Opis problemu i wizja systemu

Rozproszona platforma do tworzenia gier planszowych, przeznaczona na system Android.

*Niniejsze opracowanie powstało w trakcie i jako rezultat za­jęć dy­dak­tycz­nych z przedmiotu wymienionego na stronie tytułowej, pro­wa­dzo­nych w Akademii Górniczo-Hutniczej w Krakowie (AGH) przez oso­bę (oso­by) wy­mie­nioną (wymienione) po słowach "Pro­wa­dzą­cy zajęcia" i nie może być wy­korzystywane w jakikolwiek sposób i do jakichkolwiek ce­lów, w ca­ło­ści lub części, w szczególności pub­li­ko­wa­ne w ja­ki­kol­wiek spo­sób i w jakiejkolwiek formie, bez uzy­ska­nia uprzed­niej, pi­sem­nej zgody tej oso­by (tych osób) lub odpowiednich władz AGH.****Copyright © 2013 Akademia Górniczo-Hutnicza (AGH) w Krakowie***

**Spis treści**

1. Zakres dokumentu 4

2. Analiza problemu 5

2.1. Opis problemu 5

2.2. Rozpoznanie problemu na przykładach 5

2.2.1. Monopoly 5

2.2.2. Grzybobranie 6

2.2.3. Pędzące Żółwie – Wyścig do Sałaty 6

2.2.4. Chińczyk 7

2.2.5. Podsumowanie 8

2.3. Finalne sformułowanie problemu 9

3. Wizja systemu 11

3.1. Tworzenie logiki gry 11

3.2. Komunikacja między graczami 11

3.3. Grafika 11

3.4. Zapisywanie i wznawianie rozgrywki 12

3.5. Docelowa platforma sprzętowa 12

3.6. Możliwe przypadki użycia platformy 13

3.7. Możliwe przypadki użycia gry 15

3.8. Podsumowanie głównych wymagań 16

4. Koncepcja systemu 18

4.1. Architektura systemu 18

4.1.1. Diagram wdrożenia 18

4.1.2. Opis komponentów 19

4.1.2.1. Komponent game 20

4.1.2.2. Komponent gameResources 21

4.1.2.3. Komponent gameInitializers 21

4.1.2.4. Komponent utilities 22

4.1.2.5. Komponent network 22

4.1.2.6. Komponent platformExceptions 23

4.1.2.7. Komponent tests 24

4.2. Przepływ sterowania dla podstawowych funkcji platformy 24

4.2.1. Tworzenie nowego pokoju gry 25

4.2.2. Dołączanie do istniejącego pokoju 26

4.2.3. Przebieg gry 27

4.2.4. Inicjalizacja obiektu gry 28

5. Bibliografia 30

Indeks ważniejszych terminów 31

Spis ilustracji i tabel 31

# Zakres dokumentu

Dokument zawiera pierwotne sformułowanie problemu, powstałe w wyniku analizy przedmiotu naszego projektu. Znajduje się w nim opis przeanalizowanych przez nas przypadków, wraz z wnioskami wynikającymi z ich analizy. Wyspecyfikowany jest w nim również cel projektu i wymagania strategiczne wobec rozwiązania.

Druga część dokumentu obejmuje powstałą na bazie analizy wizję rozwiązania problemu, w szczególności jego realizowalność, oraz wynikające z niej ograniczenia problemu. Poza tym są w niej zawarte wynikające z analizy ograniczenia technologiczne i samego zakresu rozwiązania.

# Analiza problemu

## Opis problemu

Tematem przedsięwzięcia projektowego jest zaprojektowanie platformy do tworzenia gier planszowych, przeznaczonych do rozgrywki między wieloma graczami. Tworzone na niej gry są przeznaczone na system Android, a komunikacja między graczami ma odbywać się w sposób rozproszony. Podstawowym trybem graficznym jest tryb 2D, z możliwością rozszerzenia do 3D.

Kluczową funkcjonalnością naszej platformy ma być maksymalne ułatwienie twórcy gry jej implementacji.

Szczególną uwagę należy poświęcić grom typu Monopoly.

## Rozpoznanie problemu na przykładach

Aby lepiej zrozumieć istotę problemu przeprowadzona została analiza typowych gier planszowych, aby wyszczególnić podobieństwa i różnice dotyczące planszy, logiki gry, losowości rozgrywki, oraz innych istotnych elementów gier. Analizie poddane zostały cztery wybrane gry planszowe : Monopoly, Grzybobranie, Chińczyk oraz Pędzące żółwie.

W wyniku przeprowadzonego rozpoznania wyszczególnione zostały następujące cechy każdej z tych gier:

### Monopoly

- rozgrywka przebiega w sposób turowy,

- jest jedna możliwa ścieżka, którą mogą poruszać się pionki,

- jedno z pól jest oznaczone jako start,

- plansza jest cykliczna,

- przesunięcie pionka jest determinowane przez kostkę,

- na planszy można wyszczególnić dwa typy pól: pola „zwykłe” i pola specjalne

- pola „zwykłe” mają przypisaną cenę, właściciela, koszt zakupu domku, koszt zakupu hotelu,

wysokość kary dla gracza który znajdzie się na polu przeciwnika; część z tych właściwości jest przypisana na stałe do pola, pozostałe mogą ulec zmianie w trakcie trwania rozgrywki

- do pól specjalnych są przypisane niestandardowe akcje, które pozostają niezmienne w trakcie trwania rozgrywki; pole te to m.in. : więzienie, policjant, pole szansy, pole startu

- w trakcie trwania swojej tury gracz może modyfikować niektóre właściwości pola, poprzez jego zakup lub kupno budynków,

