My Fishery – Raport B: Projekt Systemu

Wymagania funkcjonalne (FR)

FR-A. Monitorowanie i Utrzymanie Jakości Wody

- **FR-A1**. System zbiera dane z czujników jakości wody: np. tlen rozpuszczony
- **FR-A2.** System steruje napowietrzaniem (z użyciem pompy).

FR-B. Analityka i zalecenia dla właściciela łowiska

- **FR-B1.** System wylicza wskaźniki „stanu łowiska” (np. kondycja jakości wody, szacowany rybostan).
- **FR-B2.** System wysyła powiadomienia:
  - zalecenie zarybienia
  - alarm dotyczący niskiej jakości wody zawierający informację o przy przekroczeniu parametrów np. tlenu rozpuszczonego
- **FR-B3.** Rejestrowanie operacji zarybiania przez właściciela (wprowadzenie danych np. dot. gatunku, ilości, daty).

FR-C. Automatyzowanie Karmienia

- **FR-C1.** Sterowanie automatycznym karmieniem: ustalanie wielkości porcji i harmonogramu karmienia.
- **FR-C2.** Sprawdzanie stanów magazynowych karmy.
- **FR-C3.** Automatyczne zamawianie karmy za pomocą maila do sklepu.

FR-D. Limity obowiązujące wędkarzy

- **FR-D1.** Definiowanie limitów: maks. liczba wędkarzy jednocześnie oraz liczba ryb do zabrania (globalnie i per-gatunek).
- **FR-D2.** Blokada zgłoszenia zarybienia obowiązująca wszystkich wędkarzy przy przekroczeniu dziennego limitu.
- **FR-D3.** Powiadomienia do wędkarzy o aktualnych limitach.
- **FR-D4.** Wędkarz zgłasza dane dotyczące połowu: gatunek, wielkość/szacowana masa, akcja (zabranie/wypuszczenie), czas.

FR-E. Optymalizacja wielkości i różnorodności rybostanu

- **FR-E1.** System integruje dane z kamer rozpoznających gatunki oraz sonarów zliczających ryby i analizuje liczbę ryb
- **FR-E2**. System uwzględnia zadaną i aktualną wielkość ryb w częstotliwości karmienia

- **FR-E3.** System sygnalizuje zalecenie zarybiania danym gatunkiem aby sterować różnorodnością rybostanu
- **FR-E4.** Sezonowość wpływa na zalecenia zarybienia (np. zalecane okna zarybiania wiosną przy odpowiedniej temperaturze).
- **FR-E5.** System analizuje, ile ryb jest wielkości S/M/L.
- **FR-E6**. Zgłoszenie połowu wpływa na szacowany stan łowiska: liczbę ryb, udziały gatunków, rozmiar ryb.

Model Ról

Wyróżniliśmy następujące role:

- Właściciel łowiska

- Wędkarz

- Wodnik – Odpowiedzialny za stan jakości wody

- Karmink – odpowiedzialny za karmienie

- Rola managera różnorodności i wielkości rybostanu (DEI)

- Rola managera wielkości ryb

Rola 1. WL - Właściciel łowiska (prezes)

Protokoły i aktywności są zapisane w snake_case, ponieważ jako język implementacji prawdopodobnie wybierzemy python.

Owner
Opis Rola odzwierciedla faktycznego właściciela, który będzie podejmował działania. Owner korzysta z danych przekazywanych przez czujniki, DEI oraz Wodnika i na ich podstawie wydaje decyzje operacyjne: - wydaje pozwolenia na wejścia wędkarzy na łowisko - śledzi aktualną liczbę wędkarzy na łowisku - wydaje pozwolenia na zabranie ryby - wydaje rekomendacje zarybienia - steruje celem różnorodności rybostanu - wydaje rekomendacje poprawy jakości wody
Pozwolenia **Czyta** Alarm o krytycznej jakości wody - *water_quality_alarm* Alarm o potrzebie zarybienia - needs_stocking_alarm Dane o rybostanie – *fish_state* Stan jakości wody – *water_quality* Liczba zabranych ryb przez wędkarza – *fishes_taken_count*

