nakláňaním trupu a úpravami záverečných fáz pohybu, ako napríklad zmenšením uhlu medzi nohami robota pri dosadnutí na zem. Vďaka úpravám sa zrýchlilo aj celkové vykonanie pohybu sadnutia. Tím vytvoril aj experimentálnu veľmi rýchlu verziu tohto pohybu. Tá ale nebola v konečnom dôsledku taká stabilná ako normálna verzia.

Optimalizovali a stabilizovali pohyb “walk_fine_fast2_optimized”. Pohyb zlepšili hlavne tým, že prerobili do “ľudskejšej formy”. Hráč viac používa kĺby a do istej miery imituje chôdzu človeka.

Vytvorili nový pohyb “walkback_slow”. Tento nový pohyb je vytvorený z pohybu “walkback3”, agent vykonáva ale menšie kroky smerom dozadu. Využíva sa najmä pri pohybu hráča okolo lopty.

Postavenie hráča za loptu. Vysokoúrovňový pohyb pohyb. Neúspešný návrh a implementácia pomocou vysokoúrovňových pohybov pohybov “WalkOld” (kráčanie za loptou) a “Lokalize” (lokalizovanie lopty).

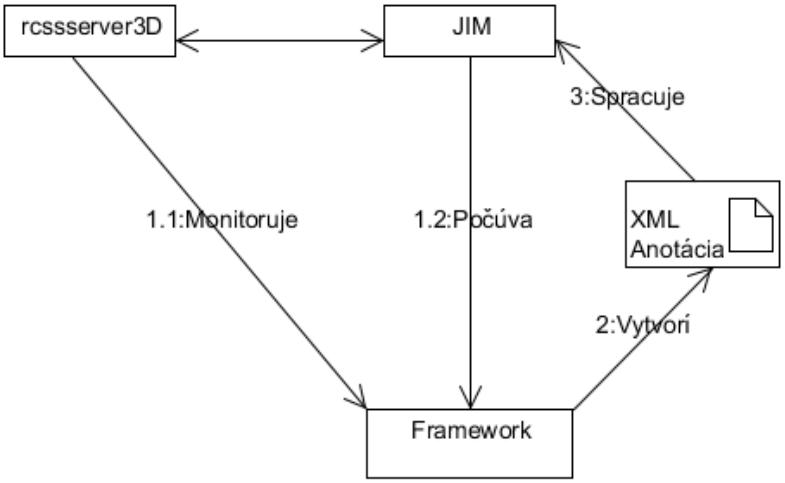
Vytvorili a optimalizovali 12 nízkoúrovňových pohybov. Ďalej vytvorili jeden komplexný vyšší pohyb, ktorého trieda je implementovaná tak, že ak z nej dedí iná trieda, musí implementovať jedine metódu pickHighSkill().

3.1.6.2.4 Anotácia pre pohyby

Tím navrhol a implementoval framework na tvorbu anotácií pohybov hráča. Anotácie slúžia na správne rozhodovanie pri plánovaní pohybov. Potrebovali k tomu počítať možnú pozíciu hráča z pohľadu hráča a nie z pohľadu ihriska.

Vypočítané hodnoty sa rozhodli uchovávať vo formáte XML.

Na obrázku je vidno proces spracovania údajov a vytváranie anotácií.



Obrázok 3.5: Proces spracovania údajov a vytváranie anotácií

Anotácie sa v druhom kroku rozšírili za účelom:

- lepšie reprezentovať vstupné a výstupné podmienky (poloha lopty)
- určiť bod najvyššej efektivity kopu do lopty

Vytvorili teda viac anotácií pre jeden pohyb a pridali rozptyl v akom sa lopta po vykonaní pohybu nachádza.

Zmenili reprezentáciu polohy lopty pomocou maximálnej, minimálnej a priemernej polohy, na kruhovú reprezentáciu.

3.1.6.2.5 Prerábanie GUI testovacieho frameworku

Navrhli a implementovali nové GUI s novými funkciami a vzhľadom. Doplnili možnosť pridávania nových hráčov, sledovania logov a možnosť ich konkrétneho výberu zo 7 kategórií (7 typov logovania).

Implementovali anotovací tab do GUI.

Vytvorili tretí tab v GUI na prenos dát sveta z Jim do testovacieho frameworku. Daný tab zahŕňa 2

podtaby, v ktorých je možné sledovať loptu alebo iného hráča z pohľadu agenta.

3.1.6.2.6 Ďalšie práce

Pomocou vykonanej analýzy hodnôt z akcelerometra a gyroskopu rozoznali hodnoty, ktoré je nutné sledovať pri navrhovaní pohybov robota, aby sa minimalizovali pády. Do budúcna sa to dá využiť ako pomôcka pri tvorení pohybov, ale aj ako navigácia pri adaptabilnej chôdzi.

Práca na vytvorení viacerých plánovačov.

3.1.6.3 Analýza tímu TÍM 17 ŽIJE...

3.1.6.3.1 Zistenie pozície stavu hráčov, ktorých agent vidí

Videný hráč stojí, ak je rozdiel Z súradnice hlavy a Z súradnice nôh väčší ako konštanta STANDING_LIMIT. Inak hráč leží.

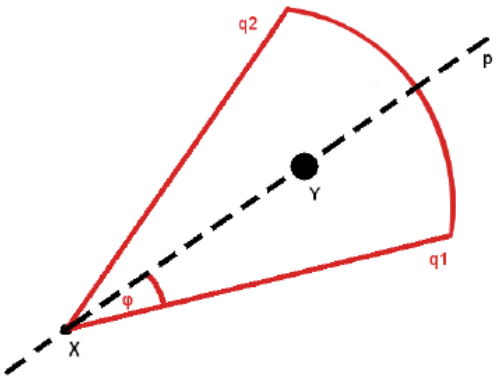
3.1.6.3.2 Vyhodnocovanie herných situácií

Pri brániacich a útočiacich situáciách boli pridané aj počty útočiacich a brániacich hráčov. Z polohy hráčov sa určuje, či je brániace mužstvo pod tlakom súpera alebo nie. Vlastník lopty je hráč s loptou od neho vzdialenou 1 meter. Hráč je pod tlakom ak je súper od neho vzdialený menej ako 2 metre.

3.1.6.3.3 Vytvorenie hernej formácie

Implementácia high skillu pre prejdenie hráč na určitú pozíciu vo formácii.

3.1.6.3.4 Zistenie vhodnosti prihrávky



X – hráč, Y – miesto kde má prihrávka skončiť.

|X,q1| = 1,5 x |X,Y|

Ak sa v červenom výseku nenachádza súper, tak je prihrávka vhodná.

3.1.6.3.5 Zadefinovanie high skillov

GetUp – hráč vstane ak leží na bruchu alebo na chrbte, ak leží inak tak vstáva ako z chrbta. Pri vstávaní z chrbta sa najprv preklápa na brucho.

Localize – keď hráč nevidí loptu dlhšie ako 5 sekúnd, tak sa otáča okolo svojej osi kým ju nenájde.

Walk – najskôr sa hráč k lopte natáča a kráča k nej, potom zaujíma vhodnú pozíciu.

Turn – smerovanie sa na cieľ a zároveň nevzdialenie sa od lopty.

Kick – pri vhodnej pozícii lopty a správnom natočení na bránku. Podľa vzdialenosti od lopty sa rozhoduje, či k nej ešte pristúpiť. Podľa polohy lopty sa rozhoduje, ktorou nohou vystrelí.

3.1.6.3.6 Prispôsobenie pohybov na zreťazenie

High skill sa môže nachádzať v jednom zo 4 stavov: initial, executing, finalizing, ended.

3.1.6.3.7 Plánovanie trajektórie

Trieda TrajectoryPlanner a zoznamy anotácií „rotate“ a „walk“ zabezpečujú výber pohybov vedúcich k želanému otočeniu a posunutiu sa po trajektórii.

3.1.6.3.8 Vylepšenie plánovača

Pri výpočte trajektórií existujú akceptovateľné odchýlky. Trieda Obstacles kontroluje prienik trajektórie s prekážkami, vypočítava bod obchádzania prekážky, kontroluje či sa bod nachádza vnútri hracieho poľa. Trieda *Trajectory* uchováva a sprístupňuje zoznam pohybov tvoriacich trajektóriu. Do triedy *Obstacles* bola pridaná metóda *getRealObstacles(),* ktorá z inštancie triedy *WorldModel* získava pozície všetkých hráčov na ihrisku a vracia ich ako zoznam prekážok. Trieda *TrajectoryRealTime* získava informácie pre výpočet trajektórie priamo z modelu agenta a modelu sveta.

3.1.6.3.9 Predikcia lopty

Predikcia pozície lopty sa počíta približne ako predikcia pozície hráča. Vyrátava sa vektor rýchlosti, pomocou ktorého sa počíta poloha hráča/lopty za X sekúnd.

3.1.6.3.10 Vytvorenie nových pohybov

Vytvorilo sa otočenie o 45 stupňov, pohyb otáčania sa okolo lopty, chôdza spojená s otáčaním, kop na diaľku.

3.1.6.3.11 Grafické zobrazenie v test frameworku

Do test frameworku bolo pridané grafické zobrazenie hry. V súčasnosti sa zobrazujú dáta z monitora pre všetkých agentov a loptu. Ďalej sa môžu zobraziť dáta z modelu sveta jedného alebo všetkých agentov, čo zahŕňa polohu a rotáciu agenta a polohu lopty.

3.1.6.3.12 Kop do lopty

Bol vytvorený silný priamy kop do lopty. Pozostáva z naváženia sa na jednu nohu, prikročenia k lopte druhou nohou, kopom prvou nohou a stabilizáciou hráča.

3.1.6.3.13 High skill – hranie futbalu

Implementovali sa metódy findTeammates pre nájdenie ostatných spoluhráčov, nearestToBall pre určenie či je hráč najbližšie k lopte čo vedie k priblíženiu sa k lopte, secondToBall ak je hráč druhý najbližší k lopte.

3.1.6.3.14 Zrýchlenie a spomalenie chôdze

Chôdza bola upravená tak, aby začala miernym prikrčením a zrýchľujúcou sa chôdzou, a končila spomaľujúcou sa chôdzou.

3.1.6.3.15 Vedenie lopty

Pri tomto pohybe sa hráč snaží zdvíhať nohy čo najmenej nad zem aby lopta neodskakovala ďaleko od agenta. Ďalej bol vytvorený high skill, ktorý zabezpečuje udržanie lopty stále pred agentom.

3.1.7 Analyzovanie RoboCup wikipedie (pp 1.7)

3.1.7.1 Analýza

Minuloročný tím vytvoril wikipediu k projektu RoboCup, ktorá popisuje najdôležitejšie časti a zhromažďuje najdôležitejšie údaje na jednom mieste. Wikipedia teda dokáže výrazne ušetriť čas pri hľadaní potrebných údajov.

3.1.7.2 Návrh

Každý člen tímu mal za úlohu analyzovať wikipediu RoboCupu a zvoliť si zopár zaujímavých skutočností, ktoré by bolo potrebné vedieť. Na nasledujúcom spoločnom tímovom stretnutí sa vytvorila diskusia, kde sa jednotlivé prístupy a informácie z wikipedie preberali a navzájom vymieňali medzi jednotlivými členmi tímu.

3.1.8 Analyzovanie zahraničných tímov (pp 1.8)

3.1.8.1 Analýza tímu Karachi Koalas

Karachi tím vznikol v polovici roka 2010 vďaka vedeckej spolupráci medzi technologickou univerzitou v Sydney (UTS) a inštitútu biznis administrácie, Karachi(IBA). UTS sa pravidelne zúčastňuje súťaže RoboCup už od roku 2003. V roku 2004 vyhrali súťaž RoboCup v Austrálii. Karachi Koalas skončil v prvej desiatke v súťaži RoboCup za rok 2012.

3.1.8.1.1 Vývojové prostredie

- C#/Mono
- TinMan pre komunikáciu zo serverom
- RoboViz na dynamické umiestňovanie lopty a hráčov, pohľad agenta

3.1.8.1.2 Architektúra

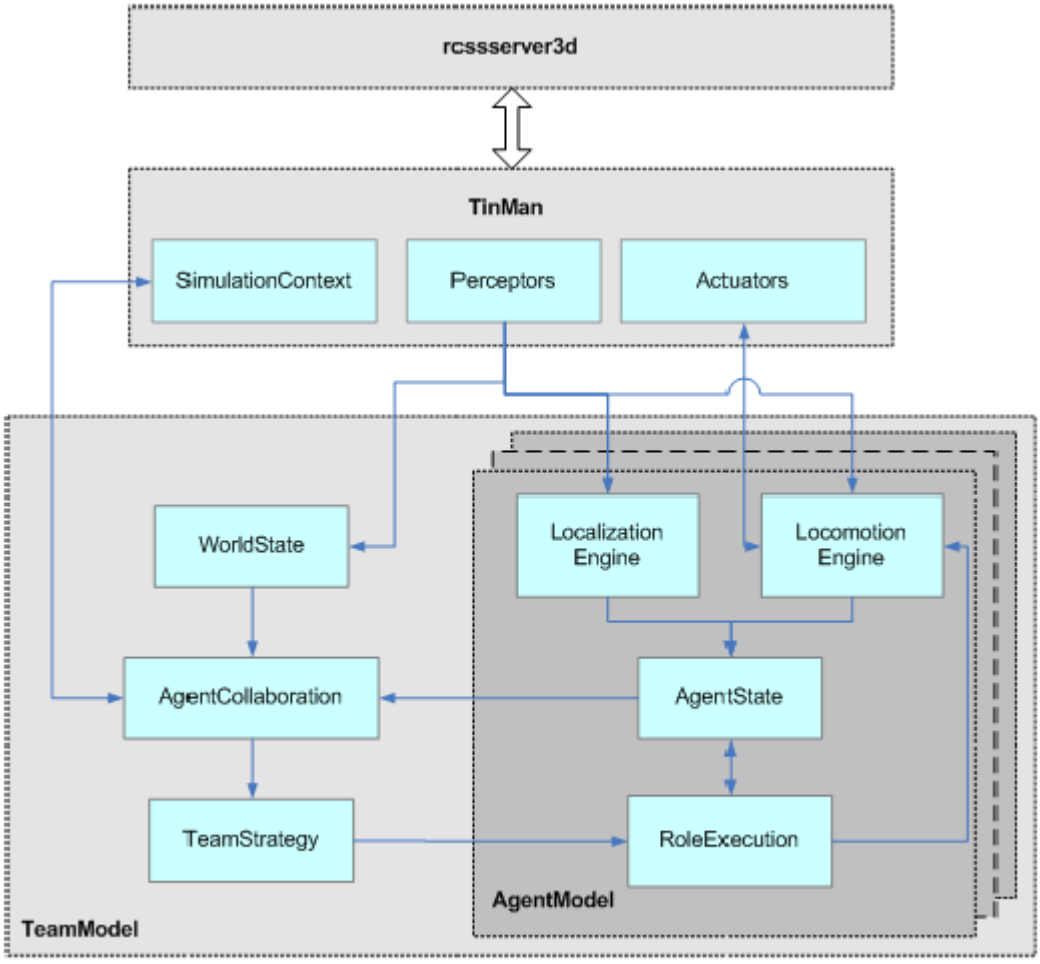


Fig. 1. Software Architecture

Obrázok 3.6: Softvérová architektúra

Ide o modulárnu architektúru. *TinMan* slúži ako rozhranie na RoboCup server *rcsserver3d*. Poskytuje abstrakciu aktivátorov(*Actuators*) a zachytávanie vnemov(*Perceptors*). *AgentModel* je zodpovedný za ovládanie jednotlivých agentov, ich stavov *AgentState*. Súčasťou agentovho modelu je *LocalizationEngine* a *LocomotionEngine.* Celkovú koordináciu medzi agentmi riadi *AgentCollaboration* a využíva pritom *AgentState* a *WorldState.* Aplikovaním heuristík rozhodne o stratégii. *TeamStrategy* má na starosti vykonanie určitej stratégie, spravuje formácie hráčov a preto poveruje *RoleExecution* zmenou roly jednotlivých hráčov. RoboCup server poskytuje priamu komunikáciu medzi agentmi cez rozhranie správ. *AgentCollaboration* používa *SimulationContext* na prijímanie a zasielanie broadcast správ.

3.1.8.1.3 Pohyblivosť

- Chôdza dopredu, dozadu a okolo
- Postavenie sa z chrbta a brucha, hádzanie brankára
- Kopy

3.1.8.1.4 Chôdza

Pre chôdzu bol navrhnutý spôsob inteligentného učenia sa pohybov kĺbov. Pohyby kĺbov boli modelované furiérovými radmi. Pri učení sa chôdze Nao robota a na optimalizáciu chôdze v simulačnom prostredí boli použité aj evolučné algoritmy. Pomocou NaoQi boli pozorovane ohyby kĺbov počas rôznych typov chôdzí. Zozbierané dáta použili v evolučných algoritmoch aby našli parametre pre furiérové rady. Rovnice, ktoré dostali boli rôznych zložitostí. Najzložitejšie mali 96 parametrov. Neskôr sa im počet týchto parametrov podarilo znížiť vďaka odhaleniu súvislostí pohybov jednotlivých kĺbov.

Fitnes funkcie evolučných algoritmov zahŕňali skutočnosti:

- Prejdená vzdialenosť robota
- Priamočiarosť jeho chôdze
- Stabilita robota pozorovaná gyroskopom

Pri otáčavých pohyboch bol dôraz na hľadanie najväčšieho počtu značiek v zadanom čase.

3.1.8.1.5 Postavenie sa z brucha, chrbta

Pomocou akcelerometra dokážu rozlíšiť polohu robota. So súradnicami x,y,z vedia rozlíšiť či robot padol na zem, či leží na chrbte alebo bruchu. Vždy keď robot padne na zem zavolá funkcie postavenia sa. Postavenie vyvoláva pohyby kĺbov v závislosti od stávania z bruchu, z chrbta.

3.1.8.1.6 Kopy

Tri typy kopov:

- Bočný kop – používaný ako nahrávka spoluhráčovi
- Uhlový kop – pre dribling alebo nahrávka hráčovi pred ním, v danom uhle
- Kop dopredu – pre strieľanie gólov alebo pasy

Kope sa pravou ako i ľavou nohou. Výber závisí na situácii.

3.1.8.1.7 Lokalizácia

Na lokalizáciu hráča používajú značky. Hráč má vnemové senzory a na základe týchto značiek sa určí poloha hráča. Vždy keď hráč vidí aspoň jednu značku tak vypočíta svoju polohu a orientáciu na ihrisku. Ak je v zábere iba jedna značka potom predpovedá pozíciu použitím rovníc a neskôr obnoví pozíciu hráča použitím priemeru súradníc. Ak nie je v dosahu žiadna značka, pohybovými rovnicami sú zistené približné súradnice. Počas chodenia hráč otáča hlavou zo strany na stranu aby vždy videl aspoň jednu značku. Ak hráč vidí loptu a je si istý jej pozíciou môže zasielať informácie iným hráčom čo loptu nevidia. Vyvinuli aj vizualizačný modul, ktorý odhaduje pozíciu hráča v poli a jeho pohyb v grafickom okne na overenie správnosti implementácie.

3.1.8.1.8 Stratégia

Dôraz riešenia bol kladený na lokalizáciu a pohyblivosť. Ich stratégia podporuje formácie tímov, prepínanie rol, vyhýbaní sa nebezpečných situácií. Vytvorili 4 roly, ktoré hráči môžu zastávať: brankár, obranca, záložník, útočník. Obrancovia sú rozložení na pravých a ľavých. Rola záložníka je dynamická a môže sa po splnení určitých podmienok stať útočníkom. Ostatní hráči sú predvolene záložníci.

3.1.8.1.9 Zdroje

- http://karachikoalas.iba.edu.pk/files/karachikoalas_TDP.pdf

3.1.8.2 Analýza tímu rUNSWift

Tím rUNSWift vznikol v roku 1999. V rokoch 1999 až 2006 sa zúčastňoval svetovej súťaže v najvyššej kategórii 4-Legged League, kde dosahoval výborné výsledky. Medzi prvými troma priečkami sa umiestnil v každom roku. Prvé miesto tím dosiahol v rokoch 2000, 2001 a 2003. Od roku 2008 súťažili v kategórii Standard Platform League, v ktorej boli štvornohý roboti nahradený robotmi dvojnohými. V roku 2010 získali druhé miesto a v roku 2012 obsadili tretiu prečku. Tím rUNSWift patrí pod univerzitu The University of New South Wales, ktorá sídli v Austrálii.

3.1.8.2.1 Výskum

Dlhodobým cieľom je vývoj všeobecne použiteľného inteligentného systému, ktorý by učil vykonávať mnoho rozličných úloh samostatne len za pomoci interakcie v prostredí.

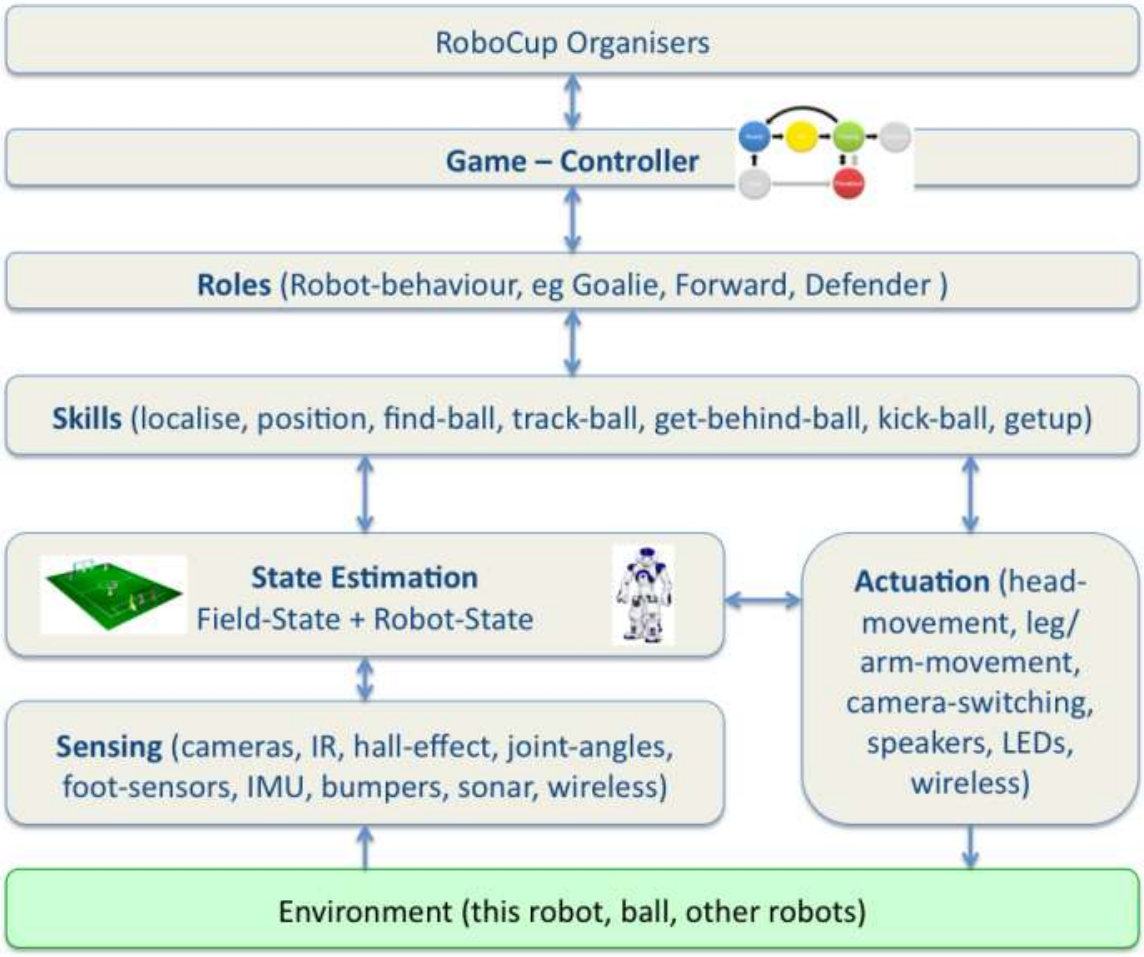
Hlavnými všeobecnými výskumnými cieľmi tímu rUNSWift v SPL (Standart Platform League) je:

- Ďalej rozvíjať rozumové metódy, ktoré zahŕňajú neurčitosť a real-timeové obmedzenia. A integrovať ich so štatistickými metódami používanými pri vnímaní.
- Vyvíjať metódy pre používanie odhadov neurčitosti a riadiť tak budúce rozhodovanie tak, aby eliminovala neistota a neurčitosť.
- Rozšíriť tieto metódy na multi-robotovú spoluprácu.
- Používať symbolickú reprezentáciu ako základ pre interakciu humanoidného robota.
- Vyvíjať učiace algoritmy pre hybridné systémy, ako je využitie znalostí logických obmedzení na vyvarovanie sa vyhľadávaniu pokus-omyl.
- Vyvíjať high-level symbolický jazyk pre robotov, ktorý poskytne abstrakciu pre veľké množstvo úvah, plánovanie a učiace techniky tza účelom zjednodušenia programovania robotov.

3.1.8.2.2 Robotická architektúra

Robotická architektúra je úlohová hierarchie pre multi-agentný tím štyroch Naos (Naos je najnovší druh robota, ktorý sa používa na svetových turnajoch). Tím používa centrickú architektúru s chybovou toleranciou. To znamená, že každý robot môže mať mierne odlišný pohľad na svet a preto aj inú rolu v tíme. Tento prístup má tú výhodu, že poskytuje čiastočnú redundanciu v prípade, že niektorý robot prestane pracovať alebo je diskvalifikovaný.

Herný ovládač vyvolá na koreňovej úrovni high-level stavy pre začatie hry. Na nižšej úrovni, generátor chôdze vykoná fázy chôdze, ktoré vyvolajú stav prechodu tvoriaci pohyb robota.



Obrázok 3.7: Robotická architektúra

3.1.8.2.3 Videnie

Systém videnia vyvinul tím ešte v roku 1999. Od začiatku používali jednoduchý výučbový systém na trénovanie systému na rozpoznávanie farieb. V roku 2001 začali používať strojové učenie na budovanie rozoznávača rozhodovacieho stromu. To sa ukázalo veľmi dôležité, pretože v súťažiach by ich predchádzajúci systém videnia nebol dostatočný.

V posledných rokoch bol ich systém aktualizovaný na rozpoznávanie ELD značenia a menej sa orientovali na farbu použitím hranových funkcií. V súčasnosti skúmajú *foveated* videnie a virtuálne kmitanie za účelom maximalizácie výpočetných zdrojov dostupných v Nao-vi. Pre eld čiary a eld hrany vyvinuli senzorový model, ktorý poskytuje viac hypotéz pre novú lokalizáciu.

3.1.8.2.4 Lokalizácia

V roku 2000 bolo prvý krát na súťaži použitá Kalman Iter lokalizačná metóda, ktorá bola v nasledujúcich rokoch vyvíjaná. Vďaka výhodnej lokalizácii a pohyblivosti v roku 2000 tím nedal nikdy menej ako 10 gólov a dostal maximálne jeden gól. Od roku 2000 bol systém lokalizácie prepracovaný a zahŕňa multi modálne distribuovanie dát cez sieťovaných robotov. V roku 2006

upustili od zdieľania informácii medzi robotmi ale zaoberali sa nimi ako s jedným tímom viacerých robotov. To umožňovalo spracúvať viac hypotéz a taktiež používať loptu pre získanie informácii ohľadne lokalizácie.

3.1.8.2.5 Pohyblivosť

V roku 2000 bol predstavený pohyb UNSW (skratka univerzity), ktorý sa používal na súťažiach. Kľúčovou myšlienkou tohto pohybu bolo popísať trajektóriu labky robota jednoduchými geometrickými prvkami, ktoré boli parametrizované. Vďaka tomu sa stali hráči rýchlejší ako mali konkurenčné tímy. Od vtedy mal takmer každý tím na turnaji podobný štýl pohybov, ktorý vychádzal z kódu rUNSWIFT. Zmena prišla v roku 2003, keď tím požil strojové učenie na vylepšenie robotovej chôdze a tým dosiahol moc rýchlejšie pohyby. Od vtedy veľa tímov používalo ich systém na strojové učenie na vylepšovanie chôdze svojich hráčov.

Náš súčasný výskum je zameraný na výučbu všestrannosti bipedálnych pohybových ovládačov použitím hierarchického posilňujúceho učenia.

3.1.8.2.6 Zdroje:

- http://cgi.cse.unsw.edu.au/~robocup/2011site/reports/Robocup2011rUNSWiftTeamDescription.pdf
- http://www.cse.unsw.edu.au/help/students/robocup/

3.1.8.3 Analýza tímu beeStambul

BeeStambul je projektom laboratória umelej inteligencie a robotiky na Technickej univerzite v Istambule. Na RoboCup projekte sa zúčastňujú od roku 2009, súťažia od roku 2010 a majú za sebou mnoho úspešných víťazstiev.

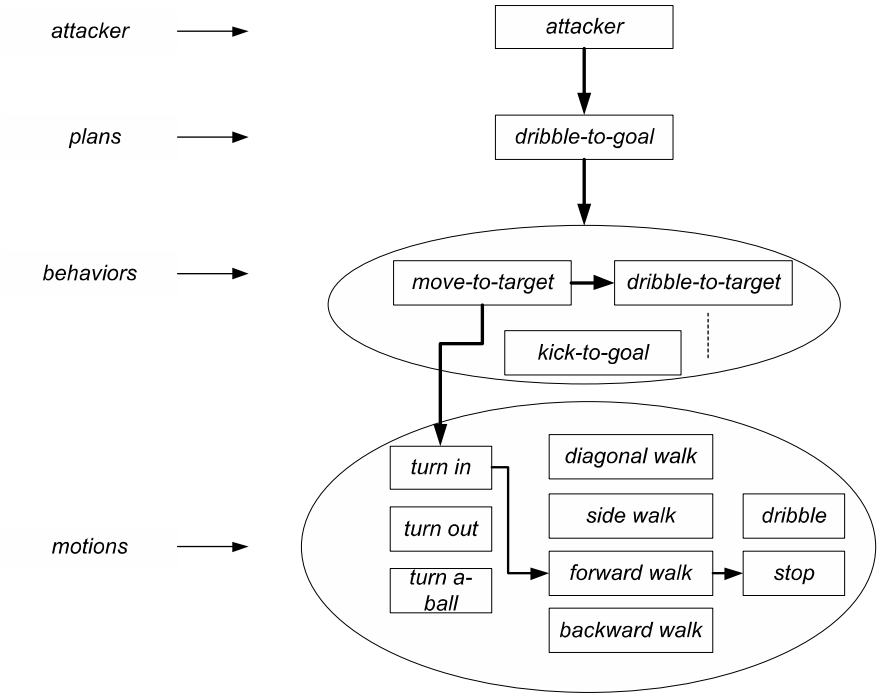
3.1.8.3.1 Návrhy

Pre pohyb v koronárnej rovine v prvotnej implementácií používali model Fourierovho radu. Táto implementácia zahŕňala sedem hlavných pohybov, špeciálne prechodové funkcie a hladký prechod medzi dvoma ľubovoľnými pohybmi. Tieto prechody pôsobili ale príliš veľké oneskorenie v reakciách agenta.

V novom návrhu implementovali statické a dynamické pohyby. Statické boli postavenie sa hráča alebo hľadanie lopty, dynamické pohyby sa generovali dynamicky podľa cieľa hráča. Dynamické plánovanie je založené na rôznych typoch chôdze, otáčok a zarovnaní. Parametre optimalizujú pomocou evolučnej stratégie CMA (Covariance Matrix Adaptation).

3.1.8.3.2 Správanie

V stratégií tímu je vrstvený návrh výberu pohybov, správania a plánovania (layered motion selection architecture). Táto hierarchia dosiahla vysokú modularitu a ľahkú údržbu. Tím vytvára pohyby a plánovanie na vyššej úrovni pomocou pohybov nižších úrovní. Príklad na schéme č.1.



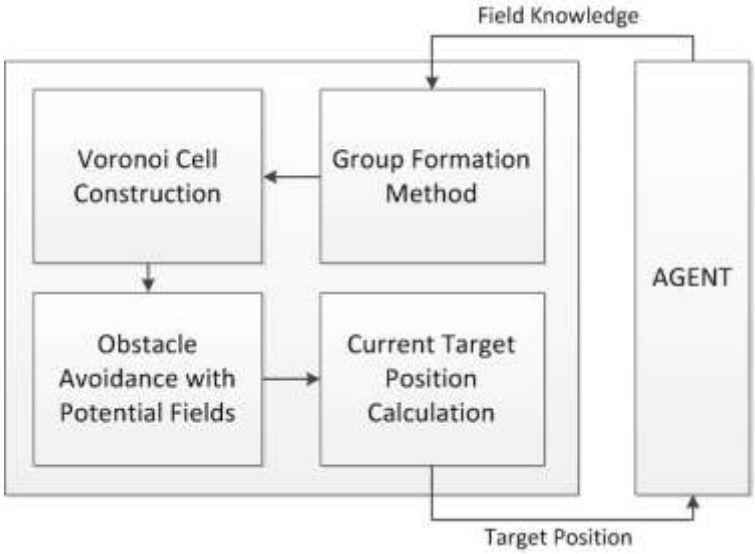
Obrázok 3.8: Pohyb dribble-to-goal

3.1.8.3.3 Tímová stratégia a plánovanie

Tímová stratégia zahŕňa štyri po sebe idúce procesy pre určenie cieľa hráča. Na začiatku sa formulujú dve skupiny, útočníci a obranári.

Majú tri rôzne plánovače pre štyri roly. Forward plánovač vedie loptu bez ohľadu na spoluhráčov a vyberá medzi akciami vedenia lopty ďalej alebo kopania na bránu. Plánovač brankára sa snaží tvoriť prekážku pre protihráčov. Finálny plánovač je používaný aj strednými aj útočiacimi hráčmi, rozdielne je iba kalkulovanie na základe cieľa. Plánovače sú aktivované na základe pridelenia roly pre hráča.

Hlavný modul pre tímovú stratégiu je na obrázku č.2.