- każdy z graczy może być właścicielem pól,

- każdy gracz ma początkowo przypisaną określoną ilość gotówki, która ulega zmianie w trakcie trwania gry; gracz, któremu skończą się pieniądze, przegrywa;

- gra kończy się w momencie, kiedy pozostaje jeden gracz – zwycięzca

- pionki poszczególnych graczy mają różny wygląd

- pola planszy różnią się od siebie wyglądem

- plansza ma kształt prostokąta

### Grzybobranie

- rozgrywka przebiega w sposób turowy,

- plansza posiada pole startu i mety,

- w niektórych wersjach gry istnieje kilka ścieżek umożliwiających dotarcie do mety,

- kształt planszy może być różny w zależności od wersji,

- każdy z graczy otrzymuje na początku koszyczek,

- grzybki mają różne kolory

- na odpowiednich polach planszy ustawiane są grzybki – jadalne lub niejadalne,

- jeżeli gracz stanie na polu na którym znajdzie się grzybek jadalny zbiera go do koszyczka,

- jeżeli gracz znajdzie się na polu z grzybkiem niejadalnym traci jeden grzybek,

- jeżeli w koszyczku gracza skończą się grzybki czeka dwie kolejki,

- występują pola specjalne, m.in. „Stare rozłożyste drzewo napędziło ci strachu” – gracz cofa się o pięć pól, „Zająłeś się zabawą z napotkanym zajączkiem, nie myślisz o zbieraniu grzybów” – gracz traci jedną turę,

- gracz, który jako pierwszy dotrze do mety wygrywa

- pionki poszczególnych graczy mają różny wygląd

- pola planszy różnią się od siebie wyglądem

### Pędzące Żółwie – Wyścig do Sałaty

- rozgrywka turowa,

- plansza posiada pole startu i mety,

- istnieje jedna ścieżka umożliwiające dotarcie do mety,

- ruch losowy determinowany przez kostkę,

- kolor gracza jest przydzielany losowo, gracz zna tylko kolor swojego żółwia,

- każdemu z graczy przyporządkowane jest 5 losowo wybranych kart,

- w każdej turze gracz wybiera jedną spośród swoich pięciu kart, które mogą określać ruch jego bądź przeciwników,

- w momencie kiedy na zajęte pole wejdzie inny żółw staje on na żółwiu który już zajmuje pole,

- w momencie ruchu żółw zabiera ze sobą wszystkie żółwie które stanęły na jego grzbiecie,

- wygrywa żółw który jako pierwszy dojdzie do mety,

- w przypadku gdy na mecie znajdzie się kilka żółwi ułożonych jeden na drugim wygrywa żółw znajdujący się na dole

- pionki poszczególnych graczy mają różny wygląd

- pola planszy różnią się od siebie wyglądem

### Chińczyk

- do gry potrzebna jest plansza, 16 pionków (cztery w kolorze żółtym, cztery w niebieskim, cztery w zielonym i cztery w czerwonym) oraz kostka do gry.

- gracze, w ustalonej wcześniej kolejności, rzucają [kostką](http://pl.wikipedia.org/wiki/Ko%C5%9B%C4%87_do_gry) po trzy razy. Kiedy któryś z graczy wyrzuci kostką [liczbę](http://pl.wikipedia.org/wiki/Liczba) 6, wtedy ustawia jeden ze swoich czterech pionków na polu startowym (kropka w "[kolorze](http://pl.wikipedia.org/wiki/Barwa) gracza") i rzuca jeszcze raz, a następnie przesuwa pionek o taką liczbę pól w kierunku wskazówek [zegara](http://pl.wikipedia.org/wiki/Zegar_(czasomierz)), ile wyrzuci kostką.

- gracze przesuwają się o taką liczbę pól, jaką liczbę wyrzucą kostką.

- pionki przesuwają się jedynie na tych polach, które na ilustracji obok są oznaczone są kolorem białym (oprócz tego na samym środku).

- jeżeli któryś z graczy wyrzuci 6, ma prawo do jeszcze jednego rzutu (pozostali czekają kolejkę). Kiedy przykładowo ktoś wyrzuci 6, a potem 5, rusza się o 11 pól. Kiedy wyrzuci 6, a potem jeszcze raz 6, rzuca jeszcze raz itd. Gracz, po wyrzuceniu 6, może także wyprowadzić ze "schowka" kolejny pionek.

- jeśli podczas gry pionek jednego gracza stanie na polu zajmowanym przez drugiego, pionek stojący tutaj poprzednio zostaje zbity i wraca do swojego "schowka".

- kiedy gracz obejdzie pionkiem całą planszę dookoła, wprowadza swój pionek do "domku" - czyli czterech pól oznaczonych własnym kolorem. Do "domku" jednego gracza nie mogą wjechać swoimi pionkami inni gracze.

- kiedy gracz wjechał swoim pionkiem do "domku", a na planszy nie ma żadnych innych pionków, nie ma już żadnej możliwości ruchu, rzuca kostką trzy razy tak, jak na początku gry, aż nie wyrzuci 6.

### Podsumowanie

Rozpatrzone przypadki pozwoliły wyszczególnić zestaw cech wspólnych dla większości gier planszowych, na które należy zwrócić szczególną uwagę.

Przede wszystkim nieodłącznym elementem gry planszowej jest plansza, na której odbywa się rozgrywka i po której przemieszczają się pionki. Mimo, że w większości przypadków ruch pionków odbywa się po jednej ścieżce, w niektórych grach planszowych istnieją rozgałęzienia na planszy. Poza tym wygląd planszy może ulec zmianie w trakcie przebiegu rozgrywki.

