

Dokumentacja projektu zaliczeniowego

Przedmiot: Inżynieria oprogramowania

Temat:	Organizator turniejów komputerów szachowych
Autorzy:	Artur Tamm
Grupa:	I1-222A
Kierunek:	informatyka
Rok akademicki:	2019/20
Poziom i semestr:	I/4
Tryb studiów:	stacjonarne

1 Spis treści

2	Odnosińniki do innych źródeł.....	4
3	Słownik pojęć	5
4	Wprowadzenie	5
4.1	Cel dokumentacji.....	7
4.2	Przeznaczenie dokumentacji	7
4.3	Opis organizacji lub analiza rynku.....	7
4.4	Analiza SWOT organizacji	7
5	Specyfikacja wymagań	8
5.1	Charakterystyka ogólna.....	8
5.2	Wymagania funkcjonalne.....	9
5.3	Wymagania нефункционалне.....	18
6	Zarządzanie projektem	19
6.1	Zasoby ludzkie	19
6.2	Harmonogram prac.....	19
6.3	Etapy/kamienie milowe projektu	19
7	Zarządzanie ryzykiem.....	21
7.1	Lista czynników ryzyka	21
7.2	Ocena ryzyka.....	21
7.3	Plan reakcji na ryzyko	21
8	Zarządzanie jakością.....	21
9	Projekt techniczny	26
9.1	Opis architektury systemu.....	26
9.2	Technologie implementacji systemu	26
9.3	Diagramy UML	26
9.4	Charakterystyka zastosowanych wzorców projektowych.....	39
9.5	Projekt bazy danych	39
9.6	Projekt interfejsu użytkownika.....	40
9.7	Procedura wdrożenia	45
10	Dokumentacja dla użytkownika.....	46
11	Podsumowanie	47
11.1	Szczegółowe nakłady projektowe członków zespołu	47
12	Inne informacje	48

2 Odnośniki do innych źródeł

https://github.com/zut-io-projekt/zadanie_projekt_artur_tamm

3 Słownik pojęć

Elo – Jedna z metod używana do obliczania relatywnych poziomów umiejętności graczy w grach, w tym przypadku używany do porównywania siły silników szachowych.

Glicko - Jedna z metod używana do obliczania relatywnych poziomów umiejętności graczy w grach, w tym przypadku używany do porównywania siły silników szachowych.

Gauntlet – system turniejów w którym gracz o numerze 1 gra mecz (lub serię meczów) z każdym innym uczestnikiem turnieju. Używany do szczegółowego testowania jednego programu.

IPC – inter process communication, tutaj sposób na komunikację z silnikiem szachowym aby pytać go o najlepsze ruchy w danej pozycji. Realizowany za pomocą potoków systemu operacyjnego.

Knockout – system turniejów w którym gracze są ustawieni we wcześniej ustalonej drabince, przegrany każdego meczu odpada. Istnieje również wariant systemu knockout zwany Double Elimination, w którym każdy przegrany ma jeszcze jedną szansę w tzw. drabince przegranych.

Książka (book) – jest to zestaw ruchów który zawiera wcześniej ustalony, subiektywny ruch dla każdej pozycji w książce. Książki mogą być narzucane programom szachowym aby zmusić je do np. grania określonych otwarć, bądź aby po prostu zaoszczędzić czas na obliczaniu ruchów. Wskazane jest zaczynanie meczów w turnieju z książki, tak aby uzyskać jak największą różnorodność pozycji, co umożliwi testowanie programów w jak największej ilości aspektów.

Likelihood of Superiority – LOS, wskaźnik który wskazuje jaka jest szansa że gracz1 jest silniejszy niż gracz2. Obliczany na podstawie wyników meczu, wskaźnik na poziomie 100% oznacza że któryś z graczy jest na pewno silniejszy od drugiego.

LLR – *Log Likelihood ratio* – Test statystyczny, używany do wcześniejszego zakończenia testu w przypadku gdy wiemy że np. aktualizacja nie przynosi korzyści dla programu.

NN engine – *Neural Network engine*, czyli w kontekście programów szachowych jest to program który używa tzw. sieci neuronowych do obliczania ruchów, w przeciwieństwie do algorytmu Alpha-Beta w tradycyjnych programach szachowych. Programy takie mocno zyskują na sile jeżeli używają kart graficznych (GPU) w przeciwieństwie do procesora (CPU).

Leela Ratio – Wskaźnik używany do porównania relatywnej szybkości sprzętu GPU do CPU. Wykalibrowany na 1.0 dla stosunku 1/875 (tj na każdą pozycję przeszukaną przez

NN, program klasyczny przeszukuje 875 pozycji). Zakłada się że jeżeli Leela Ratio jest pomiędzy 0.7-1.0 to warunki sprzętowe są mniej więcej równe. Większa liczba oznacza że GPU jest relatywnie szybsze niż CPU, a więc programy NN mają przewagę w sprzecie. Mniejsza liczba oznacza że CPU jest relatywnie szybsze od GPU, a więc tradycyjne programy Alpha Beta mają przewagę.

Round Robin, RR - system w którym każdy gra przeciwko wszystkim pozostałym przeciwnikom conajmniej raz, „każdy z każdym”. Cechuje się dokładnością wyników, jednakże zajmuje bardzo dużo czasu do zakończenia, jako że liczba gier rośnie bardzo szybko wraz z liczbą graczy.

Silnik Szachowy – program komputerowy stworzony do gry w szachy. Program taki musi przestrzegać tzw. protokoły czyli tak jakby normy które określają jaką funkcjonalność musi dany silnik posiadać. Silniki posiadają opcje do znalezienia „najlepszego ruchu” w danej pozycji (bestmove) oraz do nieskończonej analizy danej pozycji.

Swiss, system szwajcarski – system turnieju w którym nikt nie jest eliminowany, który używany jest do ustalenia względnej kolejności uczestników, jednakże bez grania meczów w systemie każdy z każdym. W każdej rundzie uczestnicy grają z innymi przeciwnikami którzy mają podobną, lub taką samą liczbę punktów, jednakże nie więcej niż raz. Turniej ten ma wcześniej ustaloną liczbę rund, jednakże może skończyć się wcześniej gdy będziemy mieli już jasnego zwycięzcę.

TC – *Time control* – Jest to licznik czasu, który ustala jak dużo czasu ma dany gracz. Może być to np. czas na ruch (np. 5 minut na ruch), czas na całą grę (np 300 minut na x ruchów bądź całą partię), oraz tzw. inkrement, czyli ilość sekund dodawana co każdy ruch.

TB – (endgame) *tablebases* – zestaw matematycznie udowodnionych pozycji do maks 7 figur, który zawiera najlepsze ruchy dla wszystkich istniejących pozycji z 7 figurami lub mniej. Do „figur” wliczają się także obaj króle. Każda pozycja zawiera informacje o najlepszym ruchu, oraz informację czy pozycja jest wygrana, przegrana lub zremisowana przy perfekcyjnej grze. TB zawierają wiedzę obiektywną z której każdy program może zawsze korzystać do zwiększenia jego siły, w przeciwieństwie do książek które zawierają widzę subiektywną.

4 Wprowadzenie

4.1 Cel dokumentacji

Celem dokumentacji jest pomoc w organizacji pracy dla programistów. Zawarte tu są szczegóły zarówno implementacyjne, jak i szczegóły dotyczące działania.

4.2 Przeznaczenie dokumentacji

Dokumentacja ta będzie niezbędnikiem dla każdego programisty który ma zamiar pracować nad tym projektem. Zawiera ona wszystkie diagramy np. klas i opis funkcjonalności. Dodatkowo zawiera ona szczegółowe objaśnienie rzeczy, tak aby osoby tworzące projekt nie miały wątpliwości co do zasad działania.

4.3 Opis organizacji lub analiza rynku

System ten jest przeznaczony do sprzedaży na rynek fascynantów szachowych. Istnieje wiele programów szachowych i wielu programistów, którzy będą mogli używać naszego organizatora turniejów do skomplikowanego testowania. Również sami gracze grający w szachy będą mogli używać programów do analizy swoich gier przy użyciu różnych programów szachowych. Popularność szachów, jak i komputerowych szachów utrzymuje się na mniej więcej stałym poziomie od wielu lat oraz aktualnie nic nie prognozuje zmiany. Prostota i zaawansowana funkcjonalność mojego programu powinna być wartościowym atutem każdego fascynanta szachowego.

4.4 Analiza SWOT organizacji

Silne:

- zapotrzebowanie na szczegółowe analizy partii szachowych wśród graczy.
- zapotrzebowanie na precyzyjne narzędzia statystyczne wśród programistów silników.

Słabe:

- dosyć niskie zainteresowanie programami szachowymi

Szanse:

- rosnące zainteresowanie szachami dzięki szachom online i streamerom

Zagrożenia

- powstanie darmowych alternatyw

5 Specyfikacja wymagań

5.1 Charakterystyka ogólna

5.1.1 Definicja produktu

Program ten umożliwi szczegółowe badanie silników szachowych w różnych turniejach.

5.1.2 Podstawowe założenia

Najważniejszymi założeniami są:

- prostota w użyciu
- szczegółowe statystyki
- duża różnorodność formatów rozgrywek
- duże możliwości analizy

Trudność użycia jest zdecydowanie największym problemem który odpycha użytkowników nie znających się na komputerach od szachów komputerowych. Dlatego program powinien zawierać wbudowane wcześniej programy, tak aby użytkownik mógł jednym lub dwoma kliknięciami wszystko zorganizować. Dodawanie nowych programów musi również być bardzo proste, np poprzez przeciągnięcie pliku wykonywalnego silnika na okienko aplikacji. Jednakże prostota ta nie może wykluczać funkcjonalności. Dla bardziej zaawansowanych użytkowników powinien być dostępny szereg możliwości, które pozwolą bardzo precyzyjne wykalibrowanie programów do ich własnego upodobania. Powinien być również dostępny szereg statystyk matematycznych do każdego turnieju, z łatwo dostępnymi okienkami pomocy które tłumaczą znaczenie każdej z nich. Ważne również jest oczywiście uwzględnienie błędów statystycznych aby dopełnić te wskaźniki.

Oprócz funkcjonalności która zadowoli fascynantów szachów komputerowych, program musi być także użyteczny dla zwykłych graczy. Powinien on zawierać szczegółowe możliwości analizy, np poprzez uruchomienie w silnikach szachowych wyszukiwania na każdej pozycji oraz wskazanie błędów które obie strony popełniły w meczu. Ocena pozycji jak i możliwe ruchy z książki są zdecydowanie niezbędne.

5.1.3 Cel biznesowy

Chcemy tchnąć nowe życie w rynek komputerów szachowych. Wiele programów do turniejów jest albo trudne w użyciu, albo brakuje im wielu funkcji. Chcemy stworzyć jeden program który będzie zawierał wszystko czego fascynant szachów może chcieć.

5.1.4 Użytkownicy

0. Gracze w szachy
1. Programiści piszący programy szachowe
2. Fascynanci programów szachowych

5.1.5 Korzyści z systemu

1. Ludzie grający w szachy mogą używać naszego programu do analizowania partii które rozegrali. Silniki szachowe są w stanie przeanalizować każdą pozycję

- podczas gry, oraz powiedzieć co możnaby zrobić lepiej. Dodatkowo istnieje możliwość pokazania np 3-4 linii na każdy ruch, tak aby lepiej zrozumieć każdą pozycję. Dodatkowo silniki szachowe są w stanie wycenić każdą pozycję przy użyciu tzw. centypionków, lub w przypadku programów NN, procentowej szansy na zwycięstwo. Dodatkowo używając łatwo dostępnych opcji handicapu, oraz mniej rozbudowanych, bardziej „ludzkich” sieci neuronowych, są w stanie zagrać partie przeciwko różnym przeciwnikom aby przeciwiczyć nauczone koncepty w akcji.
2. Programiści piszący programy szachowe są w stanie użyć naszego programu do szczegółowego testowania ich własnych dokonań. Mogą stworzyć bez problemu turnieje o wielu różnych formatach, a szczegółowe statystyki podające LOS, LLR, Elo, Glicko oraz własny, autorski system rankingu są w stanie precyzyjnie określić siłę każdego programu. Dodatkowo narzędzia do analizy używające najlepszych programów na świecie są w stanie wskazać im miejsca, w których ich programy sobie nie radzą. Turnieje typu gauntlet są idealne do testowania jednego programu przeciwko wielu przeciwnikom.
 3. Fascynanci programów szachowych są bardzo podobną grupą do programistów, jednakże oni są bardzo często mniej zainteresowani analizą gier, a bardziej wynikami które dane programy osiągają. Niezwiązani z żadnym programem, ich celem zazwyczaj jest organizowanie miniturniejów, nierzadko w bardzo nierównych ustawieniach. Nasz program pozwala narzucać komputerom ruchy z książki stworzonej przez użytkownika, zmuszając je do grania najróżniejszych otwarć, co pozwala urozmaicić rozgrywkę.