Liczba wędkarzy na łowisku - *fisherman_count*

Generuje

informację o wykonanym zarybieniu – *stocking_data*
Limity obowiązujące wędkarzy - *fish_takes_limit*
Pozwolenie wejścia na łowisko - *enter_permission*
Pozwolenie na zabranie ryby - *take_fish_permission*
Nowy cel różnorodności – *diveristy_target*
Rekomendację poprawienia jakości wody - *water_clear_recommendation*
Rekomendacje zarybienia – *stocking_recommendation*

Obowiązki

- **Żywotne:**
  - *Owner = (ManageFishermen || ManageStocks || ManageWaterQuality)*$^{\omega}$
  - *ManageFishermen = set_fisher_limit || (ManageFishermanEntrance | (ManageFishermanEntrance . ManageFishesTaken)) || ManageFishermanExit)*
  - *ManageFisherEntrance = if_can_enter_response | if_can_enter_response . set_fishermen_count*
  - *ManageFishesTaken = if_can_take_fish_response*
  - *ManageFishermanExit = register_exit_response . set_fishermen_count*
  - *ManageStocks = set_diversity_target_request || (receive_needs_stocking_alarm . recommend_stocking . register_stocking)*
  - *ManageWaterQuality = receive_water_quality_alarm . recommend_water_clear*
- ***Bezpieczeństwa:***
  - *fishes_taken_count ≥ fish_takes_limit => take_fish_permission = false*
  - *fisherman_count ≥ fisherman_limit => enter_permission = false*
  - *water_quality_alarm = true => water_clear_recommendation = true*

Protokoły

- `if_can_enter_response` - Pozwolenie na wejście
- `if_can_take_fish_response` - pozwolenie na zabranie ryby
- `set_diversity_target_request` – ustawienie celu różnorodności (z DEI)
- `register_exit_response` – odebranie informacji od wędkarza o opuszczeniu łowiska
- `receive_needs_stocking_alarm` – odebranie alarmu o wymaganym zarybieniu
- `receive_water_quality_alarm` – odebranie alarmu o krytycznej jakości wody

Aktywności

- `recommend_stocking` - Rekomendacja zarybienia (dla os. Fizycznej)
- `recommend_water_clear` - Rekomendacja poprawy jakości wody (dla os. Fizycznej)
- `register_stocking` – Zarejestrowanie zarybienia (rejestracja przez os. Fizyczna)
- `set_fisher_limit` – ustawienie limitu wędkarzy
- `set_fishermen_count` – ustawienie aktualnej liczby wędkarzy

Rola 2. Wędkarz

Fisherman
Opis Fisherman reprezentuje użytkownika łowiska. - zgłasza prośby o wejście na łowisko - zgłasza zabranie ryby - przekazuje dane o złowionych rybach - Rejestruje swoje wejście na łowisko, wyjście i połowy - współpracuje z systemem i przestrzegając limitów ustalonych przez Właściciela.
Pozwolenia **Czyta** decyzja o wejściu na łowisko - *enter_decision* decyzja o pozwoleniu na zabranie ryby – *take_fish_decision* **Generuje** dane wędkarza – *fisherman_data* rejestracja wejścia na łowisko – *register_enter* dane o złowionej rybie - *fish_data*
Obowiązki - **Żywotności**: o Fisherman = if_can_enter_request | (if_can_enter_request . register_enter . Fishing . register_exit_request) o Fishing = (if_can_take_fish_request | (if_can_take_fish_request . register_take_fish register_fish_data_request . register_fish_size_request))* - **Bezpieczeństwa**: o enter_decision = true o take_fish_decision ≠ true => fish_data = NULL
Protokoły - if_can_enter_request - Prośba do właściciela o wejście - if_can_take_fish_request - Prośba do właściciela o zabranie ryby - register_fish_data_request - Przesłanie danych o rybie do DEI - register_fish_size_request - przesłanie danych o rybie do Mng. wielkości osobników - register_exit_request – Informacja do właściciela o opuszczeniu łowiska **Aktywności** - register_enter - Wejście na łowisko (rejestracja przez użytkownika) - register_take_fish - Zarejestrowanie ryby

Rola 3. Wodnik – Manager jakości wody

WaterQualityManager
Opis Rola odpowiada za pomiar jakości wody (zbieranie danych, analizę parametrów) oraz informowanie Właściciela o aktualnym stanie wody. Steruje pompą napowietrzającą wodę.