Na planszy znajdują się pola, które posiadają różne charakterystyki i wiążą się z różnymi akcjami, określonymi w zasadach gry.

Kolejnym elementem każdej gry planszowej jest to, że każdy z graczy ma przyporządkowany jeden lub kilka pionków. Mogą się one różnić od siebie wyglądem. Dobór pionka, którym gracz wykonuje w danej turze ruch jest zależny od gry. Ilość pionków gracza może zmieniać się w trakcie trwania rozgrywki.

Rozgrywka przebiega zawsze w sposób turowy. Poszczególne gry różnią się ilością możliwych do wykonania akcji w danej turze, oraz sposobem określania ilości tych akcji.

Każda z gier ma określoną sytuację startową, tj. początkowe ustawienie pionków, wygląd planszy, przyporządkowanie elementów specjalnych do pionków, graczy, pól.

Warunki zakończenia rozgrywki, jak i sposób wyłaniania zwycięzcy są ściśle określone w zależności od gry.

Poszczególne gry różnią się między sobą dopuszczalną ilością graczy w rozgrywce.

Każda gra ma określony wygląd planszy, pionków, oraz pozostałych elementów gry. Może on ulegać zmianie w trakcie trwania rozgrywki.

## Finalne sformułowanie problemu

W wyniku przeprowadzonej analizy przykładowych gier planszowych problem został sprecyzowany.

Platforma ma przede wszystkim ułatwić i ujednolicić proces tworzenia typowych gier planszowych o następującej charakterystyce:

a) plansza

- różne rozmiary (zmienna ilość pól w zależności od gry),

- różne kształty planszy oraz różne możliwe ścieżki,

- do pól przypisane są atrybuty:

-- akcje specjalne związane z polem

-- przedmioty znajdujące się na polu

-- dodatkowe atrybuty zależne od gry

- wygląd planszy może ulec zmianie w trakcie trwania rozgrywki

b) pionek

- pionek ma przypisane atrybuty:

-- przedmioty związane z pionkiem

-- dodatkowe atrybuty zależne od gry

- każdy z pionków jest przyporządkowany do gracza,

c) logika gry

- określony stan początkowy rozgrywki:

-- początkowy wygląd i układ planszy

-- początkowe ustawienie pionków na planszy

-- początkowe atrybuty każdego z pól planszy

-- początkowe atrybuty każdego z pionków oraz graczy

- określona dopuszczalna ilości graczy, dla których może odbyć się rozgrywka

- określony sposób przemieszczania się pionków po planszy

- sprecyzowany sposób wyłaniania gracza, którego jest tura

- określone warunki zakończenia gry przez gracza:

-- warunki zwycięstwa – jeżeli spełnione, gra dobiega końca

-- warunki przegrania – jeżeli spełnione, gracz przestaje brać udział w rozgrywce

- określone konsekwencje przegrania gry przez gracza

- określone konsekwencje opuszczenia gry przez gracza

- w skład jednej tury gracza może wchodzić kilka akcji

- stan rozgrywki jest sprawdzany po każdej akcji gracza:

-- określone zmiany stanu gry spowodowane akcją podjętą przez gracza,

d) gracze

- każdy z graczy ma przyporządkowany zestaw atrybutów i przedmiotów

- każdy z graczy w stanie początkowym ma przyporządkowaną określoną ilość pionków, która może ulec zmianie w trakcie trwania rozgrywki

e) grafika gry

- zindywidualizowany wygląd elementów rozgrywki (pól, pionków, dodatkowych przedmiotów)

- zmienny wygląd planszy, oraz pozostałych elementów gry

Problem początkowy został dodatkowo poszerzony o możliwość tworzenia bardziej złożonych, strategicznych gier planszowych, co wiąże się z uwzględnieniem następujących aspektów tych gier:

- kształt planszy w trakcie trwania rozgrywki może ulec zmianie

- tworzenie grup pionków w obrębie jednego gracza

- różny wygląd planszy w zależności od gracza

# Wizja systemu

## Tworzenie logiki gry

W związku z różnorodnością zasad gier planszowych platforma musi być na tyle elastyczna, aby możliwe było definiowanie logiki każdej z nich. Z drugiej strony ważne, aby mimo tej różnorodności proces określania zasad gry był w miarę możliwości ujednolicony. Tworzenie gier przy użyciu platformy powinno składać się z ciągu ściśle określonych kroków, dzięki czemu po zapoznaniu się z procesem tworzenia gier na platformie, czynność ta stanie się rutynowa.

Przeniesienie zasad gry z wersji „papierowej” do wersji elektronicznej, z użyciem platformy musi być jak najprostsze i intuicyjne.

## Komunikacja między graczami

Platforma ma zapewniać niewidoczną dla twórcy gry bezprzewodową komunikację między graczami. Całość odpowiedzialności związanej z ustanawianiem połączenia między graczami, przekazywaniem wymaganych danych, tworzeniem i dołączaniem do pokoi należy do platformy.

## Grafika

Twórca gry ma być odpowiedzialny jedynie za dostarczenie grafik reprezentujących poszczególne elementy rozgrywki oraz przypisanie ich do odpowiednich elementów. Platforma zajmuje się wyświetlaniem graficznej reprezentacji rozgrywki u poszczególnych graczy, na podstawie określonego przez twórcę przyporządkowania elementu gry na jego grafikę.

Platforma ma dawać możliwość wyświetlania dodatkowych komunikatów lub grafik dla niestandardowych akcji.