5.1.6 Ograniczenia projektowe i wdrożeniowe

Program będzie dostarczany wraz z trzeba open source programami, Stockfish, Leela i Leelenstein. Każdy z tych programów jest oparty na licencji GPL, dlatego trzeba uwzględnić link do kodu źródłowego każdego z tych programów.

Program będzie pisany dla systemu Windows w języku C++17. IPC będzie realizowane przy użyciu potoków w WinApi na systemie Windows10 kompatybilne z Windows7. Jest to spowodowane tym, że zdecydowana większość użytkowników używa Windowsa, Linux jest zbyt mało popularny aby poświęcać ważny czas deweloperski na funkcjonalność na tym systemie. Możnaby jednak upewnić się, że program działa na WINE.

IPC jest niezbędne do komunikacji z programami szachowymi, które wypisują wyniki swoich działań na stdout. Potok musi być dwukierunkowy, musimy być w stanie powiadomić program o aktualnej pozycji w grze, tak jak i musimy być w stanie odczytać jego output.

Programy szachowe używają protokołów UCI oraz xboard, ważne jest abyśmy zaimplementowali wsparcie dla obu z nich. Xboard może już trochę odchodzić na dalszy plan, jednakże kilka czołowych programów (np Scorpio) nadal go używa.

Program musi być niezawodny w organizacji procesów programów, kończąc te które nie są już używane. W przeciwnym wypadku, może to powodować znaczne spowolnienie systemu.

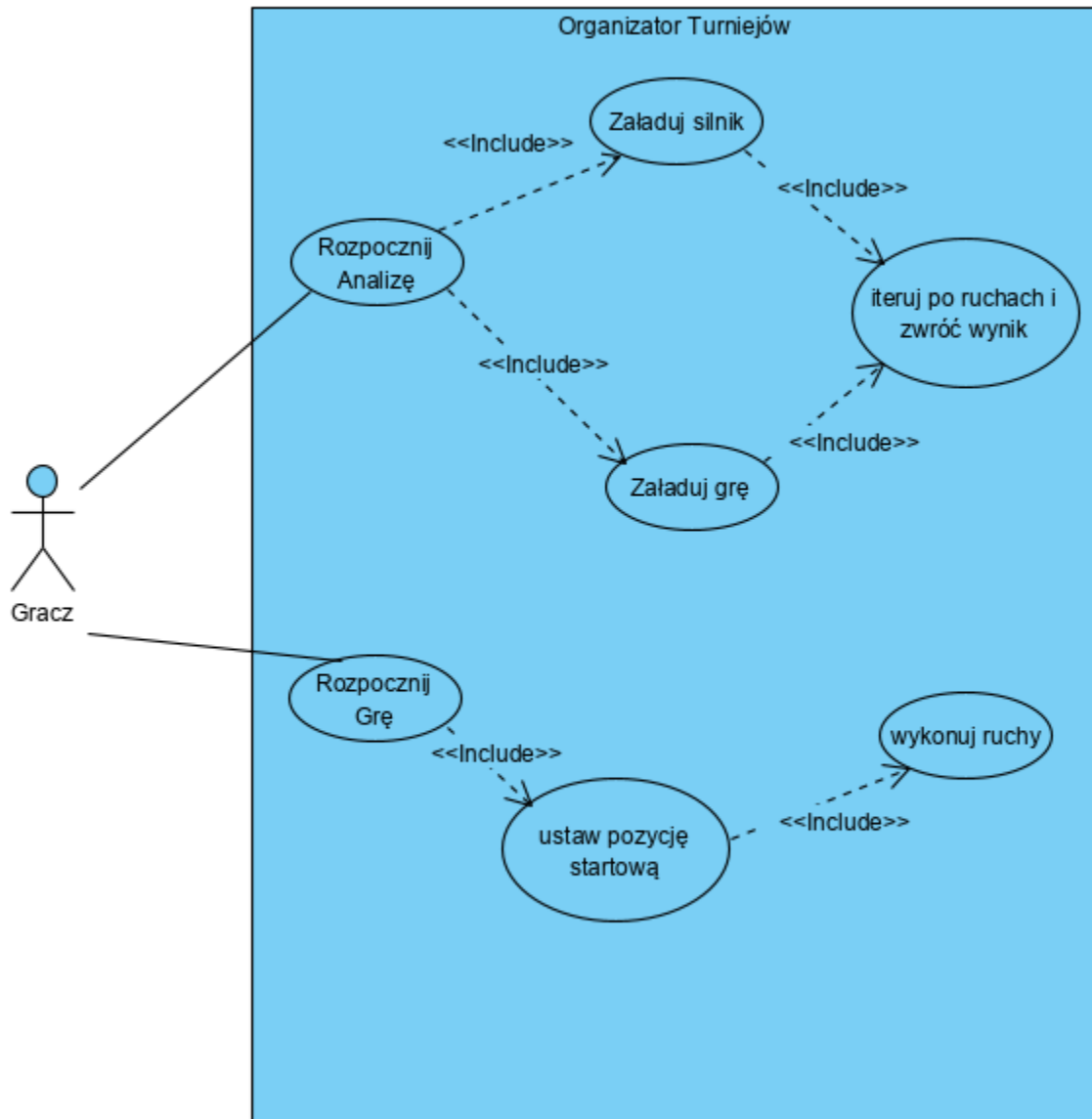
5.2 Wymagania funkcjonalne

5.2.1 Lista wymagań

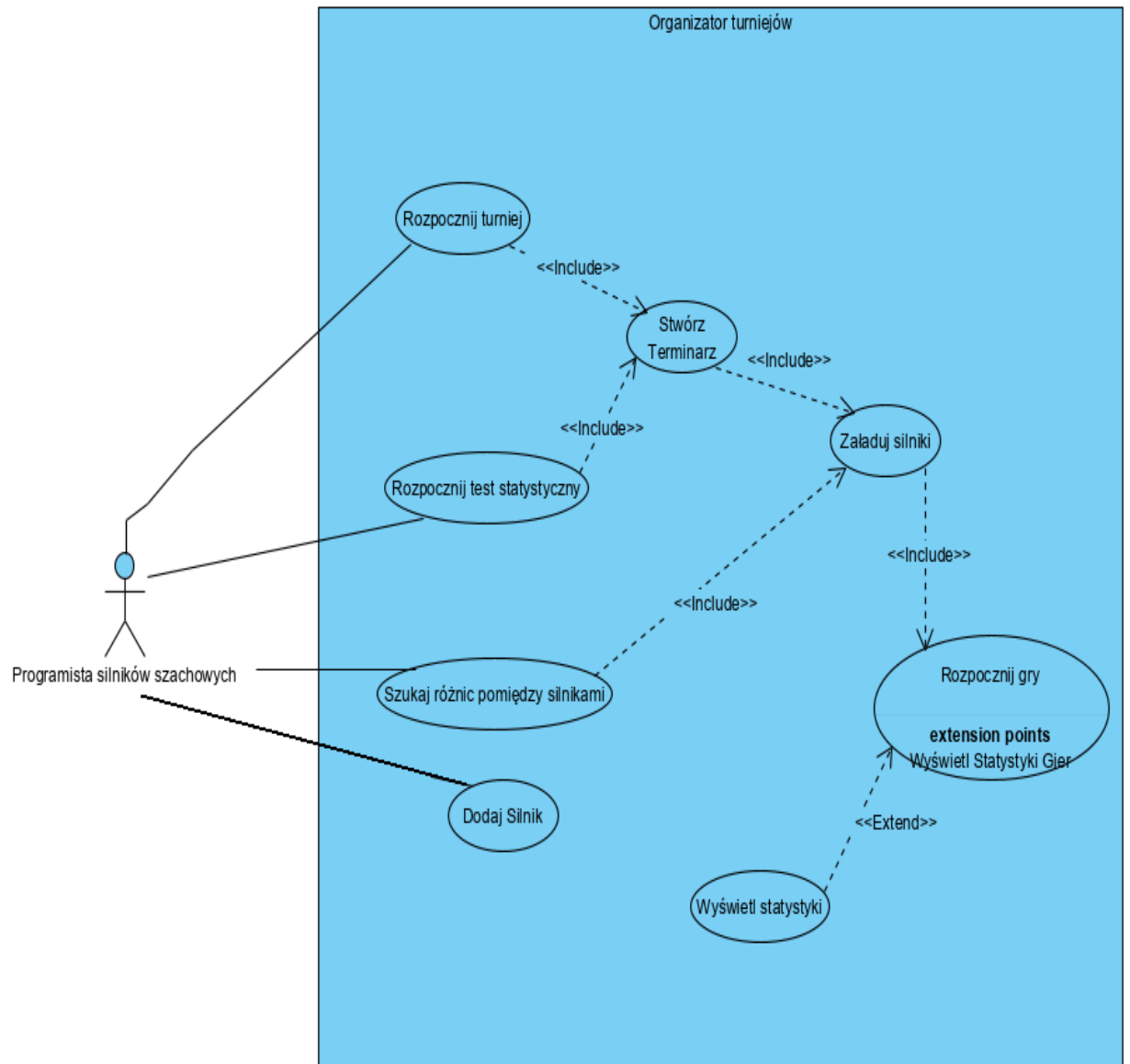
1. System powinien umożliwiać tworzenie turniejów silników szachowych
2. System powinien umożliwiać analizę partii
3. System powinien umożliwiać matematyczne statystyki do meczów
4. System powinien umożliwiać organizację silników kilkoma kliknięciami

5.2.2 Diagramy przypadków użycia

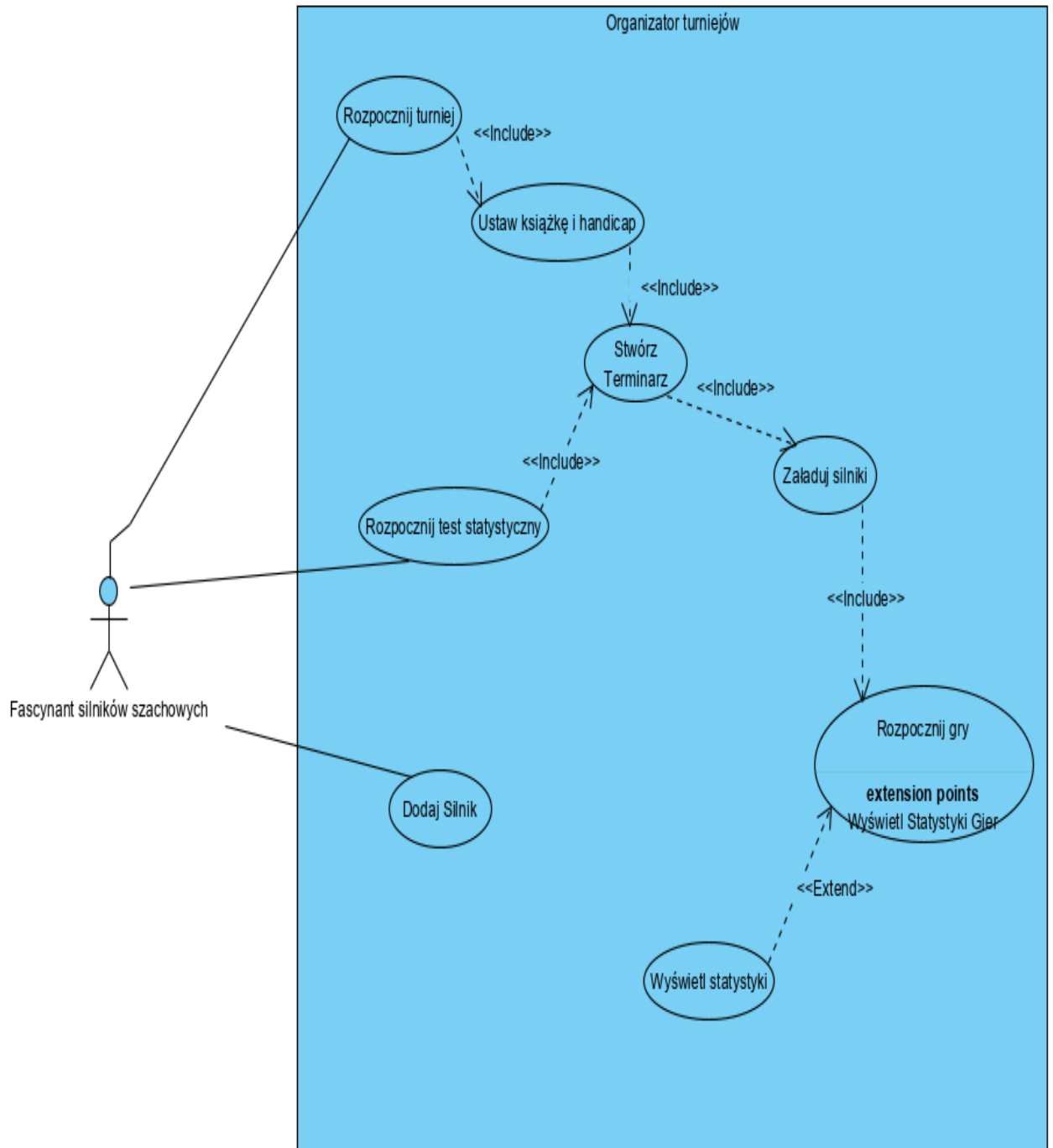
Przypadki użycia dla zwykłych graczy:



Przypadki użycia dla programistów silników szachowych:



Fascynanci programów szachowych:



1. Tworzenie turniejów

Użytkownicy: Programiści i Fascynanci (2 i 3)

Uzasadnienie biznesowe:

Programy które umożliwiają tworzenie turniejów silników szachowych są niezbędne do testowania ich siły. Programy takie zawsze znajdą swoje użycie, czy to aby sprawdzić który z nich jest najlepszy w danej sytuacji, czy to aby porównać ich siłę w danym otwarciu. Dziedzina szachów komputerowych bez przerwy ewoluje, dlatego porównywanie programów w boju nigdy nie przestanie być potrzebne. Nasz program oferuje bardzo dużo narzędzi, które pozwolą precyzyjnie określić ich siłę.

Scenariusze:

1. Rozpoczęcie turnieju

Warunek początkowy: Użytkownik znajduje się na ekranie głównym, żaden turniej nie jest rozpoczęty.

- Użytkownik najpierw klika odpowiedni guzik do rozpoczęcia turnieju

- Użytkownik wybiera programy szachowe które chce testować, oraz wybiera tryb testowania

- Użytkownik ustawia parametry turnieju, TC, książkę, oraz ewentualnie modyfikuje ustawienia silników szachowych

- Program tworzy terminarz, oraz rozpoczyna rozgrywkę

Efekt: turniej się rozpoczyna, pojawia się tabelka z aktualnymi wynikami, bądź drabinka turnieju

Wymagania niefunkcjonalne:

- Przerwa pomiędzy ruchami programów musi być możliwie mała, nie powinna być dłuższa niż 0.2 sekundy.

- Wszystkie procesy powinny zostać niezwłocznie zakończone po zakończeniu meczu.

- Wszystkie silniki szachowe powinny uzyskać komendę do startu nie dłużej niż 20 ms po rozpoczęciu odliczania czasu

- Každy ruch musi być sprawdzany czy przestrzega zasad gry w szachy, mecze powinny być kończone natychmiastowo według tychże reguł

Częstotliwość i istotność: **5, jest to główna funkcjonalność programu**

2. Analiza Partii

Użytkownicy: Gracze w szachy (1) i Programiści (2)

Uzasadnienie biznesowe:

Gra w szachy jest jedną z najbardziej popularnych gier planszowych na świecie. Gra znana jako „królewska” przysporzyła sobie wielu fanów. Wielu z tych ludzi chce nauczyć się grać w tę grę. Jednym z najlepszych sposobów jest uczenie się na błędach, oraz uczenie się od lepszych, a nie ma lepszych graczy niż silniki szachowe. Dodatkowo programiści mogą używać innych programów szachowych do szukania wad w ich własnych programach. Nasze narzędzie przyda się tym wszystkim ludziom w samoudoskonalaniu się.

Scenariusze:

1. Ładowanie gry do analizy

Warunek początkowy: Użytkownik znajduje się na ekranie głównym, dana gra jest załadowana, żadna gra nie jest rozpoczęta.

-Użytkownik klika guziki „załaduj grę”, bądź wprowadza ruchy własnoręcznie.

-Użytkownik następnie wybiera program/y do analizy, oraz klika guzik „analizuj grę”, bądź „analizuj pozycję”

-Program przekazuje pozycję/pozycje do silnika szachowego, oraz po określonym czasie analizy pokazuje wynik analizy.

-Użytkownik może wykonywać własne ruchy i po analizie program zwróci wyniki tejże analizy.

Efekt: Zostaje wypisany najlepszy ruch, oraz potencjalnie inne, możliwe ruchy. Zostaje pokazana linia którą komputer miał na myśli oceniając pozycję. Dodatkowo użytkownik może wykonywać kolejne, własne ruchy w czasie rzeczywistym

Wymagania нефunkcjonalne:

-Program powinien w czasie rzeczywistym powiadamiać użytkownika na temat aktualnych wyników analizy

-Program powinien bez przerwy analizować pozycje, tryb analizy według protokołów uci i xboard jest trybem nieskończonym dopóki użytkownik go nie zakończy

Częstotliwość:3

Istotność: 4

3. Matematyczne statystyki

Użytkownicy: Programiści (2), Fascynanci (3)

Uzasadnienie biznesowe:

Programiści nigdy nie mogą być pewni czy aktualizacja którą zamierzają wprowadzić jest dobra czy nie. Im bardziej skomplikowany silnik, tym mniej rzeczy jest oczywiste i tym bardziej potrzebne są statystyczne testy. Nasz program posiada wszystkie niezbędne narzędzia statystyczne które potrafią określić wszystko na temat siły porównywanych programów.

Scenariusze:

1. Użytkownik klika w okno ze statystykami

Wymagania początkowe: Turniej jest rozpoczęty, conajmniej jedna gra się zakończyła.

Efekt: Okienko pojawia się i wyświetla dane statystyczne aktualnego wydarzenia (turnieju).

Wymagania niefunkcjonalne:

-Program powinien po każdej grze obliczać odpowiednie testy statystyczne

Częstotliwość:1

Istotność: 2

4. Organizacja silników kilkoma kliknięciami

Użytkownicy: Programiści, Fascynanci (1,2,3)

Uzasadnienie biznesowe:

Jednym z głównych problemów dzisiejszych programów do organizacji turniejów jest trudność zarządzania silnikami dla przeciętnego użytkownika. Wiele programów wymaga grzebania w configu i męczenia się z literówkami/syntaxem komend. Nasz program automatyzuje ten proces dla przeciętnego użytkownika, podczas gdy ci bardziej zaawansowani będą mogli łatwo ustawić odpowiednie opcje.

Scenariusze:

1. Użytkownik klika przycisk dodawania nowego silnika.
Wymagania początkowe: brak, nowy silnik można dodać w każdej chwili.
 - Pojawia się okienko dodawania silnika
 - Użytkownik podaje ścieżkę do pliku LUB użytkownik przeciąga plik w nowo powstałe okienko.
 - Silnik zostaje dodany do listy silników i zostaje automatycznie skonfigurowany
 - Użytkownik modyfikuje opcje i wychodzi z okna.

Efekt: Silnik zostaje dodany do listy skonfigurowanych silników, można go już używać do gier/analizy

Wymagania niefunkcjonalne:

- Program powinien automatycznie skonfigurować każdy silnik używając ustawień domyślnych które każdy silnik dostarcza protokołem uci bądź xboard.
- Każdy silnik musi zostać zapisany na zawsze.

Częstotliwość:2

Istotność: 5

5.3 Wymagania niefunkcjonalne

1. Wydajność.

Program nie może spędzać więcej niż 50 milisekund na odczytywaniu ruchów zrobionych przez silnik. Po otrzymaniu komendy „bestmove” należy w 50 milisekund lub szybciej wykonać ruch na szachownicy, zatrzymać zegar oraz zlecić wykonanie kolejnego ruchu przeciwnikowi.

Program nie może spędzać zbyt dużo czasu na wykrywaniu rezultatu gry, powinno się to zmieścić w wyżej wymienionych 50 milisekundach.

Tworzenie terminarza powinno nastąpić szybko, nie dłużej niż 1 sekunda nawet dla największych turniejów (100 uczestników).

2. Istnieje ryzyko że wyciekną gry które użytkownik zagrał np analizując partie przygotowując się do turnieju. Reszta danych jest niegroźna i ich wyciek nie jest niebezpieczny.

Nasz system jest niegroźny dla innych programów, nie istnieje żadne inne ryzyko.

3. Aby uniknąć sytuacji wycieku danych, program będzie używał algorytmu RSA aby zaszyfrować gry w bazie danych na życzenie użytkownika. Klucz będzie przechowywany w bezpiecznym miejscu w chmurze bądź na systemie użytkownika. W takiej sytuacji, nawet jeśli nastąpi włamanie i wyciek gier to nie będzie możliwe ich odczytanie bez klucza.

4. Program powinien używać bardzo mało pamięci RAM, mniej więcej 2 MB.

Program powinien być bardzo prosty w obsłudze dla początkującego

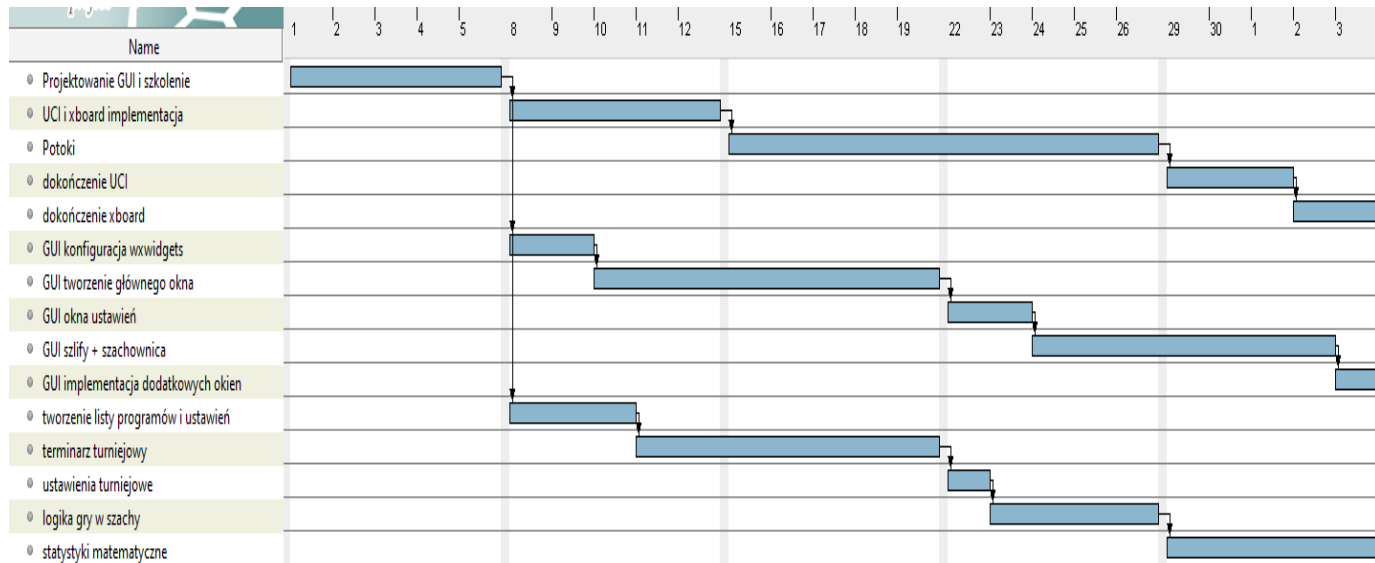
Program nie powinien nigdy się zawieszać, ani nie powinien się crashować. Powinien bezproblemowo odczytywać TB, oraz powinien bezproblemowo inicjalizować silniki NN. Ustawienie parametrów takich jak Leela Ratio powinno nastąpić w nie dłużej niż 10 milisekund.

