Wydział Matematyki i Nauk Informacyjnych Politechniki
Warszawskiej

Project Game

Sebastian Jakubiak, Mikołaj Karaś, Tomasz Koter

v2.2

22 stycznia 2017r.

Spis treści

1 Specyfikacja **2**
1.1 Opis biznesowy 2
1.2 Wymagania funkcjonalne 3
1.3 Wymagania niefunkcjonalne 8
1.4 Architektura rozwiązania 8
1.5 Opis obiektów 12
1.6 System komunikacji 13
1.6.1 Mechanizm 13
1.6.2 Protokół 14
1.6.3 Scenariusze wymiany komunikatów 16
1.7 Użytkowanie aplikacji 20
1.7.1 Uruchomienie i konfiguracja 20
1.7.2 Przebieg rozgrywki 21
1.7.3 Wyjątki 23
1.8 Strategie AI 23

Tablica 1: Lista zmian

Data	Autor	Opis zmiany	Wersja
Jan 22, 2017	Sebastian Jakubiak, Mikołaj Karaś, Tomasz Koter	Dodano sekcję Strategie AI. Dodano więcej przykładowych komunikatów. Uzupełniono wymagania niefunkcjonalne o wpis o testach.	2.2
Jan 21, 2017	Sebastian Jakubiak, Mikołaj Karaś, Tomasz Koter	Poprawiono diagramy, architekturę rozwiązania. Dodano opis prowadzenia wielu rozgrywek za pośrednictwem jednego serwera. Dodano opis podziału na aplikacje.	2.1
Jan 16, 2017	Sebastian Jakubiak, Mikołaj Karaś, Tomasz Koter	Korekcja tabeli przypadków użycia. Dodano wymagania niefunkcjonalne. Dodano sekcje: System komunikacji, Użytkowanie aplikacji.	2.0a
Nov 27, 2016	Sebastian Jakubiak, Mikołaj Karaś, Tomasz Koter	Rewizja i poprawa spójności dokumentu. Pomniejsza korekcja tekstu.	1.2
Nov 27, 2016	Sebastian Jakubiak	Dodano opis klas. Poprawiono formatowanie i wygląd tabel. Dodano diagramy w wyższej jakości.	1.1
Nov 26, 2016	Tomasz Koter	Pierwsza wersja dokumentu	1.0a

1. Specyfikacja

1.1. Opis biznesowy

Project Game jest grą mającą na celu symulację realizacji pewnego abstrakcyjnego projektu. Grać w nią może w ramach jednej sesji jednocześnie dowolna liczba ludzkich bądź komputerowych graczy w dwóch przeciwnych drużynach, zależna jedynie od upodobań założyciela gry oraz możliwości sprzętowych.

Jako dwie konkurujące drużyny gracze mają za zadanie zrealizować swój projekt jako pierwsi. Drużyna, która jako pierwsza skończy projekt, wygrywa, natomiast ta druga przegrywa.

W ramach jednego projektu wyznaczone są cele, które dana drużyna musi osiągnąć, by zrealizować projekt. Każdy cel osiąga się poprzez podjęcie się zadania i wykonanie go w ramach tego celu. Może się jednak okazać, że wykonane zadanie nijak nie było pomocne w osiągnięciu celu, w związku z

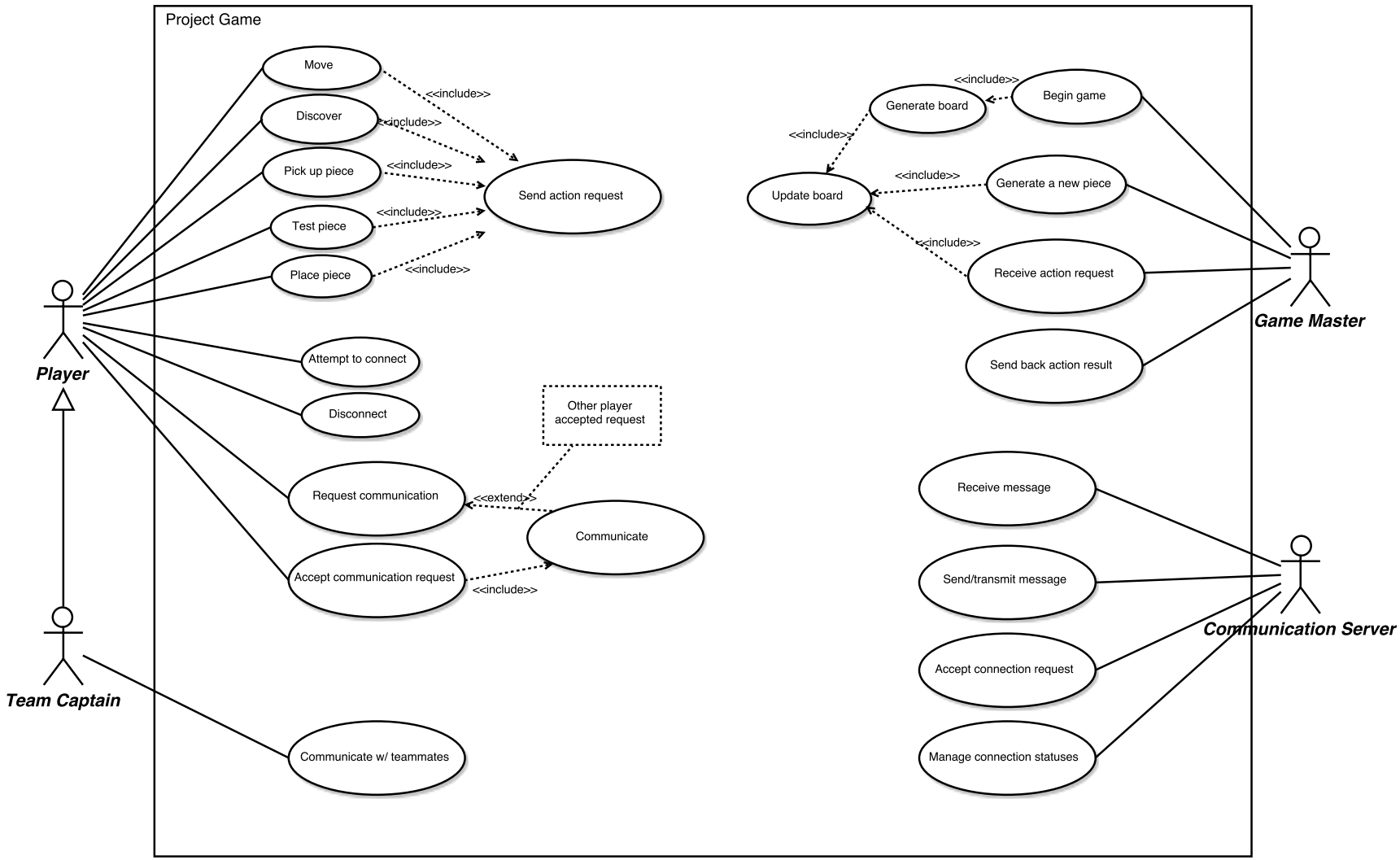
czym po wykonaniu go cel pozostaje nieosiągnięty.