Platforma może być budowana w sposób przyrostowy. Na początku musi zapewniać funkcjonalność odpowiedzialną za wyświetlanie grafii w trybie 2D, z możliwością jej rozszerzenia do 3D.

## Zapisywanie i wznawianie rozgrywki

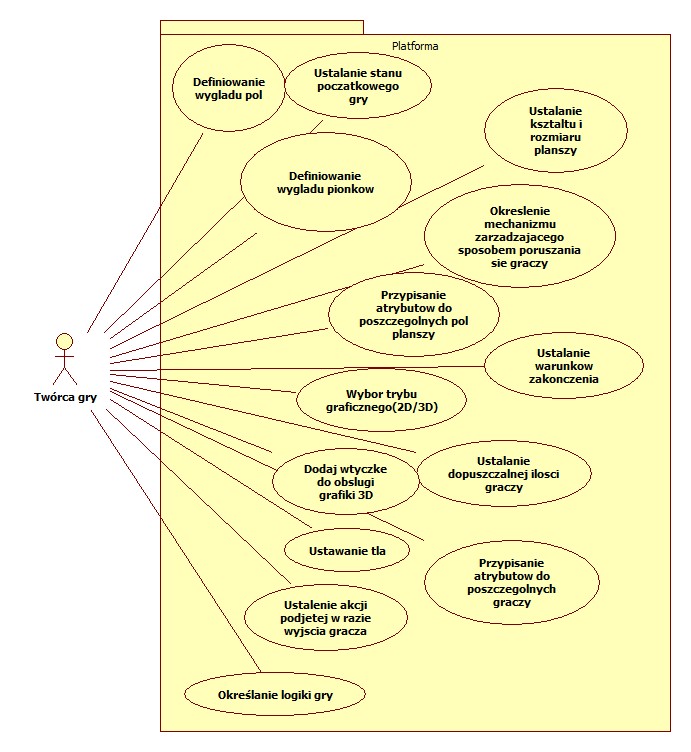
Ze względu na charakter wielu gier planszowych, dla których rozgrywka może trwać bardzo długo platforma musi dawać możliwość zapisywania i wznawiania stanu gry.

Głównym celem platformy jest ułatwienie i ujednolicenie procesu tworzenia typowych gier planszowych, jak Monopoly czy Grzybobranie, równocześnie zachowując elastyczność pozwalającą na tworzenie bardziej zaawansowanych gier.

## Docelowa platforma sprzętowa

Platforma jest przeznaczona do tworzenia gier na urządzenia mobilne z systemem Android wyposażone w interfejs Bluetooth.

## Możliwe przypadki użycia platformy



**Aktor**: Twórca Gry - osoba zajmująca sie implementacja mechanizmów rozgrywki na platformie.

**Możliwe przypadki użycia:**

1. Definiowanie wyglądu pól

Polega na określaniu sposobu graficznej reprezentacji pól planszy.

2. Definiowanie wyglądu pionków

Polega na określaniu sposobu graficznej reprezentacji.

3. Ustalanie stanu początkowego gry

Służy do określenia początkowego położenia pionków na planszy, ustawieniu kolejności ruchów, przypisaniu atrybutów początkowych do gry.

4.Ustalanie warunków zakończenia

Służy do określenia w jakich warunkach i z jakim wynikiem gra sie skończy.

5. Określenie mechanizmu zarządzającego sposobem poruszania sie graczy

Służy do wyboru sposobu określenia przemieszczenia graczy (np. rzut kostka, losowość lub nie).

6. Ustalanie kształtu i rozmiaru planszy.

Określenie ilości pól, ich ułożenie (współrzędne).

7. Ustalanie dopuszczalnej ilości graczy.

Polega na określeniu dla jakich ilości graczy możliwe jest przeprowadzenie rozgrywki.

8. Przypisanie atrybutów do poszczególnych pól planszy

Służy do przypisania atrybutów do pól (dodatkowe parametry, zdefiniowanie akcji specjalnych na polu itp.).

9. Wybór trybu graficznego

Zgodnie z nazwa służy do wyboru trybu 2D ( natywnie wbudowanego) lub trybu 3D z wcześniejszym dodaniu wtyczki do jego obsługi.

10. Dodaj wtyczkę do obsługi grafiki 3D

Pozwala na dodanie wtyczki odpowiadającej za wyświetlanie trybu 3D.

11. Przypisanie atrybutów do poszczególnych graczy

Pozwala na określenie ilości pionków związanych z danym graczem, jego atrybutów specjalnych wynikających ze specyfiki danej gry i ich położenia.

12. Ustalenie akcji podjętej w razie wyjścia gracza

Polega na rozstrzygnięciu w jaki sposób gra ma zareagować na wyjście jednego z graczy (który nie jest serwerem)

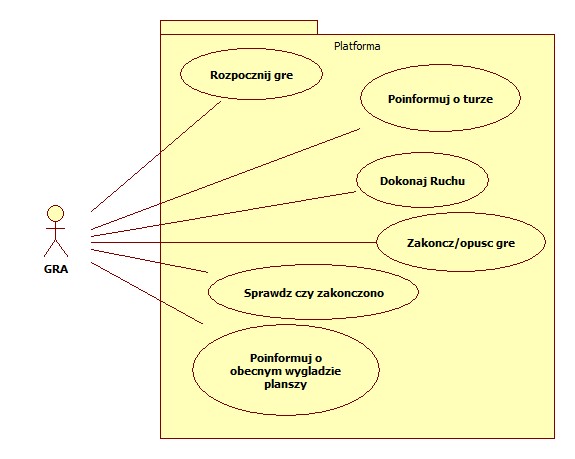
13. Ustawienie tła

Ustawia tło dla planszy i pionków.