6 Zarządzanie projektem

6.1 Zasoby ludzkie

Do programu potrzeba conajmniej trzech zespołów złożonych z około 2-3 ludzi. Jeden zespół zajmie się szatą graficzną jak i interfejsem, drugi zespół zaprogramuje turniejową powłokę, a trzeci zespół zajmie się IPC w protokołach uci i xboard.

6.2 Harmonogram prac



6.3 Etapy/kamienie milowe projektu

Etap 1. Projektowanie GUI, ewentualne przeszkolenie w normach formatów turniejów oraz protokołów uci/xboard

Etap 2. Rozpoczęcie prac nad GUI i równoległe rozpoczęcie prac nad protokołami UCI i xboard jak i rozpoczęcie prac nad turniejową funkcjonalnością.

Etap 3. Dla zespołu GUI: ukończenie ogólnego okna.

Dla zespołu turniejowego, ukończenie programu tworzącego terminarz

Dla zespołu IPC, stworzenie niezawodnego potoku do komunikacji

Etap 4. Dla zespołu gui, implementacja okien ustawień

Dla zespołu turniejowego, ukończenie części odpowiedzialnej za ustawienia turnieju

Dla zespołu IPC, ukończenie podstawowych komend uci

Etap 5. Dla zespołu GUI, szlifowanie i implementacja samej szachownicy

Dla zespołu turniejowego, implementacja samej gry w szachy i zasad

Dla zespołu IPC, dokończenie protokołu uci

Etap 6. Dla zespołu GUI, dokończenie wszystkich okien

Dla zespołu turniejowego, dokończenie zasad i implementacja statystyk matematycznych

Dla zespołu IPC, stworzenie protokołu xboard

7 Zarządzanie ryzykiem

7.1 Lista czynników ryzyka

1. Powstanie nowej technologii np nowego protokołu komunikacji z silnikiem szachowym może sprawić że nasz program nie będzie używalny z najnowszymi silnikami.
2. Zmiana zasady działania WinApi w przyszłych aktualizacjach systemu Windows sprawi że program nie będzie działał na nowszych systemach operacyjnych.
3. Powstanie programu darmowego, bądź open source podobnego do naszego programu.
4. Utrata zainteresowania szachami komputerowymi.

7.2 Ocena ryzyka

1. Jest to bardzo mało prawdopodobne, uci jest bardzo rozpowszechnione i sprawdza się bardzo dobrze.
2. Tak jak powyżej, jest to mało prawdopodobne jako iż Microsoft zazwyczaj dba o kompatybilność wsteczną jeżeli chodzi o WinApi, jednakże nie jest to gwarantowane.
3. Jest to możliwe, jednakże w dzisiejszych czasach nie ma zbyt wielu programistów silników szachowych którzy zajmowali by się programami do organizacji turniejów. Wszyscy używają tych samych, starych narzędzi co jest na pewno szansą, jednakże trzeba się liczyć z tym że być może kiedyś się to zmieni. Ludzie stojący przed wyborem płatnego i bezpłatnego programu, zawsze wybiorą ten bezpłatny.
4. Biorąc pod uwagę stałe zainteresowanie na przełomie dziesiątek lat, oraz coraz to bardziej rosnącą popularność sieci neuronowych i AI jako ogółu, szachy komputerowe mają jeszcze długi czas zanim ludzie stracą zainteresowanie. Jest to bardzo mało prawdopodobne że taka sytuacja nastąpi.

7.3 Plan reakcji na ryzyko

1. W takim przypadku będzie trzeba napisać od zera nowy protokół i wydać aktualizację, trzeba będzie również powiadomić użytkowników o istnieniu tejże aktualizacji.
2. W takim przypadku, również trzeba będzie napisać nową wersję programu dla nowszych systemów. Przez 10 lat będziemy kontynuować wsparcie dla starszej wersji na starszy system Windows, po czym wsparcie zostanie wstrzymane i kolejne aktualizacje na starszy system nie będą wychodziły
3. W takim przypadku będziemy musieli rozszerzyć funkcjonalność i wyszlifować nasz program w strefach w których program grożący nam wyparciem z rynku jest słabszy. Dodatkowo trzeba będzie mądrze manipulować ceną, tak aby potencjalni klienci jednak skłonili się ku naszemu produktowi
4. W takim przypadku będziemy musieli rozszerzyć naszą funkcjonalność dla zwykłych graczy, dodanie czegoś w stylu auto trenera dla graczy bądź inną funkcjonalność

8 Zarządzanie jakością

8.1. Scenariusze i przypadki testowe.

ID Testu	Nazwa	Tester	Termin	Opis	Kategoria
1	PERFT(8)	Zespół od turniejów	Po ukończeniu logiki gry w szachy	<p>Funkcja PERFT generuje wszystkie możliwe ruchy od pozycji początkowej do określonej głębokości (depth). Następnie wypisywana jest ilość wszystkich ruchów. Na przykład PERFT(1) = 20 PERFT(2) = 20</p> <p>Test ten ma obliczyć ilość wszystkich możliwych ruchów na depth=8. Wynik powinien być równy 84,998,978,956</p>	Testowanie modułu

Dane: Testerzy dają 8 jako parametr do funkcji PERFT, wynik zwrócony powinien wynosić 84,998,978,956. Każda inna wartość oznacza błąd w logice gry i oznacza że test nie zostaje zaliczony.

2	UCI test	Zespół od IPC	Po ukończeniu prac nad IPC	Silnik Stockfish jest silnikiem który spełnia protokół UCI w 100%. Oznacza to, że wszelkie błędy w komunikacji są spowodowane błędem w naszym programie. Należy więc rozpocząć test 20000 gier z odpowiednio zróżnicowanej książki. Warunkiem zaliczenia testu jest brak niezgodności w pozycjach we wszystkich tych grach. Stockfish powinien mieć depth 5 na ruch.	Testowanie modułu
---	----------	---------------	----------------------------	--	-------------------

Dane: Parametrem jest książka 20000 pozycji. Program zwraca stosunek zgodności pozycji w programie z pozycjami programu Stockfish na każdej pozycji która wyniknie z tych 20000 gier. Test zostaje zaliczony jeżeli będzie 100% zgodności.

3	xboard test	Zespół od IPC	Po ukończeniu prac nad IPC	Silnik Scorpio jest silnikiem który spełnia protokół xboard w 100%. Oznacza to, że wszelakie błędy w komunikacji są spowodowane błędem w naszym programie. Należy więc rozpocząć test 20000 gier z odpowiednio zróżnicowanej książki. Warunkiem zaliczenia testu jest brak niezgodności w pozycjach we wszystkich tych grach. Scorpio powinien mieć depth 5 na ruch.	Testowanie modułu
---	-------------	------------------	-------------------------------------	---	----------------------

Dane: Parametrem jest książka 20000 pozycji. Program zwraca stosunek zgodności pozycji w programie z pozycjami programu Scorpio na każdej pozycji która wyniknie z tych 20000 gier. Test zostaje zaliczony jeżeli będzie 100% zgodności.

4	GUI stress test	Zespół od GUI	Po ukończeniu prac nad GUI	<p>Bardzo wiele testów statystycznych silników odbywa się na bardzo szybkich TC. Oznacza to, że bardzo często wyniknie sytuacja w której będzie bardzo mało czasu na aktualizację GUI. Program musi pozostać stabilny w takiej sytuacji i nie crashować.</p> <p>Test będzie polegał na rozpoczęciu 1000 gier z TC 0.01 sekundy na ruch dla obu stron. Silnik Stockfish będzie idealny do tego zadania, jako że zawiera on bardzo dobrą implementację i jest w stanie poradzić sobie z tak szybkim TC. Warunkiem zaliczenia testu jest brak problemów graficznych po zakończeniu gier (GUI musi pozostać w pełni funkcjonalne i nie pomijać ruchów).</p>	Testowanie modułu
---	-----------------	---------------	----------------------------	---	-------------------

Dane: Tester rozpoczyna turniej 1000 gier stockfish vs stockfish na TC 0.01 sekundy na ruch.

9 Projekt techniczny

9.1 Opis architektury systemu

Program będzie używał komunikacji międzyprocesowej pomiędzy plikami silników a plikiem programu.

Program zawiera następujące warstwy:

- GUI
- logika gry
- logika turniejów

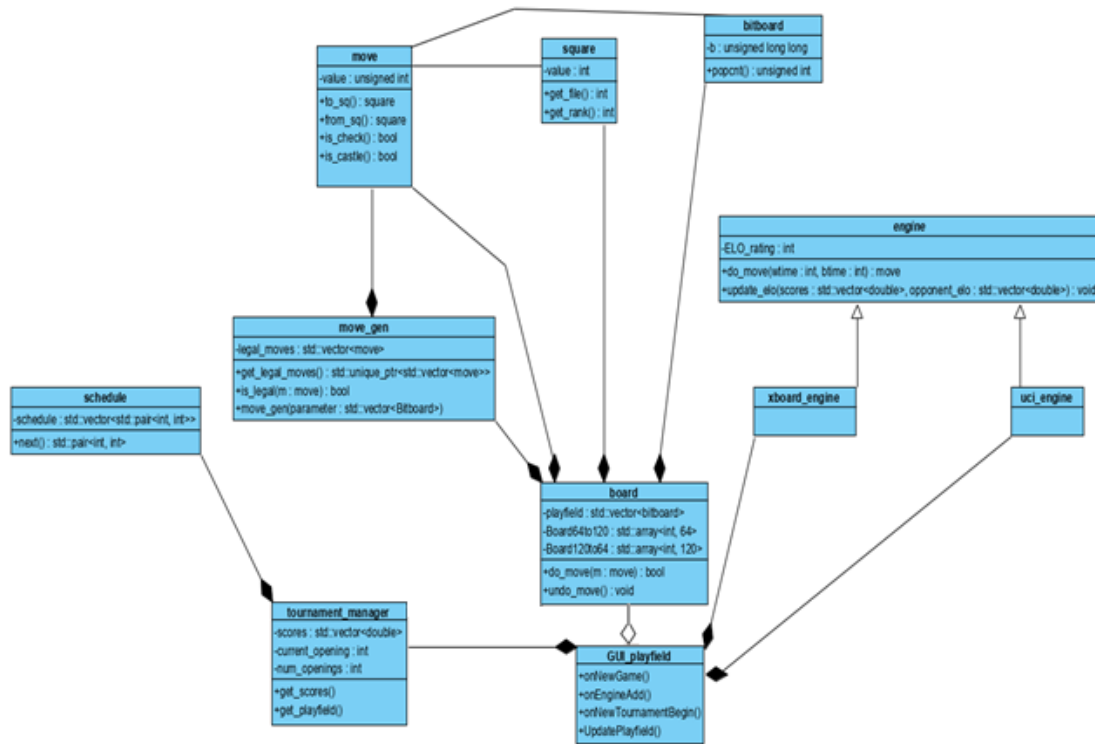
Dodatkowo program zawiera bazę danych w której znajdują się wszystkie rozegrane w programie gry, oraz wszystkie ustawienia skonfigurowanych silników.

9.2 Technologie implementacji systemu

C++17	Program będzie stworzony w języku C++17. Jest to bardzo szybki język, oraz jest obiektowy co umożliwia bardzo szybką i sprawną pracę. Wiele elementów naszego programu bardzo zależy na szybkim działaniu, język taki jak C++ jest w stanie podołać temu zadaniu.
wxWidgets	Biblioteka wxWidgets jest bardzo elastyczną i prostą w użyciu biblioteką graficznego interfejsu kompatybilną z C++. Umożliwia to dużo szybszą pracę aniżeli męczenie się z WinApi, ponieważ wiele rzeczy jest zaimplementowana bardzo prosto. Dodatkowo, jeżeli powstałoby wystarczająco duże zainteresowanie wśród użytkowników linuxa, wxWidgets jest cross-platform i możnaby bardzo łatwo przenieść GUI na linuxa nie musząc robić żadnych zmian w jego funkcjonowaniu.
Potoki w WinApi	Nasz program musi się komunikować z procesami silników szachowych. Wykorzystane do tego zostaną potoki WinApi, ponieważ są one w miarę proste w obsłudze. Ogranicza to system do działania tylko na systemach operacyjnych Windows, jednakże to właśnie na tym systemie jest zdecydowana większość zapotrzebowania na programy szachowe.