Cele i zadania reprezentowane są za pomocą prostokątnej planszy składającej się z kwadratowych pól. Jedno pole może należeć do jednego z trzech obszarów: obszaru zadań, obszaru celów pierwszej drużyny lub obszaru celów drugiej drużyny. Gracze poruszają się po tej planszy w czterech kierunkach, z pola na pole. Zadania generowane są i umieszczane w polach obszaru zadań. Następnie gracze mogą podejmować się tych zadań i wypełniać je poprzez zaniesienie ich z obszaru zadań do obszaru celów swojej drużyny. Gracze nie wiedzą, które pola celów zawierają cel, który przysłuży się zrealizowaniu projektu, a które nie są celem projektu. Odkrywają je dopiero realizując pewne użyteczne zadanie w ramach tego pola. Gracz może upewnić się, czy zadanie jest użyteczne dowiadując się więcej na jego temat, lecz zajmuje to trochę czasu.

Gracze mogą komunikować się między sobą w ramach drużyny i w ten sposób współpracować.

1.2. Wymagania funkcjonalne

Poniższy diagram oraz tabela prezentują wszystkie możliwe przypadki użycia systemu gry, co jednocześnie pokrywa postawione w ramach tego projektu wymogi funkcjonalne.


Project Game
Move
Discover
Pick up piece
Test piece
Place piece
Send action request
<<include>>
Attempt to connect
Disconnect
Request communication
Other player accepted request
<<extend>>
Communicate
Accept communication request
Communicate w/ teammates
Player
Team Captain
Begin game
Generate board
Update board
Generate a new piece
Receive action request
Send back action result
Game Master
Receive message
Send/transmit message
Accept connection request
Manage connection statuses
Communication Server

Tablica 2: Przypadki użycia

Aktor	Lp	Nazwa	Opis
Gracz/Kapitan Drużyny	1	Move	Rusz graczem w kierunku góra/dół/lewo/prawo. Aby ruch został zrealizowany, Game Master musi zostać powiadomiony o chęci wykonania tego ruchu i go zatwierdzić (patrz *# 6 Send action request*).
	2	Discover	Odkryj zawartość otaczających pól oraz poznaj ich odległości od najbliższego zadania. Aby akcja została zrealizowana, musi zostać wykonany *# 6 Send action request*.
	3	Pick up piece	Podnieś zadanie z pola, na którym stoisz. Jeśli gracz stoi na polu z zadaniem, zadanie znika z planszy; w przeciwnym przypadku nic się nie dzieje. Akcja musi zostać zatwierdzona przez Mistrza Gry (patrz *# 6 Send action request*).
	4	Test piece	Sprawdź, czy zadanie jest użyteczne. Jeśli jest bezużyteczne, zostaje wyrzucone. Akcja musi zostać zatwierdzona przez Mistrza Gry (patrz *# 6 Send action request*).
	5	Place piece	Użyj zadania do osiągnięcia celu, na którym stoisz. Jeśli zadanie nie było użyteczne, nie dzieje się nic. Jeśli zadanie było użyteczne, odkrywa cel jako osiągnięty i informuje, czy był to cel służący do zrealizowania projektu. Akcja musi zostać zatwierdzona przez Mistrza Gry (patrz *# 6 Send action request*).
	6	Send action request	Wyślij zlecenie wykonania akcji (dowolnego przypadku użycia z przedziału 1-5) przez gracza do Mistrza Gry, by mógł potwierdzić poprawność operacji oraz zaktualizować planszę. Wykonywane bez wywołania przez gracza w momencie zlecenia dowolnej z akcji 1-5. Gracz nie posiada żadnych danych poza tymi, które są mu znane. Stąd każda akcja musi być zatwierdzona przez GM, który odsyła zaktualizowaną planszę po każdej zmianie. To zapobiega kolizjom, desynchronizacji i próbom hackowania gry.

Tablica 2: Przypadki użycia

Aktor	Lp	Nazwa	Opis
	7	Attempt to connect	Spróbuj połączyć się z serwerem gry. W wypadku błędu gracz będzie próbował aż do skutku bądź osiągnięcia limitu prób. Potem dołączy do aktualnego stanu gry (powodzenie) lub się wyłączy (niepowodzenie).
	8	Disconnect	Rozłącz się z serwerem gry (opuść grę). Natychmiast po tym wyłącza aplikację gracza. Gracz rozłącza się z własnej woli jedynie w przypadku zakończenia gry, chyba że pewna implementacja sztucznej inteligencji w przyszłości będzie zamierzenie rozłączać gracza.
	9	Request communication	Wyślij do innego gracza prośbę o rozpoczęcie korespondencji. Skutkuje otwarciem kanału komunikacji między graczami w wypadku zaakceptowania prośby; w przypadku odrzucenia - informacja o tym; w przypadku braku odpowiedzi, po upływie określonego czasu - równoznaczne z odrzuceniem. Od tego momentu gracze mogą fizycznie komunikować się ze sobą.
	10	Accept communication request	Zaakceptuj prośbę komunikacji od innego gracza. Otwiera kanał komunikacji między graczami.
	11	Communicate	Komunikuj się z innym graczem z drużyny. Wyślij/przeczytaj wiadomość od innego gracza. Wymaga zaakceptowania prośby o komunikację przez docelowego członka drużyny (# 10).
Kapitan Drużyny	12	Communicate with teammates	Komunikuj się z innym członkiem drużyny. Kapitan drużyny ma większe kompetencje od zwykłego gracza i nie musi prosić o komunikację. Automatycznie otwiera dwustronne połączenie do komunikacji między kapitanem a docelowym graczem.
Game Master	13	Begin game	Wyzwala akcję generowania planszy (# 14). Ustawia stan gry na rozpoczęty. Informuje o tym graczy za pośrednictwem serwera komunikacji.
	14	Generate board	Generuje planszę po raz pierwszy według podanych w pliku konfiguracyjnym lub w programie danych wejściowych. Wymusza aktualizację planszy i powiadomienie o tym graczy (akcja # 17).

Tablica 2: Przypadki użycia

Aktor	Lp	Nazwa	Opis
	15	Generate a new piece	Generuje nowy element *piece* na planszy w obszarze *task area*. Wymusza aktualizację planszy (# 17).
	16	Receive action request	Odbiera zlecenie wykonania akcji przez gracza (wysłane przez *# 6 Send action request*). Wykonanie zleconej akcji wymusza aktualizację planszy (# 17) oraz odesłanie informacji o powodzeniu akcji (# 18).
	17	Update board	Wprowadza zmiany w planszy i rozsyła odpowiednie informacje o stanie planszy przeznaczone dla graczy.
	18	Send back action result	Odsyła informację o powodzeniu zleconej akcji (odebranej w # 16).
Communication Server	19	Receive message	Odbiera wiadomość będącą na wejściu danego połączenia.
	20	Send/transmit message	Przekierowuje odebraną wcześniej wiadomość (# 19) do odpowiedniego odbiorcy lub wysyła własny komunikat.
	21	Accept connection request	Akceptuje próbę połączenia przez danego klienta.
	22	Manage connection statuses	Kontroluj stany połączeń. Tworzy nowe tunele dla konwersacji między graczami. Zamyka tunele zakończonych konwersacji. Sprawdza i informuje o nagłych rozłączeniach.