14. Określanie logiki gry

Polega na określenie ogólnych zasad przebiegu rozgrywki

## Możliwe przypadki użycia gry



**Aktorzy**: Gra - instancja obiektu gry zaimplementowanej przez użytkownika

**Możliwe przypadki użycia:**

1. Rozpocznij grę

Służy do rozpoczęcia rozgrywki

2. Poinformuj o turze

Instancja rozgrywki na urządzeniu klienta otrzymuje notyfikacje z informacja ze jest obecna tura gracza

3. Dokonaj ruchu

Instancja rozgrywki dokonuje próby ruchu pionkami gracza w określony przez niego sposób

4. Zakończ / opuść grę

Pozwala na zamkniecie serwera gry lub w wypadku klienta opuszczenie rozgrywki

5. Sprawdź czy zakończono

Gra otrzymuje notyfikacje czy rozgrywka została zakończona i z jakim rezultatem.

6. Poinformuj o obecnym wyglądzie planszy

Każdy z klientów otrzyma informacje o obecnym modelu planszy i pionków (położenie pionków) aby następnie ja zwizualizować.

## Podsumowanie głównych wymagań

Główne wymagania stawiane platformie:

- platforma zarządza turami

- przechowuje model gry (pozycje graczy , stan pól)

- umożliwia rożne kształty planszy

- twórca gry ma możliwość ustalania wyglądu pól, wyglądu pionków

- twórca gry ma możliwość ustalenia przemieszczenia graczy w turze

- twórca gry ma możliwość ustalenia warunków zakończenia gry jak i stanu początkowego

- możliwość przypisania dowolnej liczby pionków do danego gracza

- wspieranie grafiki dwuwymiarowej z możliwością napisania rozszerzenia do 3D

- komunikacja miedzy graczami odbywa sie w oparciu o Bluetooth

- aktualizacja stanu gry u wszystkich graczy po turze

- przekazanie sterowania do użytkownika którego obecnie jest tura

- możliwość ustawienia tła dla planszy

- przykładowa implementacja gry Monopol oparta na platformie

- komunikacja w modelu klient-serwer

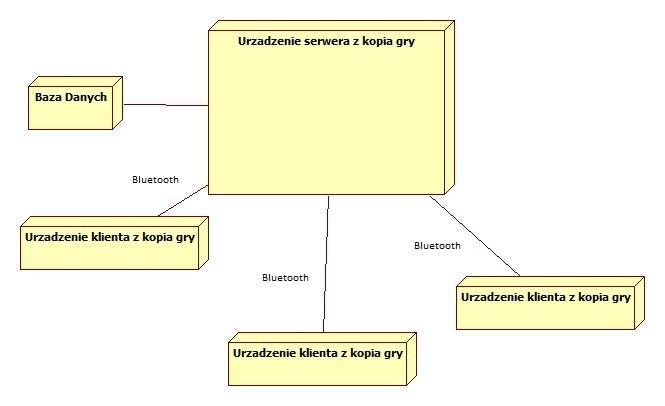
- platforma daje możliwość zapisania i wczytania stanu gry

- grafika dostosowana do ekranu urządzeń mobilnych

# Koncepcja systemu

## Architektura systemu

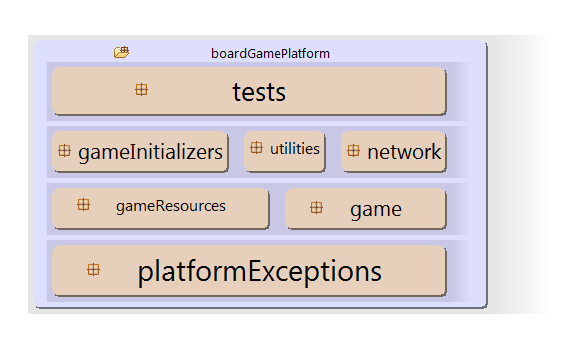
### Diagram wdrożenia



Rysunek ‑ Diagram wdrożenia

Założyliśmy ze nasza platforma będzie działała w architekturze klient serwer z przesyłaniem wiadomości przez Bluetooth. Osoba która zakłada grę jest serwerem który oczekuje na polaczenia od innych graczy którzy staja sie klientami w naszym modelu. Serwer ma dostęp do bazy danych gdzie można zapisywać i odczytywać stany gry.

### Opis komponentów



Rysunek ‑ Diagram komponentów

System składa się z następujących komponentów:

1. **Komponent game:**

Najważniejszy komponent platformy. Znajdują się w nim elementy służące do obsługi przebiegu rozgrywki.

**2.** **Komponent gameResources:**

Zawiera elementy umożliwiające reprezentujące podstawowe elementy składowe każdej gry planszowej, takie jak pionki, plansza, pole planszy, czy przedmioty dodatkowe.

**3. Komponent gameInitializers:**

Znajdują się elementy służące do określania stanu początkowego gry, tzn. wyglądu i układu planszy, pionków, przedmiotów dodatkowych oraz pól planszy. Poza tym komponent jest odpowiedzialny za inicjalizacje wcześniej określonego modelu.

**4. Komponent network:**

Zapewnia komunikację między graczami w trakcie rozgrywki.

**5. Komponent utilities:**

Zawiera klasy pomocnicze służące do ułatwienia tworzenia podstawowych elementów gry, jak i obsługi jej przebiegu, np. przesunięcie pionka o zadaną ilość pól, określenie pola na którym znajduje się któryś z pionków graczy.

