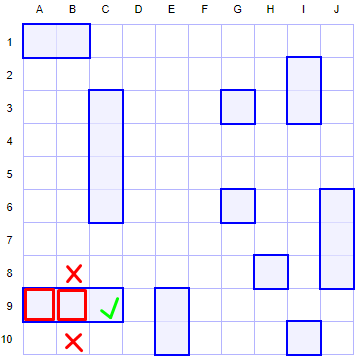
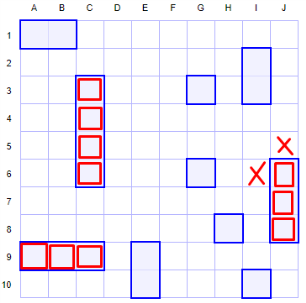
# Sistem baziran na znanju 2019/2020 - predlog projekta

# **Potapanje brodova**

* **Tim:** Filip Mladenović, SW-11/2016
* **Opis problema koji se rešava:** 
  + **Motivacija:** Sa pojavom interneta popularnost ove igrice je značajno porasla. Verujem da je svako nekad sa prijateljem odigrao barem jednu partiju *potapanja brodova*. Deluje kao da sreća igra ključnu ulogu u odnošenju pobede ali, zapravo, nije tako. Dokazaću da je, ipak, igrač sa boljom taktikom taj koji odnosi pobedu.
  + **Pregled problema:** 
    - **Pravila igre**: Igra se igra u dvoje, s tim što će u ovom slučaju čovek da igra protiv računara. Reč je o poteznoj igri na tabli od n\*n polja. Svaki igrač postavlja svoje brodove tako da ih protivnički igrač ne vidi. Brodovi se postavljaju horizontalno ili vertikalno (ne dijagonalno) i ne smeju se preklapati ili izlaziti sa mape. Nakon završetka prve faze, brodovi ne smeju menjati mesta. Nakon toga, biraju se polja na protivničkoj tabli nakon čega se otkriva da li je brod pogođen ili nije. Ako je brod pogodjen protivnik ponovo ima pravo da bira. Pobednik je onaj koji prvi potopi sve protivničke brodove.
    - **Postojeća rešenja**: [pojadnostavljena verzija](http://sr.battleship-game.org/), [složenija verzija](https://www.battleshiponline.org/).
    - **Moje rešenje**: Celokupni sistem postavljanja brodova kao i gađanja protivničkih brodova će se odvijati na osnovu nekih pravila (taktika). Takođe, ukoliko je protivnik poželeo *ponovnu igru* sistem će svoje taktike birati na osnovu poslednje odigrane partije i korigovati je u skladu sa uspešnošću tokom partije.
  + **Metodologija rada:**
    - **Očekivani ulazi u sistem:** Ulaz u sistem su protivnikova tabla (matrica svih pogođenih i promašenih polja) i poslednji odigrani potez (da li je brod pogođen ili ne i na kojim koordinatama).
    - **Očekivani izlazi iz sistema:** Sledeće polje koje se treba gađati.
    - **Opis baze znanja projekta:** Sistem će imativiše predefinisanih baza znanja kojima će se služiti da postavi brodove i da gađa protivnikova polja.
    - **Opis primera:**
      * **Gađanje brodova (primer 1):** Ako su pogođena dva polja za redom u jednom pravcu nemoj da menjaš pravac.



* + - * **Gadjanje brodova (primer 2)**: Ako su pogođena tri polja za redom, a potopljeni je brod od 3 polja i od 4 polja izaberi random poziciju.



* + - * **Prilagodjavanje taktike (primer 1):** Ako je *taktika\_1* u zadnjih 5 poteza imala 4 promašaja izaberi taktiku koja obuhvata negađana polja. (Npr taktika\_1 je ‘gađaj svako drugo polje u donjem desnom kvartu’ - taktika\_2 je ‘gađaj svako trece polje u gornjem desnom kvartu’)
      * **Prilagodjavanje taktike (primer 2):** Ako su pogođena polja takva da preostali brodovi koji treba da se pogode ne mogu da stanu u polja obuhvacena odredjenom taktikom promeni taktiku.