9.3 Diagramy UML

9.3.1 Diagram(-y) klas



9.3.2 Diagram(-y) czynności

Diagram czynności gry dwóch silników

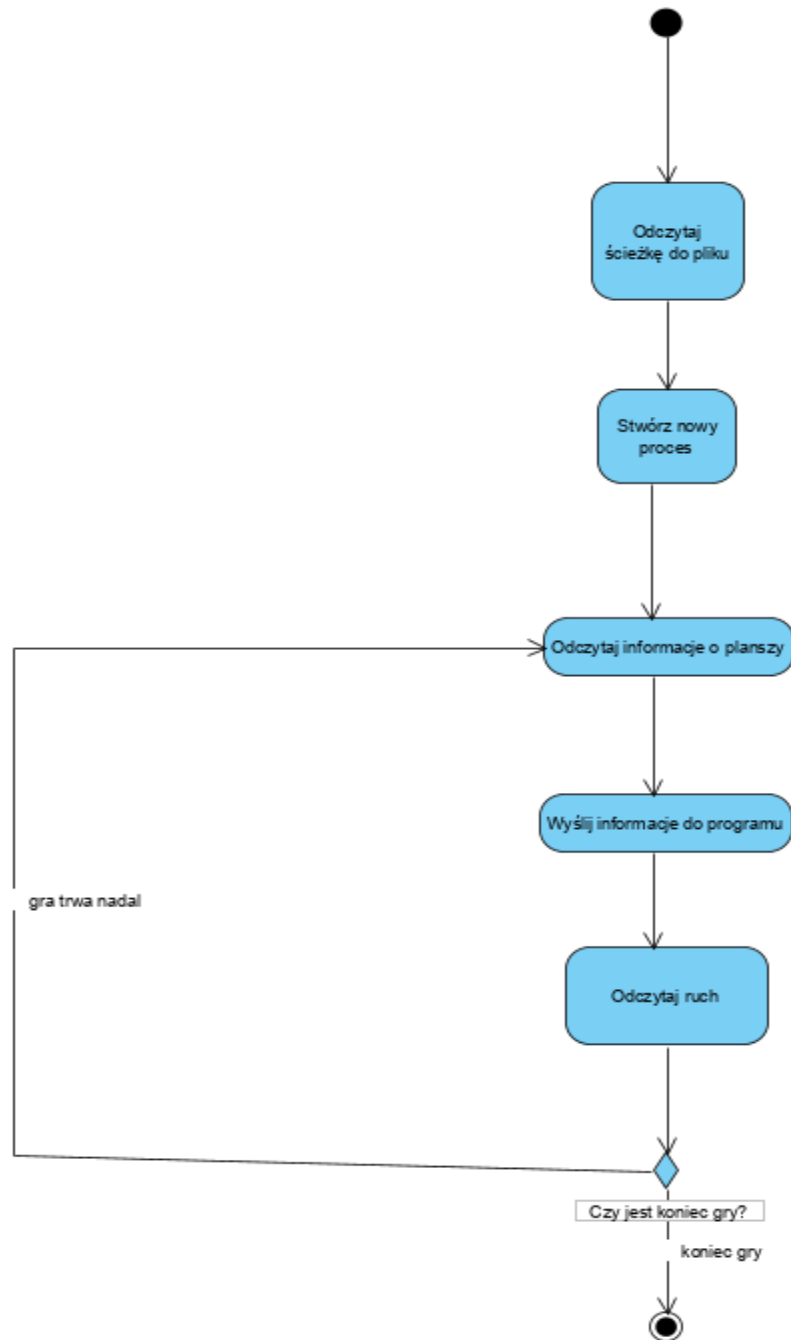


Diagram czynności rozpoczęcia nowego turnieju

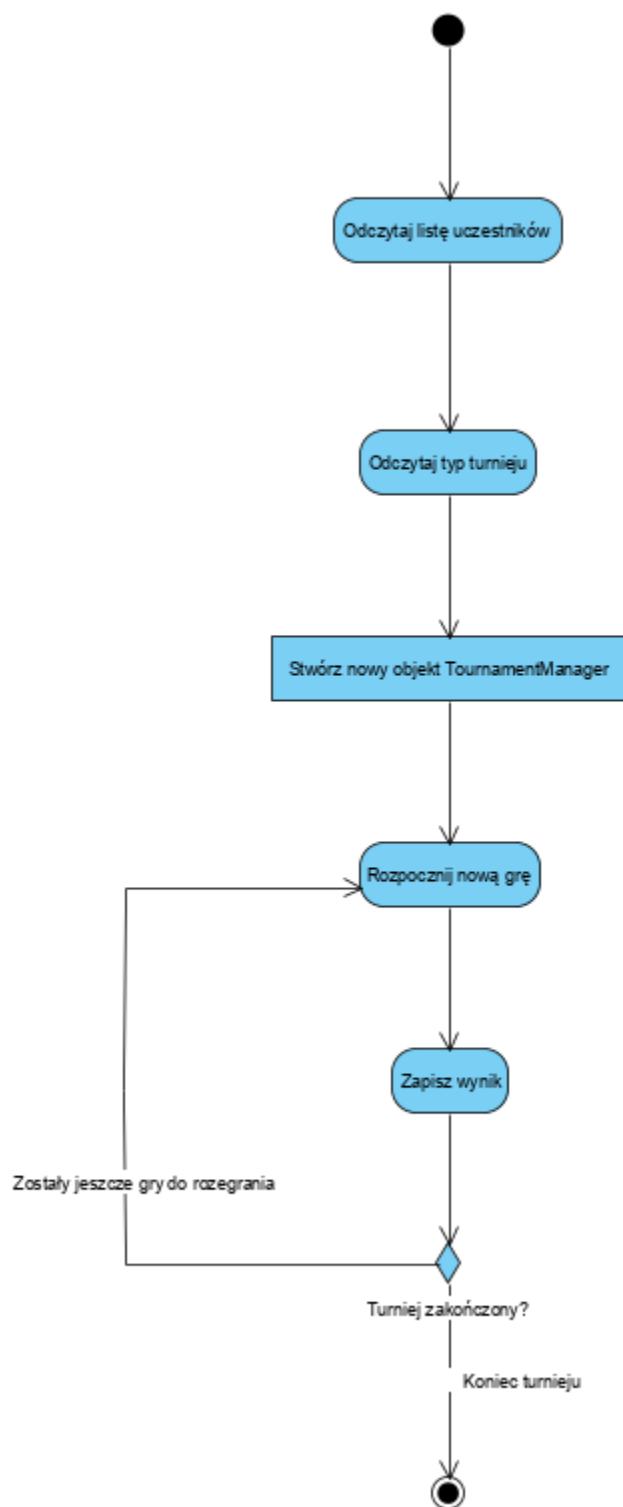
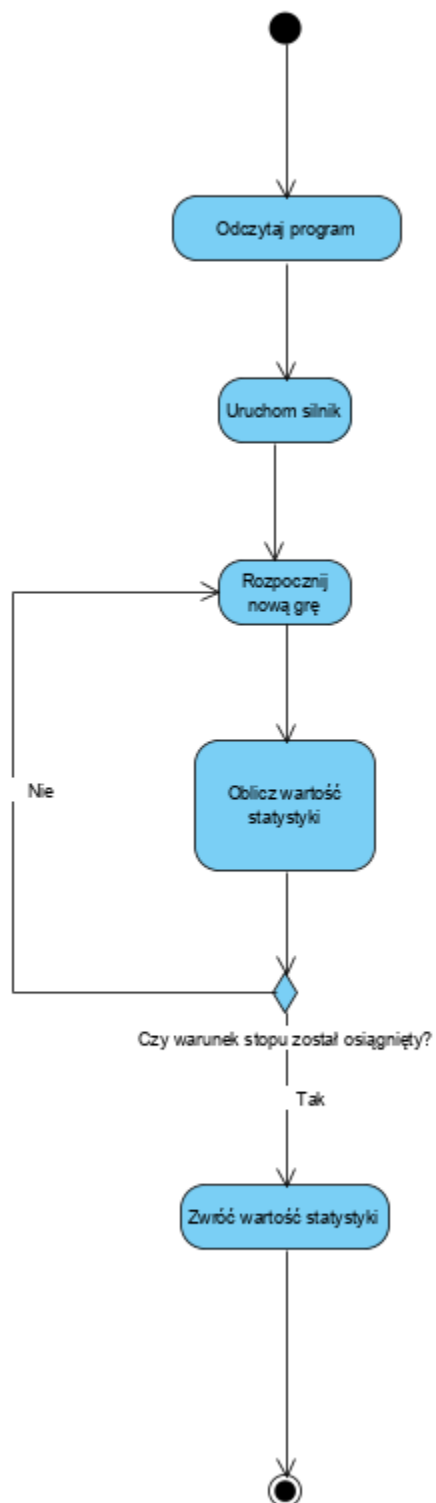