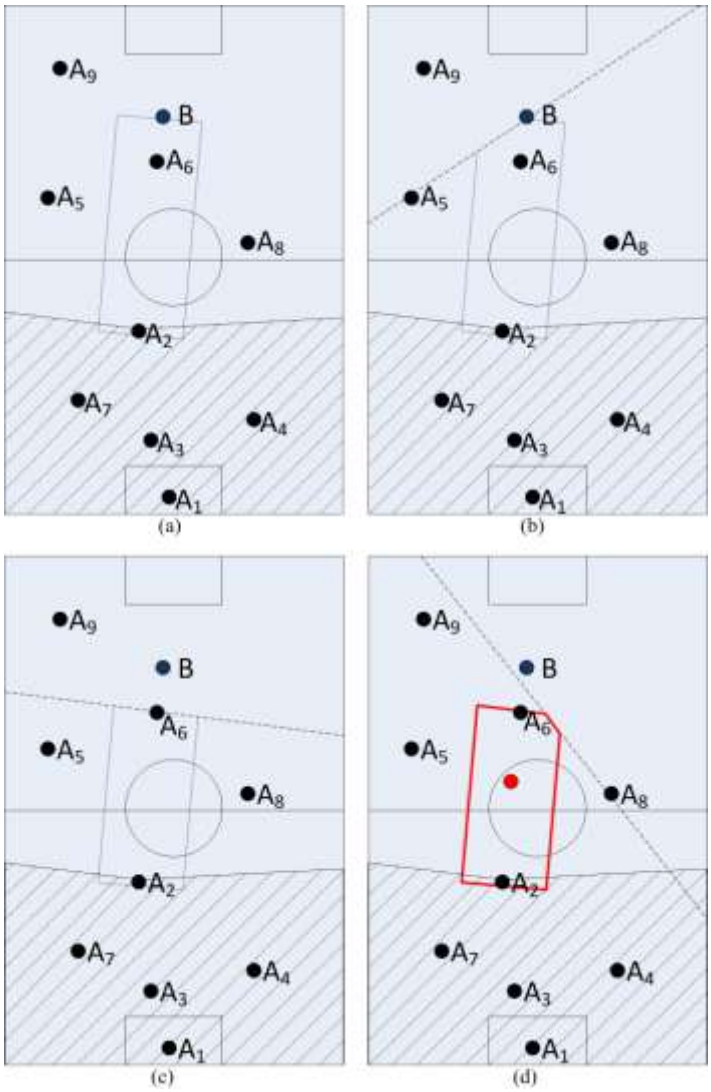


Obrázok č. 1

Počet a pomer brániacich a stredných hráčov je určený pomocou skupinového formulovania na základe stavu hry.

Obrancovia a stredné hráči sa polohujú tak, aby útočníkov mohli udržiavať čím viac vpredu. Na toto používajú výpočet svojich teritórií, ktoré si chránia. Tieto teritória hráčov sa môžu aj prekrývať.

Stredoví hráči si rátajú svoje teritórium na základe polohy lopty a spoluhráčov na ktorých vidia. Obrázok č.3. znázorňuje modifikovanie základného teritória stredného hráča na základe jeho spoluhráčov.



Obrázok č. 2

Obrázok č. 3 znázorňuje zúženie teritória hráča A2 na základe priesečníkov cez jeho spoluhráčov (A5, A6, A8) a jeho základného teritória.

Svoje teritórium si určia aj brániaci hráči podobne. Oni si ale vytvárajú svoje základné teritórium na základe stredového bodu na línie medzi loptou a stredom ihriska.

Po vykonaní týchto výpočtov sa hráči posunú na stred svojich teritórií.

3.1.8.3.4 Zdroje

- B. Demirdelen, B. Toku, O. Ulusoy, T. Sonmez, K. Ayvaz, E. Senyurek, and S. Sariel-Talay , beeStanbul RoboCup 3D Simulation League Team Description Paper 2012, url: http://air.cs.itu.edu.tr/publications-1/beeStanbul_TDP2012.pdf

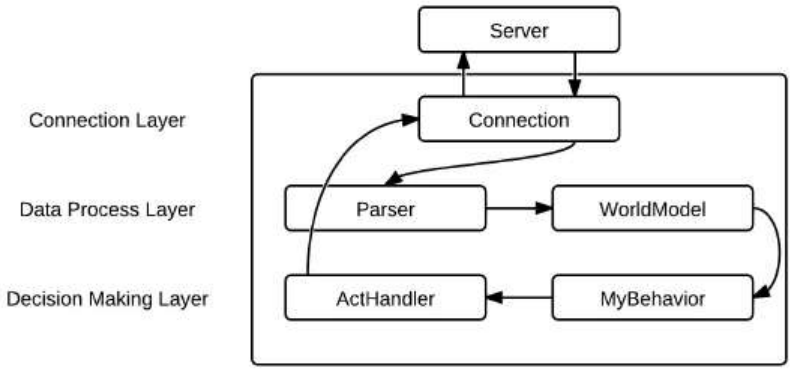
3.1.8.4 Analýza tímu Nexus3D

Tento tím patrí medzi menej úspešné v rámci medzinárodných súťaží RoboCup 3D. Pochádzajú z iránskej Ferdowsi University of Mashad. Začínali so simuláciou 2D futbalu a od roku 2004 sa snažia uspieť v poli 3D.

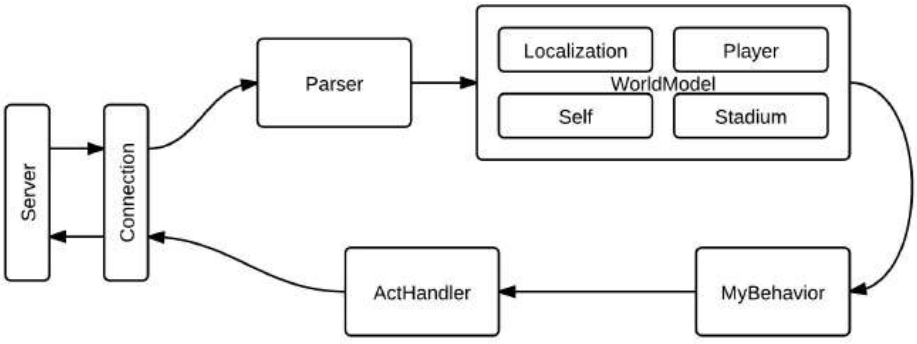
3.1.8.4.1 Architektúra

3 vrstvy:

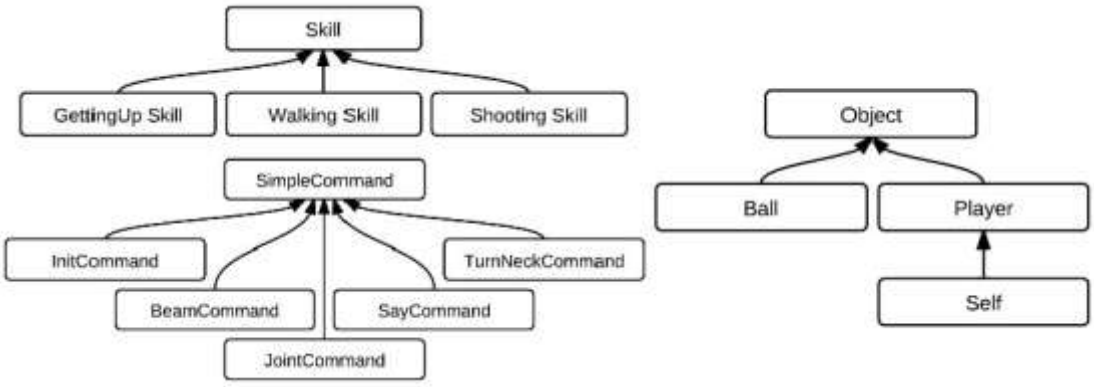
- Communication – cez perceptory a effectory
- Data - zparsované dáta sú posunuté do WorldModel (informácie o prostredí), ktorý sa delí na:
    - Localization - pozície
    - Stadium – všeobecné info o zápase
    - Aget
    - Other players
- Decision making – rozhodovanie podla informácií vo WorldModel, výsledok je Behavior pozostávajúci zo SoccerCommands



Obr.1: Vrstvy

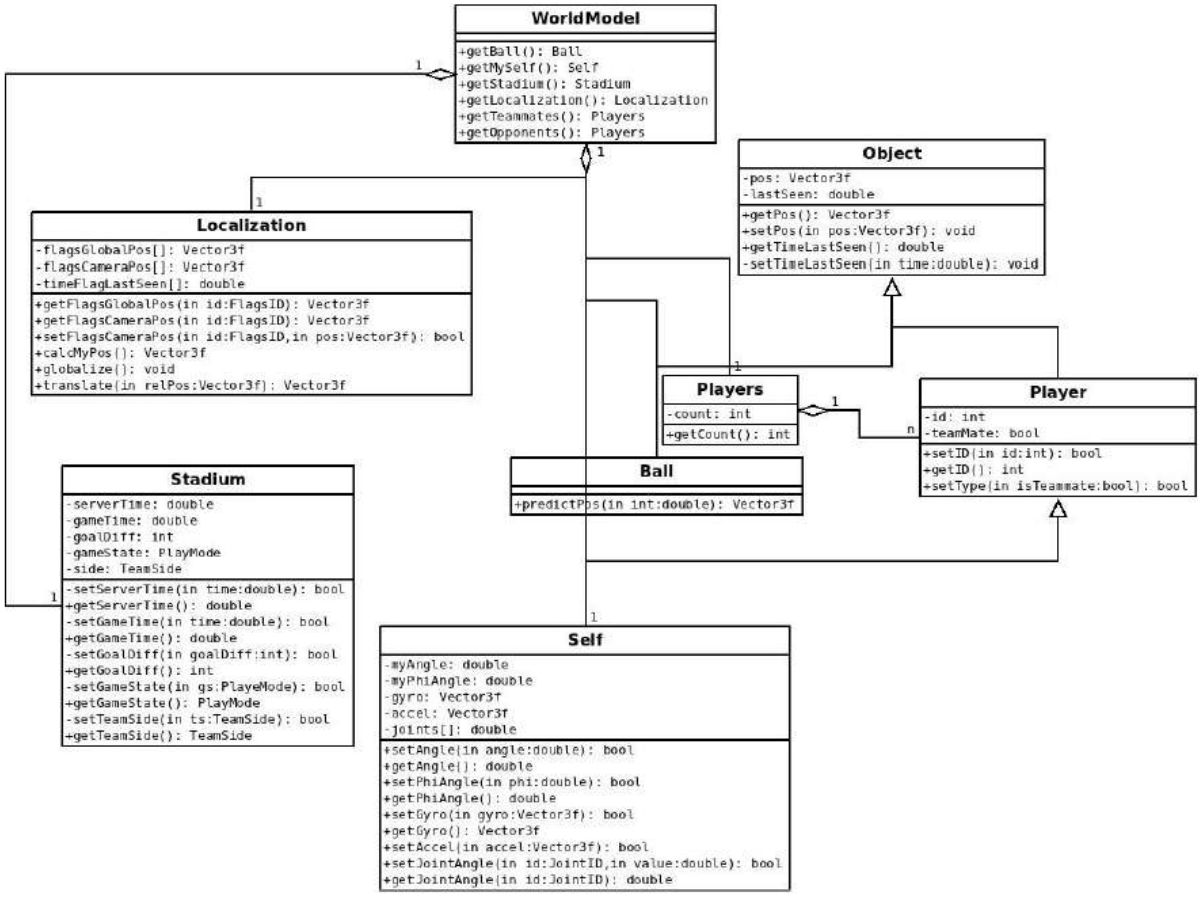


Obr.2: Data-flow diagram kodu



Obr.3: Dedenie niektorých tried

3.1.8.4.2 WorldModel



Obr.4: Štruktúra modelu sveta

3.1.8.4.3 Lokalizácia

- relatívna pozícia (relatívna k hráčovi) a globálna pozícia (relatívna k stredu ihriska)
- viditeľnosť dvoch vlajok postačuje na vytvorenie jednoduchej mapky ihriska s pozíciou hráča

3.1.8.4.4 Príkazy, schopnosti a správanie

- Command – SoccerCommand pozostáva zo SimpleCommand
- Skill – sada príkazov ústiacich do akcie (chôdza,...):
  - Prepare – vykonanie príkazov k príprave hráča na vykonanie podstaty skillu
  - Do – vykonanie príkazov podstaty skillu
  - Stop – vykonanie príkazov ktoré ukončia vykonávanie skillu (po skončení skillu alebo počas neho)
- ActHandler – vykonáva skill, ak má vykonať skill a iný sa už vykonáva, tak ten aktuálny stopne a až potom vykoná ďalší
- Behavior – sada skillov

3.1.8.4.5 Zdroje

- http://nexus.um.ac.ir/sources/NexusEN.pdf

3.1.8.5 Analýza tímu magmaOffenburg

Tento tím začal s 2D simuláciou ligy, kde ešte vystupoval pod menom magmaFreiburg. Najväčší úspech mal v roku 1999 kde na svetovom šampionáte obsadil 2. miesto. V roku 2009 prešiel k 3D simulácií. V tom roku sa aj premenoval na tím magmaOffenburg. Pod týmto menom vystupuje až dodnes. Tím magmaOffenburg je tímom Hochschule Offenburg University of Applied Sciences.

Na tvorbu správania robota majú vytvorený rámec, ktorý sa skladá z troch časti. Každý pohyb je rozdelený do pohybových fáz, ktoré pozostávajú z jednotlivých pohybov kĺbov. Medzi špeciálne funkcie patri automaticky prevod pohybu pravej časti tela do pohybu ľavej časti tela. Každý pohyb kĺbu ma definovanú rýchlosť, ktorou sa nemusí nutne pohybovať až do konca pohybovať fázy.

Keďže tím ma vytvorený dynamicky model robota, na jeho primerané pohyby použili inverzne kinematické výpočty. Na tento účel každá časť tela, dynamicky poskytuje metódu pre vytvorenie Jacobiho matice od reťazca časti tela, po koreň časti tela. Časť tela inštancie poskytuje niečo potrebné na vykonanie typického základného cyklu inverznej kinematickej metódy medzi žiadajúcou Jacobiho maticou a vykonávaným delta pohybom na príslušný kĺb. Takýmto spôsobom rôzne kinematické metódy, môžu byť použite na optimalizáciu pozície/orientácie konkrétnych časti tela, inštancia modelu tela reprezentuje ľubovoľný fyzicky model robota k cieľovej pozícii/orientácii v priestore. Aktuálne úsilie sa vynakladá na vyšetrenie rôznych kinematických metód a ich použiteľnosť v humanoidných robotoch. Skutočná sila inverznej kinematickej metódy spočíva z odobratia fyzikálnych vlastnosti z robota. Tak môžu cieľovú pozíciu a orientáciu vypočítať dynamicky pomocou ďalšieho senzora informácii a tak odovzdať tieto informácie inverznému kinetickému rámcu, ktorý adekvátne presunie robota.

3.1.8.5.1 Zdroje

- http://robocup.hs-offenburg.de/html/index.htm

3.1.8.6 Analýza tímu Austin Villa

Tím z Texaskej univerzity, ktorý sa sútredí na trening, hranie, všesmerovú chôdzu, rozpoznávanie a učenie sa farieb atď. Svojich hráčov rozdelujú na dve skupiny a to trénera a hráčov.

Tréner – pozoruje hru protivníka, učí sa a dokáže vyprodukovať strategiu.

Hráč – akceptuje trénerove taktiky a vykoná ich.

Majú aj dve stratégie. Prvá je taká, že jeden tím sa bráni a snaží si udržať loptu a druhý sa snaží ju zobrať a dať gól. Druhá stratégia je, že sa tímy navzájom snažia prekonať súperovu obranu a dať gól.

V Nao lige definujú 4 hlavné problémy videnie, lokalizácia, pohyb a koordinácia. Tieto problémy popisujú a snažia sa riešiť. Vychádzajú z obohacovaného učenia sa a predpokladajú, že robot by mal byť schopný sa viac naučiť zo skúseností z reálneho sveta ako z naprogramovania jeho vedomostí. Obohacované učenie ešte obohacujú o rozhodovacie stromy. Agent sa pomocou ich prístupu rýchlejšie učí ako pri bežných voľných učiacich modeloch.

Algoritmus RL-DT (reinforcement learning and decision tree):

RL-DT(RMax, S)
1: A Set of Actions
2: S Set of States
3: 8a 2 A : visits(s, a) 0
4: **loop**
5: a argmax$_{a' \in A}$Q(s, a′)
6: Execute a, obtain reward r, observe state s′
7: Increment visits(s, a)
8: (PM,RM, CH) UPDATE-MODEL(s, a, r, s′ , S,A)
9: exp CHECK-POLICY(PM,RM)
10: **if** CH **then**
11: COMPUTE-VALUES(RMax,PM,RM, SM,A, exp)
12: **end if**
13: s s′
14: **end loop**

Prišli s novým prístupom k lokalizácii hráča na ihrisku. Využívajú pri tom Microsoft Kinect RGB-D Sensor, čo je lacné prenosné a rýchle riešenie. Používajú to hlavne na zistenie polohy lopty a robota.

Systém je pre nich ľahko nastaviteľný a nevyžaduje nové senzory. Jednou z hlavných úloh je transformácia pointcodového výstupu z kinect senzora do mapy ihriska. Ďalším kľúčovým doplnkom je rozpoznanie farieb na ihrisku. Následná identifikácia, ktorým smerom sa má robot vybrať

vychádza zo získaných dát, ale aj z detekcie objektov na ihrisku. Robot musí vedieť identifikovať druhého robota a loptu. Napríklad pri detekcií lopty sa snažia zachytiť skupinu oranžových bodov v určitej vzdialenosti od seba. Pri testoch dokázal identifikovať druhého robota na skoro 96%.

Zaujímavosťou na záver je aj ich oblasť záujmu v štvornohej lige s robotom Sony Aibo. Štvornohí roboti v podobe psov sú obratnejší a majú odlišnú škálu pohybov. Vďaka ich konštrukcií sa nemusí riešiť koordinácia.

3.1.9 Analyzovanie robotov (pp 1.9)

3.1.9.1 Analýza robota ASIMO

ASIMO = Advanced Step in Innovative Mobility od Honda Motor Company. Je to prvý humanoidný robot, ktorý vie nezávisle chodiť a kráčať po schodoch. Dokáže rozumieť predprogramovaným gestám, hovoreným príkazom, rozoznávať hlasy a tváre. Je to robot pomocník.

Ľudia z Hondy študovali pohyby, hmyzu, ľudí, horolezca s protézami, aby lepšie pochopili fyziológii chôdze. Sledovali prenášanie váhy pri chôdzi, špeciálne pohybmi rúk. Všimli si úlohu palca na nohe pri zachovávaní rovnováhy.

Asimo má bočné, kolenné a chodidlové kĺby. Výskumníci nazvali kĺby stupňami slobody. Asimo má 34 stupňov slobody po celom tele.

Má rýchlostný senzor a gyroskopový senzor, ktoré:

- zaznamenávajú polohu a rýchlosť tela
- prenášajú prispôsobenie rovnováhy do centrálneho počítača

Tieto dva senzory pracujú podobne ako naše vnútorné ucho pre udržiavanie stability a orientácie. Asimo má tiež aj podlahové povrchové senzory v nohách a šesť ultrasonických senzorov v strede tela. Slúžia na detekovanie okolitých objektov a ich porovnávanie s dátami máp uloženými v pamäti.

Asimo má senzory zisťujúce natočenie kĺbov a osové silové senzory vnímajúce tlak vykonávaný na predmety.

Vďaka svojim otáčacím schopnostiam nemusí počas presúvania zastavovať a meniť smer alebo robí to počas toho. Na dosiahnutie takejto vyspelej chôdze skúmali vedci inerciálne sily počas chôdze. Napríklad sila gravitácie, sila rýchlosti chôdze, sila chodidla pôsobiaca na zem a pod. Pomocou ZMP „zero moment point“ sa tieto sily vyrovnávajú. Asimov postoj je ovládaný troma ovládaniami:

- floor reaction control – vyrovnáva sa s chybami povrchu
- target ZMP control – vyrovnáva časti tela, aby robot nepadol
- foot-planting location control – ovláda dĺžku krokov vzhľadom na pozíciu a rýchlosť tela

Asimo nielen predchádza pádu, ale vďaka jeho hladkému tempu sa vie otáčať bez zastavovania. Používa technológiu „predictive movement control“, ktorá zabezpečuje ako ďaleko do zatáčky sa

posunie ťažisko a ako dlho tam bude posunuté. Všetko toto sa deje v reálnom čase. Počas chôdze si nastavuje nasledovné:

- dĺžku krokov
- pozíciu tela
- rýchlosť
- smer krokov

Vďaka tomuto dokáže bežať rýchlosťou až 6 km/h. Počas behu sa dokáže ocitnúť vo vzduchu až 0,08 sekundy. Aby počas toho nestratil kontrolu, nestočil sa vo vzduchu alebo nepadol pri dopade, má Asimo v torze jeden stupeň slobody.

Asimo má v hlave dve kamery na mieste očí a používa stereoskopický pohľad.

3.1.9.1.1 Zdroje

- http://science.howstuffworks.com/asimo1.htm

3.1.9.2 Analýza pohybov

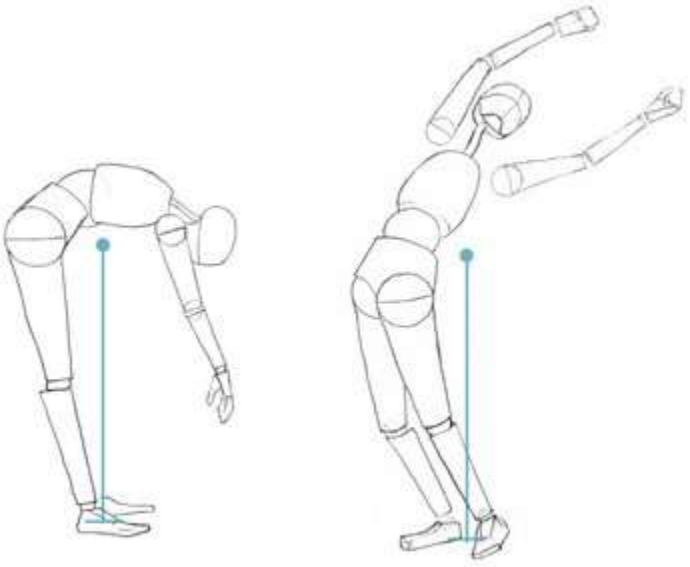
Najbežnejšie metódy pohybu, ktoré môžu byť rozdelené do 4 kategorii:

1. trajectory-based approaches (prístupy založené na trajektórii)

   Pozostáva z nájdenia kinematických trajektórií a použitia stabilizačných kritérií na zaistenie stabilnej chôdze. Stabilizačné kritéria :

   - Center of Mass – ťažisko systému častíc je bod v ktorom sa hmotnosť systému sprava ako keby bol koncentrovaný. Inak povedané ťažisko je definovane ako umiestnenie váženého priemeru jednotlivých hmotných bodov systému –

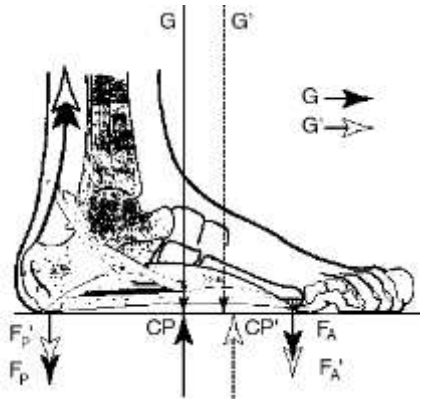
$$p_{CoM} = \frac{\sum_i m_i p_i}{M}$$



Kde, M = suma m_i, celková hmotnosť systému, m_i – množstvo itej častice, p_i - ťažisko

- Center of pressure – centrum tlaku – veľa humanoidov je vybavených senzorom krútiaceho momentu na jeho nohách. Centrum tlaku je výsledkom vyhodnotenia týchto senzorov a je definovaný ako bod na zemi kde pôsobí vyplývajúca reakcia pozemných síl

-



- Zero moment point – bod na zemi v ktorom mement všetkých aktívnych síl je nulový

2. virtual model control (virtuálny model riadenia)

Virtual Model Control (VMC) je rámec založený na heuristike, ktorý používa virtuálne komponenty, ako sú klapky pružiny, alebo hmotnosť ku generovanie spoločných momentov ktoré kontrolujú stabilitu a rýchlosť robotov. Vytvorene komponenty majú rovnaký efekt ako keby boli skutočné.

3. passive-dynamic walking (pasívna-dynamická chôdza)
    - je založená na obrátenom kyvadle (inverted pendlum)
4. central pattern generators (centrálne vzorové generátory)

3.1.9.2.1 Ďalšie algoritmy:

- Genetic algoritmus
- Inverted pendlum(obratene kyvadlo)
- Hill climbing
- Cross-entropy method
- Covariance matrix adaptation evolution strategy

3.1.9.2.2 Genetic algoritmus

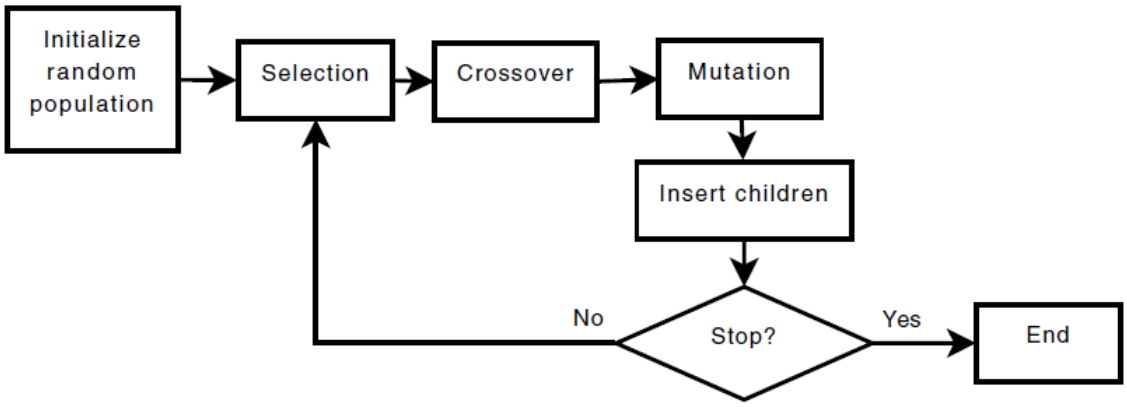


Schéma pre geneticky algoritmus

Selection – vyber rodičov pre kríženie podľa preddefinovaných pravidiel(cena funkcie/kondície)

Crossover – generovanie potomkov rodičov, výmenou niektorých génov pomocou niektorých schém: one-point, two-point, uniform(rovnomerne?). Potomkovia tak dedia niektoré vlastnosti od každého rodiča.

Mutation – generuje potomka tak, že náhodne mení jeden alebo niekoľko génov. To umožňuje hľadanie nových oblasti riešení, ktoré by neboli inak preskúmané. Mutácia preto zabraňuje GA

sústrediť sa len na lokálne vyhľadávanie, ktoré zase zvyšuje pravdepodobnosť nájdenia globálneho maxima.

Population ← *CreateInitialPopulation*()
Evaluate(*Population*)
while TerminationConditionNotMet() **do**
[Selection] *Parents* ← Selection(Population)
[Elistism] *Elite* ← Elitism(Population)
[Crossover] *Children* ← Crossover(*Parents*, *pc*)
[Mutation] *Mutants* ← Mutation(*Children*, *pm*)
Population ← *Elite* +*Mutants*
Evaluate(*Population*)
end while
return Best(*Population*)

3.1.9.2.3 Zdroje

- http://sabia.tic.udc.es/articulos/2009/BipedWalking.pdf

3.1.9.2.4 Cyklus chodenia

Majme robota s trupom a nohami. Každá noha pozostáva zo stehna, predkolenia, nohy. Majú 6 stupňov voľnosti. Tri v bedrovom kĺbe, jedna v kolene a dva v členku. Chodenie pozostáva z dvoch fáz. Buď sú obidve nohy na zemi alebo iba jedna, ktorá slúži ako opora. Počas fázy keď sú obide nohy na zemi sa postupne prenáša ťažisko zo zadnej nohy na prednú nohu. Ak je táto fáza pomalá, je náročné vytvoriť rýchlu chôdzu. Treba vedieť stanoviť správny čas na túto fázu.

Keď trajektórie pohybu bedrových kĺbov a nôh sú známe, vieme odvodiť uhly kĺbov pomocou kinematických obmedzení. Robot sa pohybuje rovno, polohy chodidiel bývajú konštante. Chôdza bude pozorovaná z boku.

Trajektória nohy popísaná $X_a = [x_a(t), z_a(t), \Theta_a(t)]^T$ kde $(x_a(t), z_a(t))$ sú súradnice členku a $\Theta_a(t)$ je uhol členku. Trajektória kĺbu popísaná vektorom $X_h = [x_h(t), z_h(t), \Theta_h(t)]^T$ kde $(x_h(t), z_h(t))$ sú súradnice bedrového kĺbu a $\Theta_h(t)$ je uhol bedrového kĺbu(Fig. 1).

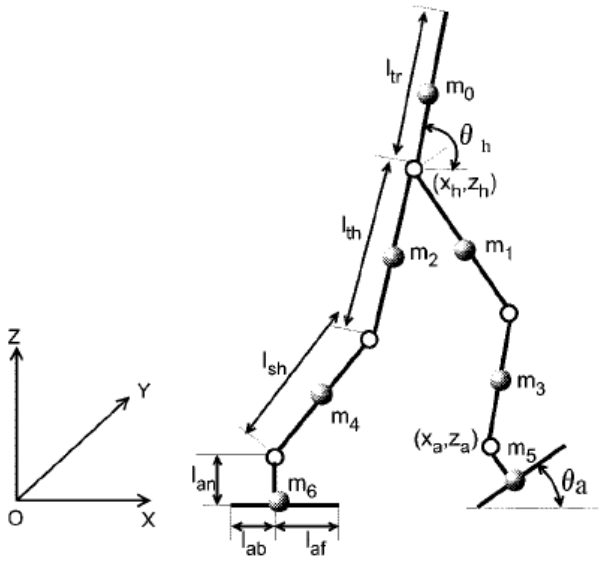


Fig. 1. Model of the biped robot.

3.1.9.2.5 Trajektória nôh

Časová perióda pre jeden cyklus chodenie je T_c. Je to čas jednej periódy cyklu chodenia z kT_c do $(k+1)T_c$. Kde k=1,2,...K, K je počet krokov. Na začiatku periódy sa päta pravej nohy dotýka zeme a odchádza, vtedy je v čase kT_c a perióda končí opäť keď sa pravá noha blíži k zemi a dotýka sa jej v čase $(k+1)T_c$, (Fig.2).

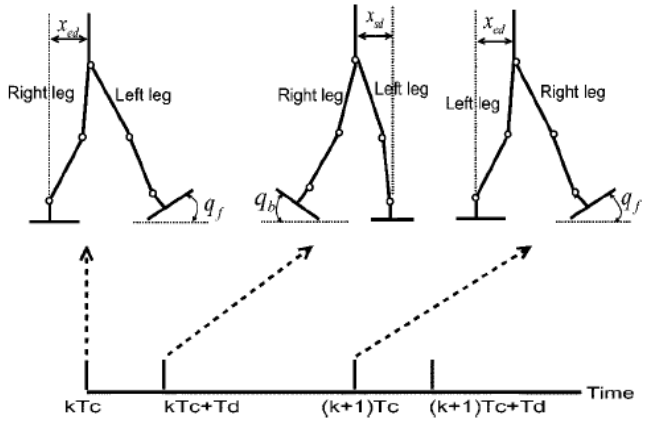


Fig. 2. Walking cycle.

Ďalej je popísané generovanie trajektórie pre pravú nohu. Trajektória ľavej nohy je rovnaká okrem trvania T_c.

Chôdza, pri ktorej sa robot dotýka celou nohou zeme hneď pri prvom kontakte nie je vhodná pre rýchlu chôdzu. Nech q_b a q_f sú uhly pravej nohy prichádzajúcej a odchádzajúcej na/zo zeme(fig. 2).

Predpokladajme, že pravá noha sa celá dotýka zeme v čase $t=kT_c$ a $t = (k+1)T_c + T_d$. Z toho dostávame

$$\theta_{\mathrm{a}}(t) = \begin{cases} q_{\mathrm{gs}}(k), & t = kT_{\mathrm{c}} \\ q_{\mathrm{b}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ -q_{\mathrm{f}}, & t = (k+1)T_{\mathrm{c}} \\ -q_{\mathrm{ge}}(k), & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (1)$$

- T_d je interval s oboma nohami na zemi
- $q_{gs}(k)$ a $q_{ge}(k)$ sú uhly medzi plochou a nohami

Na nerovnom teréne s prekážkami očakávame aby robot vedel prekážky prekonať. Nech (L_{ao}, H_{ao}) je pozícia kde sa noha vo vzduchu nachádza najvyššie(fig.3).