Pozwolenia
Generuje alarm o krytycznej jakości wody - *water_quality_alarm*
Dane/ocenę jakości wody - *water_quality*

Czyta

Obowiązki:
- **Żywotności**:
  - *WaterQualityManager = (WaterQualityMeasure)ω*
  - *WaterQualityMeasure = collect_data . calculate_quality | (calculate_quality . send_water_quality_alarm . aeration)*
- **Bezpieczeństwa**:
  - *water_quality = bad => water_quality_alarm = true*

Protokoły
- send_water_quality_alarm – alarm o krytycznej jakości wody do ownera

Aktywności
- collect_data - Zbieranie parametów stanu wody
- calculate_quality – Szacowanie oceny jakości wody
- aeration – napowietrzanie (uruchomienie pompy)

Rola 4. Karmnik

Feeder

Opis
Rola zajmuje się realizacją procesu karmienia ryb.
- odbiera parametry karmienia: wielkość porcji i częstotliwość
- steruje automatycznym karmieniem
- monitoruje zapasy karmy
- zamawia karmę

Pozwolenia
Czyta parametry karmienia - set_feeding_parameters_response
potrzebną ilość karmy - required_food_supplies
ilość karmy w magazynie - food_supplies

Generuje zmienną reprezentującą potrzebę zamówienia karmy - order_food_need

Obowiązki
- **Żywotności**:
- Feeder = (set_feeding_parameters_response || feed || (check_food_suplies | check_food_suplies . order_food))ω
- **Bezpieczeństwa**:
  - food_supplies < required_food_supplies => order_food_need = true

Protokoły
- set_feeding_parameters_response - Ustawienie parametrów karmienia

Aktywności
- feed - Karmienie
- order_food - Zamawianie karmy

- `check_food_supplies` - Sprawdzenie stanów magazynowych karmy

Rola 5. DEI - Manager różnorodności i wielkości rybostanu

DEI (Diversity, Equity and Inclusion)

Opis

Rola odpowiedzialna za sterowanie różnorodnością gatunkową i liczbą ryb w łowisku.
- reaguje na zmianę celu różnorodności
- odbiera dane o nowej rybie od wędkarza
- zbiera i analizuje dane o rybostanie, określa liczebność i udział gatunków
- ocenia, czy istnieje potrzeba zarybiania, jeśli tak, powiadamia właściciela

Pozwolenia

Czyta cel różnorodności – `diversity_target`
dane o złowionej rybie – `fish_data`
dane z kamer – `camera_data`
dane z sonarów – `sonar_data`

Generuje dane o rybostanie – `fish_state`
ocena potrzeby zarybienia – `needs_stocking`
rekomendacja zarybienia - `stocking_recommendation`

Obowiązki

- **Żywotności**:
  - `DEI = (set_diversity_target_response || register_fish_data_response || MonitorFishState || ManageRestocking )`$^{\omega}$
  - `MonitorFishState = collect_camera_data || collect_sonar_data`
  - `ManageRestocking = (if_needs_stocking | if_needs_stocking . send_needs_stocking_alarm)`
- Bezpieczeństwa:
  - `needs_stocking => stocking_recommendation = true`

Protokoły

- `set_diversity_target_response` - ustawienie celu róznorodności (od Właściciela rybostanu)
- `register_fish_data_response` - Odebranie danych o próbce ryby – gatunek + jedna ryba (od wędkarza)
- `send_needs_stocking_alarm` – alarm do Ownera o potrzebie zarybienia

Aktywności

- `collect_camera_data` - Zbieranie danych z kamer rozpoznających gatunki ryb
- `collect_sonar_data` - Integrowanie dane z sonarów zliczających
- `if_needs_stocking` - Czy jest potrzeba zarybienia

Rola 6. Manager wielkości osobników

FishHealthManager

Opis

Rola zajmuje się oceną wielkości i kondycji osobników ryb. Na podstawie przekazanych danych:

- odbiera dane od wędkarzy o złowionych rybach
- szacuje średnie rozmiary ryb
- sugeruje zmiany parametrów karmienia