**#** **Napomena:** Opisom taktika sam samo pokušao da obrazložim ideju. Taktika bi trebalo da bude barem 10ak gde će *rule-based-system* birati taktiku kojom će se sistem voditi.

## Pravila

Pravila se oslanjaju na istoriju igrača. Da bi ona došla do punog izražaja potrebno je da je igrač do kraja odigrao barem jednu partiju.

**Za prvu fazu** – fazu postavljanja brodova, rezoner dobija sledeće informacije (*FormationDecision*):

* Predefinisane formacije koje kompjuter može da izabre (*computerFormations*)
* Kompjuterova poslednja korišćena formacija (*computerLastUsedFormation*)
* Kompjuterove najčešće korišćene formacije (*computerMostUsedFormations*)
* Pozicije koje protivnik najčešće gađa (*dangerPositions*)
* Da li je rezoner spreman da donese odluku (*readyForDecision*)
* Formacija brodova koja se postavlja – odluka rezonera (*decision*)

Rezonerova logika:  
\* Postavlja se globalna lista pozicija koja služi da privremeno sačuva parove koji se ne nalaze u jednoj, a možda se nalaze u nekoj drugoj formaciji – npr: Zaključimo da je peto polje iz opasnih pozicija u našoj trenutno izabranoj formaciji, izbacujemo trenutno izabranu formaciju i vraćamo sve proverene pozicije da proverimo da li se nalaze u sledećoj izabranoj formaciji.

1. Ako nije donešena odluka – izbaci formaciju koju si koristio u poslednjoj partiji iz skupa predefinisanih formacija, zaključaj pravilo nakon aktiviranja.
2. Ako nije donešena odluka – izaberi slučajnu formaciju iz skupa preostalih predefinisanih formacija i postavi tu formaciju kao odluku.
3. Ako nije donešena odluka i trenutna odluka je u najčešće korišćenim formacijama – izbaci trenutnu odluku iz skupa predefinisanih formacija i poništi trenutnu odluku. (Ovo će ponovo aktivirati pravilo broj 2 i ostala redom)
4. Ako je donešena odluka, postoje opasne pozicije i prva od tih opasnih pozicija je u odabranoj formaciji – izbaci formaciju, vrati sva proverene opasne pozicije nazad u objekat (kako bi se proverile za sledeću izabranu formaciju) iz globalne promenljive, isprazni globalnu promenljivu i poništi odluku. (Ovo će ponovo aktivirati pravilo broj 2 i ostala redom)
5. Ako je donešena odluka, postoje opasne pozicije i prva od tih opasnih pozicija nije u odabranoj formaciji – premesti prvu opasnu poziciju u globalnu promenljivu, izbriši prvu opasnu poziciju (Ovo će aktivirati pravila 4 i 5 sve dok se ne dobije formacija u kojoj nije ni jedna opasna pozicija – dok se ne isprazne opasne pozicije)

**Za drugu fazu** – faza u kojoj kompjuter treba da bira polje koje želi da napadne.

Pre svakog odabira poteza rezoner prvo proverava koja pravila treba da se upotrebe u zavisnosti od trenutnog stanja sistema i istorije koju je imao sa protivničkim igračem – bira se agenda grupe (*AgendaGroupDecision*). Atributi klase su:

* Pozicije brodova iz poslednje partije (*lastPlayShipsPositions* )
* Najčešće pozicije na kojima se nalaze brodovi (*mostCommonShipPosition*)
* Potezi računara iz trenutne partije (*moves*)
* Da li se u nekom od poslednjih poteza desio pogodak (*checkLastHit*)
* Da li su proverene poslednje pozicije (*checkLastPositions*)
* Da li su proverene najčešće pozicije (*checkCommonPositions*)
* Odluka o pravilima koje treba primeniti – odluka rezonera (*decision*)

Rezonerova logika:

1. Ako nije odabrana agenda grupe, prvi je potez i ne postoje podaci o igraču – znači da je protivnikovo prva partija ikada i da jedino što možemo da uradimo jeste da primenimo pravila za *traženje protivnika*.
2. Ako nije odabrana agenda grupe, nije prvi potez i poslednji potez je bio pogodak – primenjivaće se pravila *nakon pogotka*.
3. Ako nije odabrana agenda grupe, nisu proverene poslednje pozicije, nisu proverene česte pozicije, postoje pozicije brodova iz poslednje partije i prebrojani dosadašnji potezi su 3 puta iskoristili strategiju *CHECK\_LAST\_GAME\_POSITIONS* (znači da smo proverili neke od brodova iz poslednje partije) – označi da su proverene poslednje pozicije brodova.
4. Ako nije postavljena agenda grupe, nisu proverene poslednje pozicije brodova, nisu proverene uobicajne pozicije brodova i postoje brodovi iz prošle partije – primeni pravila za *poslednje pozicije* brodova
5. Ako nije odabrana agenda grupe, proverene su poslednje pozicije brodova, nisu proverene najčešće pozicije brodova, postoje najčešće pozicije brodova i u dosadašnjim potezima se 5 puta primenjivala taktika *CHECK\_COMMON\_POSITIONS* (znači da smo proverili česte pozicije brodova) – označi da su proverene česte pozicije brodova.
6. Ako nije odabrana agenda grupe, proverene su poslednje pozicije brodova, nisu proverene najčešće pozicije brodova i postoje najčešće pozicije brodova - primeni pravila za proveru *najčešćih pozicija* brodova.
7. Ako nije odabrana agenda grupe, proverene su najčešće korišćene pozicije i proverene su poslednje pozicije brodova – primeni taktike za *traženje neprijatelja*.

Rezoneru se, potom, postavlja odabrana agenda grupe. Samim tim se biraju pravila koja će se primeniti na prosleđeni objekat (*MoveDecision* ). Klasu opisuju atributi:

* Pozicije brodova iz poslednje partije (*lastPlayShipsPositions* )
* Najčešće pozicije na kojima se nalaze brodovi (*mostCommonShipPosition*)
* Potezi računara iz trenutne partije (*moves*)
* Da li je spreman za odlku (*readyForDecision*)
* Konačni potez – odluka rezonera (*decision*)

Rezonerova logika ako je odabrana agenda grupe za proveru *poslednjih pozicija* brodova:  
\* Postavljaju se dve globalne liste, jedna za ponovnu proveru poteza, a druga odvaja brodove za mogući izbor.

1. Ako nije odabran potez, postoje brodovi iz prošle partije, postoje potezi, potez se nalazi u prvom brodu i sistem nije spreman da donese odluku – vrati proverene poteze da bi se proverili za sledeći brod i ukloni već proveren brod.
2. U suprotnom (ako potez nije u prvom brodu) – prebaci potez u listu za ponovnu proveru poteza.
3. Ako lista poteza ostane prazna, a postoje neprovereni brodovi – vrati proverene poteze da bi se proverili za sledeći brod, a prvi brod prebaci u listu za neproverene brodove.
4. Ako je prazna lista brodova i odluka nije donešena – sistem je spreman da donese odluku.
5. Za slučaj da smo nekim drugim taktikama proverili sve brodove rezoner će imati praznu listu brodova za odabir – u tom slučaju bira slučajnu poziciju iz slobodnih pozicija.
6. Ako ima brodove koje nije proverio – bira slučajni brod i slučajnu poziciju tog broda.

Rezonerova logika ako je odabrana agenda grupe za proveru *čestih pozicija* brodova:  
\* Postavljaju se dve globalne liste, jedna za ponovnu proveru poteza, a druga odvaja moguće poteze iz čestih pozicija (neodigrane do sada).