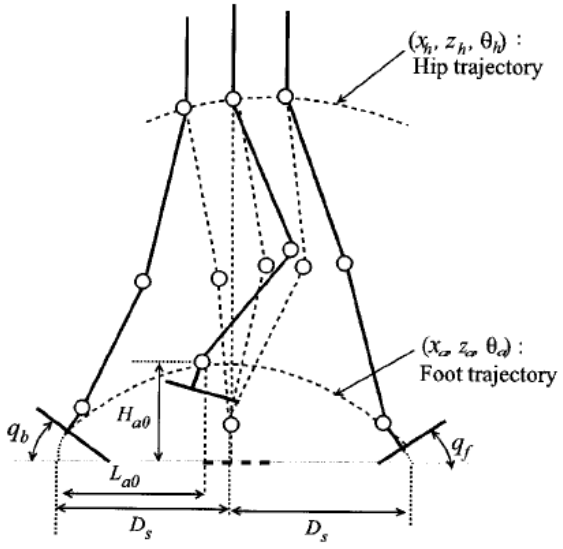


Fig. 3. Walking parameters.

$$x_{\mathrm{a}}(t) = \begin{cases} kD_{\mathrm{s}}, & t = kT_{\mathrm{c}} \\ kD_{\mathrm{s}} + l_{\mathrm{an}} \sin q_{\mathrm{b}} + l_{\mathrm{af}}(1 - \cos q_{\mathrm{b}}), & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ kD_{\mathrm{s}} + L_{\mathrm{ao}}, & t = kT_{\mathrm{c}} + T_{\mathrm{m}} \\ (k+2)D_{\mathrm{s}} - l_{\mathrm{an}} \sin q_{\mathrm{f}} - l_{\mathrm{ab}}(1 - \cos q_{\mathrm{f}}), & t = (k+1)T_{\mathrm{c}} \\ (k+2)D_{\mathrm{s}}, & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (2)$$

$$z_{\mathrm{a}}(t) = \begin{cases} h_{\mathrm{gs}}(k) + l_{\mathrm{an}}, & t = kT_{\mathrm{c}} \\ h_{\mathrm{gs}}(k) + l_{\mathrm{af}} \sin q_{\mathrm{b}} + l_{\mathrm{an}} \cos q_{\mathrm{b}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ H_{\mathrm{ao}}, & t = kT_{\mathrm{c}} + T_{\mathrm{m}} \\ h_{\mathrm{ge}}(k) + l_{\mathrm{ab}} \sin q_{\mathrm{f}} + l_{\mathrm{an}} \cos q_{\mathrm{f}}, & t = (k+1)T_{\mathrm{c}} \\ h_{\mathrm{ge}}(k) + l_{\mathrm{an}}, & t = (k+1)T_{\mathrm{c}} + T_{\mathrm{d}} \end{cases} \quad (3)$$

- D_s je dĺžka kroku
- $kT_c + T_m$ ječas kedy je noha v najvyššej pozícii
- l_{an} je výška nohy k členku (fig.1)
- l_{af} je dĺžka z členku k prstom na nohách (fig. 1)
- l_{ab} je dĺžka z členku k päte (fig.1)
- $h_{gs}(k)$ a $h_{ge}(k)$ sú dĺžky povrchu, na ktorých stojí noha

Keď celý povrch pravej nohy je na zemi v čase $t=kT_c$ a $t=(k+1)T_c + T_d$ tak platia nasledujúce obmedzenia:

$$\begin{cases} \dot{\theta}_a(kT_c) = 0 \\ \dot{\theta}_a((k+1)T_c + T_d) = 0 \end{cases} \quad (4)$$

$$\begin{cases} \dot{x}_a(kT_c) = 0 \\ \dot{x}_a((k+1)T_c + T_d) = 0 \end{cases} \quad (5)$$

$$\begin{cases} \dot{z}_a(kT_c) = 0 \\ \dot{z}_a((k+1)T_c + T_d) = 0. \end{cases} \quad (6)$$

Na generovanie hladkej trajektórie je potrebné aby

- prvá derivácia(rýchlosť) $x_a(t)$, $z_a(t)$ a $\Theta_a(t)$ boli diferencovateľné
- druhá derivácia(zrýchlenie) $x_a(t)$, $z_a(t)$ a $\Theta_a(t)$ boli spojité na t, vrátane takých t, ktoré sú rovné kT_c, $kT_c + kT_d$, $kT_c + T_m$, $(k + 1)T_c$, $(k + 1)T_c + T_d$

Splniť obmedzenia (1) až (6) bude príliš drahé a výpočet pomocou polynomiálnej interpolácie je zložitý. Trajektóriu nôh získame z 3rd-order spline interpolácie. Druhé derivácie potom vždy budú spojité. Nastavením $q_{gs}(k)$, $g_{ge}(k)$, $h_{gs}(k)$, $h_{ge}(k)$, q_b, q_f, H_{ao}, L_{ao} sa dajú vytvoriť rôzne trajektórie nôh.

3.1.9.2.6 Trajektória bedrového kĺbu

Ak nie je žiadny driekový kĺb je želateľné aby $\Theta_h(t)$ bol konštantný. $\Theta_h(t)=0.5$ Π rad. Pohyb $z_h(t)$ ovplyvňuje ZMP(zero moment point). Predpokladajme, že kĺb je v najvyššej pozícii $H_{h\ max}$v strede fázy kedy je jedna noha na zemi a v najnižšej pozícii $H_{h\ min}$ vo fáze s dvoma nohami na zemi. Potom

$$z_h(t) = \begin{cases} H_{h\,min}, & t = kT_c + 0.5T_d \\ H_{h\,max}, & t = kT_c + 0.5(T_c - T_d) \\ H_{h\,min}, & t = (k+1)T_c + 0.5T_d. \end{cases} \quad (7)$$

Trajektória $z_h(t)$ splňujúca (7) a spojitosť po druhej derivácii sa získa z 3rd-order spline interpolácie.

Zmena $x_h(t)$ je hlavný faktor ovplyvňujúci stabilitu dvojnohého robota. Existujú metódy na odvodenie trajektórie bedrového kĺbu na vykonanie ZMP. Nie všetky trajektórie sú dosiahnuteľné a akcelerácia kĺbu musí byť veľká. Na vyriešenie konfliktov boli navrhnuté opatrenia

- generovanie série hladkých pohybov $x_h(t)$
- stanoviť hranicu $x_h(t)$

Proces chodenia pozostáva z troch fáz.

- Štartovacia fáza, v ktorej rýchlosť sa mení z 0 na želanú konštantnú rýchlosť.
- Ustálená fáza s želanou konštantnou rýchlosťou
- Koncová fáza, v ktorej rýchlosť sa mení z konštantnej rýchlosti na 0.

$X_h(t)$ ustálenej fázy je získaná z procedúry. $X_h(t)$ sa dá vyjadriť dvoma spôsobmi. Počas fázy s jednou nohou na zemi a fáza s dvoma nohami na zemi. X_{sd} a x_{ed} značia vzdialenosti k osi x z bedrového kĺbu k členku opornej nohy(fig.2).

$$x_{\mathrm{h}}(t) = \begin{cases} kD_{\mathrm{s}} + x_{\mathrm{ed}}, & t = kT_{\mathrm{c}} \\ (k+1)D_{\mathrm{s}} - x_{\mathrm{sd}}, & t = kT_{\mathrm{c}} + T_{\mathrm{d}} \\ (k+1)D_{\mathrm{s}} + x_{\mathrm{ed}}, & t = (k+1)T_{\mathrm{c}}. \end{cases} \tag{8}$$

Na získanie hladkého pohybu $x_h(t)$ v ustálenej fáze musia platiť obmedzenia s deriváciami.

$$\begin{cases} \dot{x}_{\mathrm{h}}(kT_{\mathrm{c}}) = \dot{x}_{\mathrm{h}}(kT_{\mathrm{c}} + T_{\mathrm{c}}) \\ \ddot{x}_{\mathrm{h}}(kT_{\mathrm{c}}) = \ddot{x}_{\mathrm{h}}(kT_{\mathrm{c}} + T_{\mathrm{c}}). \end{cases} \tag{9}$$

$$x_{\mathrm{h}}(t) = \begin{cases} kD_{\mathrm{s}} + \dfrac{D_{\mathrm{s}} - x_{\mathrm{ed}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}^2(T_{\mathrm{c}} - T_{\mathrm{d}})}\big[(T_{\mathrm{d}} + kT_{\mathrm{c}} - t)^3 - (t - kT_{\mathrm{c}})^3 \\ \quad -T_{\mathrm{d}}^2(T_{\mathrm{d}} + kT_{\mathrm{c}} - t) + T_{\mathrm{d}}^2(t - kT_{\mathrm{c}})\big] + \dfrac{x_{\mathrm{ed}}}{T_{\mathrm{d}}}(T_{\mathrm{d}} + kT_{\mathrm{c}} - t) + \dfrac{D_{\mathrm{s}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}}(t - kT_{\mathrm{c}}), & t \in (kT_{\mathrm{c}}, kT_{\mathrm{c}} + T_{\mathrm{d}}) \\ kD_{\mathrm{s}} + \dfrac{D_{\mathrm{s}} - x_{\mathrm{ed}} - x_{\mathrm{sd}}}{T_{\mathrm{d}}(T_{\mathrm{c}} - T_{\mathrm{d}})^2}\big[(t - kT_{\mathrm{c}} - T_{\mathrm{d}})^3 - (T_{\mathrm{c}} + kT_{\mathrm{c}} - t)^3 - (T_{\mathrm{c}} - T_{\mathrm{d}})^2(T_{\mathrm{c}} + kT_{\mathrm{c}} - t) \\ \quad -(T_{\mathrm{c}} - T_{\mathrm{d}})^2(t - kT_{\mathrm{c}} - T_{\mathrm{d}})\big] + \dfrac{D_{\mathrm{s}} - x_{\mathrm{sd}}}{T_{\mathrm{c}} - T_{\mathrm{d}}}(T_{\mathrm{c}} + kT_{\mathrm{c}} - t) + \dfrac{D_{\mathrm{s}} + x_{\mathrm{ed}}}{T_{\mathrm{c}} - T_{\mathrm{d}}}(t - kT_{\mathrm{c}} - T_{\mathrm{d}}), & t \in (kT_{\mathrm{c}} + T_{\mathrm{d}}, kT_{\mathrm{c}} + T_{\mathrm{c}}). \end{cases} \tag{10}$$

Pomocou 3rd order periodic spline interpolácie získame $x_h(t)$ splňajúce (8), (9) a podmienku spojitosti druhej derivácie danej (10). Určením rôznych x_{sd} a x_{ed} dostávame rôzne plynulé trajektórie $x_h(t)$ podľa (10).

$$\begin{cases} 0.0 < x_{\mathrm{sd}} < 0.5D_{\mathrm{s}} \\ 0.0 < x_{\mathrm{ed}} 0.5D_{\mathrm{s}}. \end{cases} \tag{11}$$

Na základe (10), (11), (13), (14) je odvodená hranica stability

$$\max_{x_{\mathrm{ed}} \in (0, 0.5D_{\mathrm{s}}), x_{\mathrm{sd}} \in (0, 0.5D_{\mathrm{s}})} d_{\mathrm{zmp}}(x_{\mathrm{sd}}, x_{\mathrm{ed}}) \tag{12}$$

$d_{zmp}(x_{sd}, x_{ed})$ určujú hranice stability. Sú iba dva parametre takže ľahko získame riešenie (12) pomocou vyčerpávajúceho hľadania(fig.4).

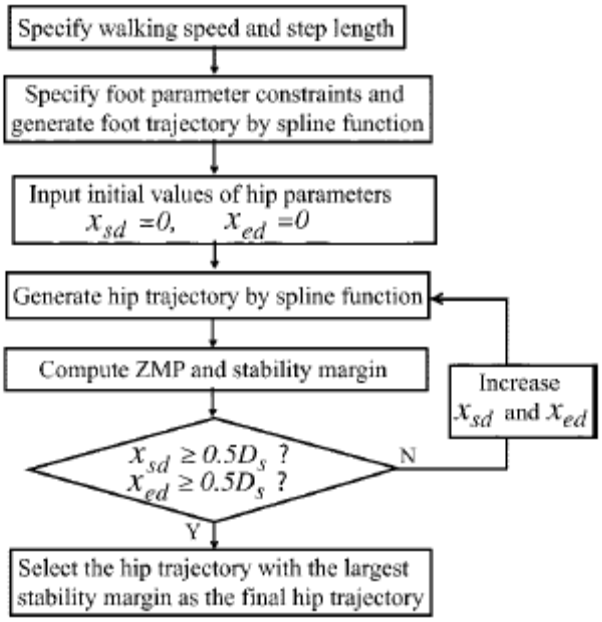


Fig. 4. Algorithm for planning walking patterns.

Určením $x_h(t)$ v ustálenej fáze sa špecifikujú koncové obmedzenia pre štartovaciu fázu a počiatočné obmedzenia pre koncovú fázu. Počiatočné obmedzenie pre štartovaciu fázu je aby derivácie $x_h(t_0)=0$ a koncové obmedzenia pre finálnu fázu $x_h(t_c)=0$. $X_h(t)$ štartovacej a koncovej fázy sa dajú získať 3rd order spline interpoláciou.

$$x_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)x_i - \sum_{i=1}^{n} m_i \ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\ddot{\Omega}_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{13}$$

$$y_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)y_i - \sum_{i=1}^{n} m_i \ddot{y}_i z_i - \sum_{i=1}^{n} I_{ix}\ddot{\Omega}_{ix}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{14}$$

3.1.9.2.7 Zdroje

- http://staff.aist.go.jp/k.kaneko/publications/2001_publications/6.pdf

3.1.10 Git (pp 1.10, pp 1.11)

3.1.10.1 Analýza

Pre verziovanie softvérového produktu sme sa rozhodli použiť git hlavne pre výhodu necentralizovaného repozitára. Možnosti spoločného repozitára boli vytvorenie repozitára na našom serveri alebo použiť externú službu ako napríklad gitbus.fiit.stuba.sk.

3.1.10.2 Návrh

Náš tím použil externú slúžbu repozitára gitbus.fiit.stuba.sk. Kde sme vytvorili tím a k tímu sme vytvorili projekt, kde sme mali náš spoločný repozitár. Každý člen tímu si musel naištalovať na svojom stroji git, následne si vygenerovať rsa klúč pre autentifikáciu so spoločným git repozitárom, ktorý si uložili v službe gitbus.fiit.stuba.sk pod svojim kontom. Nakoniec si každý člen stiahol obraz repozitáru na svoj stroj a nastavil si vzdialený pôvodný repozitár na alias origin.

3.1.10.3 Implementácia

Pri implementácii každý člen tímu postupoval návodom inštalácie a použitia gitu pre tímový projekt, ktorý je v prílohe dokumentácie. S gitom každý člen postupoval podľa metodiky verziovacich systémov.

3.1.11 Manažérsky softvér (pp 1.12)

3.1.11.1 Analýza

Analýza prebehla porovnávaním rôznych manažérskych nástrojov, ktoré spĺňali základné kritériá pre náš projekt a pre distribuovanú tímovú prácu. Kritériá:

- Vzdialený prístup z webu
- Vytváranie backlogu
- Možnosť generovania Burndown chartu
- Možnosť generovania Ganttovho diagramu

3.1.11.1.1 Redmine

Redmine je voľne dostupný open source manažérsky nástroj. Jeho výhodou je ľahká rozširovateľnosť ďalšími modulmi. Obsahuje súborový manažment a manažment dokumentov. Ponúka e-mailovú notifikáciu. Je vhodná na agilný vývoj softvéru.

3.1.11.1.2 Jira

Je komerčný program na bug-tracking a vývojový manažment. Implementovaný je v Jave. Jej používanie je veľmi ľahké. Hodí sa na agilný vývoj softvéru. Obsahuje spárovanie zdrojových kódov s úlohami.

3.1.11.1.3 dotProject

Implementovaný je v PHP spolu s MySQL. Je opensource a free softvér. Podporuje použitie Ganttovho diagramu. Obsahuje generovanie reportov aj diskusné fórum.

3.1.11.2 Výstup analýzy

Redmine nám vyhovoal najviac zo všetkých možností. Okrem opísaných výhod obsahoval všetky žiadané funkcie. Jeho prístup cez školský server bol veľmi ľahký.

3.2 Šprint 2 – „beta“

Tabuľka 3.4: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
2.1	Člen tímu	Stanoviť celkovú prácu počas projektu	Vyberať úlohy, na ktorých sa bude pracovať
2.2	Člen tímu	Importovať a skrížiť hráčov minuloročných projektov	Pracovať už na funkčnom a spojenom projekte
2.3	Člen tímu	Porovnať a zanalyzovať zdrojové kódy minuloročných projektov	Zistiť v akom sú stave a čo je alebo nie je funkčné
2.4	Člen tímu	Vytvoriť dokumentáciu projektu	Celkovú prácu zaznamenať na jedno miesto a aby bola možná tlač
2.5	Člen tímu	Analyzovať chôdzu skríženého hráča	Aby bolo možné začať prvé zdokonaľovanie prípadne vyvíjanie
2.6	Člen tímu	Vytvoriť koding guide	Jednoznačne písať a komentovať zdrojový kód
2.7	Člen tímu	Vytvoriť metodiku na používanie Gitu	Bezproblémovo verziovať zdrojový kód

Tabuľka 3.5: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
2.1	-	2.1	Návrh celkovej práce počas projektu	M. Červeňák
2.2	-	2.2	Git – importovanie a kríženie hráčov	M. Ondrejkovič
2.3	-	2.3	Analýza z. kódu hráča tímov High5 a Tím 17 žije...	M. Gregor
2.4	-	2.4	Vytvoriť dokumentáciu aj so šablónou	M. Červeňák
2.5	-	2.5	Analýza chôdze nášho (kríženého) hráča	G. Nagy
2.6	-	2.6	Koding guide (šablóna-metodika)	F. Sucháč
2.7	-	2.7	Metodika na používanie Git-u	M. Ondrejkovič

3.2.1 Návrh celkovej práce počas projektu (pp 2.1)

3.2.1.1 Analýza

Celý predchádzajúci šprint sa zaoberal inštaláciou komponentov na spustenie projektu a analýzou problematiky. A práve tá analýza bola dôležitá pre návrh celkovej časti na projekte tohto projektu. Jednotlivý členovia tímu sa oboznámili s dokumentáciami a wikipediou z minulého roka a analyzovali tímy úspešné v celosvetovej súťaži RoboCup. Medzi analyzované časti bola zahrnutá aj analýza fyzických modelov práve kvôli pochopeniu stability a pod. Predbežná analýza v tejto úlohe vyústila do spísanie základného návrhu práce, ktorým by sa bolo možné uberať.

3.2.1.2 Návrh

Na základe analýzy, ktorou sa tím zaoberal boli navrhnuté rôzne nápady, ktoré by bolo možné realizovať. Jednotlivé nápady a návrhy boli spracované do tabuľky, ku ktorým bol pridaný detailný opis. Následne po komunikácii a dohode sa z navrhnutých nápadov zostavil zoznam používateľských príbehov, ktoré sa zoradením priorít budú postupne vykonávať. Takýto priorizovaný zoznam používateľských príbehov tvorí produktový backlog projektu. Ten slúži ako základný stavebný prvok pri vývoji metodikou Scrum, ktorou sa uberá aj tento projekt.

Výstupom tejto úlohy je spísanie návrhov jednotlivých členov tímu, ktoré sa nachádzajú v tabuľke 3.6. Pod tabuľkou sú detailné opisy niektorých úloh.

Tabuľka 3.6: Navrhnuté úlohy

Návrh	Popis	Navrhol
Optimalizácia vytvorených nízkoúrovňových pohybov	• dosiahnuť kvalitu vhodnú na použitie pri vytváraní vysokoúrovňových pohybov (stabilita, rýchlosť). • testovať ich, a to najmä pomocou gyroskopu a akcelometra, ktoré obsahuje testovací framework. • vylepšenie high skillu kráčania za loptou - aby hráč vedel loptu viesť • vylepšenie úkrokov - aby sa hráč nemusel dlho otáčať a natáčať • rýchlosť základných pohybov • kop do lopty	J. Gregor M. Červeňák G. Nagy F. Sucháč
Vytvorenie nových pohybov	• beh • otáčanie hráča počas chôdze resp. chôdza po kružniciach, aby nemusel zastavovať a natáčať sa. • viac pohybov pri bránení • ochrana lopty telom	J. Gregor M. Červeňák M. Ondrejkovič
Vytvorenie univerzálneho trénera	• zozbieranie všetkých informácii z ihriska (pozícia lopty, spoluhráčov, protihráčovov) • poskytovanie informácie svojim hráčom • vyhodnotenie získaných informácií a určenie, čo majú hráči robiť (ako sa majú rozostaviť, kto ma isť za loptou, akú stratégiu majú voliť...)	J. Gregor
Nástroj pre trénovanie stratégií	• nastavovanie, teda rozloženie hráčov • zmena správania hráča	M. Ondrejkovič

Vylepšovanie editora pohybu	• identifikovanie fázy, kedy robot už stráca stabilitu • zistenie ako by sa malo zmeniť ťažisko robota alebo čo spraviť aby sa udržala stabilita v danej fáze pohybu • napovedanie ako by sa mali meniť uhly kĺbov a časy ich ohýbania, príp. algoritmus, ktorý by skúšal rôzne riešenia	M. Ondrejkovič G. Nagy
	• vylepšiť, prepracovať, poprípade napísať úplne nový editor pohybov	
Experimenty zadefinovania polôch	• dlhodobý záznam stavov kĺbov a snímačov v situáciách, kde sa stráca rovnováha a dochádza k neúmyselnému pádu • navrhnutie nového stabilizačného mechanizmu pomocou nazbieraných údajov	G. Nagy
Roboviz - vylepšenie testovacieho frameworku	• vylepšenie nášho testovacieho frameworku podľa RoboVizu • implementovanie aspoň jednej vlastnosti RoboVizu do nášho testovacieho frameworku.	M. Gregor
Positioning to Win - použitie formácií pri robotickom futbale	• náročné na implementáciu lebo vyžaduje funkčné a stabilné nízkoúrovňové pohyby • táto časť návrhu závisí od dokonale zvládnutej časti základných pohybov hráča	M. Gregor

Nástroj pre trénovanie stratégií

Vytvorenie nástroja pre trénovanie stratégií. Hlavnou častou je nastavovanie, teda rozloženie hráčov, zmena správania hráča. Tento nastroj by mal hlavne význam napr. pre analýzu danej situácie za rôznych predpokladov (priamy kop, nahrávky do priestoru, útok) alebo rôznych situácií pri rovnakej stratégii. Výhodne aj pre overovanie a testovanie stratégií.

Vylepšovanie editora pohybu

Návrh 1

Editor umožňuje vytvárať pohyby ale vytvorený pohyb nezaručuje, že pohyb robota bude stabilný a udrží sa na nohách, preto vytvoriť metódy pre vytváranie stabilných pohybov:

- identifikovanie fázy, kedy robot už stráca stabilitu(sa potáca)

- zistenie ako by sa malo zmeniť ťažisko robota alebo čo spraviť aby sa udržala stabilita v danej fáze pohybu
- napovedanie ako by sa mali meniť uhly kĺbov a časy ich ohýbania, príp. algoritmus, ktorý by skúšal rôzne riešenia prípadne nový prvok – ochrana lopty telom.

Návrh 2

Vhodné by bolo vylepšiť, prepracovať, poprípade napísať úplne nový editor pohybov. Editor by mohol pracovať aj na úplne inom princípe. Cieľom vylepšeného editora by nebolo obohatenie funkcií, skôr vylepšenie základných funkcií z dôvodu, že editor používajú začiatočníci. Pomohlo by to k rýchlejsiemu pochopeniu pohybu robota pre začiatočníkov, aby si osvojili rýchlejšie postup vytvárania pohybov, získali skúsenosť a takto sa rýchlejšie dostali na úroveň, keď už písaním XML súborov vytvárajú kvalitné pohyby.

Návrh na postup práce:

- Otestovať funkcie súčasného editora
- Otestovať kompatibilitu s operačnými systémami
- Navrhnúť nový koncept vytvárania pohybov, alebo navrhnúť zmeny súčasného riešenia
- Implementovať program
- Nový editor sprístupniť pomocou wiki RoboCupu na FIIT

Experimenty zadefinovania polôh

Dlhým testovaním by sa zadefinovali kombinácie natočenia kĺbov a stavov snímačov, pri ktorých dochádza k strate rovnováhy. Použitím výsledkov by sa dal vytvoriť kontrolný mechanizmus, ktorý by sledoval stav robota a zabraňoval by pádu a straty rovnováhy.
Návrh riešenia:

- Dlhodobý záznam stavov kĺbov a snímačov v situáciách, kde sa stáca rovnováha a dochádza k neúmyselnému pádu
- Navrhnúť nové stabilizačné mechanizmy pomocou nazbieraných údajov
- Implementovať riešenie

Roboviz - vylepšenie testovacieho frameworku

Roboviz je program na monitorovanie a podporu vývoja agenta v multiagentovom prostredí. Jeho dokumentácia, návod na inštaláciu, ukážky použitia a návrhy ďalšieho vylepšenia sú na adrese https://sites.google.com/site/umroboviz/home. Je to náhrada Simsparkového monitora robotického futbalu, ktorý navyše vykresľuje agenta s informáciami o stave sveta v 3D prostredí. Roboviz ponúka funkcionality ako programovateľné vykresľovanie rôznych útvarov do ihriska, ktoré poslúžia napríklad pri testovaní obhádzania objektov agentom. Prináša aj možnosť debugovania komunikácie agentov prostredníctvom siete. Medzi hlavné vlastnosti patrí:

- v dvojrozmernom grafickom podaní zobrazuje pod hráčom vektory smeru, kde sa agent pozerá, kde počúva a hovorí alebo aj akým smerom chce napredovať. Tiež zobrazuje kde hráč vidí loptu. Táto funkcionalita je prospešná pri dolaďovaní pohybových a lokalizačných algoritmoch.
- prostredníctvom Robovizu sa dá modelovať svet. Dajú sa napríklad pohybovať hráči, lopta, hráči sa môžu pridávať alebo odoberať.
- už spomínané kreslenie tvarov pre testovanie alebo vývoj algoritmov vyhýbania sa predmetom respektíve iným hráčom. Dá sa to využiť aj pre učenie sa nových formácií prostredníctvom strojového učenia sa a nie programovaním schopností.
- potom už len zostávajú vlastnosti ako možnosť nastavovať alebo reštartovať server, lepšia grafika zobrazovania hry, menu alebo možnosť vidieť svet z perspektívy hráča

Nevýhodu z technického hľadiska, ktorú vidíme je potreba simsparku skompilovaného bez multithreadingu, ktorého vypnutie môže spôsobiť nesprávny alebo pomalší chod servera. Náš testovací framework vylepšíme implementovaním vlastností ako sú v robovize. Pre hlbšiu anlýzu Robovizu sa dostupné zdrojové súbory.

Positioning to Win - použitie formácií pri robotickom futbale

Tím robotického futbalu Austin Villa vyvinuli pozičný a formačný systém, ktorý im dopomohol k výhre viacerých zápasov. Ich pozičný systém rozhoduje v troch krokoch:

1. Najlepšia formácia sa vypočíta podľa algoritmu prideľovania pozícií pre celý tím distribuovaním si pozícii medzi hráčmi navzájom.

2. Najlepšiu formáciu si vypočíta každý hráč podľa toho čo vidí.

3. Zo všetkých vypočítaných formácií sa vyberie tá najlepšia pre súčasnú situáciu.

Sťažujúcimi okolnosťami, s ktorými treba počítať pri výbere formácie sú obmedzenosť komunikácie a toho čo hráč vidí, motorickosť hráčov a zvládnutie nízkoúrovňových pohybov. Formácie, ktoré používa tím Austin Villa sú hlavne obranná formácia a útočná formácia. Prideľovanie pozícií formácie vychádza z pravidiel, že brankár sa do formácie nezapája lebo bráni bránkovisko. Najbližší hráč k lopte sa do formácie nezapája, ale urýchlene sa snaží dostať k lopte a nahrať ju spoluhráčovi alebo vystreliť. Ostatní hráči sa snažia dostať na určené pozície. Náš hráč po zvládnutí nízkoúrovňových pohybov bude ovládať formáciu podľa opísaných pravidiel. Formáciu sa naučí dynamickým spôsobom (strojovým učením sa) alebo naprogramovaním formácie.

3.2.2 Import a kríženie hráčov (pp 2.2)

3.2.2.1 Analýza

Zdrojové kódy je treba najprv analyzovať, čo z časti pokryje task 3.2.3. Agenti dvoch tímov sú veľmi podobní, z dôvodu, že tímy podstatnú časť projektu vyvíjali spolu. Test Framework školského projektu sa líšil iba v testovacích triedach.

3.2.2.2 Návrh

Porovnávanie zdrojových kódov prebehne vo vývojovom nástroji alebo textovom editore. Po zistení nezhody sa kódy krížia na základe niektorého z uvedených princípov:

- Použije sa funkčný kód
- Použije sa lepší kód (kvalitnejší, alebo s lepším výstupom)
- Pomocou dvoch riešení sa vytvorí tretie riešenie

Vyhodnotenie kódov prebehne testovaním, debugovaním alebo logickou simuláciou. V prípade, že sa kódy nevedia dostatočne otestovať aby sa zvolil jeden z princípov, tak do kríženého hráča doplníme oba kódy tým, že sa jeden z nich zakomentuje a posunie sa na neskoršie testovanie.

3.2.3 Analýza zdrojových kódov z minulého roka (pp 2.3)

3.2.3.1 Analýza

Na analýzu doterajších vlastností školského projektu sme si zobrali minuloročné projekty. Projekt sme analyzovali v 2 častiach a to Jim a TestFramework. Školský projekt obsahuje aj tretiu časť RoboCupLibrary, ale táto časť sa postupne prestáva implementovať v novších verziách produktu školského projektu. Preto analýza tretej časti sa dotýkala len implementovaných prvkov. Jim bol dosť rozdielny medzi tímom 5 a tímom 17 z minulého roku lebo skúšali rôzne prístupy. Testovací framework sa líšil iba vo vektorových nastaveniach v naprogramovaných testoch. Bolo potrebné zistiť funkčnosť jednotlivých častí.

3.2.3.2 Návrh

Každý člen tímu si zobral na starosť tretinu z množstva 2 projektov Jim a TestFrameworku. Funkcionalita jednotlivých funkčných prvkov sa testovala použitím Simspark servera a monitora zapínaním/vypínaním alebo modifikovaním nastavení jednotlivých prvkov. Výsledok analýzy každý člen zhrnul do preddefinovanej tabuľky, ktorá bude súčasťou zápisnice z 6. Stretnutia. Z analýzy vyplynuli nasledujúce funkčné funkcie:

- sk.fiit.jim.Settings – funguje globálne nastavenie hry
- sk.fiit.agent.AgentInfo – funguje a vracia informácie o agentovi, jeho polohe , polohe vzhľadom na loptu
- sk.fiit.agentcommunication – funguje. Zabezpečuje komunikáciu zo serverom, posielanie aj prijímanie správ.
- sk.fiit.agent.models.AgentModel – obsahuje model agenta. Údaje o polohe, natočení a výpočty z accelerometra.
- sk.fiit.agent.models.AgentPositionCalculator – počíta polohu agenta na základe vlajok, ktoré vidí.
- sk.fiit.agent.models.AgentRotationCalculator – počíta natočenie agenta podľa vlajok, ktoré vidí.

- sk.fiit.agent.models.DynamicObject – vypočíta polohu pohybujúceho sa objektu ako napríklad lopty.
- sk.fiit.agent.models.EnvironmentModel – model sveta hry, ktorý obsahuje informácie ako čas hry, stav hry, verzia servera.
- sk.fiit.agent.models.TacticalInfo – obsahuje informáciu o hernej situácii (útočíme, bránime, ...)
- sk.fiit.agent.moves – balík obsahuje triedy, ktoré definujú rozsah natočiteľnosti kĺbov, algoritmus opravy chýb natočenia kĺbov, triedy spracovania lowsklillov. Všetko funguje.
- sk.fiit.jim.log – balík obsahuje triedy na logovanie akcií v simulovanom robotickom futbale.
- sk.fiit.jim.init – balík svojimi triedami zabezpečuje inicializáciu hráča na server, čo znamená zabezpečiť vloženie agenta, načítanie ruby skriptov a ich otestovanie, načítanie xml súborov s pohybmi pre agenta
- sk.fiit.jim.gui – balík s triedami grafického rozhrania na znovu načítanie xml súborov s pohybmi aby sa nemusel reštartovať server
- sk.fiit.jim.build – balík s triedami, ktoré zabezpečujú vytvorenie a testovanie release súboru (balíka) s ruby skriptami, xml súbormi a potrebnými java knižnicami.
- sk.fiit.jim.annotation.gui – balík obsahuje triedy na rozpoznanie a načítanie xml súborov.
- sk.fiit.jim.agent.parsing – parsovanie správ a prijímanie zvukových a vizuálnych správ.
- sk.fiit.jim.agent.server – obsahuje tiredu na multi vláknový server.
- sk.fiit.agent.skills – pokus o riešenie high skillov, funguje, ale nie je to dokonalé riešenie. Je tam čo vylepšovať.
- sk.fiit.jim.annotation – správa vytvárania pohybov z anotácií. Funguje len manažér anotácii a určenie osí sveta.
- sk.fiit.testframework.init – balík, ktorý obsahuje triedy na spustenie a nastavenie testframeworku. Je možné si definovať rôzne implementácie spustenia frameworku ako napríklad na viacerých vláknach. Funguje všetko až na MpiImplementation.
- sk.fiit.testframework.monitor – balík obsahuje triedy, ktoré definujú typy správ, ktoré sa na serveri posielajú, triedy ktoré pridávajú a odoberajú vlákna agentov a monitorov a zabezpečujú pridávanie listenerov na agentov.
- sk.fiit.testframework.communication.agent – balík obsahuje triedy spracúvajúce komunikáciu medzi agentmi, ale tiež udržujú aj informácie o hráčovi ako tím, číslo dresu, pridelenie voľného tftp portu, pridanie a odobratie agenta alebo agentov. Tiež aj triedy predstavujúce model Agenta pridaného serverom a alebo agenta Jima.
- sk.fiit.testframework.communication.robocupserver – balík, ktorý obsahuje triedy na modifikáciu situácie na ihrisku. Dokážeme vykonať príkaz na serveri, posunúť hráča, loptu.
- sk.fiit.testframework.beta.ui – balík reprezentujúci grafické rozhranie testframeworku. Rozhranie je generované pomocou Jigloo.
- sk.fiit.testframework.annotator.serialization – balík na vytváranie pohybov na základe anotácii zoserializuje viacero nízkoúrovňových pohybov do jedného xml.
- sk.fiit.testframework.agenttrainer.models – triedy zodpovedné za synchronizáciu fáz pohybu, gettery a settery efektorov. Využíva sa aj pri spúšťaní pohybov z xml a to funguje.
- sk.fiit.testframework.worldPresentation – interpratácia modelu sveta a prenesenie správ do sveta, reprezentácie hráča a nastavovanie pozícií agentov

- sk.fiit.testframework.parsing - parsovanie správ, typy správ, grafy. Funguje.

Nefungujúce časti kódu:

- sk.fiit.testframework.agenttrainer – balík, ktorý má zabezpečiť trénera nových pohybov, ale neobsahuje nijaký algoritmus, ktorý by to riešil.
- sk.fiit.testframework.communication.agent.AgentJim – nefunguje reštart servera
- sk.fiit.testframework.annotator.AnnotatorTestCaseResult – trieda má zabezpečiť výpis správ pri testovaní pohybov, ale nič nevypisovala ani pri zmene granularity výpisu správ.

3.2.4 Tvorba dokumentácie (pp 2.4)

3.2.4.1 Analýza

Tímová dokumentácia pozostáva z dvoch typov dokumentácie:

- Dokumentácie k inžinierskemu dielu
- Dokumentácie k riadeniu

Obsahom prvej z nich je technická dokumentácia celého projektu a jednotlivých analyzovaných či implementačných úloh.