Pozwolenia

Czyta wielkość złowionej ryby – `fish_size`

Generuje sugestię zmiany parametrów karmienia – *`feeding_parameters_change_recommendation`*

flagę, czy należy zmienić parametry karmienia - *`is_feeding_insufficient`*

Obowiązki

- Żywotności:
  - *`FishHealthManager = ( register_fish_size_response || calculate_fish_avg_size || (revaluate_feeding | revaluate_feeding . set_feeding_parameters_request))`*$^{\omega}$
- Bezpieczeństwa:
  - *`is_feeding_insufficient = true => feeding_parameters_change_recommendation = true`*

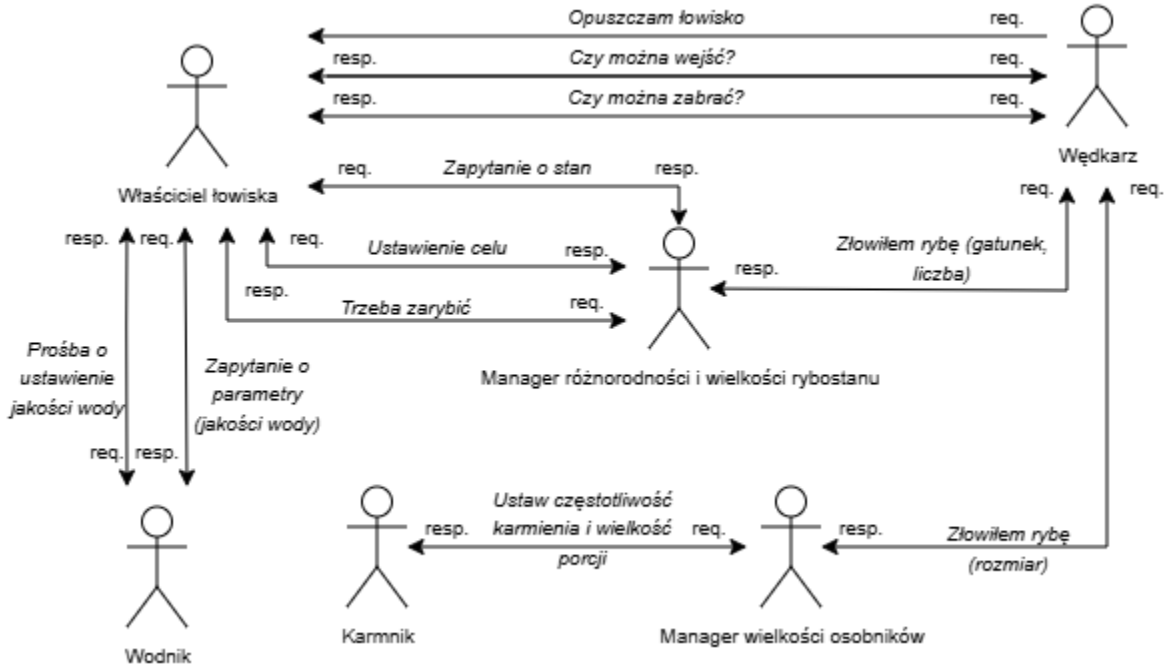
Protokoły

- `set_feeding_parameters_request` – Prośba o ustawienie parametrów karmienia
- `register_fish_size_response` – Odpowiedź na zarejestrowanie wielkości złowionej ryby

Aktywności

- `revaluate_feeding` – kalkulacja wielkości porcji karmienia
- `calculate_fish_avg_size` – kalkulacja szacowanej średniej wielkości ryb