Tablica 3: Aktorzy korzystający z systemu

Nazwa	Opis
Gracz	Gracz uczestniczący w rozgrywce. Jest programem uruchamianym automatycznie na potrzeby gry.
Kapitan Drużyny	Wyróżniony gracz, który może bez wysyłania prośby komunikować się z innymi graczami z drużyny. Każda drużyna ma dokładnie jednego kapitana.
Game Master	Aplikacja kontrolująca przebieg gry. Generuje planszę, przekazuje informacje o niej graczom, aktualizuje ją i decyduje o rozpoczęciu i zakończeniu rozgrywki.
Communication Server	Aplikacja służąca do ustanowienia komunikacji między graczami i Game Masterem. Jej jedynym celem jest przesyłanie danych i zarządzanie połączeniami.

1.3. Wymagania niefunkcjonalne

Tablica 4: Lista wymagań niefunkcjonalnych

Obszar wymagań	Lp	Opis
Użyteczność	1	Uruchomienie serwera gry wymaga jedynie posługiwania się jedną aplikacją uruchomieniową. Uruchomienie graczy następuje za pomocą aplikacji konsolowej. Uruchomienie lokalnych graczy może nastąpić za pomocą aplikacji uruchomieniowej.
	2	Aplikacja uruchomieniowa posiada prosty interfejs graficzny umożliwiający łatwe skonfigurowanie ustawień gry.
Niezawodność	3	System jest odporny na krótkie (kilkusekundowe) przerwy w łączności, o ile te przerwy nie występują zbyt często.
	4	Błędy połączeń występujące w czasie gry nie powodują wyłączenia aplikacji uruchomieniowej.
Wydajność	5	System będzie w stanie obsłużyć minimum 10 graczy w obu drużynach łącznie naraz.
Utrzymanie	6	Można skonfigurować następujące aspekty gry: liczbę graczy, strategie poszczególnych drużyn, planszę, algorytm generowania zadań (wybrać z dostępnych i/lub zmieniać parametry liczbowe).
	7	Aplikacja dostępna jest jedynie w języku angielskim.
	8	Przetestowane zostaną fragmenty kodu dotyczące nawiązywania połączeń oraz przesyłania komunikatów.

1.4. Architektura rozwiązania

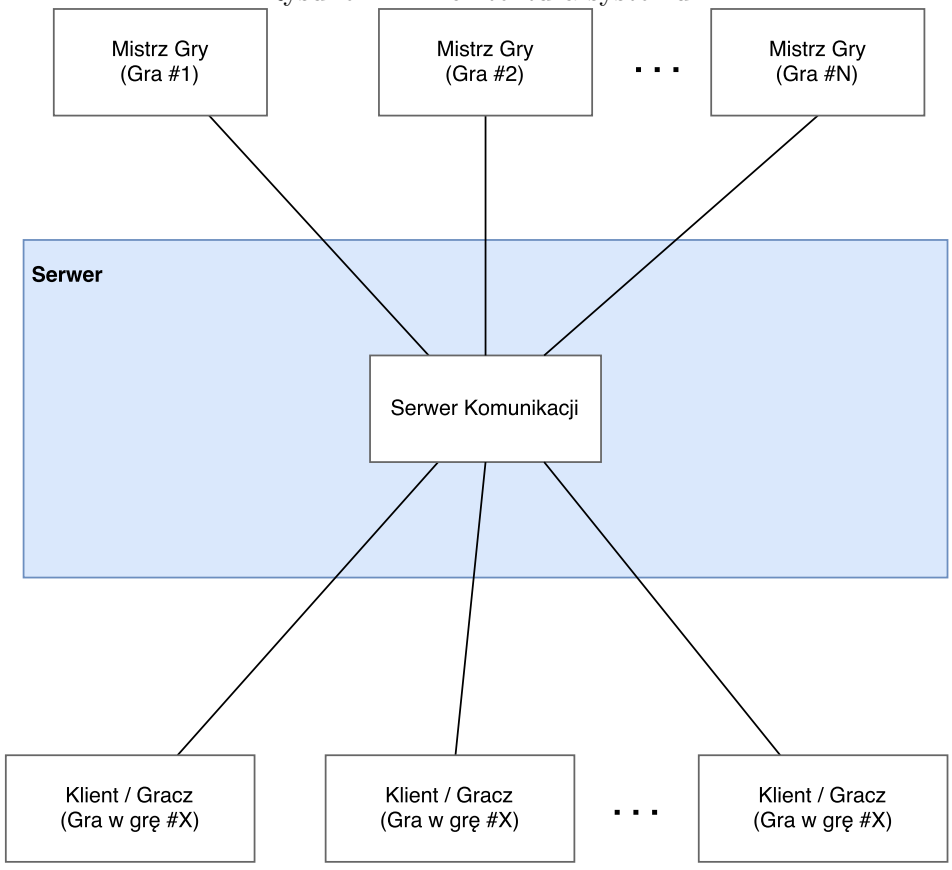
Program jest zaimplementowany w strukturze klient-serwer, gdzie klientami są poszczególni gracze korzystający z aplikacji klienckiej (możliwe będzie korzystanie z jednej maszyny przez kilku graczy jednocześnie) oraz mistrz gry. Po stronie serwera gry działa jedynie serwer komunikacji.

Jeden gracz może brać udział jednocześnie w maksymalnie jednej grze. Jeden serwer komunikacji może obsługiwać jednocześnie wiele gier. Jeden

mistrz gry może obsługiwać jednocześnie tylko jedną grę. Mistrz gry, zakładając swoją grę, wysyła prośbę do serwera komunikacji o utworzenie wpisu o jego grze. Serwer komunikacji, przyjmując zaakceptowanie prośby, odsyła unikalny identyfikator gry. Znając identyfikator danej gry oraz adres i port serwera, gracze mogą się z nią połączyć. W ten sposób realizowanych może być wiele gier równolegle za pomocą tego samego serwera, równocześnie pozostając odseparowanymi.