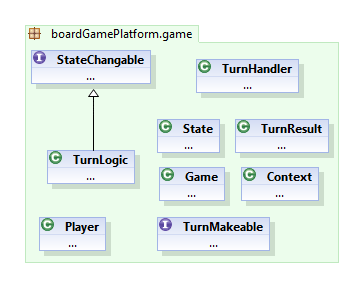
**6. Komponent platformExceptions:**

Jest odpowiedzialny za obsługę błędów.

**7. Komponent tests:**

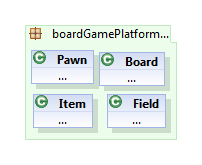
Zawiera testy tworzone w czasie implementacji platformy.

#### Komponent game



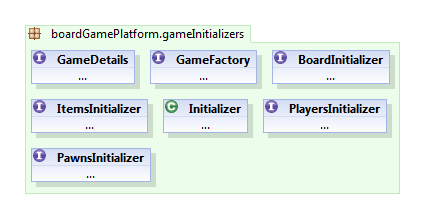
* **Interfejs StateChangable** – reprezentuje obiekt którego stan może być zmieniany
* **Klasa TurnLogic** – reprezentuje logikę tury
* **Klasa TurnHandler** – zajmuje się obsługą przebiegu tury
* **Klasa State** - reprezentuje obecny stan gry
* **Klasa TurnResult** – reprezentuje rezultat wykonania tury przez gracza
* **Klasa Game** – reprezentuje obecną instancję gry, wraz z graczami biorącymi udział w rozgrywce
* **Klasa Context** – przechowuje dodatkowe informacje przekazywane przez grę do graczy w czasie trwania rozgrywki
* **Klasa Player** – reprezentuje gracza biorącego udział w rozgrywce
* **Interfejs TurnMakeable** – reprezentuje obiekt który służy do wykonywania akcji związanych z turą

#### Komponent gameResources



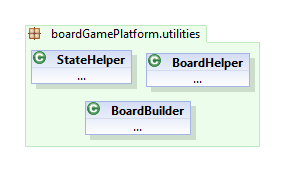
* **Klasa Pawn –** reprezentuje pionek
* **Klasa Board –** reprezentuje planszę na której odbywa się rozgrywka
* **Klasa Field –** reprezentuje pojedyncze pole planszy
* **Klasa Item –** reprezentuje przedmiot dodatkowy

#### Komponent gameInitializers



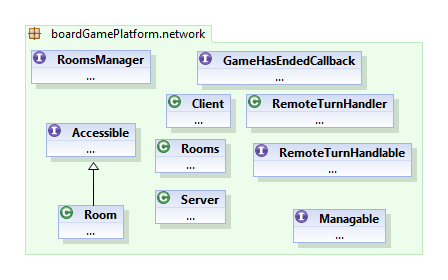
* **Interfejs GameDetails –** reprezentuje dodatkowe dane o grze
* **Interfejs ItemsInitializer -** służy do inicjalizacji przedmiotów dodatkowych
* **Interfejs PawnsInitializer -** służy do inicjalizacji pionków
* **Interfejs GameFactory –** służy do inicjalizacji gry
* **Interfejs BoardInitializer –** służy do inicjalizacji planszy
* **Interfejs PlayersInitializer –** służy do inicjalizacji graczy
* **Initializer –** obiekt inicjalizujący rozgrywkę na podstawie przekazanych danych

#### Komponent utilities



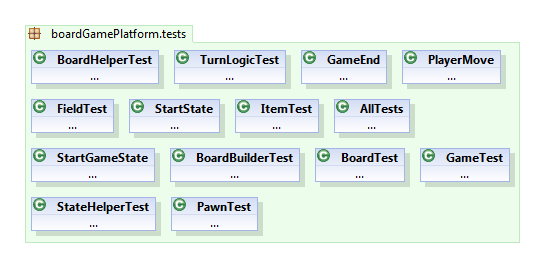
* **Klasa StateHelper –** zawiera funkcje ułatwiające definiowanie stanów gry
* **Klasa BoardHelper –** zawiera funkcje ułatwiające korzystanie z planszy
* **Klasa BoardBuilder –** zawiera funkcje ułatwiające tworzenie planszy

#### Komponent network



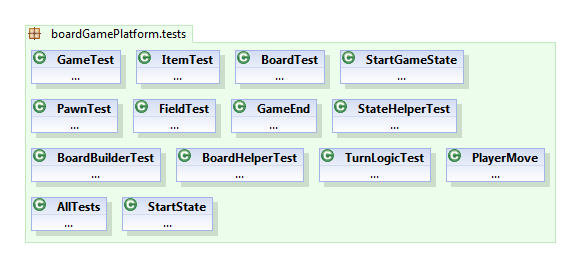
* **Interfejs RoomsManager –** służy do zarządzania pokojami w których odbywa się gra
* **Interfejs Accessible –** służy do dołączania do pokoju
* **Klasa Room –** pokój w którym odbywa się rozgrywka
* **Klasa Client –** przechowuje dane dotyczące połączenia gracza
* **Klasa Rooms –**reprezentacja listy pokoi
* **Klasa Server –** przechowuje listę pokoi i umożliwia dostęp do nich
* **Interfejs GameHasEndedCallback –** określa jakie akcje zostaną wykonane po zakończeniu gry
* **Klasa RemoteTurnHandler –** obiekt służący do przekazania informacji dotyczących obecnego stanu gry do gracza biorącego udział w rozgrywce
* **Interfejs RemoteTurnHandlable –** wykorzystywany do zdalnego odpytywania graczy biorących udział w rozgrywce o kolejne tury
* **Interfejs Managable –** służy do zarządzania pokojem

#### Komponent platformExceptions



Model służacy do obsługi błędów wewnętrznych platformy. Szczegółowy opis w jakich sytuacjach rzucane są wyjątki znajduje się w dokumentacji technicznej.