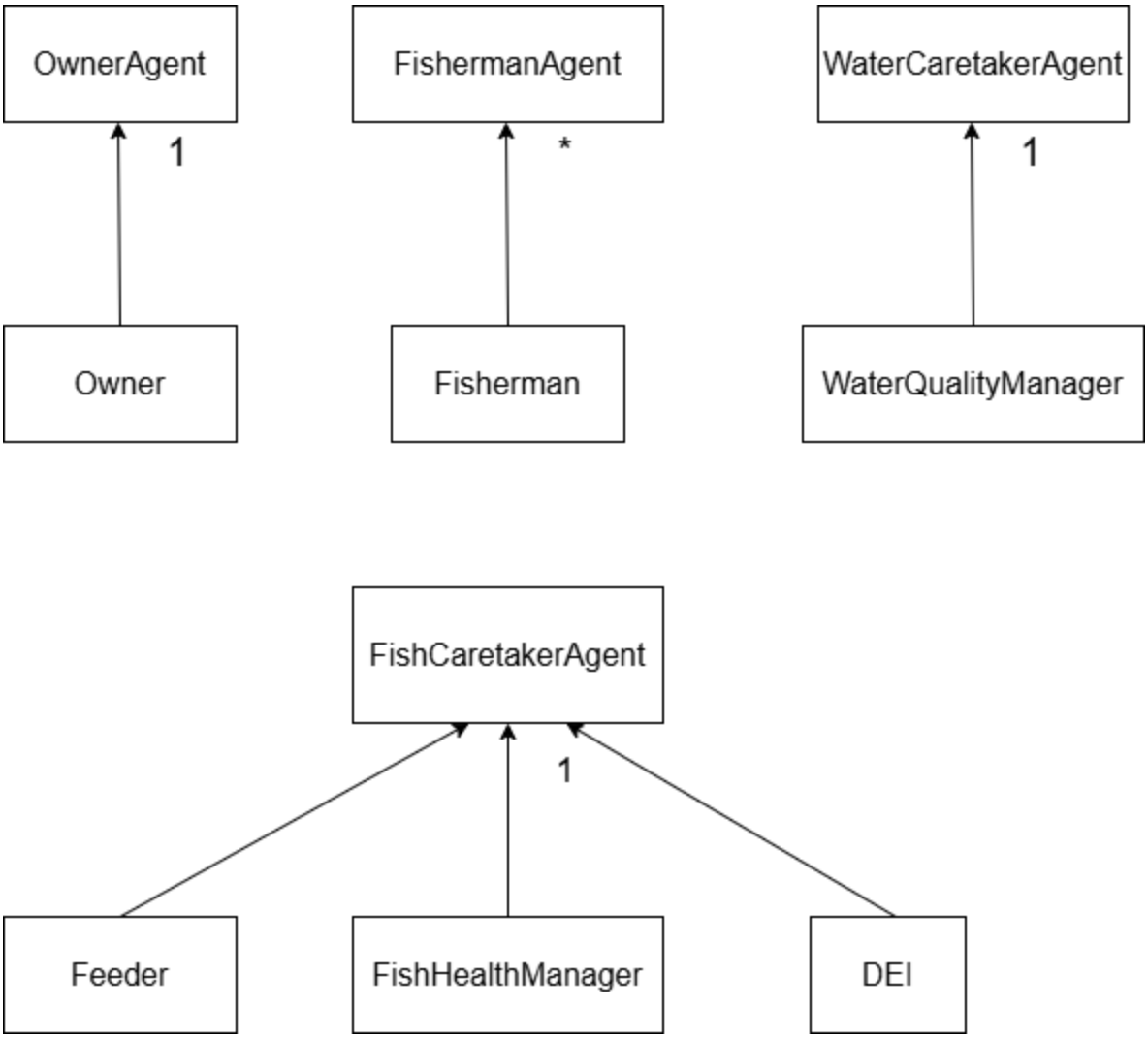
Wstępny model interakcji



Model GAIA

Model agentów

Po opisie ról kolejnym elementem było zamodelowanie istniejących w systemie agentów i przypisanie do nich poszczególnych ról. Ponieważ role *Feeder*, *FishHealthManager* i *DEI* odnoszą się do podobnych kwestii ściśle związanych z opieką ryb, postanowiliśmy przypisać je do jednego agenta – *FishCaretakerAgent*. Pozostałe role przypisaliśmy do agentów “jeden do jednego”.



Model usług

Poniżej przedstawione są usługi skojarzone z poszczególnymi agentami. Ze względu na długie ciągłe nazwy poszczególnych pól nie umieściliśmy ich w tabeli, ponieważ nazwy ty nie zmieściłyby się w kolumnach.