Obsahom druhej dokumentácie je riadenie projektu. Pozostáva z častí, ktoré sa týkajú riadenia ľudí, vývoja a pod.

3.2.4.2 Návrh

Obsahy oboch dokumentácií vznikali od začiatku projektu v podobe metodík a analýzy rôznych častí a aspektov pri vývoji. Neskôr boli vytvorené 2 veľké dokumenty, ktoré tieto dokumentácie zahrnuli. Tieto štruktúrované dokumenty sú zhromaždením všetkých informácii o projekte a samotnom riadení projektu. Dokumentácia je pravidelne dopĺňaná a aktualizovaná.

3.2.5 Návod na písanie kódu (pp 2.6)

3.2.5.1 Analýza

Analýza pre návod na písanie kódu pozostávala z analýzy existujúcich zdrojových kódov pre RoboCup a z čítania knihy Martin, R.C.: Clean Code – A Handbook of Agile Software Craftsmanship. Pearson Education, Inc. 2009. ISBN 0-13-235088-2.

3.2.5.2 Návrh

3.2.5.2.1 JavaDoc komentáre pre triedy a metódy

```
/**
 *  Settings.java
 *
```

```
 *  Encapsulates global settings that alter the behaviour of the code throughout
 *  the entire project. Can change default settings for the game. Usually is
 *  meant to be set in ./scripts/config/settings.rb.
 *
 *  @Title    Jim
 *  @author   marosurbanec
 *  @author   Androids
 */
public final class Settings { ...
```

Nad každou triedou je potrebné vytvoriť JavaDoc komentár v minimálne takejto forme:

- Názov súboru triedy
- Popis triedy (cca. 3 riadky)
- Titulok je názov projektu (Jim, TestFramework, ...)
- Autor vo formáte ako v AIS, napr: @author xsuchac
- Znovu autor, ale už s názvom tímu, teda @author A55-Kickers

```
/**
 * Checks employees and find out who is on vacation.
 *
 *  @author xsuchac
 *  @author A55-Kickers
 *
 * @param employeesToCheck - employees needed to be checked.
 * @param employeesOnVacation - employees who are on vacation.
 * @return resultEmployees - employees on vacation filtered out
 * of employeesToCheck.
 * @throws IllegalArgumentException - if any parameter is null.
 */
private static ArrayList getCheckedEmployeesOnVacation(ArrayList
employeesToCheck, ArrayList employeesOnVacation) throws
IllegalArgumentException { ...
```

Nad každou metódou:

- Popis metódy
- Autor vo formáte ako v AIS, napr: @author xsuchac
- Znovu autor, ale už s názvom tímu, teda @author A55-Kickers
- Parametre metódy (ak sú): @param názovParametra – popis parametra
- Návratová hodnota metódy: @return názovNávratovejHodnoty – popis
- Výnimky, ktoré môže metóda vyhadzovať (ak sú): @throws NázovVýnimky - popis

3.2.5.2.2 Komentáre všeobecne

Pri komentovaní metód a premenných je potrebné uvádzať nad ich definíciou JavaDoc komentár. Teda preferovať:

```
/** something */
```

pred

```
/* something */
```

Ukážka:

```
/**
 * Team of the agent
 */
public static String team;
```

Ostatné komentáre, ktoré sa neviažu na definíciu premennej, metódy, triedy a pod. môžu byť v základnom tvare `// something` alebo `/* something */`

Komentáre však používať iba v nutných prípadoch. Čisto svoj kód je potrebné radšej vytvoriť tak, aby dával zmysel aj ostatným (zmysluplnými názvami premenných, metód a pod.), namiesto aby sa len okomentoval.

3.2.5.2.3 Poznámky do kódu, zakomentovaný kód

Poznámky do kódu používať len pre osobnú krátkodobú potrebu. Pri odovzdávaní kódu do repozitára je potrebné ich zmazať alebo predtým príčinu poznámky vyriešiť. Kód sa pod vplyvom mnohých zmien a mnohých meniteľov kódu môže zmeniť tak, že poznámka úplne stratí zmysel a po čase si nikto nespomenie na to, čo znamenala a nikto ju pre istotu nebude chcieť vymazať.

Toto isté platí aj pre zakomentovanú časť kódu.

3.2.5.2.4 TODO komentár

V prípade, že si autor kódu v určitej časti kódu myslí, že by tam bolo potrebné niečo konkrétne dorobiť, ale z nejakého dôvodu to nemôže spraviť hneď, tak môže použiť TODO komentár. K týmto komentárom je potrebné sa pravidelne vracať a dať si záležať, aby bola funkcionalita na ktorú poukazujú doplnená a komentár odstránený. Výhodou TODO komentárov je, že je ich možné ľahko vyhľadať cez vývojové prostredie.

Obsah TODO komentára bude pozostávať z popisu a mena autora (tvar ako v AIS) s názvom tímu:

```
/* TODO Add special move #2 after discussion with colleagues.
 * @author xsuchac, A55-Kickers
 */
```

TODO komentár neberie do úvahy značky začínajúce zavináčom, preto môžeme použiť vyššie uvedenú skrátenú formu zápisu autora a tímu do TODO komentára.

3.2.5.2.5 Zmeny v kóde

Niektoré tímy si v kóde komentármi označujú časti kódu, ktoré zmenili. Túto praktiku robiť nebudeme. V prípade, že budeme potrebovať od predchádzajúcich autorov vysvetlenie k časti kódu, jej autorstvo zistíme podľa originál zdrojových kódov, ktoré sme od nich dostali.

3.2.5.2.6 Štruktúra triedy

Trieda by mala začínať zoznamom premenných. Public static konštanty by mali ísť najprv. Potom private static premenné, ďalej private atribúty inštancií. Len zriedkavo je dobré mať aj public premenné.

Public metódy by mali nasledovať za zoznamom premenných. Private metódy prislúchajúce k public metódam by mali ísť pod ne v poradí ako ich public metóda využíva. Toto pomáha čítať si triedu podobne ako novinový článok.

„Public“ dávať jedine premenným a metódam, ktoré budú využívať iné triedy!

3.2.5.2.7 Konvencia kódu

Používať klasickú java konvenciu. Ukážka:

```
public class ClassA {

	public int number = 1;
	private String letters[] = new String[]{“A”, “B”};

	public int meth(String text, int number) {
		if (text == null) {
			text = “a”;
		} else if (text.length() == 0) {
			text = “empty“;
		} else {
			number++;
		}
	}
}
```

Táto konvencia je defaultne nastavená v NetBeans a Eclipse. Pokiaľ ju nedodržiavate, tak je potrebné kód naformátovať:

- postup pre NetBeans:
  - pravým tlačidlom myši kliknúť do okna s kódom
  - vybrať „Format“
- postup pre Eclipse:
  - pravým tlačidlom myši kliknúť do okna s kódom
  - vybrať „Source“, kliknúť na „Format“

3.2.5.2.8 Zmysluplné názvy

Radšej použiť dlhší názov premennej, metódy a pod. ako taký čo málo povie. Napr:

Premennú, ktorá udáva uplynutý čas, pomenovať namiesto `int days;` radšej `int elapsedTimeInDays;`

3.2.5.2.9 Malé metódy a triedy

Metódy a triedy by mali byť malé, teda by nemali mať veľa riadkov. Čím menej tým lepšie, ale musí to byť funkčne použité.

Metóda by mala robiť jednu vec. Ak robí viac vecí tak sa dá rozbiť na viacero metód. Podobne aj trieda.

3.2.5.2.10 Angličtina

Je nutné v zdrojovom kóde VŽDY používať angličtinu!

3.2.6 Návod na používanie Git-u (pp 2.7)

3.2.6.1 Základ

Vytvorenie novej vetvy s menom branchname v lokálnom repozitári.

- *git branch branchname*

Pred samotnou prácou sa treba „prepnúť“ sa na danú vetvu vývoja.

- *git checkout branchname*

Pre prezeranie všetkých vetiev v lokálnom repozitári

- *git branch –a*

V príklade je uložené meno vzdialeného repozitára servera(origin) spolu s odkazom na server.

- *git remote add origin git@gitbus.fiit.stuba.sk:robocup-a55-kickers/robocup-a55-kickers.git*

Vzdialeným repozitárom môže byť aj iný repozitár na lokálnom počítači. Takýchto vzdialených repozitárov môžem v prípade potreby vytvoriť viac.

Vymazanie vetvy branchname v lokálnom repozitári

- *git branch -d branchname*

Pripadne, že som na vetve branchname a chcem ju vymazať treba prepnúť na inú vetvu.

Keď vetva obsahuje konflikty a chcem sa jej zbaviť, treba

- *git branch -D branchname*

Po zmene implementácii treba explicitne povedať, ktoré súbory podstúpili zmeny, prípadne pribudli ako nové v projekte a treba ich brať v úvahu pre nasledujúci commit.

- *git add .*

Zmeny nahrá do repozitára a spravíme ich viditeľnými v rámci lokálneho repozitára.

- *git commit -m "message"*

Pred zaslaním vetvy na vzdialený repozitár treba stiahnuť vetvu zo vzdialeného repozitára. Uvedený príklad stiahne vetvu branchname z adresy uloženej v origin a vetva je uložená s menom origin/branchname v lokálnom repozitári.

- *git fetch origin branchname*

Porovnanie súborov, odhalenie prípadných konfliktov medzi lokálnou vetvou a stiahnutou vetvou zo vzdialeného repozitára.

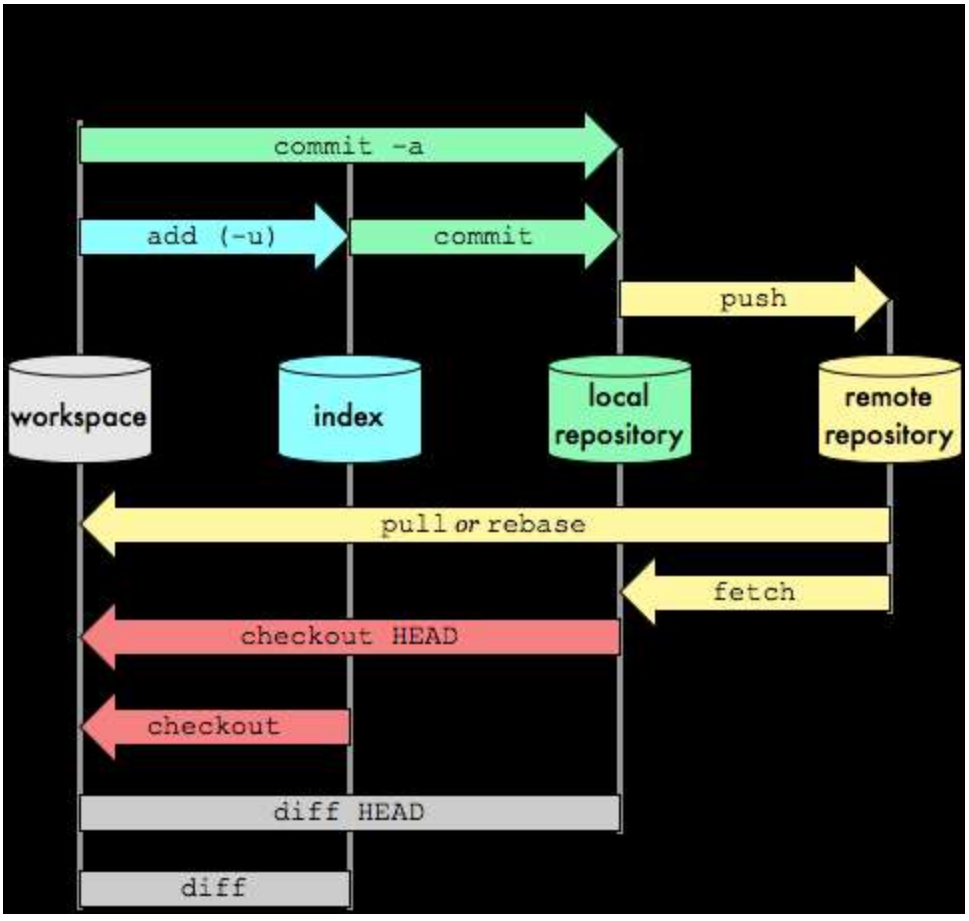
- *git diff branchname origin/branchname*

Zlúčenie vetiev, výsledok zlúčenia na vetve, na ktorej som nastavený.

- *git merge branchname origin/branchname*

Zaslanie vetvy branchname na vzdialený repozitár servera. V prípade, že vetva na vzdialenom repozitári neexistuje bude vytvorená ako nová.

- *git push origin branchname*



Obrázok 3.9: Architektúra komunikácie so vzdialeným repozitárom

Príkaz na globálne ignorovanie nechcených súborov v repozitári:
cd ~
vim .gitignore_global
git config --global core.excludesfile ~/.gitignore_global
do .gitignore_global napíšte :
*/nbproject/
build.xml
manifest.mf

3.2.6.2 Ďalšie príkazy

Fetch a merge vykonať naraz.

- *git pull origin branchname*

V tomto prípade keď vetvy obsahujú konflikty, vetva bude rozbitá. Ak chcem obnoviť stav vetvy, na ktorej som nastavený, na posledný commit.

- *git reset --hard*

Všetky zmeny od posledného commitu budú nenávratne vymazané.

Sledovanie zmien.

- git reflog

Nastavenie vetvy na vybraný minulý stav.

- git reset --hard HEAD~1

3.3 Šprint 3 – „gama”

Tabuľka 3.7: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
3.1	Člen tímu	Analyzovať chôdzu hráča	Tieto poznatky využiť pri návrhu chôdze.
3.3	Člen tímu	Analyzovať pohyby hráča	Tieto poznatky využiť pri návrhu nových pohybov.
3.6	Člen tímu	Uskutočniť prezentáciu zdrojového kódu	Aby som mal pred samotným vývojom v ňom prehľad.
3.7	Člen tímu	Sa stretnúť z minuloročnými riešiteľmi projektu	Pochopiť súvislosti, ktoré mi robili problémy.
3.8	Člen tímu	Mať všetky dôležité informácie na wiki projektu	Všetky dôležité veci nájsť na jednom mieste a aby bol zabezpečený dôkladný prehľad pre budúce generácie riešiteľov projektu

Tabuľka 3.8: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
3.1	-	3.1	Analýza chôdze nášho (kríženého) hráča	G. Nagy
3.3	-	3.3	Analýza pohybov	M. Červeňák
3.6	-	3.6	Prezentácia analyzovaného kódu	M. Gregor
3.7	-	3.7	Stretnutie s minulorčným účastníkom projektu RoboCup	G. Nagy
3.8	G. Nagy	3.8	Vytvorenie štruktúry wiki	G. Nagy
3.8	G. Nagy	3.9	Návod na inštaláciu simsparku Windows 7- 64 bit	M. Gregor
3.8	G. Nagy	3.10	Návod na inštaláciu simsparku Windows 7 - 32 bit	J. Grega
3.8	G. Nagy	3.11	Návod na inštaláciu simsparku Linux – 64 bit	M. Gregor
3.8	G. Nagy	3.12	Návod na inštaláciu simsparku Linux – 32 bit	G. Nagy
3.8	G. Nagy	3.13	Návod na inštaláciu simsparku Windows XP – 32 bit	F. Sucháč
3.8	G. Nagy	3.14	Návod na spustenie hráča v simspraku	J. Grega
3.8	G. Nagy	3.15	Vloženie dokumentov súvisiacich s RoboCup-om na wiki	G. Nagy

3.3.1 Analýza chôdze nášho (kríženého) hráča (pp 3.1)

Chôdze sa testovali optickou kontrolou priebehu chôdze. Testy sa vykonávali prejdením polky ihriska a celého ihriska. Pri vhodných chôdzach je vyrátaná reálna rýchlosť chôdze. Popisy hovoria o trajektórií, plynulosti, rýchlosti a stability chôdze. Výsledky analýzy sme zapísali a uložili na našu stránku, do manažérskeho systému a na wikipediu projektu.

3.3.2 Analýza pohybov (pp 3.3)

Na analýzu všetkých pohybov sme použili zlúčené low level pohyby od minulých tímov vo formáte xml. Pri analýze sme postupovali vložením low level pohybu do high skillu v plánovači, ktorý pohyb vykonal 5x. Počas vykonávanie sme pozorovali rýchlosť, stabilitu a presnosť. Výsledky sme zapísali a uložili na našu stránku, do manažérskeho systému a na wikipediu projektu.

3.3.3 Prezentácia analyzovaného kódu (pp 3.6)

Analyzovaný kód z predchádzajúceho šprintu bolo potrebné efektívne zdieľať medzi členmi tímu. Preto sme si dohodli tímové stretnutie, na ktorom každý z členov prezentoval svoju časť kódu.

3.3.3.1 Priebeh

Stretnutie sa uskutočnilo vo štvrtok 15.11.2012 o 14:00 v študentskom klube na FEI STU v Bratislave. Členovia postupne na svojich PC ukazovali časti kódu a pri užitočných alebo zaujímavých častiach kód objasnili ako ho využiť.

3.3.4 Stretnutie s minuloročným účastníkom projektu RoboCup (pp 3.7)

Pre rýchlejšie pochopenie niektorých častí projektu RoboCup sme sa rozhodli kontaktovať minuloročné tímy. Členovia minuloročných tímov boli ochotní komunikovať aj formou email a niektorí boli ochotní si dohodnúť aj osobné stretnutie, ktoré sme náležite využili.

3.3.4.1 Návrh

Každý člen nášho tímu si pripravil otázky ohľadom projektu. Počas stretnutia boli otázky postupne kladené, prípadne nad robocup serverom demonštrované.

3.3.4.2 Implementácia

Na každú položenú otázku členovia minuloročných tímov odpovedali a prípadne demonštrovali ukážkou kódu. Po stretnutí s členmi minuloročného tímu sa nám podarilo odstrániť mnoho nedostatkov a bugov, ktoré boli spôsobené implementačnými nedostatkami minuloročných tímov.

3.3.5 Vytvorenie štruktúry wiki (pp 3.8)

Na základe skúsenosti akú zažil náš tím pri preberaní projektu RoboCup sme rozhodli pomôcť nasledujúcim tímom, ktoré preberú projekt RoboCup po nás, v rýchlejšom spoznávaní a orientácii sa v projekte vďaka wikipedii k projektu RoboCup. Wikipedia v súčasnom stave obsahuje nedostatočné dáta k aktuálnemu stavu projektu. Preto sme sa rozhodli ju reštrukturalizovať a doplniť o najnovšie údaje.

3.3.5.1 Návrh

Štruktúra wikipedie je rozdelená do ucelených celkov s názvom Úvod do RoboCup, Úvod do Robocup na FIIT, Návody a Inštalácie a dôležitý stručný návod na prácu wiki, ktorý zabezpečí aby sa koncepcie pri dopĺňaní a vytváraní wiki dosiahli aj v budúcnosti.

3.3.5.2 Implementácia

Prvú úroveň hierarchie štruktúry vo forme hyperlinkov sme implementovali na úvodnej stránke. Po rozkliknutí konkrétnej časti sa dostanete na ďalšiu úroveň hierarchie wikipedie projektu RoboCup, kde nájdete podrobnejšie informácie ku konkrétnej časti. Obsah wikipedie sa priebežne dopĺňa.

3.3.6 Návod na inštaláciu simsparku Windows 7- 64 bit (pp 3.8)

Prvou vecou, s ktorou sa nováčikovia projektu RoboCup stretnú je inštalácia RoboCup servera na svoj počítač. Z tejto elementárnej potreby pri napredovaní v projekte sme si dali za úlohu spísať návody na inštaláciu na všetky nám dostupné operačné systémy.

3.3.6.1 Návrh

Člen tímu, ktorí sa podujal na túto úlohu presne popísal kroky inštalácie na svojom operačnom systéme. Podľa tohto návodu ho skontrolovali aj ďalší dvaja členovia tímu, či je návod funkčný. Návod bol umiestnený predovšetkým na wikipediu a do nášho manažérskeho systému.

3.3.6.2 Implementácia

Inštalácia na systém Windows 7 – 64 bit prebehla bez väčších problémov. Návod na inštaláciu bol uložený na wikipediu a do manažérskeho systému a bol overení ďalšími členmi tímu.

3.3.7 Návod na inštaláciu simsparku Windows 7 - 32 bit (pp 3.8)

Ako prvé pri stretnutí s projektom RoboCup si musí každý nováčik nainštalovať RoboCup server. Z tohto dôvodu sme sa rozhodli spísať návody na inštaláciu RoboCup servera na všetky pre nás dostupne operačne systémy.

3.3.7.1 Návrh

Člen týmu, ktorí mal túto úlohu na starosti presne popísal kroky pri inštalácií na svojom OS. Ďalší dvaja členovia tímu následne postupovali podľa tohto návodu a tak overili, čí je návod funkčný. Návod bol umiestnený do manažérskeho softvéru Redmine a na wikipediu.

3.3.7.2 Implementácia

Inštalácia prebehla bez problémov. Návod na inštaláciu bol vložení na wikipediu a do Redmine-u a bolo overený dvomi členmi tímu.

3.3.8 Návod na inštaláciu simsparku Linux – 64 bit (pp 3.8) a 32 bit (pp 3.8)

Prvou vecou, s ktorou sa nováčikovia projektu RoboCup stretnú je inštalácia RoboCup servera na svoj počítač. Z tejto elementárnej potreby pri napredovaní v projekte sme si dali za úlohu spísať návody na inštaláciu na všetky nám dostupné operačné systémy.

3.3.8.1 Návrh

Člen tímu, ktorí sa podujal na túto úlohu presne popísal kroky inštalácie na svojom operačnom systéme. Podľa tohto návodu ho skontrolovali aj ďalší dvaja členovia tímu, či je návod funkčný. Návod bol umiestnený predovšetkým na wikipediu a do nášho manažérskeho systému.

3.3.8.2 Implementácia

Inštalácia na systému Kubuntu 11 64-bit/32-bit prebehla vďaka inštalačným balíkom na serveri Kubuntu úplne bez problémov. Návod bol uložený na wikipédiu projektu RoboCup a do manažérskeho systému a bol overený ďalšími členami tímu.

3.3.9 Návod na inštaláciu simsparku Windows XP – 32 bit (pp 3.8)

Prvou vecou, s ktorou sa nováčikovia projektu RoboCup stretnú je inštalácia RoboCup servera na svoj počítač. Z tejto elementárnej potreby pri napredovaní v projekte sme si dali za úlohu spísať návody na inštaláciu na všetky nám dostupné operačné systémy.

3.3.9.1 Návrh

Člen tímu, ktorí sa podujal na túto úlohu presne popísal kroky inštalácie na svojom operačnom systéme. Podľa tohto návodu ho skontrolovali aj ďalší dvaja členovia tímu, či je návod funkčný. Návod bol umiestnený predovšetkým na wikipediu a do nášho manažérskeho systému.

3.3.9.2 Implementácia

Inštalácia na systém Windows XP – 32 bit prebehla bez väčších problémov. Návod na inštaláciu bol uložený na wikipediu a do manažérskeho systému a bol overení ďalšími členmi tímu.

3.3.10 Návod na spustenie hráča v simspraku (pp 3.8)

Ďalším krokom hneď po inštalácii servera je overenie jeho funkčnosti. Na to sme si dali úlohu vytvoriť návod na spustenie hráča v simsparku a tak overiť funkčnosť servera.

3.3.10.1 Návrh

Člen tímu, ktorí mal túto úlohu na starosti spísal presný návod na vložiť agenta na server a tak server otestovať. Návod na spustenie agenta bol vytvorení pre prostredie NetBeans a Eclipse.

Taktiež bol vytvorení návod na otestovanie servera defaultným agentom. Tento návod bol vložení na wikipediu a tiež do Redmine-u.

3.3.10.2 Implementácia

Tvorba návodu prebehla bez problémov. Návod bol otestovaní všetkými členmi tímu. Po otestovaní bol vložení na wikipediu a do Redmine-u

3.3.11 Vloženie dokumentov súvisiacich s RoboCup-om na wiki (pp 3.8)

Keďže sme sa rozhodli smerovať pri vývoji cestou najskôr prerobiť a vyrobiť dôkladné návody a pod. na wiki, tak bolo potrebné všetky už vyrobené dokumenty, návody a analýzy na wiki nahrať.

3.3.11.1 Implementácia

Vloženie súborov prebehlo v poriadku a všetky dokumenty vytvorené tímom boli vložené na wiki do zodpovedajúcich častí.

3.4 Šprint 4 – „delta“

Tabuľka 3.9: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
3.5	Člen tímu	Mať všetky dôležité informácie na wiki projektu	Všetky dôležité veci nájsť na jednom mieste a aby bol zabezpečený dôkladný prehľad pre budúce generácie riešiteľov projektu

Tabuľka 3.10: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
	-	3.2	Špecifikácia a návrh vylepšenia chôdze	F. Sucháč
3.8	G. Nagy	4.1	Celkový koncept fungovania	G. Nagy
3.8	G. Nagy	4.2	Analýza Low-level pohybov	J. Grega
3.8	G. Nagy	4.3	Analýza High-level pohybov	M. Gregor
3.8	G. Nagy	4.4	Analýza plánovačov	J. Grega
3.8	G. Nagy	4.5	Koncept modelu sveta	M. Červeňák
3.8	G. Nagy	4.7	Testframework	M. Gregor
3.8	G. Nagy	4.8	Testframework bugfix	F. Sucháč
3.8	G. Nagy	4.9	Návod na používanie wiki	G. Nagy

3.4.1 Špecifikácia a návrh vylepšenia chôdze

Pri vylepšovaní chôdze hráča simulovaného robotického futbalu našej fakulty, s názvom Jim, by sme sa chceli uberať úpravou na dynamickú chôdzu. Hráč Jim sa momentálne pohybuje staticky, to znamená že jeho pohyby sú vopred zadefinované ako pevne dané hodnoty, o ktoré sa postupne ohýbajú kĺby hráča. Statické pohyby však majú veľké obmedzenia v možnostiach ich vylepšovania. Ak by sme chceli zvýšiť rýchlosť statickej chôdze, čelili by sme problémom straty stability hráča. Pri dynamických pohyboch nie sú hodnoty natočenia kĺbov hráča stále rovnaké, ale menia sa počas pohybu. Tieto zmeny môžu byť vypočítavané napríklad tak, aby dynamický pohyb obnovil stratu stability, ktorá sa počas pohybu vyskytne. Ďalším príkladom využitia dynamického pohybu namiesto statického môže byť to, že sa pohyb bude prispôsobovať hernej situácii a tým meniť smer pohybu. Pri čisto statických pohyboch potrebuje hráč na zmenu trajektórie zastať, vykonať pohyb ktorým sa nasmeruje na nový smer a znova začať chôdzu.

Existuje viacero metód, ktoré sa používajú na vývoj dynamickej chôdze. Spomeniem Zero moment point (ZPM), fuzzy logiku a neurónové siete. V tomto návrhu sa chceme zamerať na prvú spomenutú metódu.

Zero moment point sa používa na výpočet miesta, v ktorom dosiahne humanoidný robot stabilitu. Pomocou toho je možné určiť, kam má humanoidný robot vykonať krok nohou, aby sa stabilizoval. Obnovovaním stability počas chôdze je možné pracovať na zvyšovaní jej rýchlosti.

3.4.1.1 Výpočet ZMP súradníc

Výpočet x-ovej a y-ovej súradnice pomocou ZMP vyjadrujú podľa [1] nasledovné vzorce:

$$x_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)x_i - \sum_{i=1}^{n} m_i\ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\Omega_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)}$$

$$y_{\text{zmp}} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)y_i - \sum_{i=1}^{n} m_i\ddot{y}_i z_i - \sum_{i=1}^{n} I_{ix}\Omega_{ix}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)}$$

Tento výpočet ZMP neberie humanoidného robota ako celok, ale berie do úvahy aj hybnosti jeho jednotlivých častí tela. Preto vzorec obsahuje operácie súčtov kde premenná „i“ určuje konkrétnu časť tela. „m“ je hmotnosť časti tela, „Iix“ a „Iiy“ sú inerciálne komponenty, „Ωix“ a „Ωiy“ sú absolútne uhlové rýchlosti okolo x-ovej a y-ovej osi ťažiska i-tej časti tela, „g“ je gravitačné zrýchlenie, x, y, z sú súradnice ťažiska časti tela absolútnej karteziánskej sústavy súradníc.

3.4.1.2 Inverzná kinematika kĺbov používajúca ZMP súradnice

Inverzná kinematika je metóda na určenie hodnôt natočenia kĺbov humanoidného robota tak, aby sa pomocou toho dostal na určitú pozíciu. Touto témou sa zaoberá viacero prác, ako napríklad [2], [3] a [4]. V týchto prácach uvádzajú presné vzťahy, pomocou ktorých zo vstupných súradníc bodu, kam sa má robot presunúť, určujú hodnoty natočenia viacerých alebo všetkých kĺbov robota Nao.

3.4.1.3 Zmeny potrebné v hráčovi Jim

V súčasnom hráčovi Jim je potrebné doplniť jednotlivé časti tela, tak ako to urobil aj [5] a namapovať ich na model agenta a na kĺby, ktoré ich navzájom spájajú.

3.4.1.4 Zdroje

[1] Huang, Q. et al: Sensory Reflex Control for Humanoid Walking
http://www.ent.mrt.ac.lk/iml/paperbase/TRO%20Collection/TRO/2005/october/17.pdf

[2] Graf, C. et al: A Robust Closed-Loop Gait fot the Standard Platform League Humanoid http://www.ais.uni-bonn.de/humanoidsoccer/ws09/papers/HSR09-005.pdf

[3] Jadidi, M. et al: Kinematic Modeling Improvement and Trajectory Planning of the NAO Biped Robot
http://www.academia.edu/790706/Kinematic_Modeling_Improvement_and_Trajectory_Planning_of_the_NAO_Biped_Robot

[4] Lagoudakis, M. et al: Forward and Inverse Kinematics for the Nao Humanoid Robot http://www.google.sk/url?sa=t&rct=j&q=inverse%20kinematics%20nao&source=web&cd=4&cad=rja&ved=0CEoQFjAD&url=http%3A%2F%2Fwww.intelligence.tuc.gr%2Flib%2Fdownloadfile.php%3Fid%3D430&ei=0d3HULSvKMXOsway0IGQAQ&usg=AFQjCNEa8Qa9B-f8nMjmzWydIlhl78frrA&bvm=bv.1354675689,d.Yms

[5] Hudec, J.: Motorika hráča simulovaného robotického futbalu. Ústav informatiky a softvérového inžinierstva, FIIT STU v Bratislave, 2012. 79 s.

3.4.2 Celkový koncept fungovania (pp 3.5)

Analýza a popis konceptu celkového fungovanie sú veľmi dôležité pre pochopenie práce s RoboCup projektom. Preto sme rozhodli náležite spracovať túto tému. Ide o popis súvisu medzi High Skill a Low Skill pohybmi a plánovačmi a ako sa tieto časti v systéme spúšťajú.

3.4.2.1 Riešenie

Zodpovedný člen tímu opísal koncept fungovania systému projektu RoboCup na podrobnej úrovni a opis vložil na wikipediu projektu RoboCup a do manažérskeho systému. Následne výsledky svojej práce prezentoval na stretnutí tímu, čo bol len pozitívny prínos pre tím.

3.4.3 Analýza Low-level pohybov (pp 3.5)

Low level alebo Low Skill pohyby sú dôležitou časťou implementácie robota Nao. Preto analýza existujúcich pohybov, vytváranie nových low skill pohybov a princíp fungovania sú dôležitou časťou, ktorú by mal poznať každý člen tímu projektu RoboCup.

3.4.3.1 Navrh

Cieľom tejto úlohy bolo vytvoriť prehľad o fungovaní, tvorbe a štruktúre low level pohybov. Člen tímu, zodpovedný za túto úlohu opísal ako fungujú low level pohyby. Následne opísal štruktúru XML súborov ktoré slúžia na tvorbu jednotlivých pohybov. Po spísaní štruktúry, keď už vedel čo všetko zahŕňa tvorba low level pohybov vytvoril návod pre manuálnu tvorbu pohybov a tiež návod pre tvorbu pohybov pomocou editora pohybov.

3.4.3.2 Implementácia

Popis fungovania low level pohybov a popis ich tvorby bol umiernený na wikipediu a do Redmine-u. K návodu manuálneho vytvorenia pohybu bol vytvorení a jednoduchý príklad pre lepšie pochopenie jeho tvorby ktorý je tiež vložený na wikipedii.

3.4.4 Analýza High-level pohybov (pp 3.5)

High level alebo High Skill pohyby sú logickou časťou implementácie robota Nao. Preto analýza existujúcich pohybov, vytváranie nových high skill pohybov a princíp fungovania sú dôležitou časťou, ktorú by mal poznať každý člen tímu projektu RoboCup.

3.4.4.1 Návrh

Cieľom bolo aby členovia tímu vedel kde vyhľadať high skill pohybom ich implementovať a použiť. Preto zodpovedný člen prešiel jednotlivé zdrojové súbory high skill pohybov a každý z nich popísal načo a ako funguje. Po analýze high skill pohybov, keď člen mal prehľad v high skill pohyboch, vytvoril návod a príklad na vytvorenie high skill pohybu. Následne ešte popísal princíp fungovanie high skill pohybov ako medzivrsty medzi plánovačmi a low skill pohybmi.

3.4.4.2 Implementácia

Popisy existujúcich High Skill pohybov boli umiestnené na wikipediu a do manažérskeho systému spolu s návodom a príkladom na vytvorenie high skill pohybu a konceptu fungovania high skill pohybov. Z analyzovaných pohybov existujú aj pohyby na testovanie low skill pohybov. Ďalšie pohyby sú elementárne ako kopanie, presun hráča, postavenie hráča, vyhľadávanie lopty a otočenie hráča.

3.4.5 Analýza plánovačov (pp 3.5)

Plánovače sú dôležitou časťou fungovania Nao robota. Preto analýza existujúcich plánovačov, ich vytváranie a princíp fungovania sú dôležitou časťou, ktorú by mal poznať každý člen tímu projektu RoboCup.

3.4.5.1 Navrh

Cieľom tejto úlohy bolo vytvoriť prehľad o existujúcich plánovačoch, o fungovaní a tvorbe plánovačov. Člen tímu, zodpovedný za túto úlohu opísal ako plánovače fungujú. Následne opísal ako sa plánovače vytvárajú a na ukážku a lepšie pochopenie vytvoril jednoduchý plánovač . Následne spísal existujúce plánovače a opísal načo slúžia.