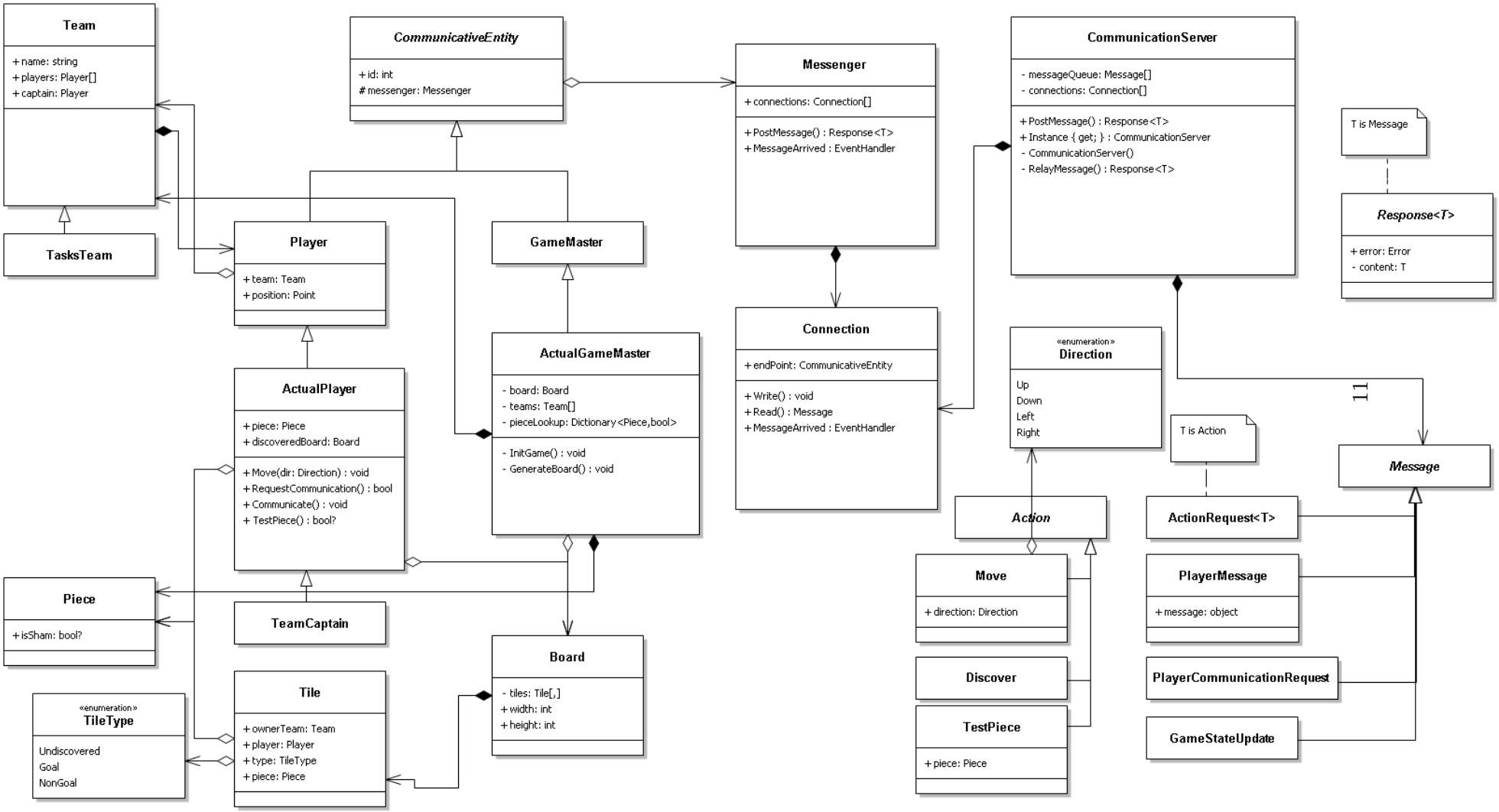
Wszelka komunikacja gracz-gracz oraz gracz-Mistrz Gry będzie odbywać się za pośrednictwem serwera komunikacji. Cała gra może również toczyć się lokalnie z lokalnymi połączeniami tak, że gracze, mistrz gry i serwer komunikacji znajdują się na jednej fizycznej maszynie. Nie ma to znaczenia, wtedy tak samo cała komunikacja odbywa się przez lokalny serwer komunikacji.

Rysunek 1: Architektura systemu



Zależności między obiektami powinny zachodzić jak na poniższym diagramie klas.

Rysunek 2: Diagram klas



1.5. Opis obiektów

Player Podstawowe informacje o graczu: drużyna, pozycja na planszy.

GameMaster Reprezentacja mistrza gry.

ActualPlayer Bieżący użytkownik systemu, jeśli jest on graczem. Posiada operacje wykonywania ruchów i akcji w grze oraz komunikacji z innymi graczami. Przechowuje zdobyte informacje o stanie gry (planszę) oraz o zadaniu, jeśli jakieś znalazł w trakcie gry.

ActualGameMaster Bieżący użytkownik systemu, jeśli jest on mistrzem gry. Posiada operacje inicjalizacji gry i generowania planszy. Przechowuje pełną informację o planszy i stanie gry oraz o tym, które zadania są użyteczne, a które nie.

Team Drużyna. Są dwie faktyczne drużyna oraz jedna sztuczna, która jest „właścicielem" pól w obszarze zadań.

Board Plansza, czyli stan gry — faktyczny (w przypadku mistrza gry) lub widziany przez gracza.

Field Pojedyncze pole na planszy. Posiada odwołanie do drużyny-właściciela; w przypadku pola w obszarze celów będzie to prawdziwa drużyna, a w przypadku pola w obszarze zadań — instancja TasksTeam. Pole posiada informację o graczu, który na nim stoi. Ponadto dla pól w obszarze zadań jest informacja o zadaniu, które może się znajdować na danym polu, a dla pól w obszarze celów informacja, czy na polu znajduje się cel (o ile użytkownik to wie).

Piece Zadanie. Może być użyteczne lub nie (użytkownik może nie wiedzieć).

Action Akcja wykonywana przez gracza, taka jak ruch, test zadania, rozpoczęcie komunikacji z innym graczem.

CommunicativeEntity Użytkownik systemu mogący komunikować się z innymi użytkownikami. Do komunikacji używa instancji klasy Messenger.

Messenger Obiekt wykorzystywany przez użytkownika do komunikacji z innymi. Umożliwia wysyłanie komunikatów i reagowanie na komunikaty przychodzące. Utrzymuje listę połączeń do innych użytkowników (tunelowanych przez serwer komunikacji).

Message Komunikat nadawany lub odbierany przez użytkownika. Wyróżnia się kilka rodzajów:

ActionRequest<T> Żądanie zatwierdzenia pewnej akcji gracza, wysyłane przez niego do mistrza gry.

PlayerMessage Wiadomość od jednego gracza do drugiego.

PlayerCommunicationRequest Prośba o rozpoczęcie wymiany informacji, kierowana przez jednego gracza do drugiego.

GameStateUpdate Informacja (być może częściowa) o zmianach stanu gry (czyli o zmianach na planszy), wysyłana przez mistrza gry do gracza (np. wskutek wykonania przez gracza poprawnej akcji).

Connection Połączenie z użytkownikiem. Obiekt odpowiadający za komunikację niskopoziomową.

ComunicationServer Serwer komunikacji przekazujący wiadomości między użytkownikami. Singleton. Posiada listę bezpośrednich połączeń do użytkowników. Nadchodzące wiadomości są gromadzone w kolejce, z której serwer je pobiera i przekazuje do adresatów.

1.6. System komunikacji

1.6.1. Mechanizm

Wszelka komunikacja może występować w następujących płaszczyznach:

Mistrz Gry - Serwer Komunikacji Służy ustanowieniu połączenia. Ponadto serwer komunikacji informuje mistrza gry o zaburzeniach toku gry w postaci rozłączeń.

Gracz - Serwer Komunikacji Służy ustanowieniu i utrzymaniu połączenia.

Gracz - Mistrz Gry Mistrz Gry informuje graczy o zmianach w układzie planszy, przekazuje im konsekwencje ich działań oraz o statusie gry. Gracze zlecają Mistrzowi Gry wykonanie operacji na rzecz gracza.

Gracz - Gracz Gracze mogą komunikować się między sobą, by przekazać informacje o planszy i stanie gry z ich perspektywy.