Model dla agenta OwnerAgent:

==========

Usługa: `check_if_fisherman_can_enter`

Wejścia: `fisherman_count, fisherman_limit`

Wyjścia: `if_can_enter_response`

Warunki początkowe: `check_if_can_enter_response = NULL`

Warunki końcowe: `if_can_enter_response ∈ {true, false}`

Opis: Podejmij decyzję, czy rybak może wejść na łowisko.

==========

Usługa: `inform_fisherman_about_entrance_possibility`

Wejścia: `if_can_enter_response`

Wyjścia: `NULL`

Warunki początkowe: `if_can_enter_response ∈ {true, false}`

Warunki końcowe: `fisherman_acknowledge`

Opis: Powiadom wędkarza, czy może wejść na łowisko

==========

Usługa: `check_if_can_take_fish`

Wejścia: `taken_fish_data, fishery_stock`

Wyjścia: `if_can_take_fish`

Warunki początkowe: `check_if_can_take_fish`

Warunki końcowe: `if_can_take_fish ∈ {true, false}`

Opis: Podejmij decyzję, czy złowiona przez wędkarza ryba może być przez niego zabrana.

==========

Usługa: `inform_fisherman_about_fish_take_possibility`

Wejścia: `if_can_take_fish`

Wyjścia: `NULL`

Warunki początkowe: `if_can_take_fish ∈ {true, false}`

Warunki końcowe: `fisherman_acknowledge`

Opis: Powiadom wędkarza, czy może zabrać ze sobą rybę

==========

Usługa: `ask_for_water_quality_parameters`

Wejścia: `NULL`

Wyjścia: `water_parameters`

Warunki początkowe: `is_water_caretaker_available = true`

Warunki końcowe: `water_parameters ≠ NULL`

Opis: Poproś o parametry jakości wody.

==========

Usługa: `request_for_water_quality_change`

Wejścia: `water_parameters`

Wyjścia: `NULL`

Warunki początkowe: `true`

Warunki końcowe: `water_caretaker_acknowlegde`

Opis: Poproś o interwencję w jakość wody.

==========

Usługa: `set_stock_diversity_policy`

Wejścia: `stock_diversity`

Wyjścia: `NULL`

Warunki początkowe: `is_fish_caretaker_available`

Warunki końcowe: `fish_caretaker_acknowlegde`

Opis: Poinformuj agenta dbającego o stan zarybienia o nowej polityce (wielkość populacji karpi, okoni, itp.)

==========

Usługa: `ask_for_fishery_stock`

Wejścia: `NULL`

Wyjścia: `fishery_stock`

Warunki początkowe: `is_fish_caretaker_available`

Warunki końcowe: `fish_caretaker_acknowlegde`

Opis: Zapytaj agenta FishCaretakerAgent o stan zarybienia.

==========

Usługa: `request_for_restocking_recommendation`

Wejścia: `NULL`

Wyjścia: `restocking_recommendation`

Warunki początkowe: `is_fish_caretaker_available`

Warunki końcowe: `restocking_recommendation ≠ NULL`

Opis: Zapytaj agenta dbającego o to, czy, jego zdaniem, powinno się wpuścić do łowiska więcej ryb.

Model dla agenta FishermanAgent:

==========

Usługa: `ask_for_entrance_permission`

Wejścia: `NULL`

Wyjścia: `if_can_enter_response`

Warunki początkowe: `is_owner_available`

Warunki końcowe: `if_can_enter ∈ {true, false}`

Opis: Zapytaj właściciela, czy możesz wejść na łowisko.

==========

Usługa: `ask_for_fish_take_permission`

Wejścia: `fish_data`

Wyjścia: `if_can_fish_take_response`

Warunki początkowe: `is_owner_available`

Warunki końcowe: `if_can_take_fish ∈ {true, false}`

Opis: Zapytaj właściciela, czy możesz zabrać ze sobą złowioną rybę.

==========

Usługa: `inform_owner_about_leaving_fishery`

Wejścia: `NULL`

Wyjścia: `NULL`

Warunki początkowe: `is_owner_available`

Warunki końcowe: `owner_acknowledge`

Opis: Poinformuj właściciela, że opuszczasz łowisko.

==========

Usługa: `inform_fish_caretaker_about_taken_fish`

Wejścia: `fish_data, fish_size`

Wyjścia: `NULL`

Warunki początkowe: `is_fish_caretaker_available`

Warunki końcowe: `fish_caretaker_acknowledge`

Opis: Poinformuj agenta dbającego o stan zarybienia o rodzaju i wielkości złowionej ryby.