Diagram czynności testów statystycznych



9.3.3 Diagramy sekwencji

Diagram sekwencji analizy partii szachowych

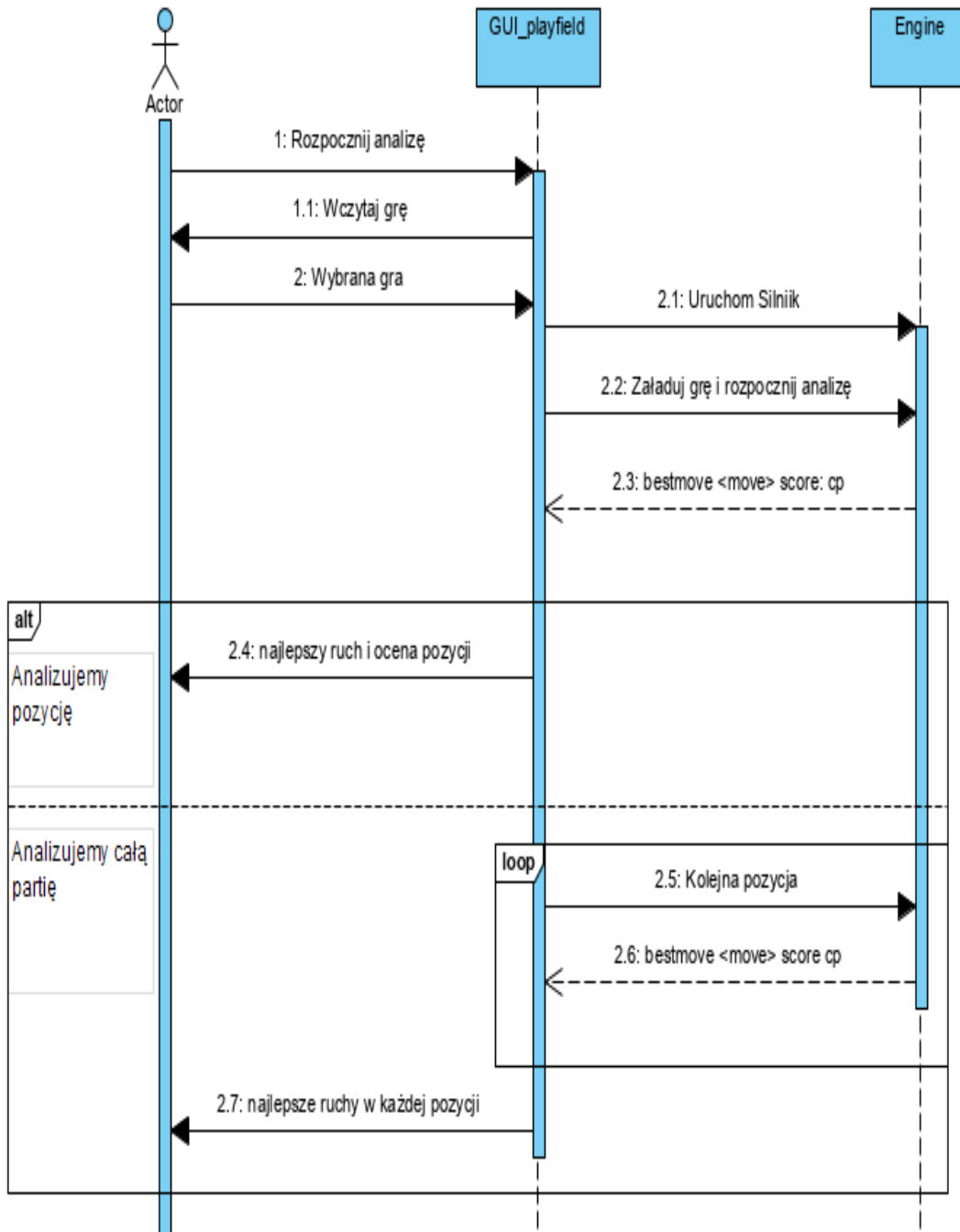


Diagram sekwencji rozgrywania gier

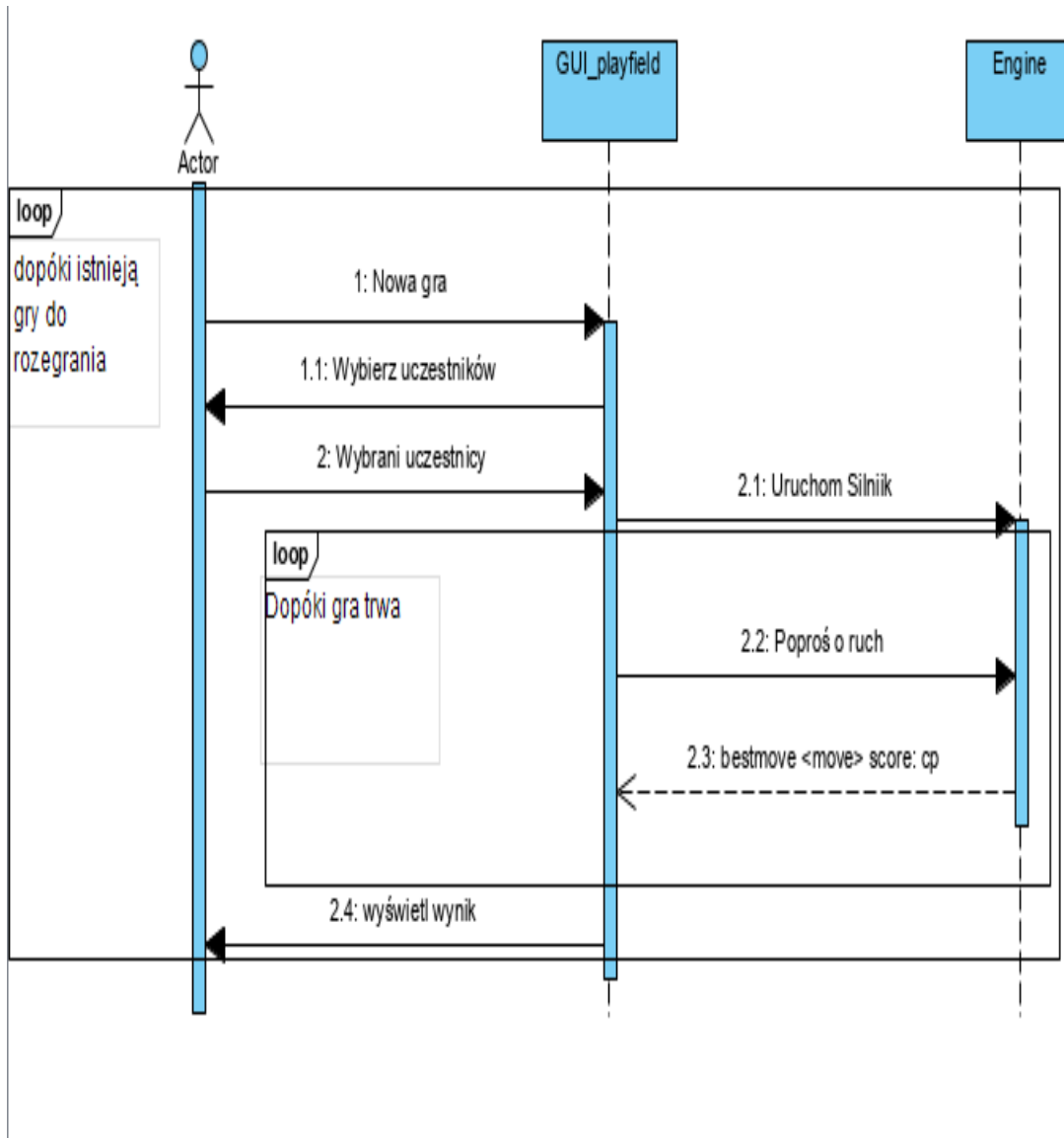


Diagram sekwencji szukania różnic pomiędzy silnikami

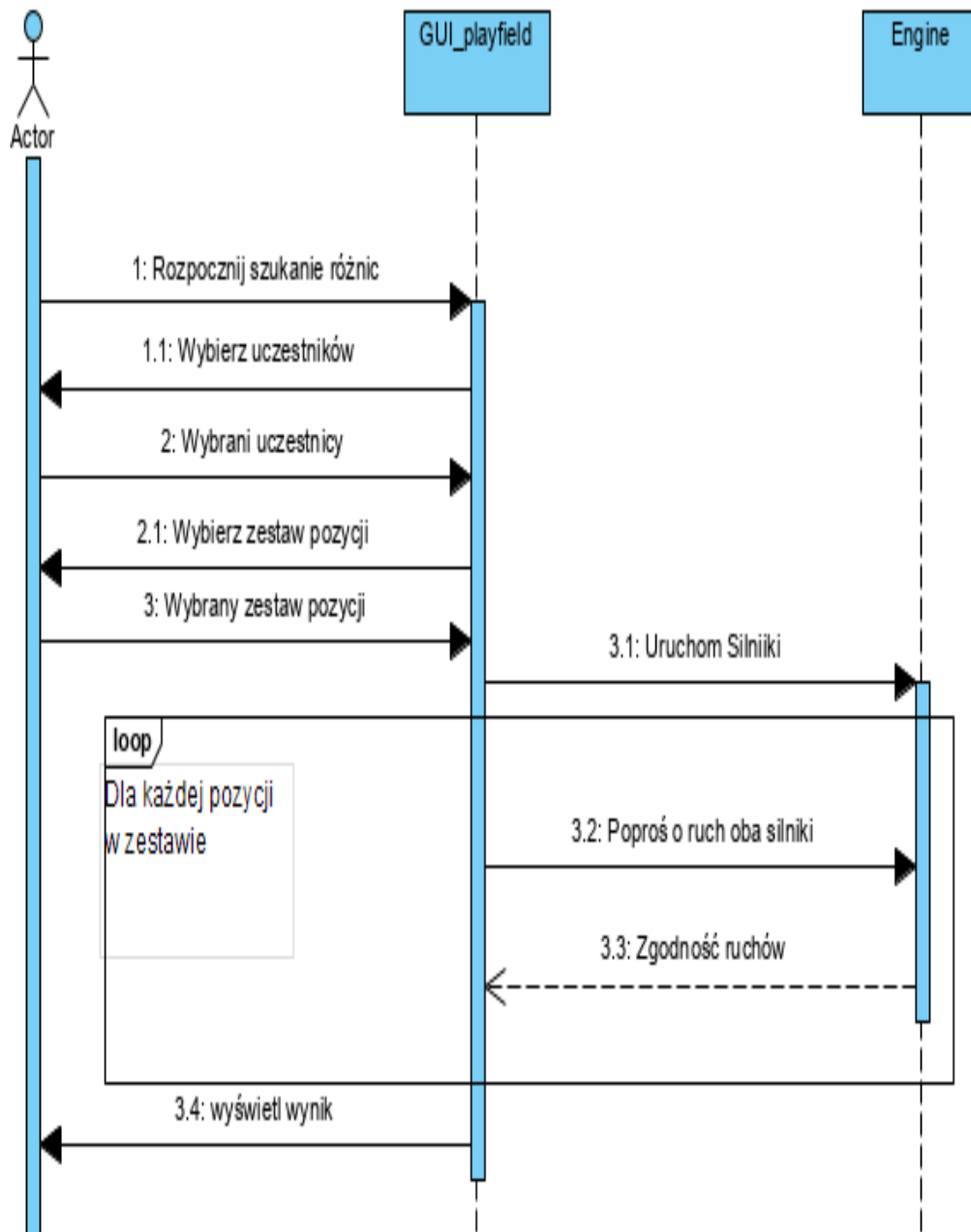


Diagram sekwencji dodawania nowego silnika

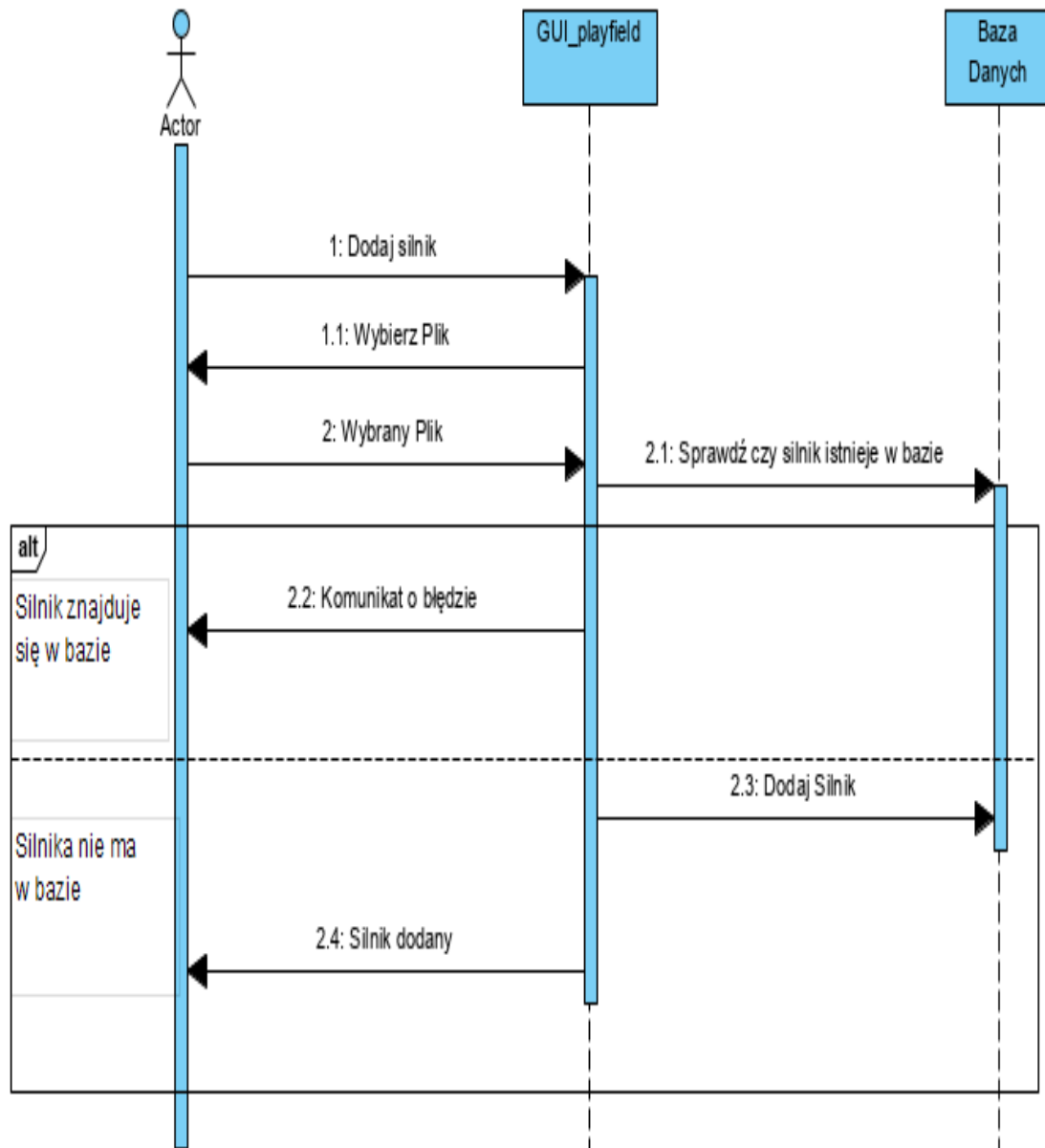
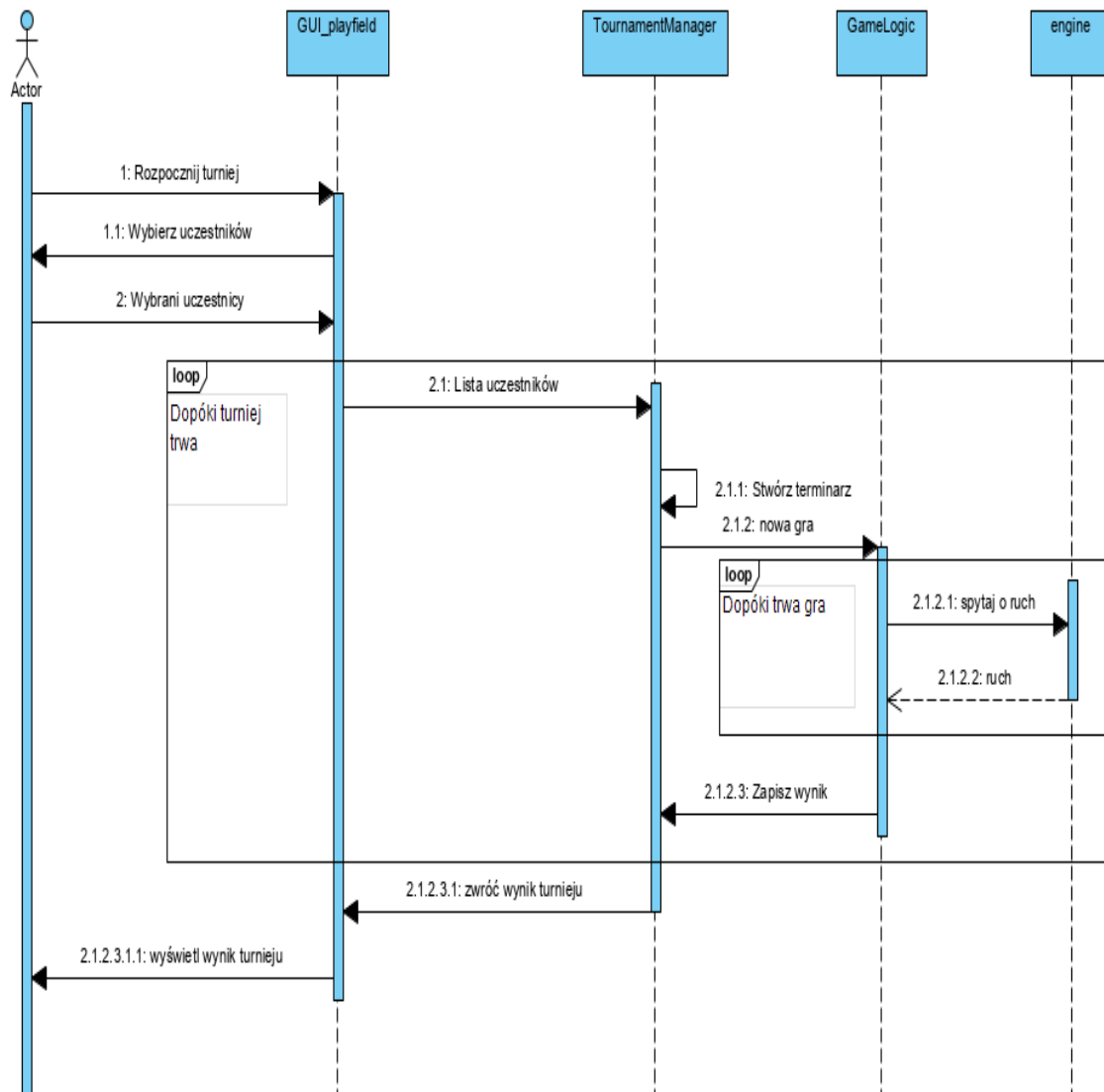