Niezależnie od płaszczyzny, **serwer komunikacji** zawsze jest ogniwem w drodze przekazywania komunikatów. Albo on sam komunikuje się (MG - Serwer lub Gracz - Serwer), albo przekazuje komunikaty między graczami a mistrzem gry lub graczami a graczami. Nie istnieją połączenia bezpośrednie, omijające **serwer komunikacji**.

Pomimo tego, że serwer komunikacji pośredniczy w każdej wymianie informacji, każda jednostka zdolna komunikować się (*CommunicativeEntity*) posiada moduł (*Messenger*), który wykonuje wszystkie działania potrzebne do komunikacji. Z punktu widzenia takiej jednostki komunikacja jest bezpośrednia, ponieważ dla każdej pary jednostek komunikujących się **serwer komunikacji** tworzy połączenie w postaci wirtualnego tunelu. Tunel posiada dwa punkty wejścia/wyjścia, to jest *Messengery* dwóch jednostek komunikujących się ze sobą. Z zewnątrz nie widać żadnych mechanizmów wewnętrznych, można jedynie pisać do tunelu i odczytywać wiadomości z niego. W rzeczywistości jednak tunel prowadzi przez **serwer komunikacji**, który przekazuje wiadomości na bieżąco do *Messengerów*.

1.6.2. Protokół

Wszystkie wiadomości zostają przekonwertowane na format JSON. Wybór ten jest motywowany niezależnością od języka programowania, powszechnością parserów oraz choćby tym, że format JSON jest bardziej zwięzły niż XML. Następnie łańcuch znaków jest pakowany w pakiety TCP/IP i transmitowany. TCP, mimo że powolniejszy niż UDP, bo wymaga potwierdzeń i uwzględnia retransmisję, jest wygodniejszy w użyciu. Ponadto prędkość UDP nie zostałaby w pełni wykorzystana, ponieważ gra nie jest na tyle złożona.

Komunikat JSON 1: Przykładowy komunikat - zlecenie ruchu o jedno pole w górę

```
{
  "type":"ActionRequest<Move>",
  "data":{
    "dir":"0"
  }
}
```

Komunikat JSON 2: Informacja od gracza o widzianej planszy

```
{
  "type":"PlayerMessage",
  "data":{
    "message":{
      "board":{
        "width":"12",
        "height":"36",
        "tiles":[
          {
            "ownerTeam":"2",
            "player":"",
            "type":"1",
            "piece":""
          },
          { ... /* another tile info */ },
          ... // more tiles
        ]
      }
    }
  }
}
```

Komunikat JSON 3: Prośba o rozpoczęcie komunikacji

```
{
  "type":"PlayerCommunicationRequest",
  "data":{ }
}
```

Komunikat JSON 4: Zlecenie odkrycia planszy wokół gracza

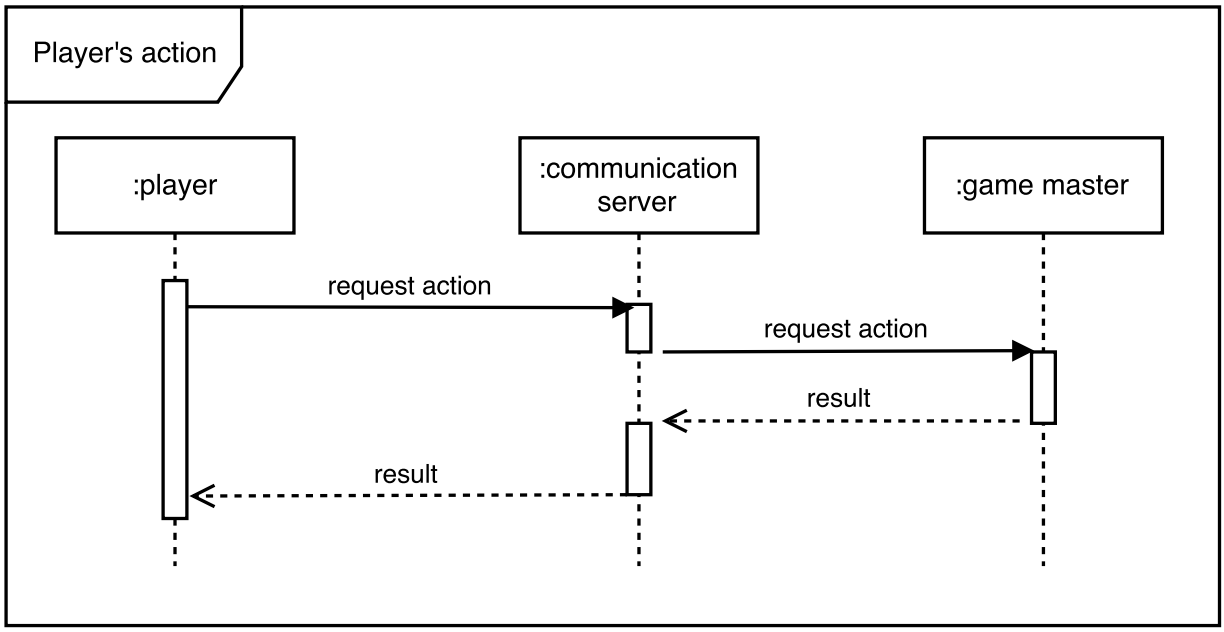
```
{
  "type":"ActionRequest<Discover>",
  "data":{ }
}
```

1.6.3. Scenariusze wymiany komunikatów

1.6.3.1 Akcja gracza

Sytuacja, gdy gracz zamierza wykonać jakąś akcję, taką jak: przemieszczenie się, sprawdzenie lub wykorzystanie posiadanego zadania, sprawdzenie sąsiadujących pól. Gracz wysyła żądanie do mistrza gry. Mistrz gry odsyła wynik akcji lub informację, że jest nieprawidłowa.