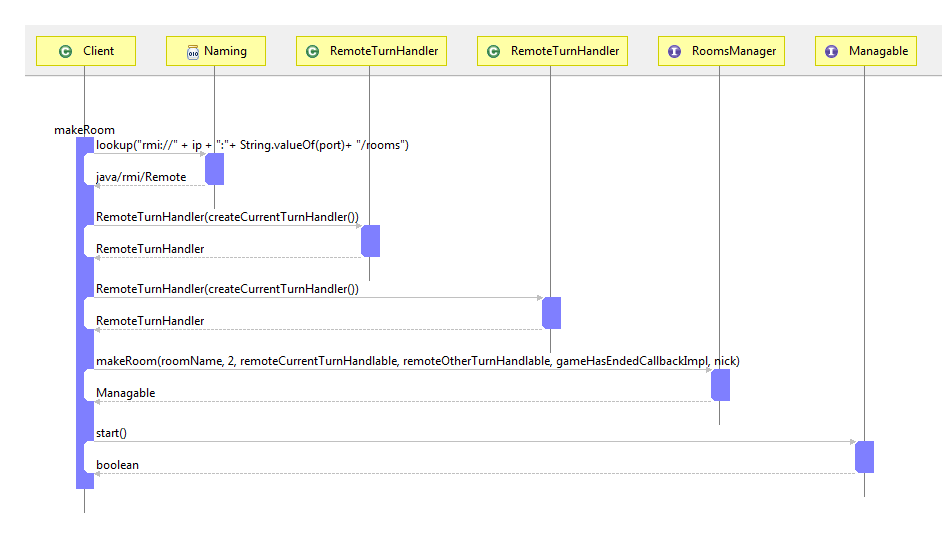
#### Komponent tests



Moduł zawiera testy tworzone na potrzeby sprawdzania poprawności działania platformy z pakietów boardGamePlatform.game , boardGamePlatform.gameResources, boardGamePlatform.utilities.

## Przepływ sterowania dla podstawowych funkcji platformy

### Tworzenie nowego pokoju gry

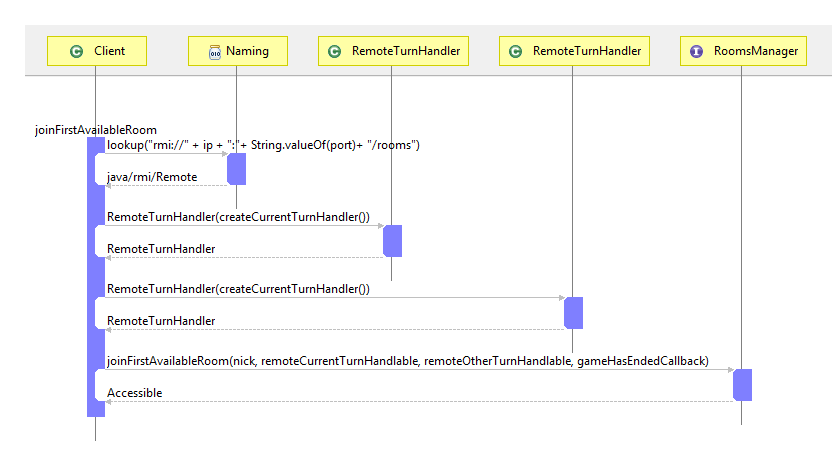


W celu utworzenia pokoju klient odpytuje serwer o listę pokoi. Kiedy ją otrzyma inicjalizowane są obiekty do obsługi tur gracza (zarówno do obsługi gracza w przypadku gdy jest jego tura jak i wtedy gdy jest tura przeciwnika) oraz jego obiekt callback, który zostanie przekazany serwerowi i posłuży do informowania gracza o zakończeniu rozgrywki.

Następnie na obiekcie przechowującym listę pokoi, zwróconym przez serwer, wywoływana jest metoda makeRoom, zwracająca obiekt implementujący interfejs Managable. Pozwala on na rozpoczęcie rozgrywki.

W momencie wywołania na obiekcie przechowującym listę pokoi metody makeRoom klient przekazuje obiekty służące serwerowi do odpytywania go o ruchy w kolejnych turach oraz obiekt callback to informowania o końcu gry.

### Dołączanie do istniejącego pokoju



Aby dołączyć do pokoju klient musi najpierw odpytać serwer o listę istniejących pokoi. Następnie na obiekcie przechowującym listę pokoi, zwróconym przez serwer, wywoływana jest metoda joinFirstAvailableRoom, zwracająca obiekt implementujący interfejs Accessible. Pozwala on na opuszczenie pokoju, jeżeli klient sobie tego zażyczy.

### Przebieg gry

## Przebieg gry.png

W momencie wywołania na obiekcie Game metody start() rozpoczyna się główna pętla programu. W każdej iteracji sprawdzane jest czy gra została zakończona (turnLogic.gameEnded()). Jeżeli nie, obiekt zarządzający logiką gry jest odpytywany o gracza którego obecnie jest tura oraz o obecny kontekst gry.

Następnie przygotowywane są obiekty, które będą przechowywać rezultaty tury. Dla gracza którego obecnie jest tura jest to pojedynczy obiekt TurnResult, natomiast dla pozostałych graczy lista obiektów tej klasy.