3.4.5.2 Implementácia

Popis fungovania plánovačov, popis ich tvorby a analýza starých plánovačov bol umiernený na wikipediu a do Redmine-u. K návodu vytvorenia plánovača bol vytvorení a jednoduchý príklad pre lepšie pochopenie jeho tvorby ktorý je tiež vložený na wikipedii.

3.4.6 Koncept modelu sveta (pp 3.5)

Model sveta zhromažďuje informácie, ktoré sú potrebné pri vývoji a testovaní agenta. Preto bolo potrebné vykonať analýzu dát ktoré sa tam vyskytujú aby ich bolo možné neskôr pri vývoji používať.

3.4.6.1 Návrh

Cieľom úlohy bolo zanalyzovať kód, ktorý sa týka modelu sveta. Na základe analýzy bolo potrebné spísať jednotlivé zistenia do dokumentu. Obsahom dokumentu a analýzy sú najmä dáta, ktoré sú v modeli sveta dôležité a ktoré sa tam priamo používajú. Jednotlivé dáta bolo potrebné popísať.

3.4.6.2 Implementácia

Bol vytvorený dokument, ktorý obsahuje rozbor kódu modelu sveta, kde sú definované funkcie a dáta, ktoré model sveta poskytuje. Jednotlivé dáta sú rozobrané detailnejšie a sú zakreslené v grafe dát, ktorý znázorňuje ich štruktúru. Dokument s analýzou je obsahom wiki a takisto je umiestnený ako výstup v manažérskom softvéri Redmine.

3.4.7 Testframework (pp 3.5)

Cieľom úlohy bolo zdokumentovanie fungovania testframeworku, vytváranie test cases na testovanie jednotlivých pohybov, formácii a modelových situácii sveta a možnosti nastavenia testframeworku.

3.4.7.1 Návrh

Výstup bol umiestnený na wikipedii a do manažérskeho systému. Výstup bol rozdelený do nasledujúcich častí a to Návod na vytváranie testcases, TestFramework: Spätná väzba od hráča, Nastavenie TestFrameworku.

3.4.7.2 Implementácia

Spomínané časti už bol napísané v pôvodnej wikipedii, preto boli len skontrolované či dané informácie sedia. Skontroloval sa návod na písanie testcases a vyskúšali sa už naprogramované test cases. Pôvodné návody a informácie boli začlenené do štruktúry wikipedie.

3.4.8 Testframework bugfix (pp 3.5)

Dôkladnejšou analýzou testovacieho nástroja „testframework“ sa zistilo, že predtým identifikované chyby sú neplatné, teda nie sú to chyby.

3.4.9 Návod na používanie wiki (pp 3.5)

Z dôvodu potreby práce s wiki, či už nášho tímu, alebo tímov, ktoré budú na projekte pracovať po nás bolo potrebné vytvorenie návodu na prácu s wiki.

3.4.9.1 Návrh

Na tímovom stretnutí sme sa dohodli, čo má návod obsahovať. Teda bola vytvorená kostra návodu.

3.4.9.2 Implementácia

Návod sa spísal a je napísaný priamo na wiki. Rovnako je uložený aj v manažérskom softvéri Redmine a na stránke tímu.

4 Celkový pohľad

Projekt RoboCup prebral náš tím v nevhodne zdokumentovanom stave. Preto ako tím sme si prešli rôznymi úskaliami, ktoré ovplyvnili celkový postup projektu. Na začiatku projektu bolo dôležité analyzovať existujúce riešenia z minuloročných tímov, zahraničných tímov, ale aj reálnych robotov. Analýzy sú zahrnuté v časti 3.1 tohto dokumentu a obsahujú analýzy minuloročných tímov RoboCup na FIIT STU ale i svetových tímov, ktoré sú na vysokej úrovni. Z týchto analýz vznikla potreba analyzovať možnosti zdokonalenia high skill pohybov a to napríklad cestou dynamického vytvárania chôdze. Pre kvalitný postup práce na projekte RoboCup bolo dôležité zlúčiť výhody dvoch minuloročných projektov RoboCup na FIIT STU. Boli to tímy High 5 a Tím 17. Výsledok zlúčenia výhod je na priloženom CD nosiči.

Po analyzovaní projektu v úvodných fázach sme zistili, že návody nie sú pre nás a tiež pre budúce generácie použiteľne. Preto sme sa rozhodli spísať dôkladné dokumentácie a návody na to, aby sa nové tímy vedeli lepšie zorientovať v projekte. Ako prvé sme sa rozhodli prerobiť wikipediu a vkladať do nej aktuálne dokumenty a návody pre celý projekt. Hlavnou požiadavkou pre túto úlohu bolo vytvoriť prehľadnú stránku, na ktorej sa každý ľahko zorientuje (pozri úlohu 3.8). Ďalšou častou bolo vytvorenie návodov na inštaláciu servera, spúšťanie hráča, editácia a tvorba schopností hráča. Tejto časti sme sa venovali v tretom a štvrtom šprinte (pozri kapitoly 3.3 a 3.4).

5 Plán projektu na letný semester

V letnom semestri je naplánovaných 6 šprintov. Prvé štyri budú zamerané na návrh a implementáciu nových riešení a následne dva na testovanie a integráciu projektu. Počas 4. šprintu v letnom semestri (šprint 8) prebehne prezentácia projektu na konferencii IIT.SRC. Po ukončení posledného šprintu sa uskutoční turnaj RoboCup.

Šprint	Úlohy	Obdobie
5. šprint	Návrh výšších pohybov a taktiky Špecifikácia videnia sveta Návrh testframeworku Zaučenie nových členov	18.02.2012 - 04.03.2012
6. šprint	Implementácia taktiky (vedenia na bránu) Implementácia videnia sveta (šanca získať loptu) Špecifikácia a návrh vylepšenia pohybov	04.03.2012 - 18.03.2013
7. šprint	Implementácia videnia sveta (úspešnosť prihrávky) Implementácia nových pohybov (úkroky, otočky) Implementácia highskill pohybov (prihrávanie, kopanie na bránu, prídenie k lopte, taktika)	18.03.2013 - 08.04.2013
8. šprint	Vylepšenie taktiky Dokumentácia Unit testy	08.04.2013 - 22.04.2013
9. šprint	Príprava na RoboCup turnaj Vylepšenie hráčov	22.04.2013 – 06.05.2013
10. šprint	Integrácia riešení Aktualizácia wiki Dokumentácia a príprava na odovzdanie a obhajobu RoboCup turnaj	06.05.2013 – 20.05.2013

6 Šprinty

6.1 Šprint 5 – „epsilon“

Tabuľka 6.1: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
5.1	Nový člen tímu	Získať prehľad o celom projekte	Pochopiť a zistiť, čo je alebo nie je implementované
5.2	Nový člen tímu	Získať prehľad o celom projekte	Pochopiť a zistiť, čo je alebo nie je implementované
5.3	Nový člen tímu	Funkčný server na RoboCup pre nových členov tímu	Spúšťanie hráča a testovacieho frameworku
5.4	Nový člen tímu	Funkčný server na RoboCup pre nových členov tímu	Spúšťanie hráča a testovacieho frameworku
5.5	Nový člen tímu	Vytvoriť nový pohyb hráča	Ľahšie pochopiť nasledujúci vývoj
5.6	Nový člen tímu	Vytvoriť nový pohyb hráča	Ľahšie pochopiť nasledujúci vývoj
5.7	Člen tímu	Vylepšiť nižšie pohyby	Zabezpečiť lepšiu stabilitu hráča
5.8	Člen tímu	Navrhnúť vylepšenia vyšších pohybov	Tieto vyššie pohyby realizovať v ďalšej fáze
5.9	Člen tímu	Navrhnúť hernú taktiku	Navrhnutú taktiku implementovať a využívať pri hre
5.10	Člen tímu	Analyzovať a navrhnúť videnie sveta	Využiť tieto znalosti pri ostatných vyvíjaných súčastiach
5.11	Člen tímu	Pripraviť test framework na turnaj	Využiť test framework bez problémov na turnaji
5.12	Člen tímu	Fungujúcu stránku a wiki	Naďalej pracovať s touto podporou

Tabuľka 6.2: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
5.1	-	5.1	Štúdium wiki	M. Chylik
5.2	-	5.2	Štúdium wiki	M. Škoda
5.3	-	5.3	Spustenie servera a hráča	M. Chylik
5.4	-	5.4	Spustenie servera a hráča	M. Škoda
5.5	-	5.5	Vytvorenie pohybu	M. Chylik
5.6	-	5.6	Vytvorenie pohybu	M. Škoda
5.7	-	5.7	Špecifikácia a návrh nižších pohybov	G. Nagy
5.8	-	5.8	Návrh vylepšenia vyšších pohybov	J. Grega
5.9	-	5.9	Návrh vylepšenia taktiky	M. Červeňák
5.10	-	5.10	Analýza a návrh videnia sveta	M. Gregor
5.11	-	5.11	Príprava Test Frameworku na turnaj	F. Sucháč
5.12	-	5.12	Oživenie stránky + wiki	G. Nagy

6.1.1 Špecifikácia a návrh nižších pohybov (pp 5.7)

Krátky prehľad stavu existujúcich pohybov ku dňu začatia šiesteho šprintu. Prehľad vychádza z už vykonaných analýz nižších pohybov počas zimného semestra [**Chyba! Nenašiel sa žiaden zdroj dkazov.**][**Chyba! Nenašiel sa žiaden zdroj odkazov.**].

6.1.1.1 Chôdze

Existuje veľa typov chôdzí, ktoré boli používané predchádzajúcimi tímami. Pre náš tím je potrebné aby sme mali čo najmenší počet schôdzí a tie potrebujú byť spoľahlivé. Chôdzu dopredu je potrebné vylepšiť. Chôdze smerom dozadu sú stabilné a rovné. Keďže prakticky hráči sa veľmi málo pohybujú smerom dozadu, pri tomto type chôdze sa uprednostňuje stabilita oproti rýchlosti.

Vhodné chôdze na vylepšenie a ďalšiu prácu:

- walk_fast_fine2_optimized2acc
- walkback3

Kritériá pre nové chôdze:

1. Rýchlejšia a stabilnejšia chôdza dopredu
2. Chôdza dopredu a chôdze s otáčaním rovnakého typu

6.1.1.1.1 Návrh vylepšenia chôdze

Potrebné je vytvoriť jednu rýchlu a stabilnú chôdzu, z ktorej sa odvedie aj chôdza s otáčaním. Keď rovná chôdza a chôdza s otáčaním budú rovnakého typu, hráč bude schopný prepínať medzi nimi bez zastavenia a zmeny štýlu chôdze vďaka čomu dosiahne vyššiu stabilitu.

Návrh výstupných chôdzí:

- walk_fast_fine2_optimized3
- walk_turnR
- walk_turnL

6.1.1.2 Kopnutia

Existujúce kopnutia sú vhodné na používanie. Základné kopnutia na ďalšiu prácu:

- kick_left_normal
- kick_right_normal
- kick_straight_edge_l
- kick_straight_edge_r

Kritériá pre väčšiu flexibilitu:

1. silnejšie a slabšie kopnutia
2. kopnutie na prihrávanie

6.1.1.2.1 Návrh vylepšenia kopnutia

Z dôvodu, že sú veľmi stabilné na ďalšiu prácu si zvolíme: *kick_left_normal* a *kick_right_normal,* ktoré sú najstabilnejšie a najrýchlejšie. Na vytvorenie použiteľných prihrávok budeme taktiež vychádzať z už existujúcich pohybov.

Návrh výstupných kopnutí:

- kick_left_normal_optimized
- kick_right_normal_optimized
- kick_left_strong
- kick_right_strong
- pass_right_1 - dlhý cca. ¼ ihriska
- pass_right_2 - dlhý cca. pol ihriska
- pass_right_3 - dlhý cca. ¾ ihriska
- pass_left_1
- pass_left_2
- pass_left_3

6.1.1.3 Otáčania

Otáčania sú pohyby, pri ktorých je prioritou najmä presnosť. Po preverení údajných a reálnych stupňov otáčania už existujúcich pohybov je potrebné:

1. Premenovať otáčania na základe reálneho stupňa otáčania
2. Doplniť chýbajúce stupne otáčania

Základné otáčania na ďalšiu prácu:

- turn_left_cont_5_faster_2
- turn_right_cont_5_faster_2
- turn_left_cont_20_faster_2
- turn_right_cont_20_faster_2
- turnleft45_2
- turnright45_2
- turnleft90_faster_2
- turnright90_faster_2

6.1.1.3.1 Návrh vylepšenia otáčaní

Otáčania majú byť doplnené o ďalšie, aby bol širší výber v rozsahu otáčania.

Rozstup navrhovaných otáčaní: 5°, 10°, 20°, 30°, 45°, 60°, 90°

Návrh výstupných otáčaní:

- turn_left_cont_5

- turn_right_cont_5
- turn_left_cont_10
- turn_right_cont_10
- turn_left_cont_20
- turn_right_cont_20
- turnleft30
- turnright30
- turnleft45_2
- turnright45_2
- turnleft60_2
- turnright60_2
- turnleft90_faster_3
- turnright90_faster_3

6.1.1.4 Úkroky

Existujúce úkroky sú stabilné a presné. Je potrebné doplniť existujúce úkroky s úkrokmi po kružnici.

Úkroky vhodné na ďalšiu prácu:

- stepleft1
- stepright1

Kritériá pre nové úkroky:

1. Zvýšenie rýchlosti
2. Vytvorenie úkrokov po kružnici

6.1.1.4.1 Návrh vylepšenia úkrokov

Úkroky po kružnici budú vychádzať z rovných úkrokov, aby typ pohybu bol rovnaký a prechod medzi nimi bol stabilný.

Návrh nových úkrokov:

- stepleft1_fast
- stepright1_fast
- circleStep_left
- circleStep_right

6.1.1.5 Pády

Pády sú vhodné na používanie.

6.1.1.6 Sadanie a vstávanie

Pohyby sú vhodné na používanie.

6.1.1.7 Implementácia

V rámci úlohy sa implementovali otáčania o rôzne stupne, ostatná časť implementácie sa presunula do ďalšieho šprintu.

Otáčania sa implementovali na základe uvedeného návrhu.

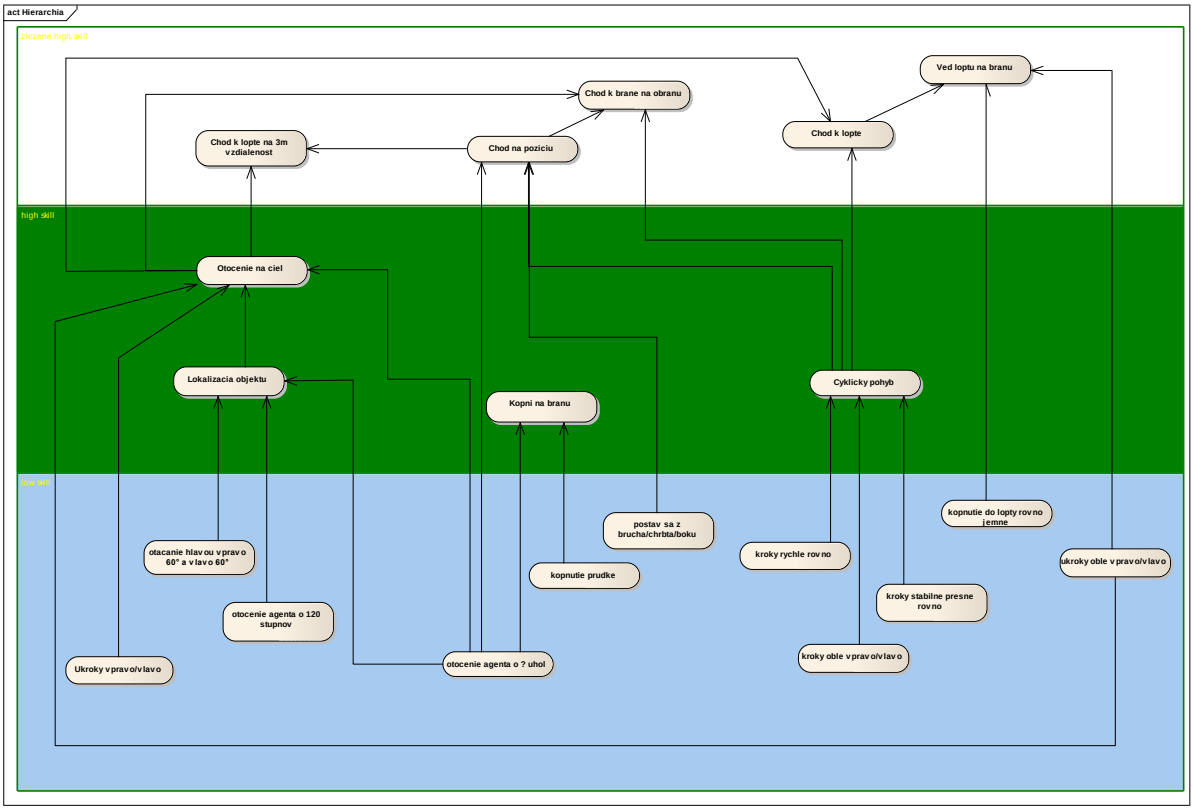
Rozsah novovytvorených: 4.5°, 6.5°, 10°, 20°, 30°, 45°, 60°, 90°

6.1.2 Návrh vylepšenia vyšších pohybov (pp 5.8)

Cieľom tejto úlohy bolo preštudovať existujúce vyššie pohyby a navrhnúť ich možné vylepšenie.

6.1.2.1 Návrh

Pri návrhu vyšších pohybov sme vychádzali z analýzy vyšších pohybov, ktoré boli vykonané v šprinte 4 (3.4.4 Analýza High-level pohybov). Po naštudovaní všetkých existujúcich vyšších pohybov a vytvorení predbežnej taktiky sme spísali zoznam potrebných vyšších pohybov a ďalej sa zaoberali len nimi. Pre lepšiu orientáciu sme vytvorili hierarchiu pohybov (Obrázok 6.1). Zistili sme, že základom pre kvalitné vyššie pohyby bude vytvorenie stabilných nižších pohybov.



Obrázok 6.1 Diagram aktivít: Hierarchia pohybov

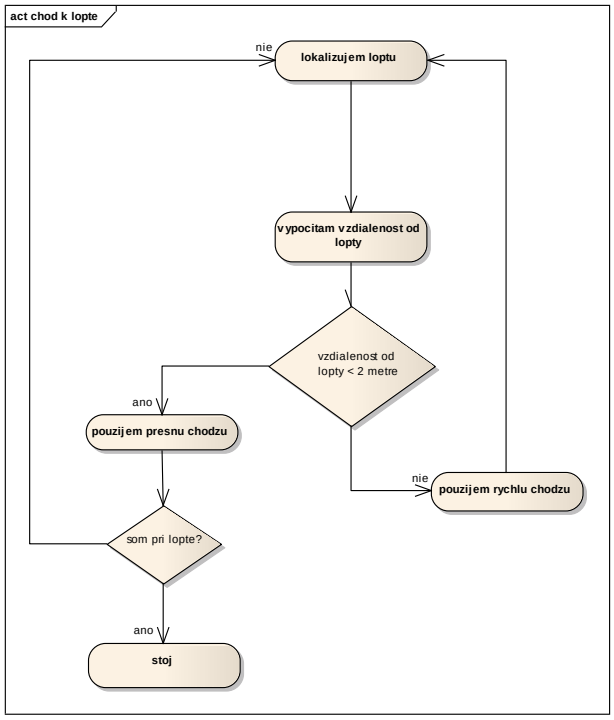
6.1.2.2 Implementácia

Po spísaní potrebných pohybov sme začali tvoriť diagramy, ktoré popisujú vykonávanie vyšších pohybov. Každý pohyb ma svoj diagram a k nemu vysvetlenie jednotlivých aktivít.

6.1.2.2.1 Choď k lopte

Lokalizujem loptu – HS lokalizácia objektu(lopty)
Vypočítam vzdialenosť od lopty – potrebne informácie z modelu sveta (presná pozícia lopty)
Použijem presnú chôdzu – LS presná chôdza na priblíženie sa k objektu na veľmi malú vzdialenosť (0,1 m od lopty) - cyklické vykonávanie presných stabilných krokov
Použijem rýchlu chôdzu – LS rýchla chôdza nemusí byť veľmi presná, pomocou nej je potrebne sa dostať čím skôr na požadovanú vzdialenosť od objektu - cyklické vykonanie krokov, cyklus tvorí X krokov

Postup krokov vykonávania pohybu je znázornený na obrázku 6.2.

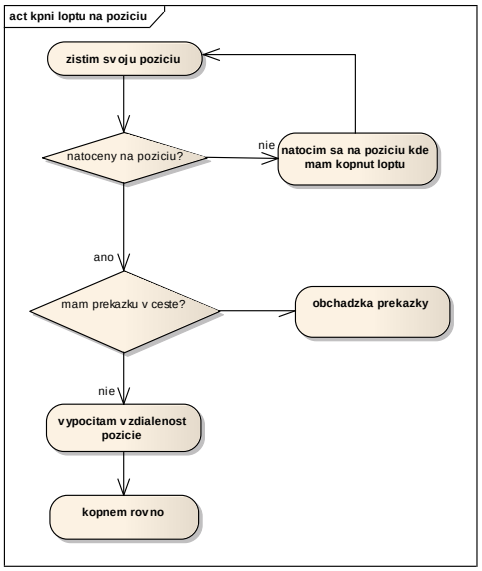


Obrázok 6.2 Diagram aktivít: Choď k lopte

6.1.2.2.2 Kopni loptu na pozíciu

Zistím svoju pozíciu – info z modelu sveta
Natočím sa na pozíciu, kde mám kopnúť loptu – zistím o koľko sa mám natočiť a natočím sa LS - otočenie o požadovaný uhol tolerancia 5°
Obchádzka prekážky – HS obchádzka prekážky
Vypočítam vzdialenosť pozície – výpočet vzdialenosti medzi mnou a pozíciou, závisí na tom sila kopu
Kopnem rovno – LS kopni (podľa vzdialenosti)

Postup vykonávania krokov pohybu je na opísaný na obrázku 6.3.

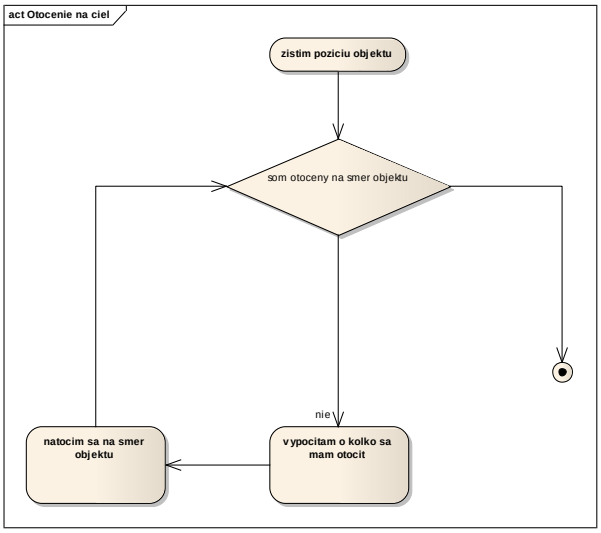


Obrázok 6.3 Diagram aktivít: Kopni loptu na pozíciu

6.1.2.2.3 Otočenie na cieľ

Zistím svoju pozíciu – info z modelu sveta
Vypočítam o koľko sa mam otočiť – zistím o koľko sa mám natočiť
Natočím sa na smer objektu - natočím sa LS - otočenie o požadovaný uhol tolerancia 5°

Postup vykonávania vyššieho pohybu je znázornený na obrázku 6.4.



Obrázok 6.4 Diagram aktivít: Otočenie na cieľ

6.1.2.2.4 Choď k bráne na obranu

Tento vyšší pohyb je zhodný s pohybom chod na pozíciu. Pohyb je opísaný v kapitole 6.1.2.2.6.

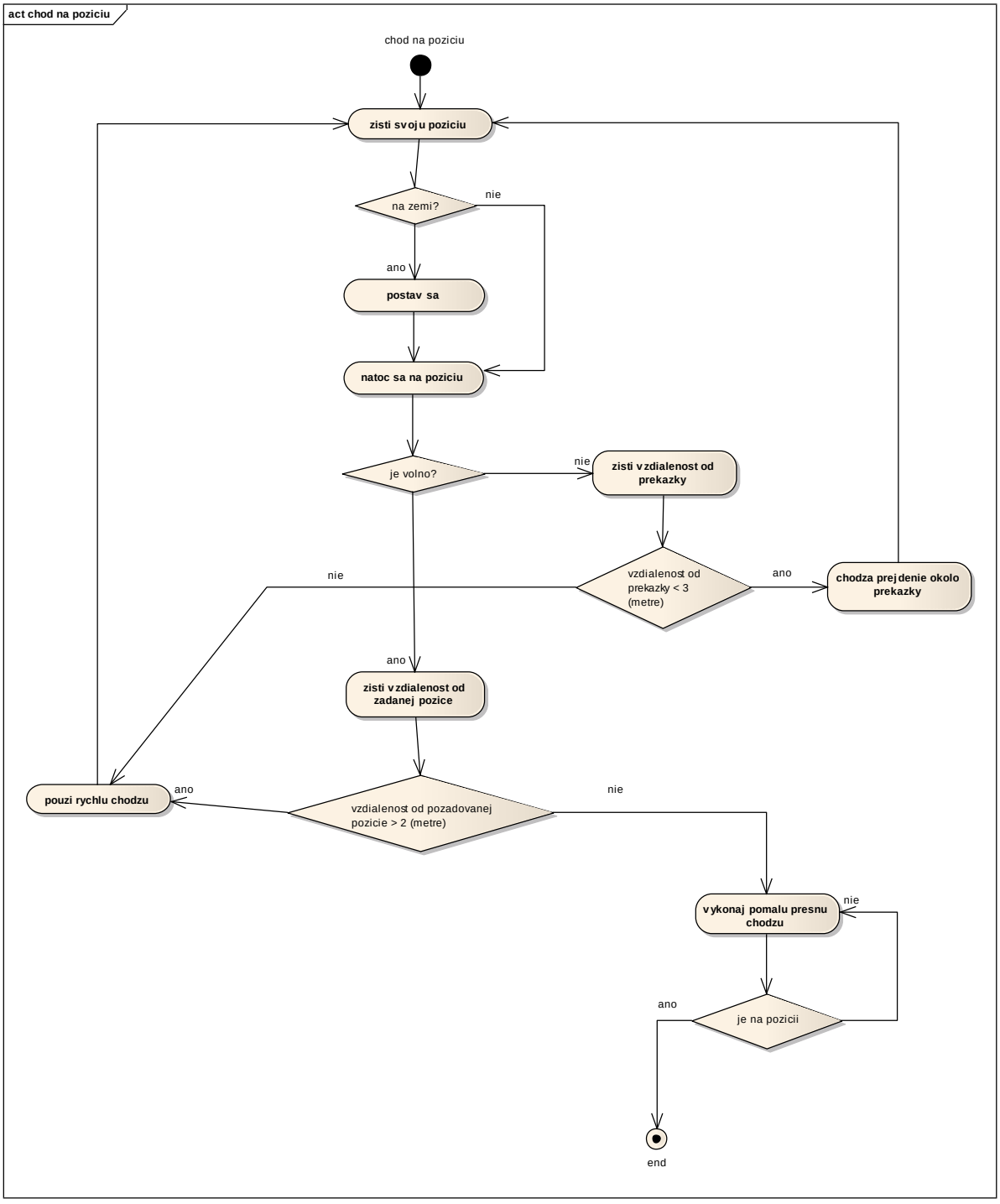
6.1.2.2.5 Choď k lopte na 3m vzdialenosť

Tento pohyb je zhodný s pohybom choď na pozíciu, ktorý je opísaný v kapitole 6.1.2.2.6.

6.1.2.2.6 Choď na pozíciu

Zistí svoju pozíciu – info z modelu sveta
Postav sa – LS na postavenie sa z brucha/chrbta
Natoč sa na pozíciu – LS otočenie celým agentom o požadovaný uhol
Zistí vzdialenosť od prekážky – info z modelu sveta
Chôdza prejdenie okolo prekážky – LS obchádzka pomocou chôdze po kružnici cyklická chôdza vyskladaná z krokov ktoré tvoria kružnicu so zadaným polomerom
Zisti vzdialenosť od zadanej pozície – info z modelu sveta vypočíta rozdiel medzi mojou a požadovanou pozíciou
Použi rýchlu chôdzu - LS rýchla chôdza nemusí byť veľmi presná, pomocou nej je potrebne sa dostať čím skôr na požadovanú vzdialenosť od objektu - cyklická chôdza vyskladaná z krokov po priamke
Vykonaj pomalú presnú chôdzu - LS presná chôdza na dostavenie sa na požadovanú pozíciu- cyklická chôdza vyskladaná z krokov po priamke
Je volno? – zisti, čí ma pred sebou prekážku

Postup vykonávania jednotlivých krokov vyššieho pohybu je zobrazený na obrázku 6.5.



Obrázok 6.5 Diagram aktivít: Choď na pozíciu

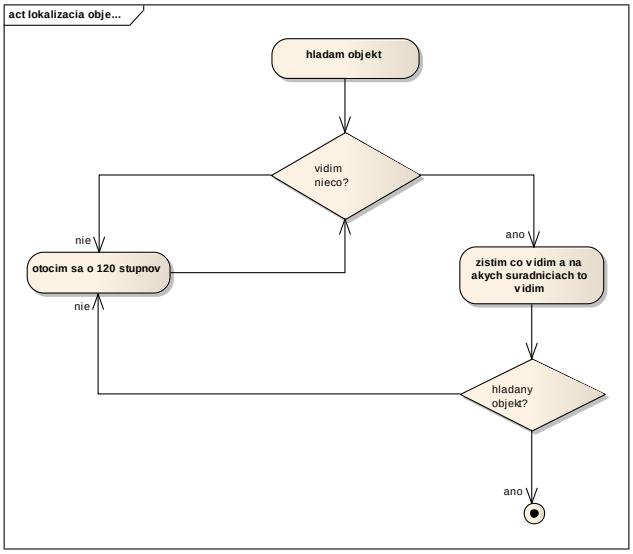
6.1.2.2.7 Kopni na bránu

Tento vyšší pohyb je zhodný s pohybom kopni loptu na pozíciu, ktorý je opísaný v kapitole 6.1.2.2.2.

6.1.2.2.8 Lokalizácia objektu

Hľadám objekt – LS - otáčam hlavou vpravo a vľavo
Otočím sa o 120 stupňov – LS - otočím celého agenta o 120 stupňov (otočenie na mieste)
Natočím sa na smer objektu – otočím celým agentom o požadovaný uhol
Zistím čo vidím a na akých súradniciach to vidím – informácie z modelu sveta
Vypočítam o koľko sa mam otočiť – výpočet uhla o ktorý sa potrebujem otočiť
Som otočený na smer objektu? – info z modelu sveta, tolerancia ± 5°

Postup vykonávania krokov pohybu je znázornený na obrázku 6.6.



Obrázok 6.6 Diagram aktivít: Lokalizácia objektu

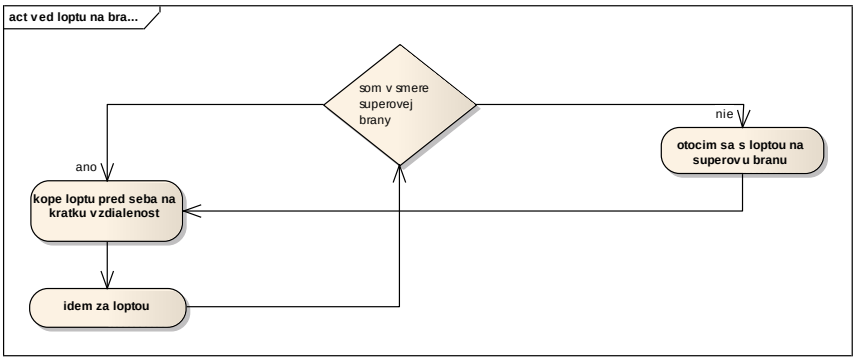
6.1.2.2.9 Prihraj loptu

Pohyb prihraj loptu je zhodný s pohybom kopni loptu na pozíciu, ktorý je opísaný v kapitole 6.1.2.2.2.

6.1.2.2.10 Veď loptu na bránu

Kopnem loptu na krátku vzdialenosť – LS - kopnutie do lopty rovno (malou silou – prekopávanie na tri, štyri kroky)
Idem za loptou – HS chod k lopte
Otočím sa s loptou na súperovu bránu – agent sa otočí okolo lopty tak aby bol nasmerovaný do stredu súperovej brány LS – úkroky vpravo/vľavo po kružnici s polomerom lopty

Nižšie zobrazený diagram (Obrázok 6.7) popisuje postup vykonávania pohybu.



Obrázok 6.7 Diagram aktivít: Veď loptu na bránu

6.1.3 Návrh vylepšenia taktiky (pp 5.9)

Pre hru viacerých hráčov bolo potrebné vyvinúť hernú taktiku kde má každý hráč určenú svoju úlohu.

6.1.3.1 Analýza

Cieľom úlohy bolo vytvorenie taktiky, podľa ktorej by bolo možné spravovať viacero tímových hráčov na ihrisku a ktorá bude fungovať. Do teraz nebola vytvorená taktika, ktorá by naozaj fungovala a dokázala pomocou tímovej hry dostať loptu do brány.