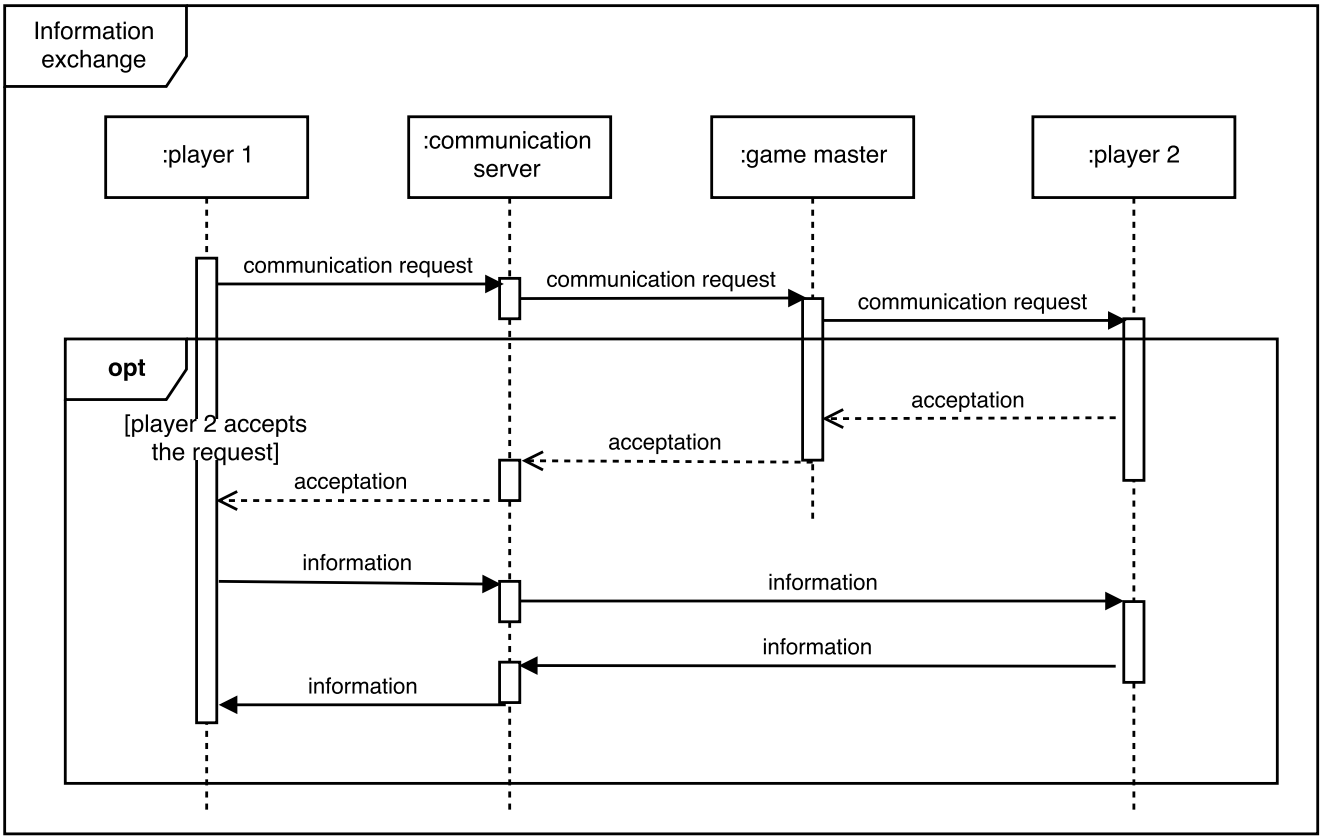
Rysunek 3: Diagram sekwencji



1.6.3.2 Komunikacja graczy

Sytuacja, gdy jeden gracz chce wymienić informacje z drugim. Pierwszy gracz musi zapytać drugiego o pozwolenie. Musi w tym pośredniczyć mistrz gry, ponieważ jest to traktowane jako rodzaj akcji. Drugi gracz może prośbę zaakceptować lub nie. W przypadku akceptacji obie strony przesyłają sobie nawzajem informacje.

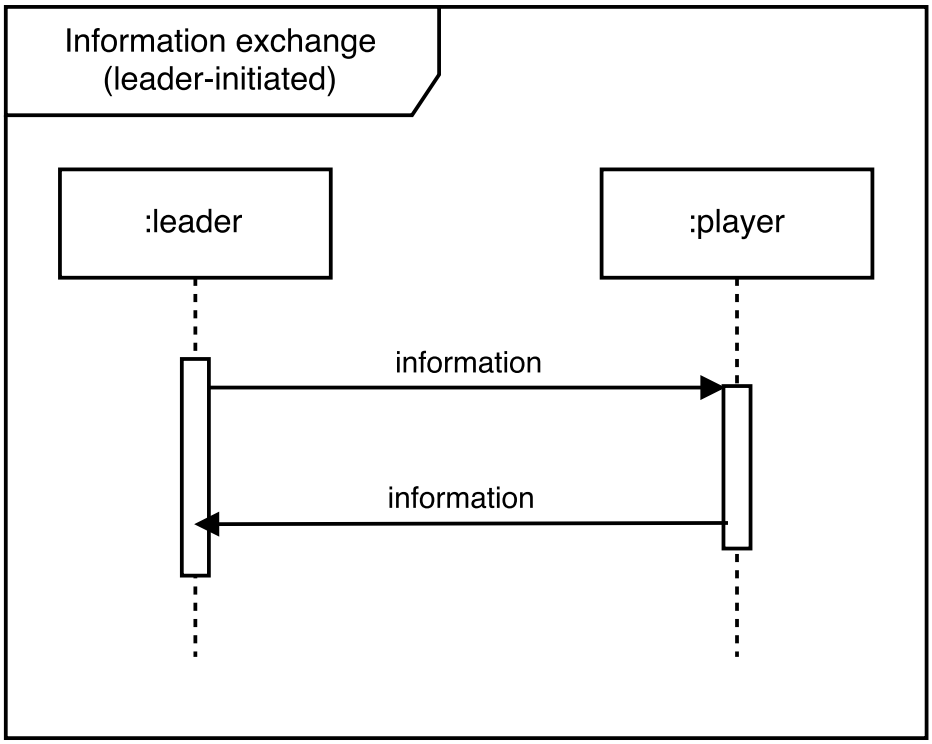
Rysunek 4: Diagram sekwencji



1.6.3.3 Komunikacja kierownika z zespołem

Sytuacja, gdy kierownik zespołu chce wymienić informacje innym członkiem drużyny. Kierownik nie musi pytać o pozwolenie.