W kolejnym kroku gracze są odpytywani o wykonanie tury. W tym celu wykorzystywana jest metoda obiektów RemoteTurnHandler przekazanych przez graczy w momencie dołączania. Zostaje jej przekazany także obecny kontekst gry dzięki czemu wszyscy gracze otrzymują informacje o obecnej sytuacji tj.:

- obecnym stanie planszy,

- liście aktywnych graczy biorących udział w rozgrywce,

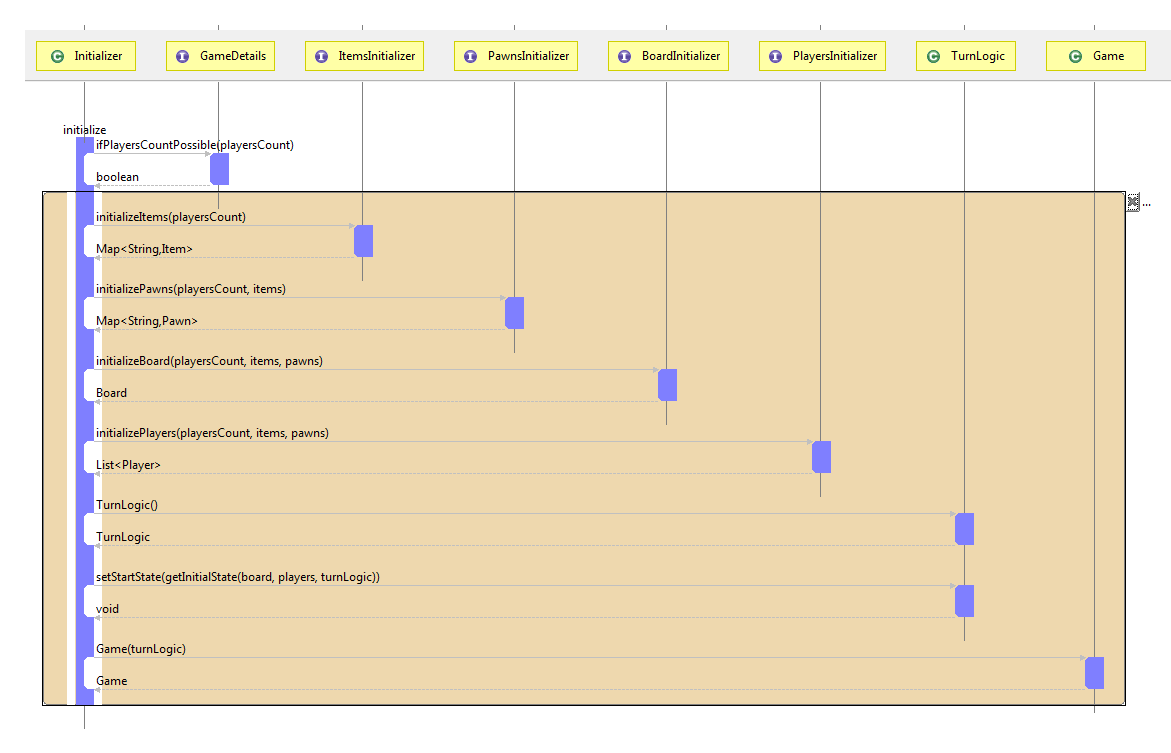
- obecnym stanie gry.

Sposób w jaki obecna sytuacja zostanie zinterpretowana po stronie każdego z klientów (rozróżnienie na gracza którego obecnie jest tura i pozostałych) jest zależny od implementacji gry przez twórcę gry.

Po wykonaniu wymaganych operacji po stronie graczy każdy z nich zwróci wynik tury.

Wyniki te zostaną następnie przekazane do logiki gry, która zinterpretuje je w sposób zdefiniowany przez gracza.

### Inicjalizacja obiektu gry



Za inicjalizację obiektu gry odpowiedzialna jest obiekt klasa Initializer. W momencie tworzenia obiektu tej klasy w konstruktorze przekazywany jest do niej obiekt implementujący GameFactory, implementowany przez twórcę gry, zawierający wszystkie informacje konieczne do inicjalizacji początkowego stanu gry, tzn:

- planszy,

- listy graczy, wraz z ich atrybutami i przedmiotami dodatkowymi,

- pól wraz z ich atrybutami i przedmiotami dodatkowymi,

- dodatkowych przedmiotów koniecznych do prawidłowego przebiegu rozgrywki,

- pionków wraz z ich dodatkowymi atrybutami.

Poza tym obiekt implementujący GameFactory pozwala na uzyskanie informacji o nazwie implementowanej gry, początkowym stanie gry oraz możliwych ilościach graczy, dla których może zostać przeprowadzona rozgrywka.

Dzięki takiemu rozwiązaniu sama inicjalizacja ogranicza się do wykonania prostego algorytmu, wykorzystującego fabrykę dostarczoną przez twórcę. Wykonywane są następujące kroki:

1) sprawdzenie czy możliwe jest zainicjalizowanie rozgrywki dla wymaganej ilości graczy

2) jeżeli nie zwracany jest false

3) jeżeli tak następuje inicjalizacja kolejnych elementów:

- przedmiotów dodatkowych

- pionków

- planszy

- graczy

- logiki gry, z określonym przez twórcę gry stanem początkowym

- samego obiektu gry.

# Bibliografia

1. Brett D. McLaughlin, Gary Police & David West: *Object Oriented Analysis and Design*, O’Reilly, 2007

Indeks ważniejszych terminów

**Nie można odnaleźć żadnych haseł indeksu.**

1. Spis ilustracji i tabel

Ilustracje

Rysunek 2‑3 Diagram wdrożenia 8

Rysunek 2‑4 Diagram komponentów 9

Tabele