Diagram sekwencji rozpoczęcia nowego turnieju



9.3.4 Inne diagramy

co najmniej trzy – komponentów, rozmieszczenia, maszyny stanowej itp.
Diagram komponentów:

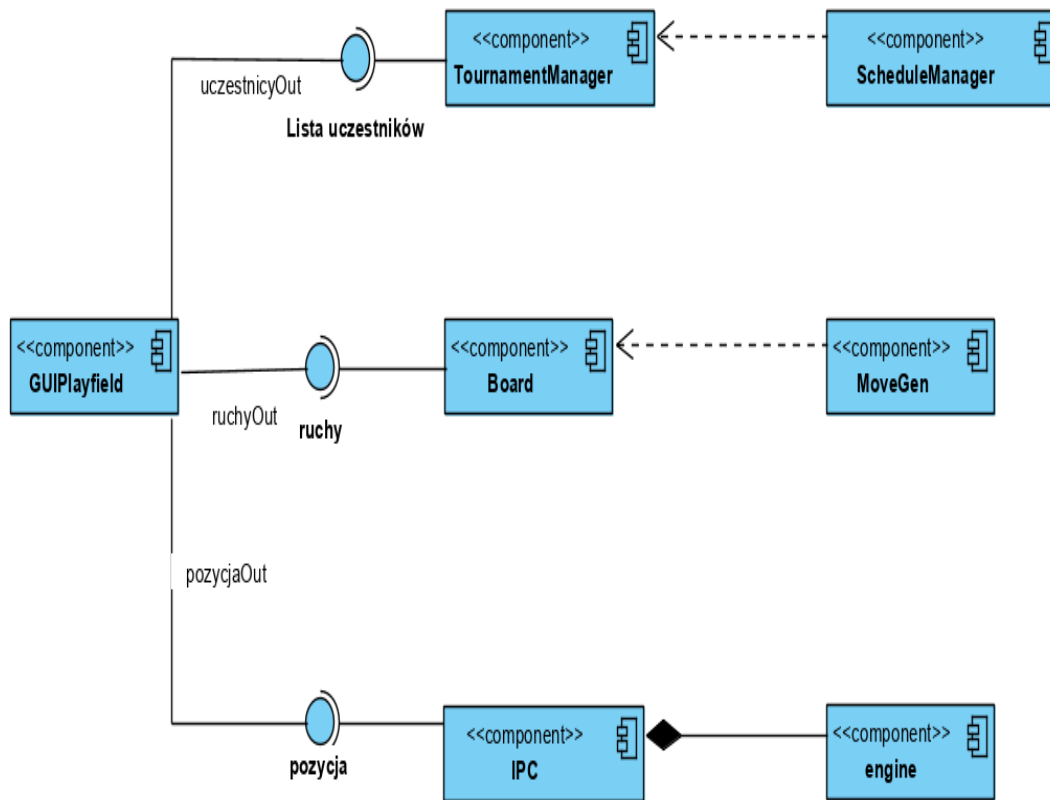


Diagram rozmieszczenia:

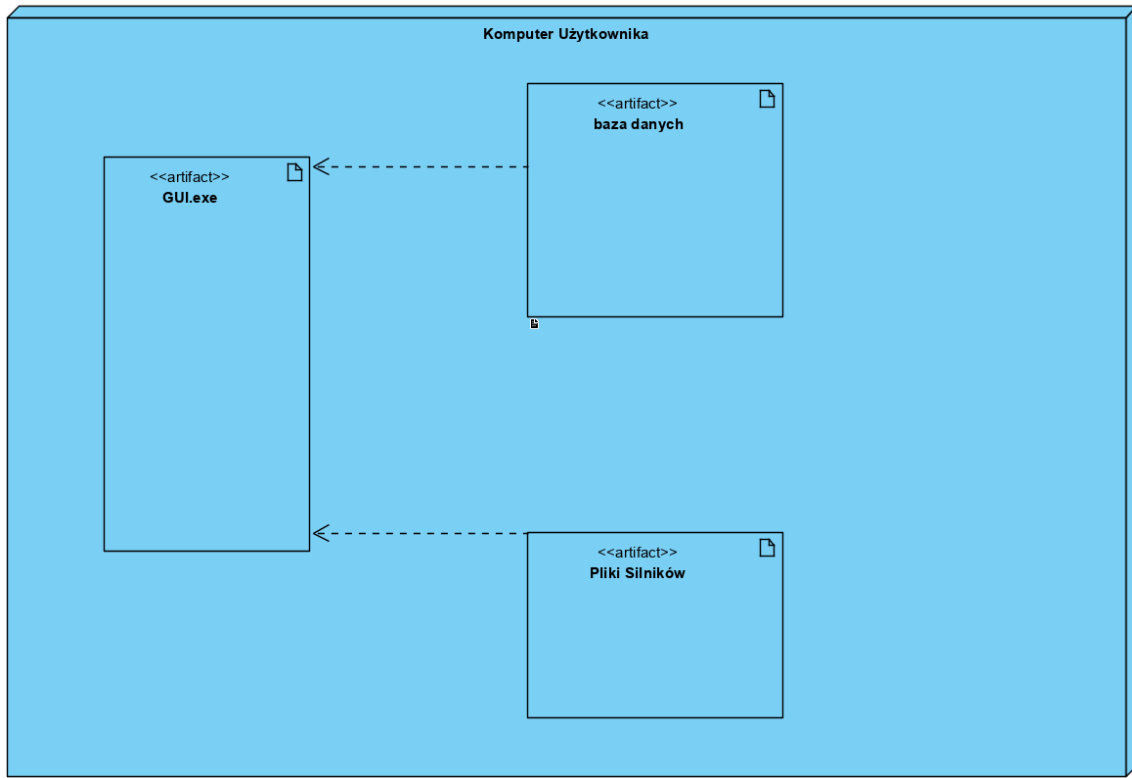
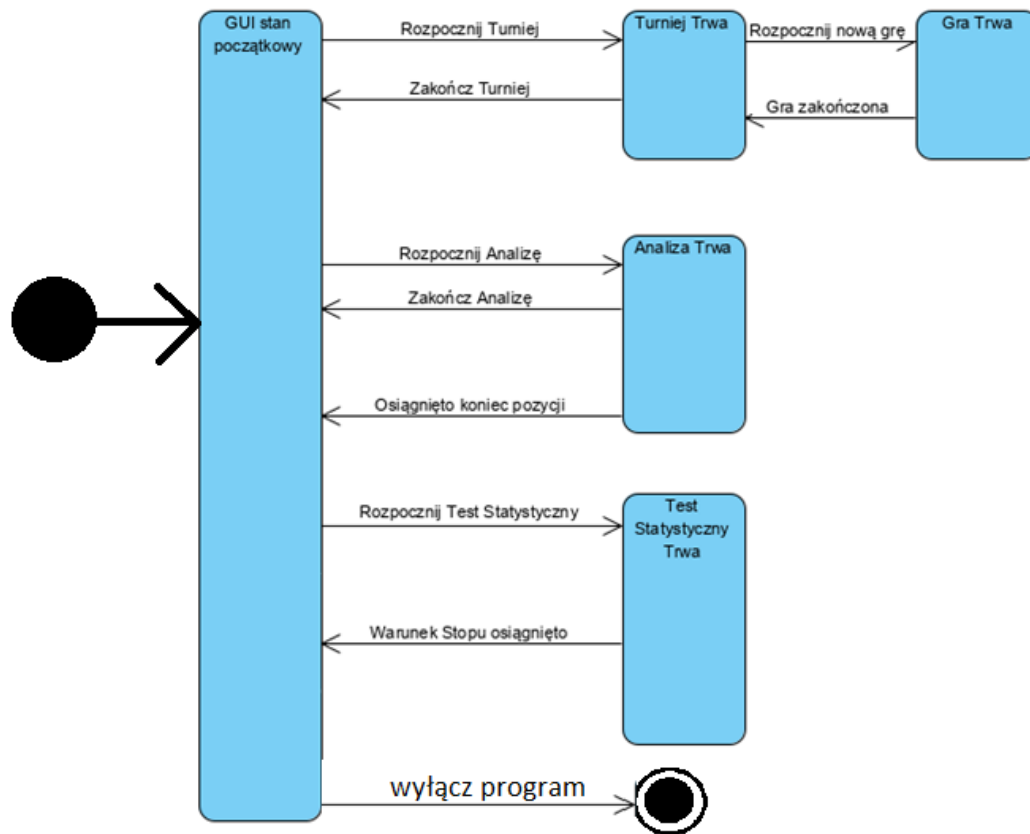


Diagram maszyny stanów:



9.4 Charakterystyka zastosowanych wzorców projektowych

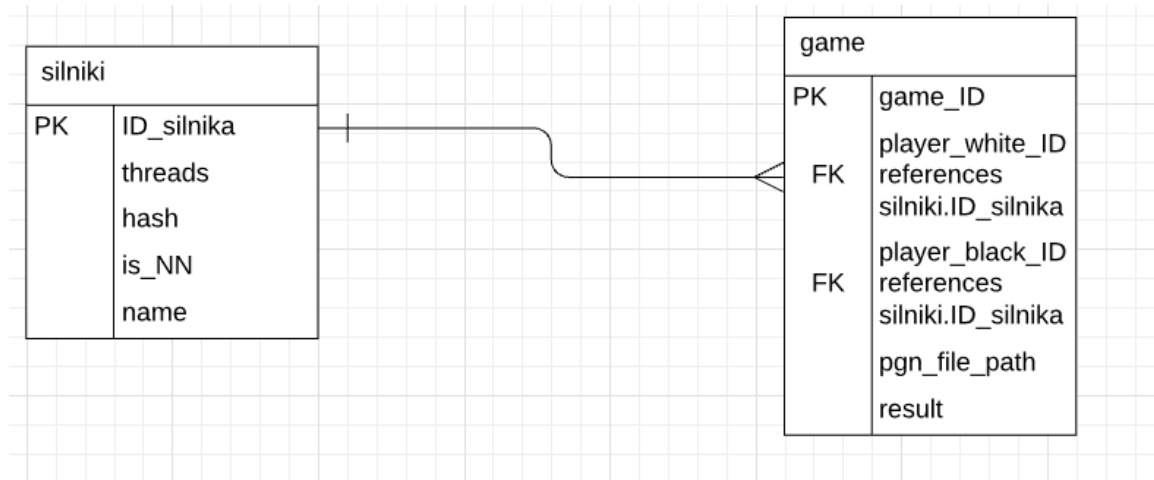
Mam zastosowaną architekturę 3 warstwową (widać to na diagramie klas)

GUI -> Silniki (engine) -> silniki uzyskują ustawienia z bazy danych

Mam również zastosowane potoki, na diagramie rozmieszczenia widać komunikację pomiędzy GUI.exe a plikami silników.

9.5 Projekt bazy danych

9.5.1 Schemat



9.5.2 Projekty szczegółowe tabel

W tabeli **silniki** znajdują się wszystkie informacje dotyczące pojedynczego gracza.

W tabeli **games** znajdują się szczegóły wszystkich rozegranych gier.

Więcej danych nie jest wymagane, większość rzeczy jest obliczana bądź generowana w trakcie run-time.

9.6 Projekt interfejsu użytkownika

9.6.1 Lista głównych elementów interfejsu

Główne okno wyświetlające statystyki aktualnego testu/turnieju oraz przebieg aktualnej gry

Okno ustawień w którym można dodać silniki.

Okno nowej gry.

Okno nowego turnieju.

9.6.2 Przejścia między głównymi elementami

Wszystkie okna są dostępne z głównego okna jako guziki. Po wciśnięciu danego guziki następuje automatyczne przejście (do nowego okna, bądź powrót do okna głównego).

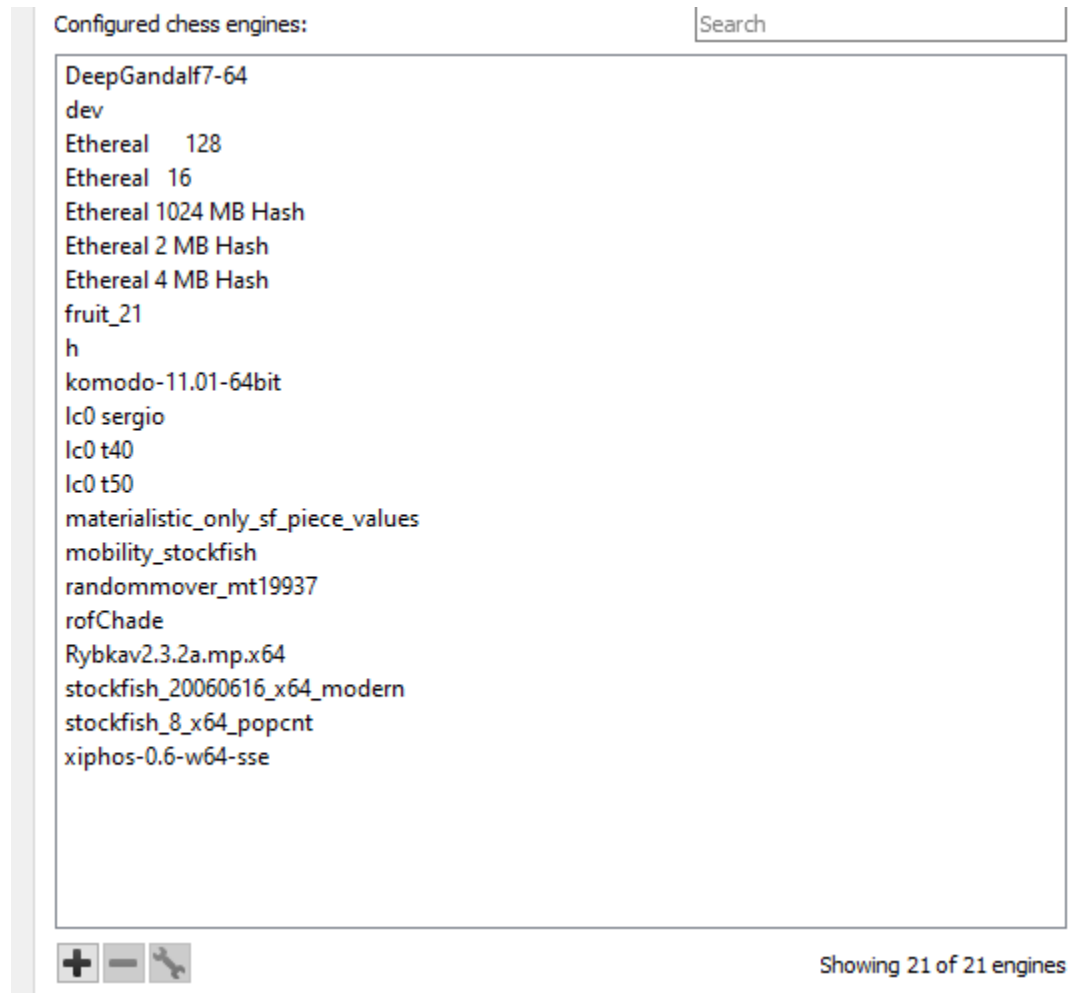
9.6.3 Projekty szczegółowe poszczególnych elementów

1. Główne okno



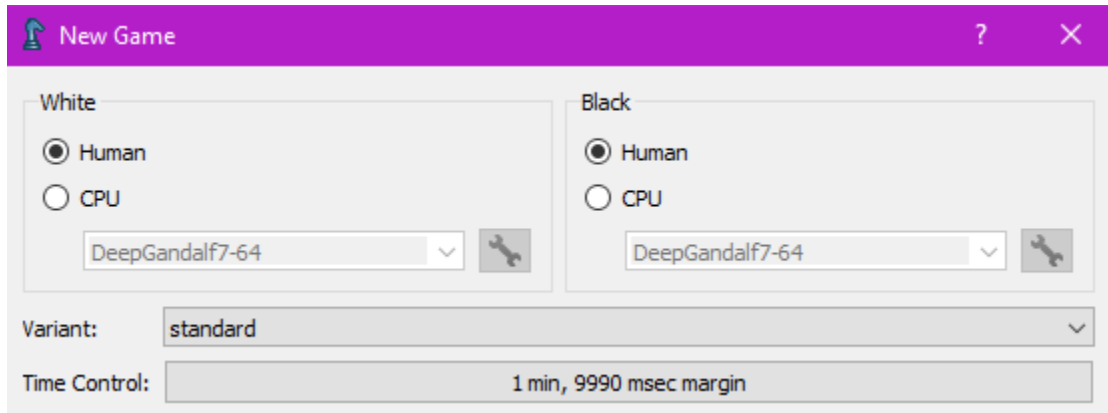
W tym oknie widzimy przebieg aktualnej czynności wykonywanej przez program, oraz możemy rozpocząć nową czynność.

2. Okno ustawień silników



Program odczytuje skonfigurowane programy z bazy danych i wyświetla je tutaj. Istnieje opcja dodania nowego programu.

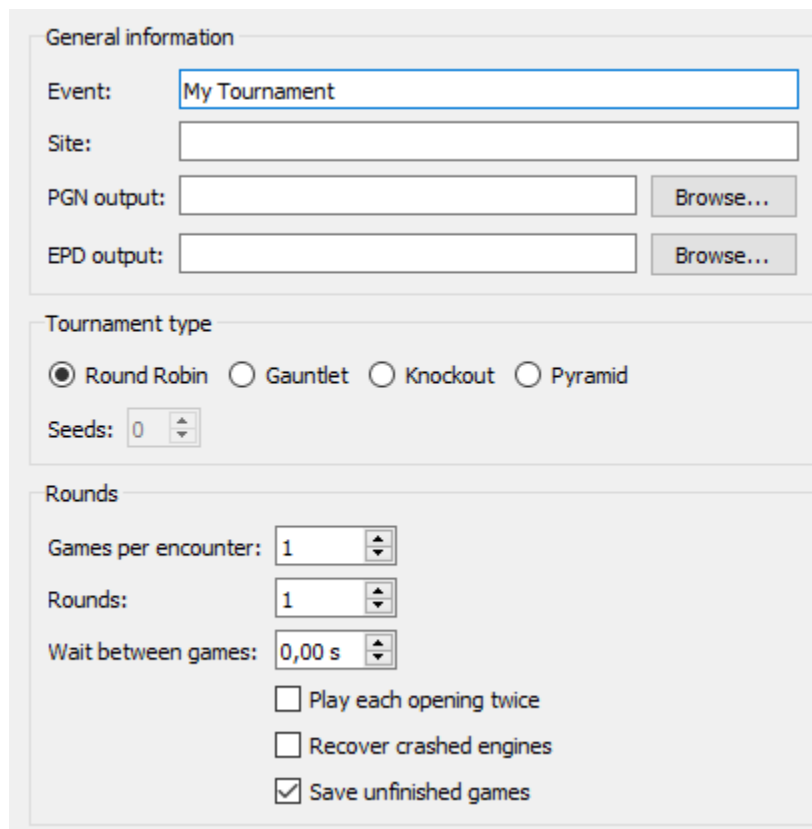
3. Okno nowej gry



Można wybrać rodzaj rozgrywki (np. zwykłe szachy, szachy fischera) oraz dwóch uczestników. Można również wybrać TC.

Po rozpoczęciu nowej gry, okno się zamyka i gra zaczyna się wyświetlać w głównym oknie.

4. Okno nowego turnieju



General information

Event:

Site:

PGN output:

EPD output:

Tournament type

☒ Round Robin ☐ Gauntlet ☐ Knockout ☐ Pyramid

Seeds:

Rounds

Games per encounter:

Rounds:

Wait between games:

☐ Play each opening twice

☐ Recover crashed engines

☒ Save unfinished games

W oknie tym można wybrać ustawienia nowego turnieju. Następnie wybiera się uczestników, a po tym zaczyna się gra.

9.7 Procedura wdrożenia

Terminy są bardzo elastyczne, przewidziane jest 25 dni roboczych na stworzenie całego programu.

Każdy zespół przejdzie 5 dni szkolenia na temat podstaw technologii zastosowanych oraz zasad działania wszystkich protokołów i zasad.

GUI może odbiegać wyglądem od założeń, jednakże funkcjonalność musi być ta sama.

Dodatkowo harmonogram może zostać przedłużony o 5 dni, dni te są poświęcone na wykonywanie dodatkowych testów.

Bardzo ważne jest jednak trzymanie się terminów. Zostały one tak ułożone, żeby bez żadnych problemów udało się wszystko zdążyć zrobić.

10 Dokumentacja dla użytkownika

- Nie jestem chętny

11 Podsumowanie

11.1 Szczegółowe nakłady projektowe członków zespołu

Autorzy : Artur Tamm 100%, około 50 godzin pracy (wliczając pracę na laboratoriach)

12 Inne informacje

przydatne informacje, które nie zostały ujęte we wcześniejszych punktach