Pri hre nastávajú 3 hlavné situácie, ktoré treba ošetrovať samostatne a sú to situácie keď:

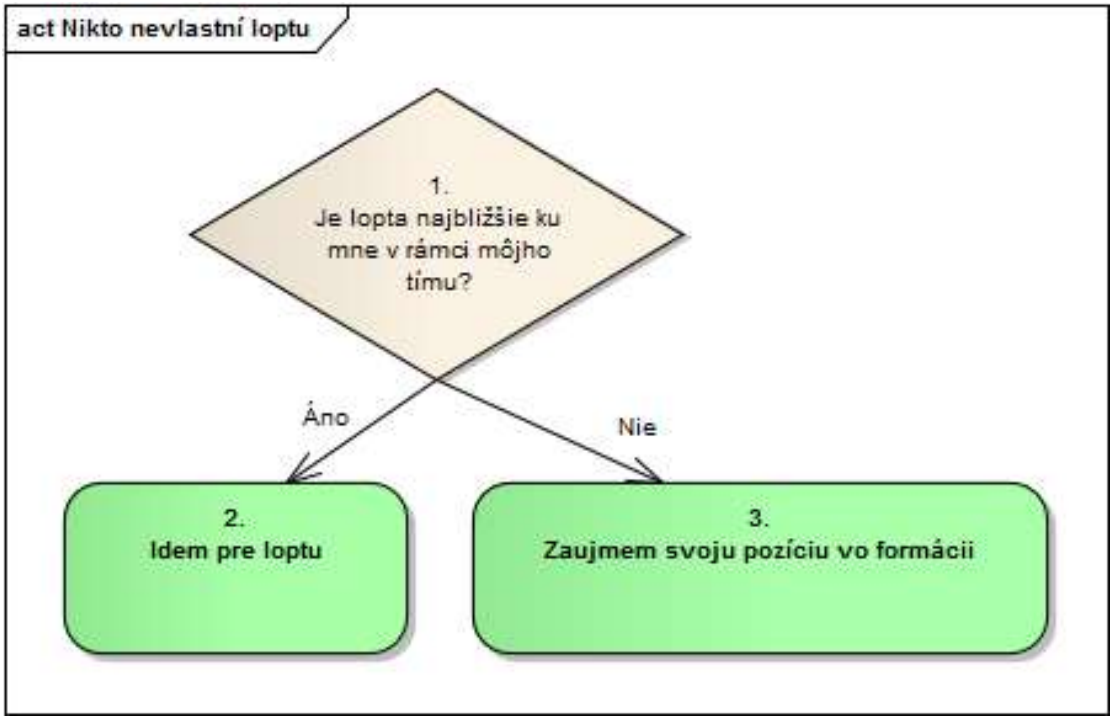
- loptu nevlastní nikto z hráčov na ihrisku
- loptu vlastní niektorí z protivníkov
- loptu vlastní niektorí z nášho tímu

6.1.3.2 Návrh

Spočiatku bol návrh vytvorený len pre jedného hráča. Nedokázal teda ošetrovať tímovú hru aj keď išlo o tímovú taktiku. Následne sa do návrhu zakomponovali i formácie, ktoré boli vytvorené predchádzajúcim tímom. Našim tímom boli otestované a doplnené o správnu funkcionalitu. To ale nebolo náplňou tejto úlohy. Formácie napomohli tomu, aby mohli byť riešené prípady viacerých hráčov na ihrisku podobne, ako tomu v skutočnosti na ihrisku pri hre futbalu je.

Pri návrhu bolo treba brať do úvahy teda 3 situácie popísané vyššie v analýze. Pre jednotlivé situácie vznikli 3 diagramy aktivít. Ak by sa diagram spojil do jedného, predstavoval by celú hernú taktiku. Rozdelený na 3 časti sa diagram zaoberá každým prípadom hernej situácie samostatne. Jednotlivé obrázky diagramov aj s opisom sú popísané ďalej.

6.1.3.2.1 Nikto nevlastní loptu



Obrázok 6.8 Diagram aktivít: Nikto nevlastní loptu

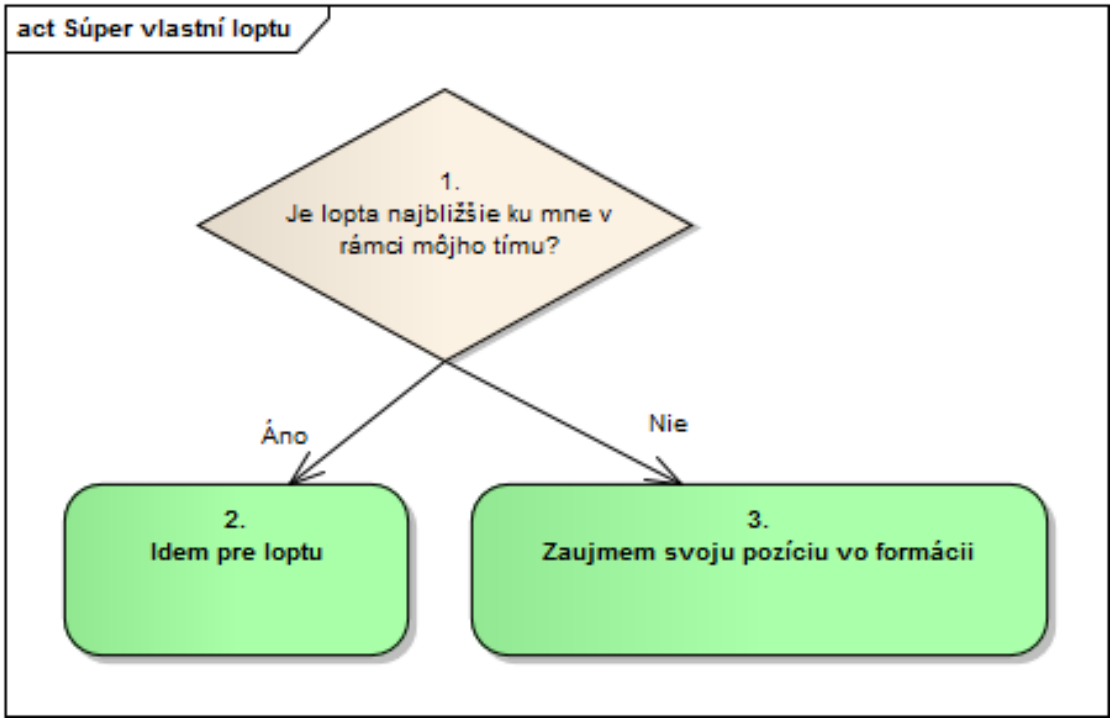
Prvým prípadom, ktorý na ihrisku nastáva je situácie, keď žiadny hráč, či z nášho alebo zo súperovho tímu nevlastní loptu. Postup ako prebieha takáto taktika je znázornený na Obrázku 6.8 a je popísaný nasledovne:

1: Výpočet, ktorý pre každého tímu vypočíta, či je najbližšie k lopte v rámci svojho tímu.
Pre tento účel bola použitá metóda *bIsBallNearestToMeInMyTeam()*, ktorá je súčasťou triedy *TacticalInfo*.

2: Ak je hráč najbližšie k lopte tak ide preloptu.
Momentálne použitý high skill je w*alk2Ball.rb*

3: Ak hráč nie je najbližšie k lopte, ide na svoju pozíciu vo formácii. Pozície formácie sú určené na základe čísel hráča. Každé číslo hráča má pridelenú pozíciu vo formácii. Formácia sa posúva postupne vzhľadom na posúvanie lopty
Pre tento účel sa volá trieda *FormationHelper* so statickou metódou *getHighSkillToGoToFormation,* ktorá vracia high skill *GoToPosition.rb*. Tento high skill posiela hráča na jeho danú pozíciu vo formácii.

6.1.3.2.2 Súper vlastní loptu



Obrázok 6.9 Diagram aktivít: Súper vlastní loptu

Druhým prípadom, ktorý na ihrisku nastáva je situácie, keď vlastní loptu súper, teda niektorí zo súperových hráčov. Postup ako prebieha takáto taktika je znázornený na Obrázku 6.9 a je popísaný nasledovne:

1: Výpočet, ktorý pre každého tímu vypočíta, či je najbližšie k lopte v rámci svojho tímu.

Pre tento účel bola použitá metóda *bIsBallNearestToMeInMyTeam()*, ktorá je súčasťou triedy *TacticalInfo*.

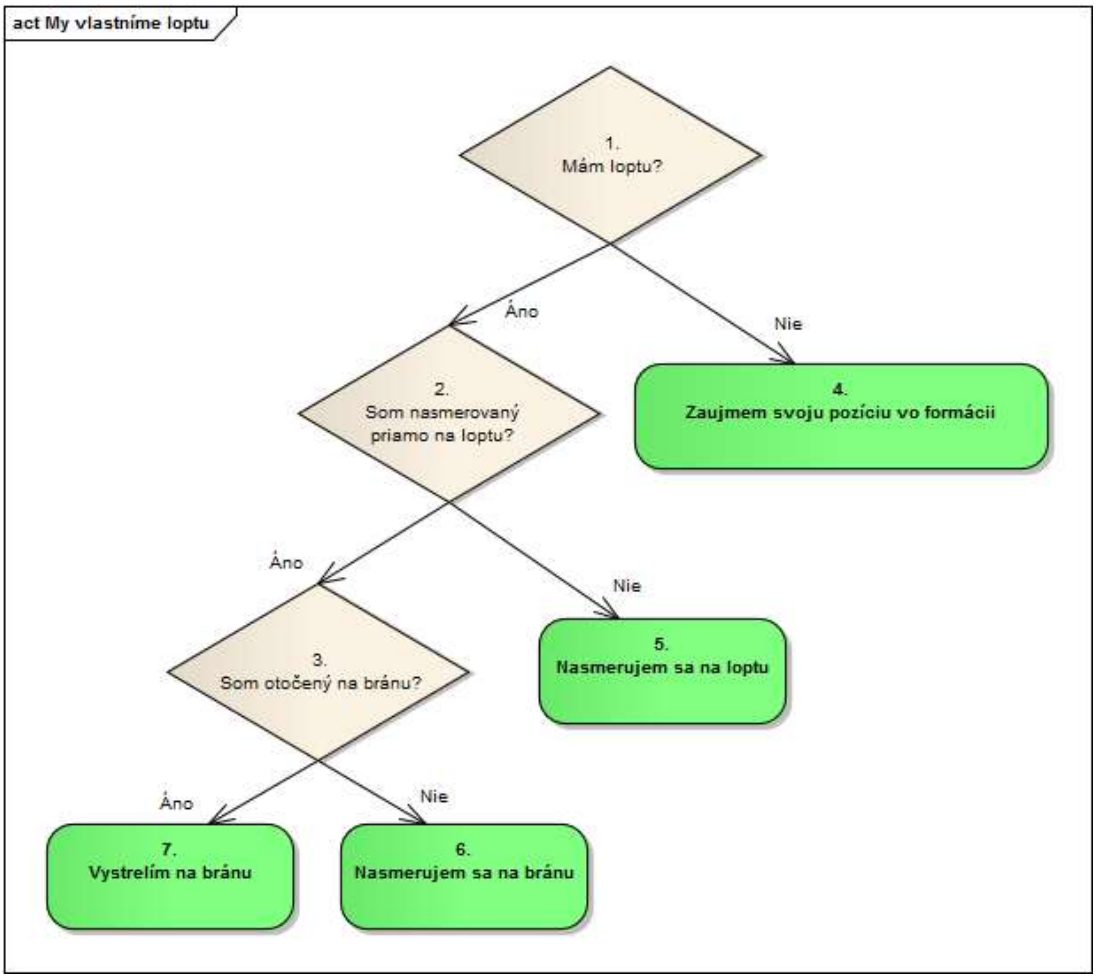
2: Ak je hráč najbližšie k lopte tak ide preloptu.

Momentálne použitý high skill je w*alk2Ball.rb*

3: Ak hráč nie je najbližšie k lopte, ide na svoju pozíciu vo formácii. Pozície formácie sú určené na základe čísel hráča. Každé číslo hráča má pridelenú pozíciu vo formácii. Formácia sa posúva postupne vzhľadom na posúvanie lopty

Pre tento účel sa volá trieda *FormationHelper* so statickou metódou *getHighSkillToGoToFormation,* ktorá vracia high skill *GoToPosition.rb*. Tento high skill posiela hráča na jeho danú pozíciu vo formácii.

6.1.3.2.3 My vlastníme loptu



Obrázok 6.10 Diagram aktivít: My vlastníme loptu

Tretím prípadom, ktorý na ihrisku nastáva je situácie, keď vlastní súper náš tím, teda niektorí z hráčov nášho tímu. Postup ako prebieha takáto taktika je znázornený na Obrázku 6.10 a je popísaný nasledovne:

1: Zistí, či mám loptu.
Použitá metóda je *isBallMine()* , ktorá je súčasťou triedy *AgentInfo*.

2: Zistí, či som nasmerovaný priamo na loptu, resp. mám loptu priamo pred sebou.
3: Zistí, či som nasmerovaný priamo na súperovu bránu.
4: Ak hráč nemá loptu, ide na svoju pozíciu vo formácii. Pozície formácie sú určené na základe čísel hráča. Každé číslo hráča má pridelenú pozíciu vo formácii. Formácia sa posúva postupne vzhľadom na posúvanie lopty
Pre tento účel sa volá trieda *FormationHelper* so statickou metódou *getHighSkillToGoToFormation,* ktorá vracia high skill *GoToPosition.rb*. Tento high skill posiela hráča na jeho danú pozíciu vo formácii.

5: Ak hráč nie je nasmerovaný priamo na loptu, tak nasmerovanie vykoná.
Použitý high skill je *walk2Ball.rb*

6: Ak hráč nie je natočený priamo na bránu, tak natočenie vykoná.
Použitý high skill je *turn.rb*

7: Ak má hráč loptu, je nasmerovaný na loptu resp. má loptu pred sebou a je natočený na bránu tak vystrelí.
Použitý high skill je *kick.rb*

6.1.4 Analýza a návrh videnia sveta (pp 5.10)

Tím potreboval na vylepšenie hráča analyzovať a vylepšiť videnie sveta.

6.1.4.1 Aká funkcionalita videnia sveta funguje

Natočenie agenta - AgentRotationCalculator -> calculateRotation , ale je potrebné volať pravidelne updateRotation zo získaných údajov zo servera, čo sa vykonáva v registrovaných observerov v skripte scripts/config/dependencies.rb. Údaje musia obsahovať statické objekty ako vlajky (minimálne 3 kusy).

Pozícia agenta – v agent.position v AgentModeli. Vypočíta to

AgentPositionCalculator.updatePosition

z údajov zo servera. Údaje musia obsahovať minimálne videné vlajky ako statické objekty (s hodnotou absolutnej pozície).

Pozicia dynamickeho objektu – v triede DynamickObject je dokoncene sledovanie lopty, ale treba dokončiť sledovanie hráčov. Vieme zistiť pozíciu dynamického objektu metódou getPosition(), vieme tiež získať relatívnu pozíciu vzhľadom na čas videnia objektu. Je daný minimálny čas potrebný na zachytenie dynamického objektu v konštante LOW_TIME. Je tam pokus o predikciu pozície na zaklade času kedy bol dynamický objekt naposledy videný a jeho rýchlosti. Ale nie je to dokončené a presné. Možné vylepšenie predikcie.

Údaje prostredia – sú uchovávané v EnvironmentModel. Obsahuje údaje ako čas hry, čas simulácie, hrací mód (BEFORE_KICK_OFF, PLAY_ON, KICK_OFF_LEFT, KICK_OFF_RIGHT, KICK_IN_LEFT, KICK_IN_RIGHT, CORNER_KICK_LEFT, CORNER_KICK_RIGHT, GOAL_KICK_LEFT, GOAL_KICK_RIGHT, OFFSIDE_LEFT, OFFSIDE_RIGHT, GAME_OVER, GOAL_LEFT, GOAL_RIGHT, FREE_KICK_LEFT, FREE_KICK_RIGHT), časový krok, ktorým je čas zvyšovaný, stav hry (BEFORE_KICK_OFF, GAME_OVER, GOAL_LEFT, GOAL_RIGHT, KICK_OFF_LEFT, KICK_OFF_RIGHT), verzia servera (podpora pre poslednu verziu pridaná 0_6_5)

Pozície statických objektov – su pozície vlajok. Pre každú verziu servera rozdielna implementácia. Tiež viem získať pozíciu bránok (našej a súpera).

Taktické informácie o herných situáciách – TacticalInfo

- isOffensiveSituation
- isDefSituation
- isBallOurs
- isBallTheir
- isStartAttackSituation
- isEnemyStartAttack
- positionInformation – ostatných
- pos
- relPos

Tiež existujú metódy na zistenie či je lopta najbližšie ku mne. Zistíme aj, že je lopta najbližšie ku mne z môjho tímu. Existuje metóda, ktorá zistí či náš tím zakladá útok a či je pod tlakom (podľa vzdialenosti hráčov od súperov), alebo za akú dobu ho založí. Tiež existuje metóda, ktorá zisťuje či súperov tím zakladá útok a či je pod tlakom. Máme aj metódu, ktorá zisťuje stav hry či útočíme alebo či bránime podľa toho či je nepriateľ na našej polovici a je náš hráč na našej polovici. Vieme zistiť počet našich obrancov. Je to ale dosť nepresné lebo to berie len na základe pozície našich hráčov na osi X oproti pozícii súperov. Vieme zistiť počet našich aj súperových útočníkov. Vieme zistiť či je lopta naša na základe vzdialenosti lopty od hráča (<1). Obdobne vieme zistiť či loptu má súper. Formácie – máme metódy, ktoré nastavujú pozície hráča vo formácii vzhľadom na pozíciu lopty. Je to implementované ako stavový automat. Určuje pozíciu pevne na ID hráča. Vylepšiť to vieme tak že budeme poziciu vo formácii určovať podľa vzdialenosti k bodom vo formácii. Ďalej brať do úvahy vzdielenosť od lopty (niekto zakričí, kto ide po lopte). Formácia je dynamická pri útoku a sekvenčná pri obrane, čiže musíme brať do úvahy čas a v čase meniacu sa pozíciu formácie.

Prezentovanie agentov ako hráčov na ihrisku – Player je trieda s vlastnosťami hráčov. Vlastnosti vieme čítať a zapisovať. Vieme si vypočítať vzdialenosť lopty od bodu, lopty od hráča. Vzdialnosť nôh od seba a uhol. Tiež vieme zistiť či je agent v dosahu iného hráča. Vieme nastaviť chodidlá , ruky a hlavu, vypočítať pozíciu a uhol chodidiel vzhľadom na X os ihriska.

Model sveta – vo WorldModel dokončiť autogenerovanie logovacieho suboru vo fixtures. Drží a updatuje informácie o našich a superových hráčoch. Vie a zisťuje informáciu o pozícii lopty a robí aj predikciu lopty na základe už zistených anotácii (nie je to dokonalé). Vieme kontrolovať pozíciu hráča na dostrel do brány. Stavový automat počíta na základe vzdialenosti hráča od súperovej brány. Určuje to pre nášho aj súperového hráča.

Model agenta – AgentModel – objekt triedy uchováva uhol každého kĺbu hráča. Vieme aj získať pozíciu hráča (globálnu a relatívnu). Spracovať správu zo servera a podľa toho prisposobovať kĺby, natočenia,… v objekte. V agent modeli sú aj matematické funkcie na prácu s maticami a vektormi (opisujúcimi agenta). Vieme tiež získať rýchlosť a zrýchlenie hráča. Vieme zistiť či agent stojí alebo je na zemi, či agent leží na bruchu alebo na chrbte. Vieme zistiť kedy videl agent naposledy vlajku. Pomocou anotácii vieme upraviť objekt sučasného agenta.

6.1.4.1.1 Aké metódy treba vytvoriť alebo čo treba vylepšiť

Odhadové videnia sveta

- šanca získať loptu našim hráčom = dotknuť sa jej
- kto rýchlejšie kopne (som chrbtom, na zemi) – posielanie správ s časmi kopov, odhadnem vzdialenosť a svoju rýchlosť = to isté ako predchádzajúce, len špec. prípad
- som brankár, obranca, stred, útočník = určovanie pozícií
- prekážka? z matematického hľadiska
- diery medzi prekážkami k súperovej bráne ak mám loptu
- ak mám loptu a diery medzi prekážkami, kde je najbližší hráč = môžem prihrať?
- bezpečný bod na priamke súpera k bránke = kde najlepšie bránim
- Čo treba dokončiť
- je potrebné dokončiť sledovanie hráčov a držanie si informácie v dynamickom objekte
- možné vylepšenie je pomocou predikcie zisťovanie pozície lopty

6.1.4.1.2 Rozdelenie údajov sveta

Dynamické objekty – možné vylepšenie dokončením predikcie pozície lopty na základe získaných anotácii alebo rozšírením zachytávania do dynamického objektu aj hráčov. Lopta a hráč sa v údajoch nelíšia v pozícii, zrýchlení, rýchlosti a času posledné videnia, ale hráč by mohol mať navyše údaje ako id a team.

Statické objekty – pozície vlajok alebo bránok.

Agent – objekt obsahuje informácie o mne. Moje kĺby, pozícia (absolútna alebo relatívna), moja poloha, rýchlosť, zrýchlenie

Hráč – objekt reprezentujúci hráčov na ihrisku. Obsahuje informácie potrebné iba o hráčoch a to je ich pozícia, uhly nôh, rúk a hlavy a či je hráč na dosah iného hráča alebo vzdialenosť lopty od hráča. Je to prezentácia dynamického objektu hráča, ale chýba zrýchlenie, rýchlosť a možnosť predikcie.

Prostredie – objekt reprezentujúci stav hry, mód hry, čas hry, informácie o hráčoch na ihrisku a o lopte a to pozície. Nedokončená a nepresná predikcia pozícia hráčov a lopty.

Taktické informácie – objekt reprezentujúci informácie o stave tímu či útočí alebo obraňuje, aká je formácia a kde je moje miesto vo formácii. Sú to dôležité informácie pre vyššie pohyby. Obsahuje tiež informácie o počte útočníkov a obrancov na oboch stranách ihriska alebo či je lopta naša. Formácia sa udáva vzhľadom na pozíciu lopty či je na našom alebo na súperovom ihrisku a či loptu vlastníme my alebo súper.

6.1.4.2 Návrh vylepšení

Tím navrhol 2 vylepšenie, ktoré chcel implementovať v najbližších šprintoch.

6.1.4.2.1 Vylepšenie taktiky pridelovaním najbližšej pozície vo formácii

Pozícia vo formácii sa hráčovi prideľuje na základe jeho id. To môže byť nevhodné pokiaľ hráč bude z nejakých dôvodov na inom konci ihriska a daná pozícia môže byť obsadená bližším hráčom skôr.

Preto sme navrhli vylepšiť algoritmus počítaním si vzdialeností k jednotlivým pozíciam, výberom najbližšej voľnej pozície, zahlásením obsadenej pozície a príchodom do pozície.

Nasledovný pseudo algorirtmus využíva už existujúcu funkcionalitu. Základná časť algoritmu je nasledujúca:

```
foreach formPosition in positions
        dist[][0] = formPosition.vector – my.position.vector;
        dist[][1] = formPosition.id
end

dist.sort();
have_formPosition = falsel;
while ( !have_formPostion)
        myPosition = dist[0];
        if ( !positioHold(dist[0][0]) )
                Communication.transmit(„(say Position: „ + dist[0][1] + “ taken)“);
                have_formPosition = true;
        else
                dist[0].pop();
        end
end

positionHold( positionId )
        foreach position in holdPositions
                if ( positionId == position.id)
                        return true;
                else
                        return false;
                end
        end
end
```

Návrh neprešiel lebo mohol spôsobovať chaotické presúvanie hráčov na ihrisku.

6.1.4.2.2 Šanca získať loptu našim hráčom

Keď loptu nevlastní žiaden hráč, je dôležité si určiť kto loptu získa skôr. Či to bude náš tím alebo súperov. Preto sme navrhli algoritmus kde hráč zistuje poziciu hráča najbližšieho k lopte vrátane seba. Počíta to z pozícii hráčov ktorých vidí a z pozície lopty, ktorú vidí. Táto informácia pomôže hráčovi sa následne rozhodnúť či isť po loptu alebo sa pripraviť na bránenie súperovmu hráčovi. Algoritmus berie do úvahy aj či je hráč otočený chrbtom k lopte. Štruktúra algorimtu je popísaná nasledovným pseudokódom:

```
ballPos = WorldModel.getInstance().getBall().getRealtivePosition();
min = infinity;
foreach player in players
        if ( player is turned ) player.relPosition += turnDdistance;
        if ( palyer.relPosition – ballPos < min )
                min = player.relPosition;
                playerId = player.id;
```

```
                playerTeam = player.team;
        end
        if ( playerTeam == ours ) return 100%;
        else return 0%;
end
```

Návrh bol prijatý a upravený. Implementácia je popísaná nižši v metóde chanceToGetBall().

6.1.5 Príprava Test Frameworku na turnaj (pp 5.11)

Pre potreby turnaja sme potrebovali pripraviť Test Framework, aby počas turnaja nevznikali problémy.

6.1.5.1 Analýza

Pri analyzovaní tohto problému sme si overili momentálny stav v akom sa nachádza Test Framework vzhľadom na pripravenosť vyhodnocovať turnaj. Turnaj predstavuje akciu „RoboCup at FIIT“, ktorá pozostáva z viacerých presne zadefinovaných disciplín. Pravidlá pre jednotlivé disciplíny, zhľadom na ktoré sme analyzovali a pripravovali Test Framework pre turnaj je možné nájsť na stránke http://www2.fiit.stuba.sk/robocup/2012/turnaj_pravidla_3D.htm.

Po spustení meraní jednotlivých disciplín v Test Frameworku (disciplíny sú v ňom nazvané ako „test case“), skúšaní rôznych situácií ako splniť alebo nesplniť disciplíny, sledovaní vyhodnotení disciplín a porovnaní tohto všetkého s pravidlami sme usúdili, že momentálny stav Test Frameworku nebol vyhovujúci pre použitie na oficiálnom turnaji. Dôvodom boli chaotické a nejasné výpisy, chybné vyhodnocovanie niektorých disciplín a viacero neošetrených prípadov, ktoré mohli nastať a s ktorými Test Framework nepočítal.

6.1.5.2 Implementácia

V ďalších podkapitolách uvádzame jednotlivé disciplíny a zmeny, ktoré sme v nich vykonali.

6.1.5.2.1 Vstávanie

Pri disciplíne vstávanie má hráč v pravidlách udanú počiatočnú pozíciu, na ktorej má disciplínu vykonávať. Táto pozícia (-3; 0) bola nastavená tak, aby sa pri inicializácii disciplíny na ňu hráč teleportoval.

Podľa pravidiel disciplíny sa meranie zastaví buď po 20 sekundách alebo 1 sekundu po tom, ako sa robot dostane do vzpriamenej polohy (vstane) a prestane sa hýbať. Vtedajší stav bol taký, že sa meranie zastavilo buď po 20 sekundách alebo hneď ak je hráč vo vzpriamenej polohe. Do metódy „isStopCriterionMet“, ktorá sleduje hráča a na základe istých podmienok zastavuje disciplínu, bola pridaná kontrola, ktorá sledovala presné hodnoty súradníc hráčovej hlavy a meranie zastavilo vtedy, ak bolo hráčovo telo vo vzpriamenej polohe a jeho hodnoty ostali nezmenené počas 1 sekundy. Zabezpečil to nasledujúci kód:

```
TransformationMatrix currentBodyState = p.getBody().getHead().getTransformation();

if (standing) { // is standing
       if ((elapsedTime - lastIsStandingTime) < 0) {  // first standed up
```

```
                lastIsStandingTime = elapsedTime;
                lastBodyState = currentBodyState;
        } else if (currentBodyState.compareWith(lastBodyState)) {
                if ((elapsedTime - lastIsStandingTime) >= 1) {
                        successTime = (elapsedTime - startTime);
                        return true;
                }
        } else {
                lastBodyState = currentBodyState;
                lastIsStandingTime = elapsedTime;
        }
} else {
        lastBodyState = currentBodyState;
        lastIsStandingTime = 5000;
}
```

6.1.5.2.2 Chôdza (stabilita)

Cieľom tejto disciplíny je prejsť na vzdialenejšiu polovicu ihriska. Počas nej sa pripočítavajú trestné sekundy, teda 10 sekúnd za dotyk počas 1 sekundy inou časťou tela ako chodidlom.

Vtedajší kód mal znova zadané nesprávne počiatočné súradnice hráča a preto ich bolo potrebné zmeniť podľa pravidiel na (-5; 1).

V prípade, že hráč spadne a nebude sa vedieť postaviť, server automaticky premiestňuje hráča za hranice ihriska. V pravidlách je zadefinované, že ak tento prípad nastane, disciplína má byť vyhodnotená ako keby hráč ostal ležať na zemi do uplynutia časového limitu disciplíny. Vtedajší kód tejto disciplíny toto neošetroval, preto sme to zapracovali nasledujúcou metódou, ktorá kontroluje, či sa hráč nenachádza mimo ihriska:

```
/**
 * Checks if player is out of the field.
 *
 * @author xsuchac
 * @author A55-Kickers
 *
 * @param player
 * @return true if player is behind any border of field, else false.
 */
private boolean playerBehindBorder(Player player) {
        double playerLocationX = player.getLocation().getX();
        double playerLocationY = player.getLocation().getY();

        if (Math.abs(playerLocationX) > (fieldLength / 2)
                        || Math.abs(playerLocationY) > (fieldWidth / 2)) {
                return true;
        }
        return false;
}
```

V metóde „evaluate“ bola chybne počítaná lineárna aproximácia výsledného času podľa prejdenej vzdialenosti, ktorá sa využíva vtedy keď hráč nestihne v stanovenom limite prejsť za polku ihriska. Táto metóda bola nami viac-menej celá prerobená a bolo v nej aj sprehľadnené vypisovanie výsledkov. Bol ošetrený aj stav, keď hráč prešiel nulovú dĺžku a hrozilo delenie nulou pri vyhodnocovaní. Lineárna aproximácia bola urobená nasledovne:

```
double maxTime = 180;
```

```
double traveled = Math.abs(initPos.getX()) - Math.abs(currentPlayerLocationX);
if (Math.abs(playerLocationXBeforePutBehindBorder) <= (fieldLength / 2)) {
	traveled = Math.abs(initPos.getX()) -
Math.abs(playerLocationXBeforePutBehindBorder);
}
double result = -5000;
if (traveled != 0) {
	result = (Math.abs(initPos.getX()) * maxTime) / traveled;
}
```

6.1.5.2.3 Chôdza (rýchlosť)

Cieľ tejto disciplíny je podobne ako v predchádzajúcej dostať sa na druhú stranu ihriska, avšak podstatnejšia ako stabilita je rýchlosť hráča a preto tu hráč nedostáva trestné sekundy.

Ako v predchádzajúcom prípade tu bola zadefinované nesprávna počiatočná pozícia hráča a tiež neexistovala kontrola prípadu, keď server premiestni spadnutého hráča za hranice ihriska. Taktiež sme tu našli chybné počítanie lineárnej aproximácie času pre prípad, keď hráč nestihne prejsť na druhú stranu ihriska v časovom limite. Opravili sme to podobne ako v predošlej disciplíne.

6.1.5.2.4 Kopanie do lopty (vzdialenosť)

Cieľom tejto disciplíny je kopnúť loptu čo najďalej v rozmedzí piatich pokusov. Disciplíny, kde sa vyžaduje kopanie hráča do lopty sa od ostatných odlišujú, že ich merania sú odštartované v hracom režime „KickOff_Left“ namiesto „PlayOn“. Vo vtedajšom kóde sme to podľa tohto upravili.

Znova sme pridali kontrolu premiestnenia hráča serverom za hranice ihriska.

Pravidlá k disciplíne udávajú aj to, že pokiaľ hráč počas kopnutia spadne, vydelí sa výsledná vzdialenosť dvomi. Pridali sme kontrolu pádu hráča a ak sa tak stane, je to zaznamenané v premennej:

```
if(!playerFalled && ss.getScene().getPlayers().get(0).isOnGround()){
	playerFalled = true;
}
```

Starý kód zastavoval meranie kopu vtedy, keď lopta prešla určitú malú vzdialenosť. Toto však nebolo výhodné a nahradili sme tým, že meranie sa zastaví vtedy, keď sa lopta prestane hýbať po tom, čo sa už hýbala:

```
if(ballMoved && !ss.getScene().isBallMoving()){
	return true;
}
```

Starý kód vyhodnocoval disciplínu tak, že sa až na konci pozrel, či je hráč na zemi a vtedy delil výsledok dvomi. Tento stav bol nevyhovujúci, pretože hráč mohol počas disciplíny padnúť, vstať a potom kopnúť do lopty a skončiť na konci disciplíny vo vzpriamenej polohe. Toto sme nahradili podmienkou, v ktorej figuruje už spomínaná premenná, v ktorej je uložené, či hráč niekedy počas kopu padol na zem.

6.1.5.2.5 Kopanie do lopty (presnosť)

Podobne ako v predošlej disciplíne sme potrebovali upraviť štart merania do fázy „KickOff_Left“. Znova sme pridali kontrolu premiestnenia hráča serverom za hranice ihriska.

Ďalšie pravidlá sú kopnutie lopty smerom k niektorému rohy ihriska, penalizácia pádu násobením odchýlky kopu od rohu ihriska dvomi a prejdenie lopty minimálne 1,5 metra.

V tomto prípade už kód obsahoval zastavenie merania v prípade, že sa lopta hýbala a prestala sa hýbať. Ostatné veci boli taktiež v poriadku.

6.1.5.2.6 Otočenie hráča

Cieľom tejto disciplíny je otočenie sa o 180 stupňov s toleranciou 5 stupňov, pričom sa z dvoch pokusov započíta ten rýchlejší.

Znova bolo potrebné pridať iniciálnu pozíciu hráča. Znova sme pridali kontrolu premiestnenia hráča serverom za hranice ihriska.

Kontrola toho, kedy sa hráč úspešne otočil bola vo vtedajšom kóde robená veľmi komplikovane a vo veľa prípadoch vyhodnocovala disciplínu nesprávne. My sme si na začiatku určili hodnoty uhlov natočení, medzi ktorými je možné vyhodnotiť, že sa hráč úspešne otočil. Jednoduchými podmienkami sme to potom v metóde „isStopCriterionMet“ kontrolovali spolu s tým, či je hráč zároveň aj v stojacej polohe:

```
boolean standing = false;
if (p.getBody().getHead().getTransformation().getValues()[14] >= 0.45) {
	standing = true;
}
boolean turned = false;
if (exptectedRotation1 < exptectedRotation2) {
	if (actualPlayerRotation >= exptectedRotation1 && actualPlayerRotation <=
exptectedRotation2) {
		turned = true;
	}
} else {
	if (actualPlayerRotation >= exptectedRotation2 && actualPlayerRotation <=
exptectedRotation1) {
		turned = true;
	}
}
if (standing && turned) {
	return true;
}
```

Bolo potrebné ošetriť aj vyhodnotenie nesplnenia disciplíny, ktoré sme vykonali vrátením veľmi veľkej hodnoty 5000s.

6.1.5.2.7 Zmeny spoločné pre všetky disciplíny

Vo všetkých disciplínach sme rozumne upravili ich výpisy. Nechali sme iba tie najdôležitejšie, teda tie, ktoré hovoria o začiatku, konci, vyhodnotení disciplíny s vypisovaním penalizácií a podobne. Zmenili sme level výpisov na INFO, čo znamená, že Test Framework ich zobrazuje

používateľovi v jeho spodnej časti bez nutnosti nastavovať level výpisov. Pretým bola väčšina výpisov levelu FINE a na ich zobrazovanie bolo potrebné Test Framework dodatočne nastavovať.