1. Ako nije odabran potez, postoje potezi, postoje najčešće korišćene pozicije, trenutni prvi potez je odigran za trenutnu prvu najčešće korišćenu poziciju i sistem nije spreman da donese odluku – izbaci proverenu čestu poziciju brodova i postavi poteze da se provere ponovo.
2. Ovo pravilo je u suprotnosti sa pravilom broj 1, ako trenutno prvi potez nije odigran na trenutno prvoj poziciji – prebaci prvi potez u listu za ponovnu proveru poteza.
3. Ako smo proverili sve odigrane poteze za prvu poziciju često korišćenih pozicija – to znači da ta pozicija još nije odigrana. Prebacujemo trenutno prvu poziciju često korišćenih pozicija u listu slobodnih pozicija i postavljamo poteze da ih opet proverimo.
4. Ako smo ispraznili listu često korišćenih pozicija – to znači da imamo spremnu listu pozicija za odabir. Postavljamo informaciju da je rezoner spreman da donese odluku.
5. Za slučaj da smo nekim drugim taktikama proverili sva polja rezoner će imati praznu listu pozicija za odabir – U tom slučaju bira slučajnu poziciju iz slobodnih pozicija.
6. Ako ima često korišćenih pozicija za odabir – bira slučajnu poziciju od preostalih pozicija.

Potrebno odraditi:

Odabir agende grupe *nakon pogotka* i pravila koja će određivati koje polje sledeće treba birati nakon pogotka. Za slučaj da se desi promašaj i u sledećih par poteza treba koristiti strategije vezane za ovu agendu grupe kako bismo istražili okolinu pogotka i uspeli da, do kraja, potopimo pogođeni brod.

Po odabiru agende grupe za *traženje neprijatelja* pravila koja će smenjivati regione i načine odabira polja.

***Complex event processing***

Pravila koja zabranjuju igraču da igra igricu ako 3 puta u roku od 10 minuta pogreši lozinku svog akaunta (U primeru je vreme čekanja skraćeno zbog lakšeg testiranja).   
\* Event klase su deklarisane unutar .drl dokumenta zbog potencijalnog menjanja pravila od strane administrator (da se unose dozvoljeni broj promašaja u određenom vremenskom intervalu i da se čeka određeni interval)

Kako bi se upravljalo prijavljivanjem igrača napravljena je klasa *SignInDecision* koja sadrži atribute:

* Identifikator igrača (*playerId*)
* Nadimak igrača (*playerNick*)
* Da li je lozinka uneta pravilno (*success*)
* Da li je prijavljivanje zabranjeno – odluka rezonera (*forbiden*)
* Razlog za zabranu prijavljivanja (*reason*)

Rezonerova logika za zabranu prijavljivanja:

1. Ako postoji zabrana za prijavu igrača sa tim identifikatorom – zabrani prijavu.
2. Ako ne postoji zabrana za prijavu igrača i lozinka je pravilno uneta – dozvoli prijavu.
3. Ako je pokušana prijava sa pogrešnom lozinkom – kreira se novi *WrongPasswordEvent.*
4. Ako postoji više od 3 pokušaja prijave igrača sa pogrešnom lozinkom i ne postoji već postojeća zabrana – dodaj zabranu prijave (*PlayerFailToSignIn3TimesEvent*) za tog igrača.

Sistem, takođe, ima pravila koja ograničavaju igračev potez. Nakon jednog minuta sitem obaveštava igrača da treba odabrati potez, a nakon dodatnih 30 sekundi mu oduzima potez. (nije implementiran na frontendu ova funkcionalnost, ali pravila postoje i rade):

1. Kada igrač otpočne potez sistem kreira *PlayerMoveEvent* i *WarningEvent*
2. Kada istekne igračevih jedan minut za potez u sledećih deset sekundi u memoriji se nalazi samo *WarningEvent* – to stanje detektuje sistem i upozorava igrača, u memoriju ubacuje *SwitchMoveEvent* i *PlayerMoveAfterWarningEvent.*
3. Po istom principu se vrši i oduzimanje poteza.
4. Ostala pravila služe da očiste memoriju.

Mislim da zbog same problematike projekta unit testovi nisu adekvatan način za validiranje pravila. Trebalo bi dodatno prilagođavati sistem da bi se iz njih mogla izvući bilo kakva informacija, a pitanje je koliko bi ona bila od koristi. Predlažem da se snimi više ubzanih klipova koji bi jasnije pokazivali rezultate rezonera.