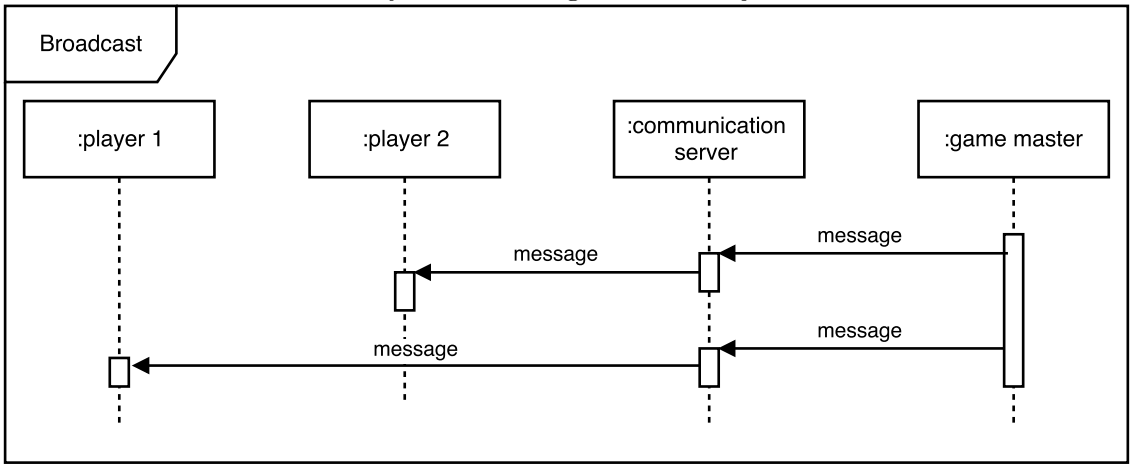
Rysunek 5: Diagram sekwencji



1.6.3.4 Rozgłaszanie komunikatów

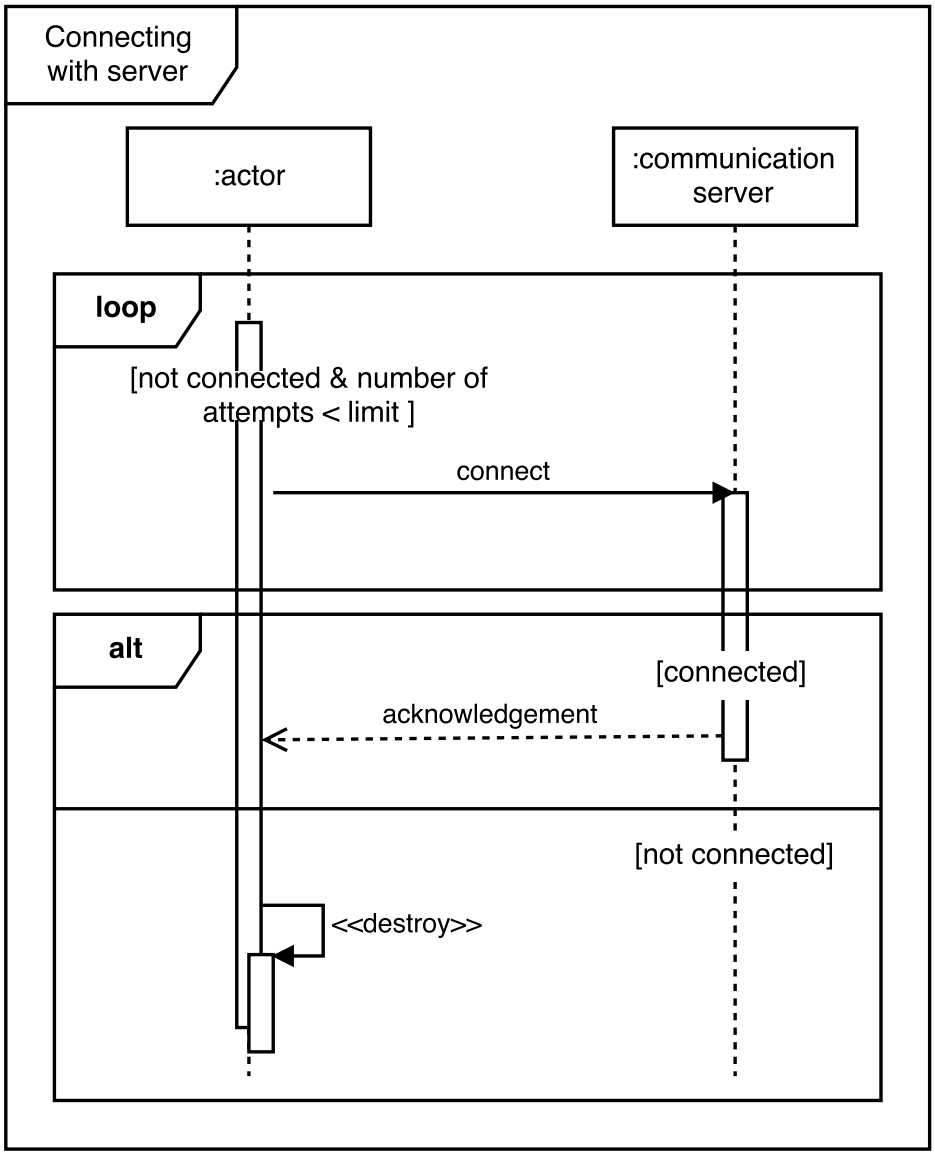
Scenariusz, w którm mistrz gry musi o czymś powiadomić wszystkich graczy, np. o zakończeniu gry. Do każdego gracza jest wysyłana taka sama wiadomość.

Rysunek 6: Diagram sekwencji



1.6.3.5 Łączenie z serwerem

Łączenie jest dokonywane przez graczy i mistrza gry tuż po wystartowaniu. W przypadku niepowodzenia podejmowane są kolejne próby aż do wyczerpania ich limitu. Udana próba połączenia kończy się odbiorem potwierdzenia od serwera.



Rysunek 7: Diagram sekwencji

1.7. Użytkowanie aplikacji

1.7.1. Uruchomienie i konfiguracja

Aby uruchomić grę, potrzebne jest uruchomienie i skonfigurowanie serwera komunikacji, mistrza gry oraz graczy. Serwer komunikacji jest aplikacją

konsolową pobierającą bieżącą konfigurację z edytowalnego pliku konfiguracyjnego. Mistrz gry jest aplikacją konsolową lub aplikacją z interfejsem graficznym umożliwiającą konfigurację, uruchomienie i śledzenie gry. Gracz jest również aplikacją konsolową przyjmującą adres i port serwera oraz identyfikator gry. Brakujący gracze mogą być również uruchomieni przez mistrza gry w wypadku potrzeby zapełnienia miejsc.

Zdalnych graczy i mistrzów gier należy uruchomić z poszczególnych maszyn. Serwer nie jest w stanie skontaktować się z własnej inicjatywy z komputerem w sieci niebędącym jakimś serwerem.