6.1.5.3 Testovanie

Testovanie prebiehalo pravidelne počas implementácie a po jej skončení, aby sa overilo, či všetky zmeny fungujú tak ako majú.

6.1.6 Oživenie stránky + wiki (pp 5.12)

Počas riešenia 5. šprintu tímového projektu prebiehalo sťahovanie školských serverov, na ktorých sa nachádzala aj stránka a wikipedia tímového projektu RoboCup. Stránka po spustení serverov nefungovala, bolo potrebné získať novú IP adresu servera a reštartovať sieťovú kartu. RoboCup Wiki bežala na inom serveri, na ktorý sme nemali prístup, ale nastavením pravidelného denného zálohovanie RoboCup wiki do git repozitára, sme RoboCup Wiki spustili zo záloh na našom serveri. Stránka aj wikipedia fungujú na adresách:

team15-12.ucebne.fiit.stuba.sk

team15-12.ucebne.fiit.stuba.sk/teamwiki/index.php/Hlavná_stránka

6.2 Šprint 6 – „zeta“

Tabuľka 6.3: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
6.1	Člen tímu	Vyhodnotiť úspešnosť prihrávky	Prihrávanie navrhnúť a implementovať
6.2	Člen tímu	Vedieť vypočítať pravdepodobnosť získavania lopty	Zapracovať túto funkciu do taktiky
6.3	Člen tímu	Implementovať navrhnutú taktiku	Využívať taktiku počas hry
6.4	Člen tímu	Implementovať vedenie lopty na bránu	Viesť loptu k súperovej bráne, keďže takáto funkcionalita zatiaľ nebola
6.5	Člen tímu	Implementovať prihrávanie	Nie len strieľať na bránu ale aj prihrávať spoluhráčom
6.6	Člen tímu	Zoptimalizovať kopanie do lopty	Prídenie k lopte zo správnej strany
6.9	Člen tímu	Prísť k lopte zo správnej strany	Vyvarovať sa natáčaniu hráča pri lopte

Tabuľka 6.4: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
6.1	-	6.1	Vyhodnotenie úspešnosti prihrávky	M. Gregor
6.2	-	6.2	Pravdepodobnosť získania lopty	M. Gregor
6.3	-	6.3	Implementovať diagramy taktiky	M. Červeňák

6.4	-	6.4	Implementovať vedenie na bránu	M. Chylik
6.5	-	6.5	Implementovať prihrávanie	J. Grega
6.6	-	6.6	Optimalizovať kopanie do lopty	J. Grega
6.9	-	6.9	Prídenie k lopte zo správnej strany	M. Škoda

6.2.1 Vyhodnotenie úspešnosti prihrávky (pp 6.1)

Tím A55Kickers pre vylepšenie taktiky hráča potreboval získať informáciu o úspešnosti prihrávky na určité miesto. Preto sme analyzovali možnosti situácie kedy hráč chce prihrať, navrhli riešenie v podobe metódy fitnessOfPass(), ktorú sme úspešne implementovali a otestovali.

6.2.1.1 Analýza

Hráč potrebuje informáciu o úspešnosti prihrávky tesne pred výkopom. Ak by bola úspešnoť prihrávky veľmi nízka môže sa hráč rozhodnúť pre inú akciu. Hráč má informácie o svete a tiež vie kam chce prihrať. Z informácii o svete vie pozíciu hráčov, prípadne si vie určiť predpovedanú pozíciu hráča o jednu sekundu, čo ale nie je presná informácia s vysokou časovou granularitou. Prihrávka na pozíciu nemusí trvať viac ako sekundu a hráč za sekundu môže byť úplne inde ako je dráha lopty. Lopta do cieľa nemá konštantnú rýchlosť. Závisý to od sily kopu. V tomto prípade by pomohli anotácie jednotlivých kopov, z ktorých by sme si sekvenčne získavali informáciu pozície lopty.

6.2.1.2 Návrh

Navrhli sme metódu, ktorá vektor prihrávky rozdelila na 10 rovnako vzdialených bodov a v každom bode vypočítala pravdepodobnosť získania lopty súperovým hráčom. Súčet jednotlivých pravdepodobností nakoniec videlila 10 a vratila úspešnosť prihrávky v hodnote od 0 do 1 vrátane. Ak aj len jeden hráč by stál v čase prihrávky v ceste, tak metóda vráti nulovú úspešnosť prihrávky. Metóda sa dá parametrizovať v akej vzdialenosti súpera od lopty je ako lopta ohrozená. Môžme určiť napríklad, že lopta začína byť ohrozená ak je hráč od nej vzdialený na 1m.

6.2.1.3 Implementácia

Metódu sme implementovali v triede TacticalInfo v modeli agenta. Metóda získa údaje o súperových hráčoch, predpovie ich pozíciu o jednu sekundu a pre každý z 10 rovnomerne od seba vzdialených bodov vektora prihrávky vypočíta ohrozenie lopty súperom.

6.2.1.4 Testovanie

Metódu sme otestovali s dvoma hráčmi, kde jeden stál s loptou v strede ihriska počítal vhodnosť prihrávky do stredu brány oproti sebe. Súperovho hráča sme posúvali pozdĺž vektora prihrávky. Úspešnosť prihrávky nadobúdala podľa pozície súpera zlé výsledky.

6.2.1.5 Záver

Možné vylepšenie metódy vidíme v zabezpečení menšej časovej granularite predpovedi pozície hráča, a ak by existovali kopy s rôznou silou a alebo dokopovou vzdialenosťou a boli by správne anotované a existovalo by pravidlo, že na akú vzdialenosť sa aký kop použije, tak by sme mohli orhozenie vektora prihrávky presnejšie vypočítať. Metódu prerobíme podľa špecifikácie úspešnosti prihrávky tímu 17, ktorá je popísaná v kapitole 3.1.6.3.4.

6.2.2 Pravdepodobnosť získania lopty (pp 6.2)

Náš tím pre vylepšenie taktiky hráča potreboval získať informáciu pravdepodobnosti získania lopty jeho tímom. Preto sme navrhli, implementovali a otestovali metódu, ktorá vracia hodnotu z intervalu <0,1>. Hodnota vyjadruje pravdepodobnosť získania lopty našim tímom. Vyššia hodnota znamená väčšiu šancu získať loptu našim tímom.

6.2.2.1 Šanca získať loptu našim tímom

Keď loptu nevlastní žiaden hráč, je dôležité si určiť kto loptu získa skôr. Či to bude náš tím alebo súperov. Preto sme navrhli algoritmus kde hráč zistuje pozíciu hráča najbližšieho k lopte vrátane seba. Počíta to z pozícii hráčov ktorých vidí a z pozície lopty, ktorú vidí. Štruktúra algorimtu je popísaná nasledovným pseudokódom:

```
ballPos = WorldModel.getInstance().getBall().getRealtivePosition();
myDistance = WorldModel.getMyDistFromBall();
mySpeed  = AgentInfo.mySpeed;
myTime = myDistance / mySpeed;

oponentPlayers = WorldModel.getOponentPlayers();
ourPlayers = WorldModel.getOurPlayers();

foreach player in oponentPlayers
                dist = player.distanceFromBall;
                speed = player.speed;
                time_array.add(dist/speed);
end

foreach player in ourPlayers
                dist = player.distanceFromBall;
                speed = player.speed;
                our_time_array.add(dist/speed);
end

sort(time_array);
sort(our_time_array)

if (!oponentsPlayers.isEmpty() && bIsClosestToMeInMyTeam())
        return myTime / (myTime + time_array[0]);
end

if (!oponentsPlayers.isEmpty() && !bIsClosestToMeInMyTeam())
        return our_time_array[0] / (our_time_array[0] + time_array[0]);
```

end

return 1;

Metóda určí časy superových a našich spoluhráčov na to aby sa dostali k lopte. Výpočet prebieha podľa vzorca *t = s/v*, kde s je vzdialenosť hráča od lopty a v je odpozorovaná rýchlosť hráča. Ak je hráč najbližšie zo svojho tímu k lopte tak vypočíta šancu získania lopty svojim tímom z najlpešieho času hráčov súperovho tímu a svojho času k dostaniu sa k lopte podľa vzorca:

sanca = hracovCas / (hracovCas + najlepsiCas)

Ak nie je najbližšie zo svojho tímu tak šancu získania lopty svojim tímom vypočíta z najlepšieho času súperových hráčov a najlepšieho času našich hráčov podľa vzorca:

sanca = najlepsiCasNasihHracov / (najlepsiCasNasichHracov + najlepsiCasSuperovychHracov)

Ak je hráč na ihrisku sám vráti metóda 100% šancu získať loptu hodnotou 1.

6.2.2.2 Moja šanca získať loptu

Pre lepšie videnie sveta sme hráčovi implementovali metódu, ktorá mu vypočíta jeho pravdepodobnosť získania lopty. Algoritmus sme navrhli tak, že hráč zistuje poziciu hráča najbližšieho k lopte vrátane spoluhráčov. Počíta to z pozícii hráčov ktorých vidí a z pozície lopty, ktorú vidí. Táto informácia pomôže hráčovi sa následne rozhodnúť či ísť po loptu alebo sa pripraviť na bránenie súperovmu hráčovi. Štruktúra algorimtu je popísaná nasledovným pseudokódom:

```
ballPos = WorldModel.getInstance().getBall().getRealtivePosition();
min = infinity;
myDistance = WorldModel.getMyDistFromBall();
mySpeed  = AgentInfo.mySpeed;
foreach player in players
        if ( palyer.relPosition – ballPos < min )
                bestTime = player.timeToGetToBall;
                playerId = player.id;
                playerTeam = player.team;
        end

        myTime = myDistance / mySpeed;
        return myTime / (bestTime + myTime);
end
```

Hráč z údajov v modeli získa vzdialenosti všetkých hráčov od lopty a ich rýchlosti, ktoré zaznamenal. Medzi hráčmi má aj súperových aj svojich spoluhráčov. Z rýchlosti a vzdialenosti každého hráča vypočíta hráčov čas potrebný na to aby sa dostal k lopte podľa vzorca *t = s/v*. Nakoniec vyberie najlepší čas, vypočíta svoj vlastný čas a vráti jeho šancu získania lopty podľa vzorca:

sanca = hracovCas / (hracovCas + najlepsiCas)

6.2.2.3 Implementácia

Metódy sme implementovali v existujúcej triede TacticalInfo v modeli hráča v balíku sk.fiit.jim.agent.models. Implemetácia metód bola priamočiara podľa navrhnutých algoritmov ako

public metódy s názvami chanceToGetBall a chanceToGetBallByMe. Obe metódy majú návratovú hodnotu double. Metóda chanceToGetBall zisťuje pozíciu lopty volaním WorldModel.getInstance().getBall().getPosition(), ktoré vráti 3D vektor pozície lopty. Ďalej získa pole všetkých súperových a vlastných hráčov volaniami WorldModel.getInstance().getOpponentPlayers()
a WorldModel.getInstance().getTeamPlayers(). Jednoduchov iteráciou nad každým hráčom vypočíta čas podľa uvedeného vzorca v návrhu. Rýchlosť hráča je určená v 3D vektore a preto rýchlosť sme prepočítali pomocou pytagorovej vety podľa vzorca $v = \sqrt{x^2 + y^2}$. Získané časy sme uložili do dvoch rôznych polí, ktoré sme usporiadali od najmenšieho po najväčšie volaním java.util.Collections.sort(). Nakoniec jednoduchým if vetvením či hráč je najbližšie k lopte zo svojho tímu a či sú nejaký súperový hráči videný sme určili šancu získania lopty vzorcom z návrhu.

Počas implementácie sme mali problém kde model hráča neobsahoval informáciu o súperových hráčoch. Problém bol, že ak súperov hráč mal rovnaké id ako hráč, ktorý informáciu prijímal, tak informácia bola odfiltrovaná a neuložila sa do hráčovho modelu. Problém bol opravený.

6.2.2.4 Testovanie

Testovanie prebehlo simuláciou s dvoma protihráčmi, ktorý hlásili do logov svoju vypočítanú šancu. Ak hráči stáli rovnako ďaleko od lopty tak hlásili približne 50% šancu. Pri zmene vzdialenosti súpera alebo seba sa šanca prepočítavala a menila. Ak bol hráč z ihriska odobratý a následne pridaný, kde súper sa pohyboval smerom k lopte, tak hráč vypočítal vysokú pravdepodobnosť spôsobenú nedostatkom informácií o súperovi. Tento problém sa nazýva *ColdStart*. Na unit testoch sa pracuje, keďže celý projekt trpý nedostatočným pokrytím testov.

6.2.2.5 Záver

Metódu by bolo možné vylepšiť zisťovaním rýchlosti hráča samého a započítaním rotácie hráča do času dorazenia k lopte, kde je ale problém, že otáčanie súpera nie je možné presne zachytiť a preto nie je možné zachytiť trvanie otáčania. Pri predpoklade, že hráči sú počas zápasu od začiatku do jeho konca stále na ihrisku, tak nie je zatiaľ potrebné ošetrovať ColdStart problém.

6.2.3 Implementovať diagramy taktiky (pp 6.3)

Cieľom úlohy bolo vytvorenie plánovača, ktorý bude implementovať navrhnutú taktiku. Implementácia bola vykonaná v jazyku ruby.

6.2.3.1 Implementácia

Taktika bola implementovaná podľa návrhu (Návrh vylepšenia taktiky (pp 5.9)). Bola implementovaná do novo vytvoreného plánovača *planTactic.rb* ako rubyscript.

6.2.3.1.1 Som najbližšie k mojej bránke

Pre vylepšenie taktiky sme potrebovali implementovať v hráčovi videnie sveta, či je najbližšie k bránke, aby sa vedel rozhodnúť či má zostať brániť našu futbalovú sieť.

6.2.3.1.2 Ako zistím či som najbližšie k bráne?

Metódu sme navrhli tak, že hráč si z videných spoluhráčov pre každého vypočíta jeho vzdialenosť od vlastnej brány. Získané vzdialenosti zoradí. Vyberie tú, ktorá je najmenšia a porovná ju zo svojou vzdialenosťou. Ak jeho vzdialenosť je menšia, tak sa vyhodnotí ak najbližší hráč. Inak sa vyhodnotí, že nie je najbližší hráč.

6.2.3.1.3 Ako to vypočítam

V modeli hráča sme implementovali metódu videnia sveta isINearestToGoal() v triede TacticalInfo. Metóda má návratovú hodnotu typu boolean. Implementácia bola jednoduchá a pozostávala so získania zoznamu videných spoluhráčov pomocou volania

WorldModel.getInstance().getTeamPlayers();

Následne iterovaním cez zoznam získaných hráčov porovnávam ich vzdialenosti od našej brány s mojou. Ak vzdialenosť niektorého z hráčov od brány je menšia ako moja, tak metóda vráti *FALSE*. Ak prejdem celý zoznam videných hráčov a žiaden nemal hodnotu vzdialenosti od našej brány menšiu, tak vrátim *TRUE*.

6.2.3.2 Testovanie

Testovanie prebiehalo podľa potrieb jednotlivých krokov v znázornených diagramoch. Zväčša boli na testovanie použitý 2-3 hráči. Následne bola herná situácia nasimulovaná pomocou testovacieho frameworku. Takto bolo možné pozorovať chyby a nedostatky spôsobené návrhom či implementáciou. Tak sa dosiahla súčasná taktika, ktorá má všetky herné situácie otestované a funkčné.

Simuláciu nebolo možné lokálne vykonať na viacerých hráčoch, pretože to nedovoľovali hardvérové kapacity. Zväčša ale boli 2-3 hráči dostačujúci.

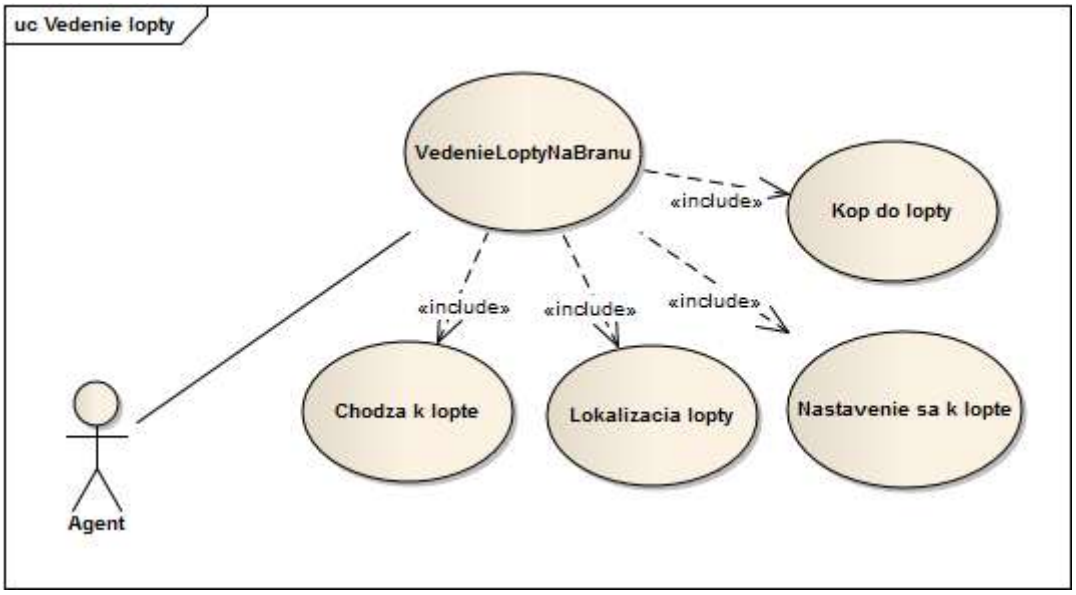
Pri vytvorení počítačovej siete bola formácia otestovaná i na viacerých hráčoch.

6.2.4 Implementovať vedenie na bránu (pp 6.4)

Cieľom úlohy bolo vytvorenie high skill pohybu, ktorého úlohou je doviesť loptu do brány. Primárnym cieľom je samotné dovedenie lopty do brány, sekundárnym cieľom je zoptimalizovanie tohto pohybu tak, aby bol čo najrýchlejší.

6.2.4.1 Analýza

Samotný high skill sa skladá z viacerých častí, ako je možné vidieť z obrázku č.6.11.



Obrázok 6.11 Diagram prípadov použitia

6.2.4.1.1 Lokalizácia lopty

Úlohou tejto fázy je objaviť loptu a natočiť sa správnym smerom tak, aby bolo možné prejsť do fázy chôdza k lopte.

Problém lokalizácie lopty nie je triviálny. V prípade, že je lopta v zornom poli hráča, je natočenie hráča len matematickým príkladom, kde je potrebné vypočítať správny uhol natočenia a natočiť sa vhodným smerom. Ak však nie je lopta v zornom polo hráča, je potrebné vykonať nejaký vybraný pohyb a pokúsiť sa tak dostať loptu do zorného pola. Hráč by mal vykonávať tieto pohyby tak, aby nimi doplnil zorné pole na plných 360°, resp. toľko, koľko je potrebné na objavenie lopty.

Tento problém sa snaží riešiť existujúci high skill localize.rb.

6.2.4.1.2 Chôdza k lopte

Úlohou tejto fázy je priblížiť sa čo najrýchlejšie k lopte do určitej vzdialenosti, kde by sa hráč už mal začať pripravovať na kop. Pre tieto účely je preto potrebné použiť čo najrýchlejší pohyb, pretože je možné, že hráč bude musieť prekonávať väčšie vzdialenosti.

Chôdza dopredu, priamym smerom, je implementovaná vo viacerých existujúcich low skilloch. Najnovším, ktorý by mal byť najrýchlejším a dostatočne stabilným, je walk_fine_fast2_optimized3.xml

6.2.4.1.3 Nastavenie k lopte

Úlohou tejto fázy, ktorá začína po tom, ako sa hráč dostane do blízkosti lopty, je nastavenie sa k lopte tak aby bolo možné vykonať kop ľavou alebo pravou nohou smerom na bránu.

Keď sa hráč dostane do dostatočnej blízkosti lopty, je potrebné nastaviť hráča pomocou krokov v štyroch smeroch(dopredu, dozadu, doľava, doprava) a otáčania hráča po kružnici. Na posúvanie hráča v rôznych smeroch existujú urobené low skilly. Na otáčanie po kružnici, high skill turn.rb.

6.2.4.1.4 Kop do lopty

Úlohou tejto fázy je vykonanie kopu do lopty, ľavou alebo pravou nohou tak, aby lopta smerovala smerom na bránu.

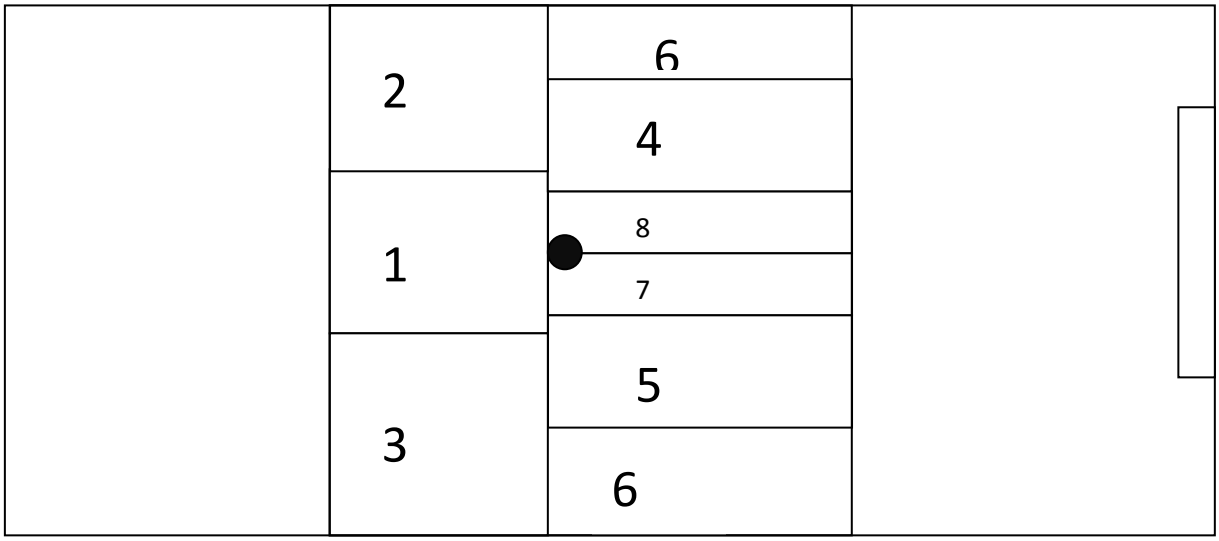
Na kopanie do lopty existuje high skill kick.rb, ktorý okrem samotného kopania správnou nohou sa stará aj o úplne presné nastavenie hráča pred kopom, nastavenie jednej či druhej nohy hráča pred loptu.

6.2.4.2 Návrh

Návrh riešenia sa skladá z návrhu riešení jednotlivých fáz vedenia lopty. Pri návrhu sa kládol dôraz na opätovné použitie existujúcich častí kódu, ktoré fungujú korektne. Pre fázu lokalizácia lopty bol použitý existujúci high skill localize. Tento high skill rieši komplexne problém lokalizácie lopty, teda zahŕňa oba prípady spomenuté v analýze.

Pre fázu chôdza k lopte je vhodné použiť najnovší a teda najrýchlejší priamočiary pohyb dopredu. Tým je low skill walk_fine_fast2_optimized3.xml. Ak sa hráč odchýli od priameho smeru k lopte, pre jeho správne natočenie sa používajú otáčania vľavo a vpravo o 6.5, 20 alebo 45 stupňov.

Fáza nastavenie k lopte nebola komplexne zatiaľ spracovaná v samostatnom high skille, čiastočne sa však rieši v high skille walk2.rb. Priestor okolo lopty je rozdelený do 8 zón, 3 zóny sú za loptou(teda lopta je medzi hráčom a bránou na ktorú chce útočiť), 5 pred ňou(Teda hráč je medzi loptou a bránou na ktorú chce útočiť). Rozdelenie priestoru je načrtnuté na obrázku 6.12.



Obrázok 6.12 Zóny okolo lopty

Pre jednotlivé zóny sú potom naplánované prislúchajúce pohyby podľa tabuľky 6.5.

Tabuľka 6.5 Pohyb podľa zón

Zóna	Pohyb
1	Pomaly dopredu
2	Úkrok doprava

3	Úkrok doľava
4	Dozadu
5	Dozadu
6	Dozadu
7	Úkrok doprava
8	Úkrok doľava

V prípade, ak hráč nie je natočený po tom, ako sa dostal do bezprostrednej blízkosti lopty, smerom na bránu, použije sa high skill turn.rb, ktorý vykoná natočenie po kružnici.

Pre kopanie do lopty sa použije high skill kick.rb.

6.2.4.3 Implementácia

Vedenie lopty bolo implementované podľa návrhu. Vznikol tak high skill walk2Ball.rb. Ten je potrebné momentálne použiť s plánovačom planmike.rb.

6.2.4.4 Testovanie

Pri testovaní sa zistili nedostatky v niektorých použitých low skilloch aj high skilloch.

Vo fáze lokalizácie sa občas stalo, že hráč loptu nenašiel, hlavne ak sa nachádzala za jeho chrbtom, resp. nevykonal pohyby tak aby bol schopný objaviť loptu.

Pri chôdzi k lopte bol hlavný problém v stabilite hráča, keď strácal stabilitu hlavne v prípadoch ukončenia pohybu a prechodu na iný pohyb. Ďalším problémom, ktorého pôvod sa mi nepodarilo odhaliť bolo zaseknutie sa hráča pri natáčaní sa. V takomto prípade pomohlo znovunačítanie xml.

Hlavným problémom nastavenia sa k lopte sú veľmi pomalé pohyby hráča pri pohybe do strán a dozadu. Nastavovanie k lopte trvá neúmerne dlho, nehovoriac o pomerne častých pádoch. Z tejto fázy sú uspokojivé len výsledky z otáčania sa po kružnici.

Kopanie do lopty je pomerne úspešné, nie je však možné nastaviť želanú dĺžku kopu. Dĺžka jednotlivých kopov je veľmi rôzna.

Celkovo bol hráč schopný vsietiť za polčas, trvajúci 300 sekúnd, maximálne jeden gól a ani to sa mu nepodarilo vždy.

6.2.4.5 Záver

Pohyb vedenia lopty do brány je použiteľný, aj keď by mohol byť efektívnejší. V prvom rade je potrebné doriešiť niektoré low skilly a high skilly vstupujúce do vedenia lopty. Tým sa zvýši efektivita celého pohybu a bude umožnená ďalšia optimalizácia pri spájaní už zaručene efektívnych pohybov. V súčasnom stave je hráč schopný reálne dosiahnuť maximálne jeden gól za 300 sekundový polčas.

6.2.5 Implementovať prihrávanie (pp 6.5)

Náš tím potreboval na vytvorenie kvalitnej taktiky, ktorá by simulovala reálny futbal implementovať prihrávky. Preto sme navrhli implementovali a otestovali vyšší pohyb pre kopnutie

na miesto, ktorý je možne použiť vo viacerých modifikáciách ako kopnutie na bránu, prihrávka a ďalšie.

6.2.5.1 Implementácia

Pri implementácií tohto pohybu sme využili návrh z predchádzajúceho šprintu, ktorý je opísaný v kapitole 6.1.2.2.2 Návrh kopnutia lopty na miesto.

Samotná implementácia spočívala v úprave už existujúcich pohybov. Upravili sme pohyb Kick, ktorý sme parametrizovali. Pridali sme mu parameter, slúžiaci na zadanie miesta na ktoré treba kopnúť. Následne sme vytvorili metódu, ktorá slúži na vyber sily kopu. Na to sme využili exitujúcu metódu , ktorá vypočíta vzdialenosť lopty od požadovaného bodu a na základe tejto vzdialenosti vyberie požadovaný nižší pohyb pre kopnutie.

Ďalším krokom bolo úprava vyššieho pohybu Turn. Keďže pohybom sa agent vedel správne natočiť len na bránu, bolo potrebne ho upraviť. Keďže miesto natočenia bolo definovane ako miesto brány, bolo potrebné zabezpečiť, aby sa toto miesto menilo podľa požiadaviek planera. Ďalšou úpravou bola implementácia výpočtu správneho uhla medzi agentom a zadanou pozíciou.

6.2.5.2 Testovanie

Testovanie bolo vykonane pomocou planera, kde sme zadávali rôzne pozície miesta pre kopnutie a sledovali, čí sa do miesta trafí. Taktiež sme na testovanie využili testovací framework, kde sme využili testcase na testovanie presnosti kopu. Tu sme sledovali uhol vychýlenia od pozície kopnutia. Týmto testom sme zistili, že uhol vychýlenia od požadovaného miesta ja 6,83°. Tiež sme testovaním zistili, že agent na zadanú pozíciu nevie kopnúť jedným kopnutím. Pre kopnutie na zadanú pozíciu jedným kopnutím je nutne vytvoriť viacero nižších pohybov s rôznou dĺžkou kopnutia. Aktuálne je maximálna dĺžka kopnutia 3,2 metra.

6.2.6 Optimalizovať kopanie do lopty (pp 6.6)

Náš tím sa rozhodol optimalizovať kopanie do lopty. Toto rozhodnutie pramenil z dlhého nastavovania sa na správnu pozíciu pri lopte a malej sily kopu.

6.2.6.1 Návrh

Pre vylepšenie kopnutia do lopty sme navrhli novy vyšší pohyb prídenie k lopte zo správnej strany, ktorý je opísaný v kapitole 6.2.7. Pre zvýšenie sily kopu sme sa rozhodli prehodnotiť vzdialenosť medzi agentom a loptou.

6.2.6.2 Implementácia

Implementácia tejto úlohy sa vykonávala dvojfázovo. Základom bolo vytvoriť stabilne nižšie pohyby pre pohyb agenta. A následné vytvorenie vyššieho pohybu Prídenie k lopte zo správnej strany pomocou nižších pohybov. Vytvorenie vyššieho pohybu Prídenie k lopte zo správnej strany je opisany v kapitole 6.2.9. Tvorba nižších pohybov je opísaná v kapitole 6.1.1, 6.3.1, 6.3.3. Pre optimalizáciu sily kopu sme zmenšili vzdialenosť medzi agentom a loptou. Túto vzdialenosť sme zmenšili o 0,05 metra.

6.2.6.3 Testovanie

Testovanie tejto úlohy sa vykonávalo odmeraním dĺžky kopu. Na testovanie sme využili testovací framework. Pomocou neho sme zistili, že dĺžka kopu sa zvýšila približné o 0,2 metra. Doba nastavovania sa na správnu pozíciu nebola odmeraná kvôli stále prebiehajúcim prácam na zdokonaľovaní nižších pohybov a ďalšom optimalizovaní vyššieho pohybu pre pradenie k lopte zo správnej strany.

6.2.7 Prídenie k lopte zo správnej strany (pp 6.9)

Cieľom tejto úlohy bolo vytvorenie high skillu, ktorý by umožňoval prísť k lopte, pričom by bolo možné uviesť aj bod na ihrisku, na ktorý by bol hráč nasmerovaný. Bolo potrebné, aby bol daný high skill spoľahlivý a efektívny.

6.2.7.1 Analýza

Ak má hráč kopnúť loptu na nejaké určené miesto, je potrebné aby bol k lopte nasmerovaný tak, že keď do lopty kopne tak lopta pôjde žiadaným smerom. K tomuto bolo potrebné identifikovať, či existujú všetky potrebné low skilly.

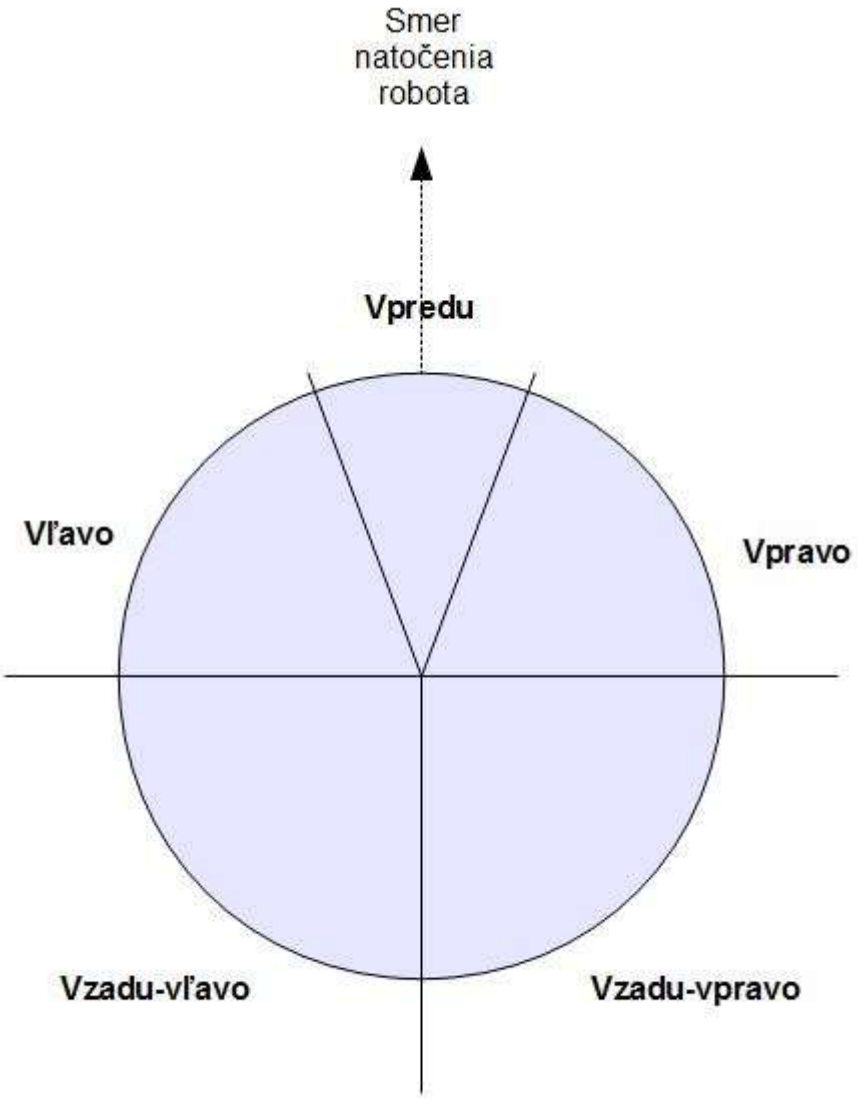
Ako odrazový mostík poslúžil high skill „GoToPosition.rb“. V ňom bol totiž použitý systém rozdelenia okolia na oblasti. Tento systém bol použiteľný aj pre prichádzanie k lopte zo správnej strany.

6.2.7.2 Návrh

Návrh riešenia je založený na rozložení okolia agenta na niekoľko oblastí, vzhľadom na relatívnu polohu agenta na ihrisku:

- Vpredu – uhol od 335 do 25 stupňov
- Vľavo – uhol od 270 do 335 stupňov
- Vpravo – uhol od 25 do 90 stupňov
- Vzadu-vľavo – uhol od 180 do 270 stupňov
- Vzadu-vpravo – uhol od 90 do 180 stupňov

Toto rozdelenie je možné vidieť znázornené na obrázku nižšie (obr. 6.13).



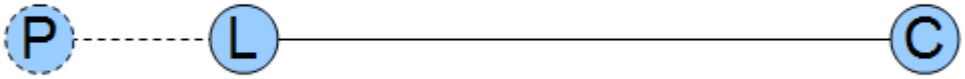
Obrázok 6.13 Rozdelenie uhlov pre otáčanie agenta

Aby sa zefektívnila chôdza agenta k lopte, zadefinovalo sa niekoľko bodov, ktoré je možné vidieť na obrázku(obr. 6.14). Týmito bodmi sú:

C – pozícia cieľa, na ktorý treba kopnúť loptu

L – pozícia lopty

P – pozícia, na ktorú sa má presunúť hráč, aby bol správne umiestnený pre nasledujúci pohyb



Obrázok 6.14 Body pre určenie smeru pohybu

Hráč sa teda presúva najprv na pozíciu P, z ktorej sa následne priblíži k pozícií L dostatočne blízko, aby bol schopný dosiahnuť na loptu.

6.2.7.3 Implementácia

Algoritmus, ktorý bol pre high skill implementovaný je nasledovný:

1. Výpočet pozície na ktorú sa treba presunúť
2. Agent je padnutý na zemi – ukonči high skill (presunúť sa naspäť do plánovača)
3. Agent nevidí loptu – ukonči high skill (presunúť sa naspäť do plánovača)
4. Agent sa zatiaľ nepresunul na pozíciu pred loptou – vyberie sa low skill na základe kombinácií týchto kritérií:
   4.1. Poloha ktorú agent dosiahol : pred loptou alebo pri lopte
   4.2. Vzdialenosť agenta od pozície na ktorú sa presúva
   4.3. V ktorom z definovaných uhlov sa nachádza pozícia na ktorú sa agent presúva
5. Agent je na mieste – koniec high skillu

Low skilly ktoré sú vyberané pre pohyb hráča, kým je ďaleko od pozície na ktorú sa má presunúť, sú vyberané rýchlejšie aj keď menej presné. Keď je hráč blízko pri lopte tak sa vyberajú low skilly s menšou rýchlosťou, ale väčšou presnosťou.

High skill, ktorývznikol sa nazýva „*goToBall.rb*“.

6.2.7.4 Testovanie

Testovanie prebiehalo spúšťaním hráča a sledovaním vykonávania tohto high skillu. Pre lepšiu prehľadnosť toho, ako sa vykonáva boli použité pomocné výpisy. Premiestňovanie agenta a lopty, bolo kvôli zjednodušeniu testovania vykonávané cez test framework.

Low skilly, ktoré sú používané boli postupne zoptimalizované a poskytovali dostatočnú stabilitu. Napriek tomu hráč občas spadol, ale množstvo pádov sa výrazne zredukovalo. Vytvorený high skill je teda použiteľný, napriek tomu, že občas hráč stratí informáciu o tom kde sa nachádza lopta (prestane ju vidieť) a chvíľku trvá kým ju zasa nájde.

6.3 Šprint 7 – „eta“

Tabuľka 6.6: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
7.1	Člen tímu	Prepojiť nižšie pohyby	Aby hráč pri prepínaní medzi pohybmi nepadal
7.2	Člen tímu	Vytvoriť nižšie pohyby s možnosťou kopu do rôznych vzdialeností	Počas hry využívať rôzne dĺžky kopu
7.3	Člen tímu	Zlepšiť nižšie pohyby s úkrokmi do strán a po kružnici	Tieto rýchlejšie a stabilnejšie pohyby využiť
7.4	Člen tímu	Vytvoriť vyšší pohyb držania formácie	Aby táto funkcionalita mohla byť využitá v taktike
7.5	Člen tímu	Opraviť chybu v lokalizovaní lopty	Po postavení hráča zo zeme možné lokalizovať loptu
7.6	Člen tímu	Vyriešiť bug v plánovači	Spúšťať hráča bez zasekávania
7.7	Člen tímu	Vyriešiť bug v testframeworku	Pridaného hráča vidieť na monitore

Tabuľka 6.7: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
7.1	-	7.1	Low skill: Prepojenie low skillov	G. Nagy
7.2	-	7.2	Low skill: Kopy do rôznych vzdialeností	G. Nagy
7.3	-	7.3	Low skill: Úkroky do strán a po kružnici	G. Nagy
7.4	-	7.4	High skill: Drž formáciu	F. Sucháč
7.5	-	7.5	High skill: Lokalizácia lopty (FIX)	M. Chylik
7.6	-	7.6	Vyriešiť problém s plánovačom	M. Chylik
7.7	-	7.7	TestFramework - pridanie hraca	M. Gregor

6.3.1 Low skill: Prepojenie low skillov (pp 7.1)

Z dôvodu zasekávania hráčov bolo potrebné vykonať prepojenie low skillov tak, aby išli plynulo. To bolo náplňou tejto úlohy.

6.3.1.1 Analýza

Analýza prebiehala testovaním vytvorených nízkoúrovňových pohybov. Zistilo sa, že počas prechodov dochádza k strate stability z dôvodu, že ukončovacie a začiatočné fázy pohybov sú rôzne. Najväčší problém pôsobí to, keď agent bezvýznamne zdvíha ruky pred seba.

6.3.1.2 Návrh

Pre vyriešenie identifikovaných problémov sme zadefinovali novú základnú polohu, do ktorej sa agent vracia po každom dokončenom pohybe a takto si stráži svoju stabilitu.

Samotný základný pohyb sme zmenili tak, aby agent nezdvíhal ruky pred seba do výšky hlavy. Základná poloha je zadefinovaná tak, aby bola vyhovujúca pre začatie aj prepájanie všetkých nízkoúrovňových pohybov.

Novovzniknutá základná poloha sa nakoniec líši od pôvodnej najmä v tom, že agent si zdvihne ruky nižšie pred seba a to do 10° uhla. Ďalej agent má pokrčené kolená aby mal bod váhy nižšie položený.

6.3.1.3 Implementácia

Parametre novovzniknutej základnej polohy sú nasledovné:

```
<he2 end="0" />
<rae1 end="-90"/>
<lae1 end="-90"/>
<rae2 end="-10"/>
<lae2 end="10"/>
<rae3 end="80"/>
<lae3 end="-80"/>
<rae4 end="60"/>
<lae4 end="-60"/>
<rle1 end="0" />
<lle1 end="0" />
<rle2 end="0" />
<lle2 end="0" />
<rle3 end="40" />
<lle3 end="40" />
<rle4 end="-60" />
<lle4 end="-60" />
<rle5 end="30" />
<lle5 end="30" />
<rle6 end="0" />
<lle6 end="0" />
```

6.3.1.4 Testovanie

Testovanie prebiehalo rovnakým spôsobom ako analýza. Testovanie odhalilo, že nová základná poloha zvýšila celkovú stabilitu hráča počas prechodov medzi jednotlivými pohybmi.

6.3.2 Low skill: Kopy do rôznych vzdialeností (pp 7.2)

Špecifikácia a návrh boli vykonané v rámci úlohy „Špecifikácia a návrh vylepšenia pohybov“

6.3.2.1 Implementácia

Implementovali sa kopy do štyroch rôznych vzdialeností. Prvé tri úrovne kopnutí vychádzajú z pôvodných pohybov:

- kick_left
- kick_right

Štvrté a zároveň najsilnejšie kopnutie vychádza z pohybov:

- left_strong_kick2
- right_strong_kick2

Prvé tri úrovne kopnutí sú vykonané počas státia agenta na jednom mieste. Najsilnejšie kopnutie je vykonané z pohybu, hráč pred kopnutím spraví krok dopredu a tak si naberá silu pre silnejšie kopnutie.

Pohyby sa oproti pôvodným pohybom optimalizovali, stabilizovali a zrýchlili.

Výsledný zoznam kopnutí:

- kick_left_slow
- kick_left_nomal
- kick_left_faster
- kick_step_strong_left
- * podobne pre pravú nohu

6.3.3 Low skill: Úkroky do strán a po kružnici (pp 7.3)

Špecifikácia a návrh boli vykonané v rámci úlohy „Špecifikácia a návrh vylepšenia pohybov“

6.3.3.1 Implementácia

Implementácia úlohy v prvej fáze prebiehala vytvorením nových rovných úkrokov. Nové úkroky sú veľmi jednoduché a stabilné, dokonale spĺňajú účel.

Úkroky po kružnici vychádzajú z rovných úkrokov, do ktorých boli pridané také ohyby kĺbov, ktoré zabezpečujú vybáčanie hráča z rovného pohybu.

Fázy jednotlivých úkrokov pozostávajú zo začiatočnej a ukončovacej fázy (základná poloha) a stredovou fázou, ktorá zabezpečí samotný krok.

Stredová fáza rovného úkroku (doprava):

```
<lle1 end="20"/>
<lle2 end="45"/>
<rle5 end="30"/>
<lle5 end="31"/>
```

Stredová fáza úkroku po kružnici (doprava):

```
<lle1 end="20"/>
```

<lle2 end="45"/>
<rle5 end="40"/>
<lle5 end="42"/>

Výsledný zoznam úkrokov:

- stepleft_g
- stepleft_circ
- *podobne pre pravú stranu

6.3.4 High skill: Drž formáciu (pp 7.4)

Pre potreby taktiky bolo potrebné mať spravenú formáciu. To bolo náplňou tejto úlohy.

6.3.4.1 Analýza

Už v minulom roku bol pokus zapracovať držanie hráčov vo formácii. Stopy po tom sa nachádzali v kóde konkrétne na týchto miestach:

- scripts/plan/PlanCreateFormation.rb – plánovač volajúci high skill GoTo
- scripts/high_skills/GoTo.rb – high skill volajúci metódy v TacticalInfo a pomocou low skillov zabezpečujúci dopravenie hráčov na pozíciu vo formácii
- sk.fiit.jim.agent.models.TacticalInfo.java s metódami:
    - setMyFormPosition() – výpočet súradníc pozícií hráčov vo formácii
    - getFormPosition() – získanie pozície vo formácii pre hráča
    - isInPositionArea() – zisťovanie, či je hráč blízko svojej pozície vo formácii
    - isOnPosition() – zisťovanie, či je hráč na pozícii vo formácii

Plánovač „PlanCreateFormation“ mal však v sebe chybu, ktorá znemožňovala využitie formácií.

6.3.4.2 Návrh

Rozhodli sme sa upraviť tieto existujúce kódy do funkčnej podoby, aby sme ich vedeli používať. Najprv bolo potrebné opraviť existujúci plánovač, odskúsať ako to celé funguje a nakoniec spraviť použiteľným high skill, ktorý by mohol využívať akýkoľvek plánovač.

Keďže high skill „GoTo“ slúžil len na dostatie sa hráča na pozíciu formácie, pre širšie a možno aj neskoršie využitie sme sa rozhodli z neho spraviť získať dva určité prvky. Prvým prvkom by bol high skill „GoToPosition“, ktorý by po zavolaní s parametrom súradníc umožnil presun hráča na tú pozíciu. Druhým prvkom by bola trieda poskytujúca metódu, ktorá by s využitím „GoToPosition“ poslala hráča na pozíciu vo formácii. Tento druhý prvok sme nazvali „FormationHelper“ s metódou „getHighSkillToGoToFormation“, ktorá vracia high skill „GoToPosition“ nastavený na chod do formácie.

Neskôr sme zistili, že počas vykonávania high skillu sa môže stať, že isté podmienky pri ktorých bol high skill zavolaný už neplatia a vykonávanie high skillu by bolo potrebné zrušiť. Zaviedli sme preto ďalší parameter, ktorým si high skill pravidelne kontroluje, či je stále validný. Pokiaľ high skill vyhodnotí, že už nespĺňa podmienky validity, tak pomocou „return nil“ odovzdá riadenie plánovaču.

6.3.4.3 Implementácia

Najprv sme naimplementovali high skill „GoToPosition“. Jeho konštruktor má dva už spomínané parametre: globálna pozícia kam sa má hráč presunúť a proces valídnosti. Keď high skill volá svoju metódu „pickLowSkill“, najskôr zavolá proces valídnosti, ktorý skontroluje, či má zmysel high skill ďalej vykonávať. Ďalej nasleduje zrelativizovanie globálnej pozície na pozíciu vzhľadom na hráča pomocou Java triedy „AgentModel“ a jej metódy „reletivize“. Nasledujú volania jednotlivých low skillov zabezpečujúcich pohyb. Tu je ukážka začiatku metódy „pickLowSkill“:

```
def pickLowSkill
        if not @validity_proc.call
              return nil
        end

        if @target_position_global == nil
              return nil
        else
              @agentModel = Java::sk.fiit.jim.agent.models.AgentModel.getInstance
              @target_position = @agentModel.relativize(@target_position_global)
        end
```

Ďalej sme naimplementovali tzv. helper triedu „FormationHelper“ so statickou metódou „getHighSkillToGoToFormation“. Táto trieda umožní vrátiť plánovaču high skill, ktorý zabezpečí dopravenie hráča na pozíciu vo formácii. Jej metóda potrebuje ako parameter len proces valídnosti. V tele metódy sa potom vypočíta pozícia vo formácii a zavolá sa „GoToPosition“:

```
class FormationHelper

# @param validity_proc - process for calling method from plan to check
#      if GoToPosition is still valid or not
def FormationHelper.getHighSkillToGoToFormation validity_proc
       tacticalInfo = Java::sk.fiit.jim.agent.models.TacticalInfo.getInstance
       target_position = tacticalInfo.getFormPositionGlobal()
       return GoToPosition.new(target_position, validity_proc)
end
end
```

Príklad volania v plánovači s procesom validity takým, čo kontroluje či hráč vidí loptu, môže byť takýto:

@plan << FormationHelper.getHighSkillToGoToFormation(Proc.new{see_ball?})

Keďže v „GoToPosition“ sa globálne súradnice relativizujú, potrebovali sme z Java triedy „TacticalInfo“ dostávať vypočítané pozície do formácie v globálnom tvare. Preto sme do „TacticalInfo“ pridali metódu „getFormPositionGlobal“:

```
public Vector3D getFormPositionGlobal(){
       setMyFormPosition();
       return pos;
```

```
}
```

Čo sa týka Java metódy „setMyFormPosition“, na začiatku sme sa domnievali, že je v nej potrebné prepočítavať súradnice vzhľadom na stranu tímu. Čiže ak má hráč pravú stranu tímu, mysleli sme si že mu treba v súradniciach invertovať x-ovú a y-ovú os (teda násobiť tie časti súradníc číslom -1). Nakoniec sa táto domnienka ukázala ako mylná, pretože čudné chodenie hráčov do formácií spôsobovala chyba v tejto metóde, ktorú sme zdedili od minuloročného tímu. V jednej „else“ vetve, v ktorej sa počítali súradnice pre všetkých sedem hráčov bolo taká chyba, že za kľúčovým slovom „else“ okolo bloku riadkov priraďujúcim súradnice chýbali množinové zátvorky. Preto pri obrannej formácii išiel prvý hráč na dobrú pozíciu a ostatní na útočnú. Túto chybu sme samozrejme odstránili.

6.3.4.4 Testovanie

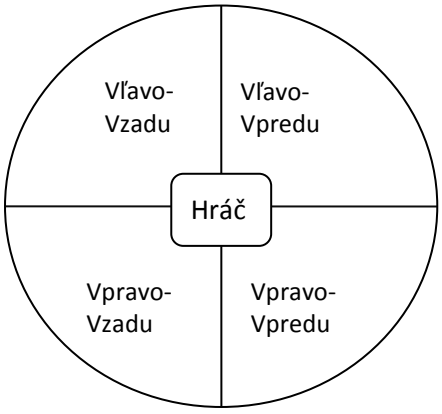
Testovanie prebiehalo pravidelne počas implementácie, ako aj po implementácie pri odľaďovaní chýb a riešení taktiky.

6.3.5 High skill: Lokalizácia lopty (FIX) (pp 7.5)

Cieľom tejto úlohy je identifikácia chyby pri lokalizácii lopty a jej odstránenie. Lokalizácia lopty je implementovaná v high skille s názvom localize.rb.

6.3.5.1 Analýza

V rámci analýzy bolo potrebné podrobne preštudovať existujúci zdrojový kód high skillu localize.rb a pochopiť jeho logiku. Priestor okolo hráča je rozdelený na 4 zóny tak, ako je to na obrázku 6.15.



Obrázok 6.15 – Zóny okolo hráč pri lokalizácii

Hráč sa obzrie hlavou doľava a doprava a následne vyhodnotí v ktorej z uvedených zón sa nachádza lopta. Následne podľa toho v ktorej zóne zbadal loptu spraví otočenie tak, ako je to uvedené v tabuľke 6.8.

Tabuľka 6.8 Otočenie hráča podľa zóny

Zóna	Otočenie
Vľavo vpredu	Doľava o 20°
Vpravo vpredu	Doprava o 20°
Vľavo vzadu	Doľava o 90°
Vpravo vzadu	Doprava o 90°

Ak nastane prípad, že napriek otočeniu hlavy doľava aj doprava sa lopta nedostane do zorného poľa hráča, nasleduje otočenie hráča doprava o 90°. Tým sa zabezpečí, že hráč sa nezasekne v situácii, keď nevidí loptu.

Podľa zdrojového kódu je teda funkcionalita naprogramovaná správne a je potrebné skúmať konkrétnu situáciu, kedy sa hráč pri lokalizácii lopty správa inak, ako by sa mal správať.

6.3.5.2 Testovanie

V rámci testovania sme chceli odhaliť chybné správanie hráča pri lokalizácii lopty. Napriek opakovanému testovaniu sa nepodarilo nasimulovať hlásený problém s lokalizáciou lopty. Testovanie bolo preto ukončené s tým, že high skill localize.rb sa správa korektne a podľa očakávania.

6.3.5.3 Záver

Táto úloha bola vybratá do šprintu po hlásení nekorektného správania hráča v situáciách, keď by mal lokalizovať loptu. Pri testovaní samotného high skillu localize.rb sme zistili, že je naprogramovaný korektne a správa sa podľa očakávania. Problém teda bol zrejme v inom high skille, resp. prepojení high skillov. Po skončení 7.šprintu a optimalizácii ostatných high skillov sa problém pri lokalizácii už neopakoval.

6.3.6 Vyriešiť problém s plánovačom (pp 7.6)

V rámci tejto úlohy bolo potrebné vyriešiť problém, ktorý spôsoboval zaseknutie, resp. zacyklenie hráča, ktorý nebol schopný pokračovať vo vykonávaní želaného pohybu.

6.3.6.1 Analýza

Prvou stopou k odhaleniu zdroju problému bol fakt, že zaseknutie hráča sa vyskytovalo pri špecifických pohyboch, konkrétne pri hráčovom otáčaní sa. Neskôr sme došli na fakt, že sa to stane v prípade, ak hráč chce vykonať dvakrát za sebou rovnaké otočenie.

V tejto chvíli už všetko nasvedčovalo tomu, že zdrojom problému budú low skilly otáčania. Pre istotu sme však súčasne s testovaním skúšali nasimulovať túto chybu aj pri iných pohyboch. Zistili sme však, že problematickými sú naozaj iba otáčania rôznych typov. Vzhľadom nato, že týmto problémom trpela drvivá väčšina low skillov otáčaní, usúdili sme, že chyba nastala pri vytváraní niektorého zo starších low skillov, z ktorého vychádzali potom ostatné a tak vlastne dedili túto chybu.

6.3.6.2 Návrh

Návrhom riešenia, ktoré by pomohlo odstrániť vyššie popísaný problém, bolo porovnať nefungujúce otáčania s niektorým zo starších otáčaní, ktoré sa nezacyklovali, napríklad s test_turn_left. Problémom mala byť prvá alebo posledná fáza pohybu, resp. ich možné prepojenie.

6.3.6.3 Implementácia

V rámci implementácie bolo potrebné opraviť poslednú fázu pohybu tak, aby na ňu bolo možné znovu napojiť rovnaký pohyb. Tento krok bolo nutné spraviť pre všetky low skilly, v ktorých sa vyskytol problém.

6.3.6.4 Testovanie

Testovanie bolo veľmi intenzívne a dalo by sa rozdeliť do 4 fáz:

1. Testovanie pre potreby zistenia príčiny problému
2. Testovanie pre potreby zistenia zoznamu low skillov, ktorých sa týka daný problém
3. Testovanie jednotlivých low skillov po opravách
4. Testovanie low skillov po ich nasadení do high skillov, priamo v hre.

Každá fáza bola dôkladne pretestovaná a vyžadovala veľa času.

6.3.6.5 Záver

Po zapracovaní opráv do jednotlivých low skillov sme dokázali odstrániť zacyklovanie hráča, ktoré predstavovalo veľmi vážnu prekážku plnej funkčnosti hráča.

6.3.7 TestFramework - pridanie hraca (pp 7.7)

V tejto úlohe bolo potrebné odstrániť bug, ktorý nastával pri spúšťaní hráčov cez test framework.

6.3.7.1 Analýza

Pri čistom pullnutí repozitára nebolo možné súštať hráčov z testframeworku. Chyba sa neprejavovala v projektoch, ktoré neboli čisto pullnute z repozitára, ale nakopírované priamo z projektu minuloročného tímu.

6.3.7.2 Návrh

Postup riešenia bol navrhnutý na chľadaní chyby debugingom v projekte. Ak sa chybu nepodarí nájsť, je potrebné porovnať priečinky projektov projektu kde pridávanie hráča cez testframework funguje s projektom, kde to nefunguje.

6.3.7.3 Implementácia

Klasickým debugingom testframeworku chybu nebolo možné odhaliť, čo znamenalo, že chyba sa nachádza niekde v projekte Jim pri naviazaní komunikácie s testframeworkom. Preto sme použili porovnanie priečinkov prokjektov Jim a zistili sme, že v priečinku Jim/scripts chýbal v repozitári priečinok testframework. Chyba prečo chýbal je v tom, že priečinok neobsahoval nič a git nekomituje prázdne priečinky. Riešenie bolo pridanie súboru .gitignore do priečinku, ktorý ignoroval všetko okrem seba.

6.4 Šprint 8 – „theta“

Tabuľka 6.9: Príbehy v šprint backlogu

ID pp	Ako	Chcem	Aby bolo možné
8.1	Člen tímu	Integrovať naše súčasné riešenie	Mať v repozitári fungujúceho hráča
8.2	Člen tímu	Spravovať zintegrované riešenie	Mať v repozitári fungujúceho a vyladeného hráča
8.3	Člen tímu	Zabezpečiť technickú stránku prezentácie výsledku	Rozbehať zápas na našich počítačoch po sieti
8.4	Člen tímu	Pripraviť materiály na IIT.SRC	Aby sme tak splnili náležitosti a požiadavky konferencie
8.5	Člen tímu	Mať aktualizovanú dokumentáciu	Mať všetky náležitosti projektu na jednom mieste

Tabuľka 6.10: Detailný opis úloh

ID pp	Zodpovedný pp	ID úlohy	Úloha	Zodpovedný
8.1	-	8.1	Integrovať naše nové funkcie	G. Nagy
8.2	-	8.2	Vyladenie zintegrovaného riešenia	F. Sucháč
8.3	-	8.3	Zabezpečenie technickej stránky prezentácie výsledku	M. Škoda
8.4	-	8.4	Príprava materiálov na ITSRC	J. Grega
8.5	-	8.5	Aktualizovať dokumentáciu (šprinty 5-8)	M. Červeňák

6.4.1 Integrovať naše nové funkcie (pp 8.1)

Obsahom tasku bolo spojenie všetkých našich vyvíjaných častí do jedného fungujúceho celku.

6.4.1.1 Analýza

Aktuálny stav pred začatím tejto úlohy bol nasledovný:

- V Git repozitári sa nachádzali pridané funkcionality do štvrtéo až piateho šprintu.
- Ostatné nové funkcie sa nachádzali na lokálnych strojoch členov vo fáze testovania

6.4.1.2 Návrh

Nové funkcie bolo treba zjednotiť do jedného kódu a aktualizovať repozitár. Po aktualizovaní repozitáru bolo potrebné aktualizovať lokálne verzie kódov u každého člena tímu.

6.4.1.3 Implementácia

Členovia tímu postupne aktualizovali repozitár a pridali novovytvorené funkcionality. Po dokončení aktualizovania repozitára sa aktualizovali lokálne stroje.

6.4.2 Vyladenie zintegrovaného riešenia (pp 8.2)

Po integrácii sa očakávalo s chybami a potrebnými vyladeniami. To bolo náplňou tohto tasku.

6.4.2.1 Analýza

Zadaním predošlej úlohy sme sa rozhodli, že nastala chvíľa kedy dáme dokopy úplne všetko. Čiže k tomu čo sme mali v rezitári bolo potrebné pridať všetky ostatné dôležité vetvy, súbory či kódy, ktoré sme mali inde, hlavne ktoré boli len na našich lokálnych počítačoch. Očakávali sme preto rôzne problémy, ktoré budú nasledovať a bude ich potrebné vyladiť.

6.4.2.2 Implementácia

Po zintegrovaní nášho riešenia sa vyskytli chyby. Asi najvážnejšia boli občasné výnimky hádzané nepravidelne pri hre hráča. Objavili sa aj konkrétne výnimky hádzané pri kope hráča do lopty. Po opravení high skillu „Kick“, ktorý sme mali v tom momente nedokončený sa všetky problémy s výnimkami vyriešili.

Ďalej bolo potrebné nahradiť low skilly volané high skillmi. Vytvorili sme si kvôli tomu google dokument s tabuľkou, kde sme si rozpísali high skilly volané v používaných plánovačoch, low skilly používané v high skilloch a tie sme postupne nahrádzali za novšie, prípadne iné vhodnejšie. Na tomto mieste sme to mali uložené prehľadne a uľahčovali nám to robotu:

Tabuľka 6.11 Používané high skilly a k nim prislúchajúce low skilly

Pouzivany High Skill	Stare Low Skilly	Nove Low Skilly	Gabo: Treba nahradiť za	Nahradil
Beam	rollback	pridal som ho zo starych do novych pohybov		Filip: pridane k novym pohybom.
GetUp	stand_back	existuje v novych pohyboch		
	stand_front	existuje v novych pohyboch		
Localize	head_left_120	existuje v novych pohyboch		
	head_right_120	existuje v novych pohyboch		
	turn_left_cont_20	existuje v novych pohyboch		
	turn_right_cont_20	existuje v novych pohyboch		
	turnleft90_faster_4	existuje v novych pohyboch		
	turnright90_faster_4	existuje v novych pohyboch		
Walk2Ball	stepleft_g	existuje v novych pohyboch		
	stepright_g	existuje v novych pohyboch		
	walk_fine_slow	existuje v novych pohyboch		
	kick_right_faster	existuje v novych pohyboch		
	walkback3	existuje v novych pohyboch		
	walk_fine_fast3	existuje v novych pohyboch		
	turnright45_2	existuje v novych pohyboch		
	turn_right_cont_20	existuje v novych pohyboch		
	turn_right_cont_6.5	existuje v novych pohyboch		
	turnleft45_2	existuje v novych pohyboch		
	turn_left_cont_20	existuje v novych pohyboch		
	turn_left_cont_6.5	existuje v novych pohyboch		
	rollback	pridal som ho zo starych do novych pohybov		Filip: pridane k novym pohybom.
Turn	test_turn_left	existuje v novych pohyboch		
	test_turn_right	existuje v novych pohyboch		

Kick	ball_control3	existuje v novych pohyboch		
	stepright1	pridat zo starych alebo nahradit za niektory z novych step... ? stepright_g?	stepright_g	Filip: Nahradene za stepright_g.
	stepleft1	pridat zo starych alebo nahradit za niektory z novych step... ? stepleft_g?	stepleft_g	Filip: Nahradene za stepleft_g.
	kick_right_faster	existuje v novych pohyboch		
	kick_left_faster	existuje v novych pohyboch		
	kick_right_fast	pridat zo starych alebo nahradit nejakym novym? kick_right_faster?	Máme 4 úrovne ponovom. Slow, normal, faster a "step strong".	Filip: Nahradene za kick_right_normal.
	kick_left_fast	pridat zo starych alebo nahradit nejakym novym? kick_left_faster?	Je to na tvorcovi High-skillu, ktorá sila mu vyhovuje. Za starý fast asi treba dať nový normal.	Filip: Nahradene za kick_left_normal.
	kick_right_normal	existuje v novych pohyboch		Filip: Nahradene za kick_right_slow.
	kick_left_normal	existuje v novych pohyboch		Filip: Nahradene za kick_left_slow.
GoToPosition	stepleft1	pridat zo starych alebo nahradit za niektory z novych step... ? stepleft_g?	stepleft_g	Filip: Nahradene za stepleft_g.
	stepright1	pridat zo starych alebo nahradit za niektory z novych step... ? stepright_g?	stepright_g	Filip: Nahradene za stepright_g.
	walk_fine_back	pridat zo starych alebo nahradit walkback3?	walkback3	Filip: Nahradene za walkback3.
	walk_fine_fast2	pridat zo starych alebo nahradit walk_fine_fast3?	walk_fine_fast3	Filip: Nahradene za walk_fine_fast3.
	turn_left_cont_20	existuje v novych pohyboch		
	turn_right_cont_20	existuje v novych pohyboch		

Na základe týchto všetkých zmien sa vyskytli chyby aj v plánovači našej taktiky. Chyby sme postupne opravili.

6.4.2.3 Testovanie

Testovali sme priebežne pri vyľaďovaní a aj po ňom, aby sme sa presvedčili, že v projekte nemáme žiadne kritické chyby.

6.4.3 Zabezpečenie technickej stránky prezentácie výsledku (pp 8.3)

Na prezentáciu výsledku bolo potrebné zabezpečiť, aby ho bolo možné prezentovať. To si vyžadovalo najmä zaobstaranie technického vybavenia. Týmto vybavením bol router a jednotlivé počítače, na ktorých sa spúšťali jednotliví hráči.

Počítače sa museli nachádzať v rovnakej sieti aj pracovnej skupine. Bolo potrebné povoliť komunikáciu vo firewalloch na jednotlivých počítačoch.

V súbore „*settings.rb*“ bolo navyše potrebné nastaviť nasledujúce hodnoty:

- *PlayerId* : na každom počítači iné číslo, číslovať sme začínali číslom 1
- *TestFramework_monitor_enable* : TRUE
- *TestFramework_monitor_port* : port počítača, kde bol spustený TestFramework

- *TestFramework_monitor_address* : ip adresa počítača, kde bol spustený TestFramework
- *Tftp_port* : na každom počítači iný port, číslovať sme začínali číslom 3071
- *Server_ip* : ip adresa počítača, kde bol spustený server

Sieť sa pred samotným predvedením produktu odskúšala a fungovala bez problémov.

6.4.4 Príprava materiálov na ITSRC (pp 8.4)

Cieľom tejto úlohy bolo zabezpečiť materiály na prezentáciu nášho projektu.

6.4.4.1 Návrh

Na dôstojnú prezentáciu nášho projektu sme sa rozhodli vytvoriť plagát a video. Tieto materiály poukazovali na našu dosiahnutú prácu na projekte.

6.4.4.2 Implementácia

Implementácia spočívala vo vytvorení plagátu, ktorý sme navrhli. Ďalším krokom bolo vytvorenie videí z hrania futbalu, testovania kopov a ich následné spracovanie.

6.4.5 Aktualizovať dokumentáciu (šprinty 5-8) (pp 8.5)

Pred kontrolným bodom bolo potrebné zapracovať nové navrhnuté, implementované alebo zmené veci do dokumentácie. Každý člen tímu zapracoval do dokumentácie svoje úlohy a popisy. Takto vznikla aktualizovaná dokumentácia pripravená na odovzdanie a kontrolu.

6.5 Zhrnutie

Počas riešenia posledných 4 šprintov sa nám podarilo vylepšiť hráča v plánovaných smeroch. Hráč má aktuálne stabilnejšiu chôdzu, je inteligentnejší a vie dať gól. V aktuálnej etape riešenia projektu sme čelili problému zaučenia nových členov tímu, čo sa nám úspešne a bez problémov podarilo a nový členovia sa zúčastňovali práce na úlohách projektu. Implementácia vylepšení hráča odhalila chyby v kóde a v projekte, ktoré sa nám podarilo úspešne odstrániť. Vylepšenia zhrnieme v nasledujúcich bodoch:

- Po úspešnej analýze videnia sveta sme vylepšili model agenta o nové videnia ako šanca získania lopty, úspešnosť prihrávky, som najbližšie k bráne.

- Nové lowskill pohyby a stabilizácia. Máme celkový nový súbor optimalizovaných pohybov, ktoré sú doplnené o ďalšie hodnotné funkcionality. Vznikli takto nové variácie prepojení, ktoré zabezpečujú sofistikovanejší celkový pohyb a taktiku hráča.

- Nové highskill pohyby ako vedenie lopty do brány, prídenie k lopte zo správnej strany, prihrávka na cieľ a držanie formácie.

- Zaoberali sme sa aj prípravám testframeworku na turnaj v robotickom futbale čo pozostávalo z otestovania súčasného stavu a vyriešenia nájdených chýb.

- Nakoniec sa nám úspešne podarilo navrhnúť a implementovať taktiku hráča, ktorá zahŕňa správu viacerých hráčov a využíva pozície hráčov.

- Prebiehali testy integrácie našich vylepšení cez sieťové rozhranie, kde sme zapojili pomocou routera viac počítačov do siete a zahrali sme si futbal.

Celkovo hodnotíme túto etapu projektu ako produktívnu a úspešnú.