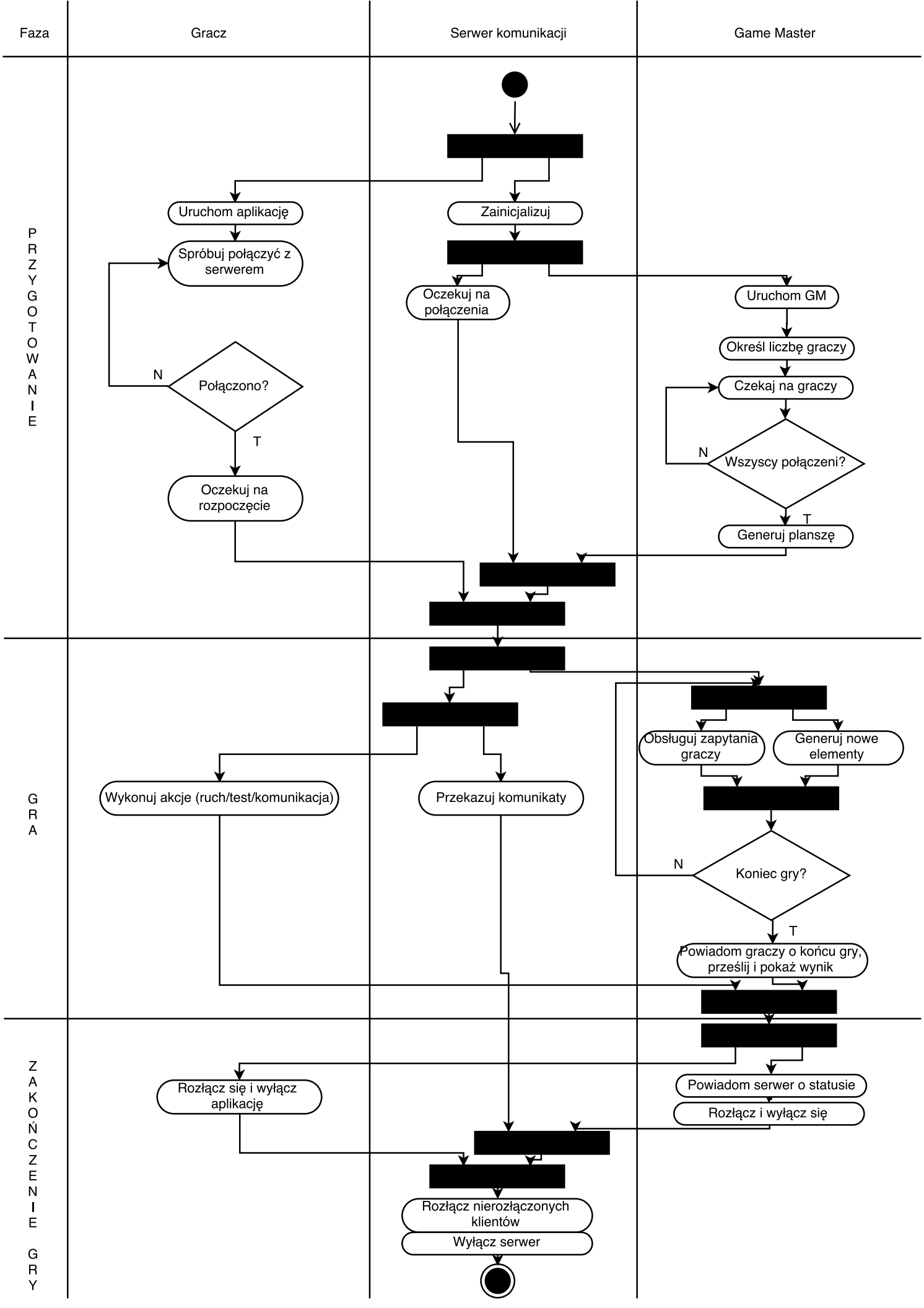
1.7.2. Przebieg rozgrywki

Gracze są w stanie połączyć się z serwerem gry dopiero w momencie, gdy serwer komunikacji posiada wpis w słowniku związany z danym identyfikatorem gry. Proces mistrza gry ustala liczbę graczy według konfiguracji, a następnie czeka aż wszyscy z nich się połączą z serwerem komunikacji. Gdy to nastąpi, generuje planszę i rozpoczyna grę.

Kolejną fazą działania programu jest sama gra. Aplikacja gracza wykonuje kolejne ruchy natychmiast po otrzymaniu odpowiedzi o powodzeniu poprzedniego ruchu i podjęciu decyzji o bieżącym. Decyzje gracz komputerowy podejmuje na podstawie przypisanej mu strategii. Komunikat o decyzji ruchu (lub innej akcji) jest przekazywany do mistrza gry poprzez serwer komunikacji. Mistrz gry zaś obsługuje i odpowiada na zlecenia graczy oraz generuje nowe elementy, jednocześnie sprawdzając, czy nie wystąpił warunek zakończenia gry. Dopóki gra może być kontynuowana, każdy z graczy kontynuuje swój cykl postępowania.

W przypadku zakończenia gry, mistrz gry wysyła graczom poprzez serwer komunikacji informację o zakończeniu gry i jej wynikach. Po ogłoszeniu wyników i decyzji o zakończeniu sesji następuje faza czyszczenia, klienci rozłączają się z serwerem komunikacji, a aplikacje graczy są zamykane. Mistrz gry pozostaje uruchomiony, jako że jest aplikacją posiadającą wszystkie dane na temat gry. Te dane użytkownik (człowiek) może przejrzeć i zapisać do pliku. Serwer komunikacji należy wyłączyć osobno.

Po ogłoszeniu wyników, zanim gracze się rozłączą, można podjąć decyzję o kontynuacji sesji i ponowieniu gry na tych samych bądź innych (wyspecyfikowanych przez użytkownika) ustawieniach.


Faza
Gracz
Serwer komunikacji
Game Master
PRZYGOTOWANIE
GRA
ZAKOŃCZENIE GRY
Uruchom aplikację
Spróbuj połączyć z serwerem
Połączono?
N
T
Oczekuj na rozpoczęcie
Zainicjalizuj
Oczekuj na połączenia
Uruchom GM
Określ liczbę graczy
Czekaj na graczy
Wszyscy połączeni?
N
T
Generuj planszę
Wykonuj akcje (ruch/test/komunikacja)
Przekazuj komunikaty
Obsługuj zapytania graczy
Generuj nowe elementy
Koniec gry?
N
T
Powiadom graczy o końcu gry, prześlij i pokaż wynik
Rozłącz się i wyłącz aplikację
Powiadom serwer o statusie
Rozłącz i wyłącz się
Rozłącz nierozłączonych klientów
Wyłącz serwer

1.7.3. Wyjątki

Podczas rozgrywki może zdarzyć się wiele nieprzewidzianych błędów. Wynikają one przede wszystkim z wadliwych połączeń. Poniżej opisane są akcje podejmowane w wypadku możliwych scenariuszy błędów.

1.7.3.1 Utrata łączności z graczem

W momencie, gdy serwer stwierdzi brak połączenia z graczem, podejmie zdefiniowaną w czasie konfiguracji akcję. Możliwe będą:

1. Zakończenie gry bez podania wyniku
2. Zakończenie gry, przegrywa drużyna, która straciła gracza
3. Kontynuacja gry bez rozłączonego gracza (jeśli zostanie 0 graczy w danej drużynie, koniec gry - drużyna przegrywa walkowerem lub gra kończy się bez podania wyniku)
4. Próba uruchomienia i dołączenia nowego gracza, by kontynuował w miejscu rozłączonego gracza (w przypadku kilku nieudanych prób dołączenia można zakończyć grę z błędem)

1.7.3.2 Utrata łączności z serwerem

Jeśli to mistrz gry traci łączność z serwerem, to serwer informuje o tym graczy, którzy sami powinni się rozłączyć. Po chwili ucina pozostałe połączenia i się wyłącza. W tym czasie mistrz gry wyłącza się. Aplikacja gry zwraca komunikat o błędzie i proponuje ponowne uruchomienie.

Jeśli gracz traci łączność z serwerem, próbuje połączyć się ponownie z maksymalną liczbą prób określoną podczas konfiguracji. Jeśli mu się to nie uda, wyłącza się.

1.7.3.3 Utrata łączności z mistrzem gry

W przypadku utraty łączności z mistrzem gry patrz par. *1.7.3.2 Utrata łączności z serwerem* - akapit 1.

1.8. Strategie AI

Każda drużyna przed rozpoczęciem rozgrywki ma przyporządkowaną strategię dla jej graczy. Implementacja programu zawiera więcej niż jedną strategię możliwą do wyboru przez użytkownika aplikacji. Poniżej przedstawiony jest schemat jednej z nich.

Algorytm 1: Strategia brute-force

1. Odkryj pola wokół siebie, po każdym ruchu.
2. Przesuń się w kierunku najbliższego Piece (najmniejszy Distance).
3. Co losową liczbę ruchów (z przedziału [3, 6]) wyślij request komunikacji z innych graczem ze swojej drużyny, lub w 20% przypadków z drużyny przeciwnej.
4. W przypadku requestu od gracza ze swojej drużyny, zawsze się zgódź, wpp zgódź się w 20% przypadków
5. W przypadku podniesienia Piece, komunikuj się z team leaderem, lub w przypadku team leadera, z losowym graczem ze swojej drużyny.
6. Umieść piece w losowym nieodkrytym polu Task.
7. Zakomunikuj efekt kapitanowi