Model dla agenta FishCaretakerAgent:

==========

Usługa: `inform_owner_about_stocking`

Wejścia: `fishery_stock`

Wyjścia: `NULL`

Warunki początkowe: `is_owner_available`

Warunki końcowe: `owner_acknowledge`

Opis: Poinformuj właściciela o stanie zarybienia.

==========

Usługa: `give_restocking_recommendation`

Wejścia: `fishery_stock, stock_diversity`

Wyjścia: `restocking_recommendation`

Warunki początkowe: `is_owner_available`

Warunki końcowe: `owner_acknowledge`

Opis: Zarekomenduj właścicielowi dodania do łowiska więcej ryb – lub niedodania.

Model dla agenta WaterCaretakerAgent:

==========

Usługa: `give_water_quantity_parameters`

Wejścia: `NULL`

Wyjścia: `water_parameters`

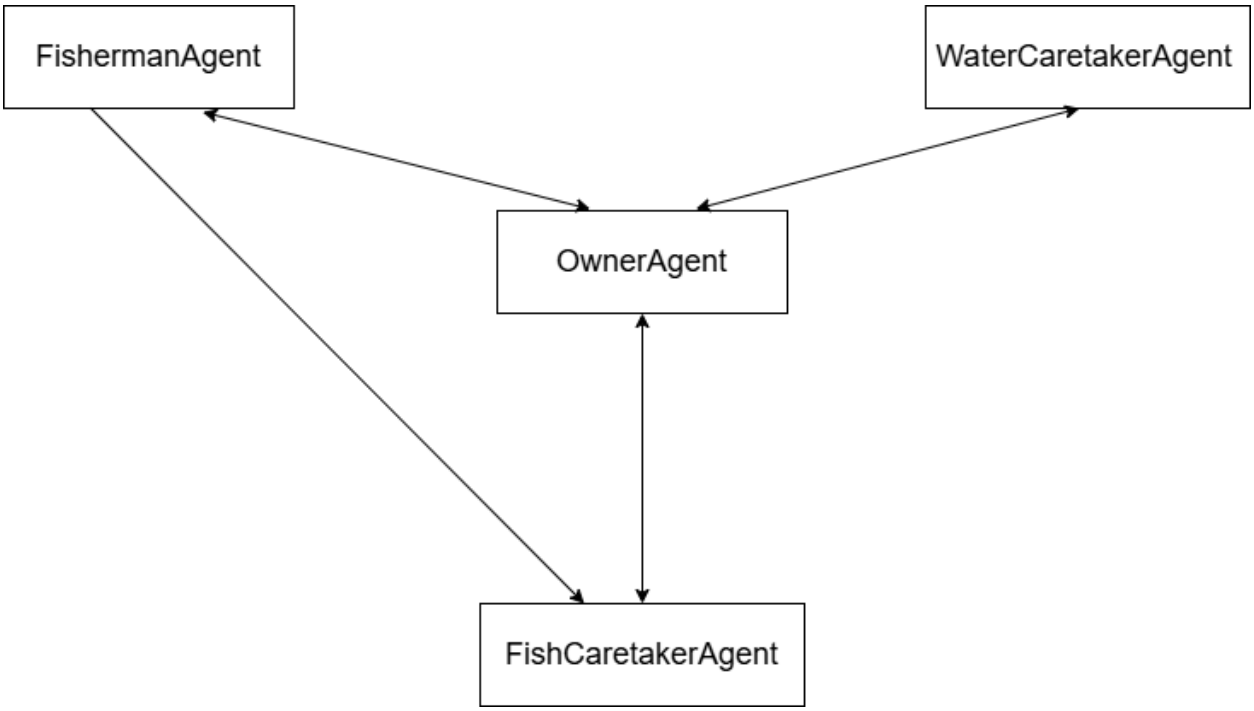
Warunki początkowe: `is_owner_available`

Warunki końcowe: `water_parameters ≠ NULL AND owner_acknowledge`

Opis: Zarekomenduj właścicielowi dodania do łowiska więcej ryb – lub niedodania.

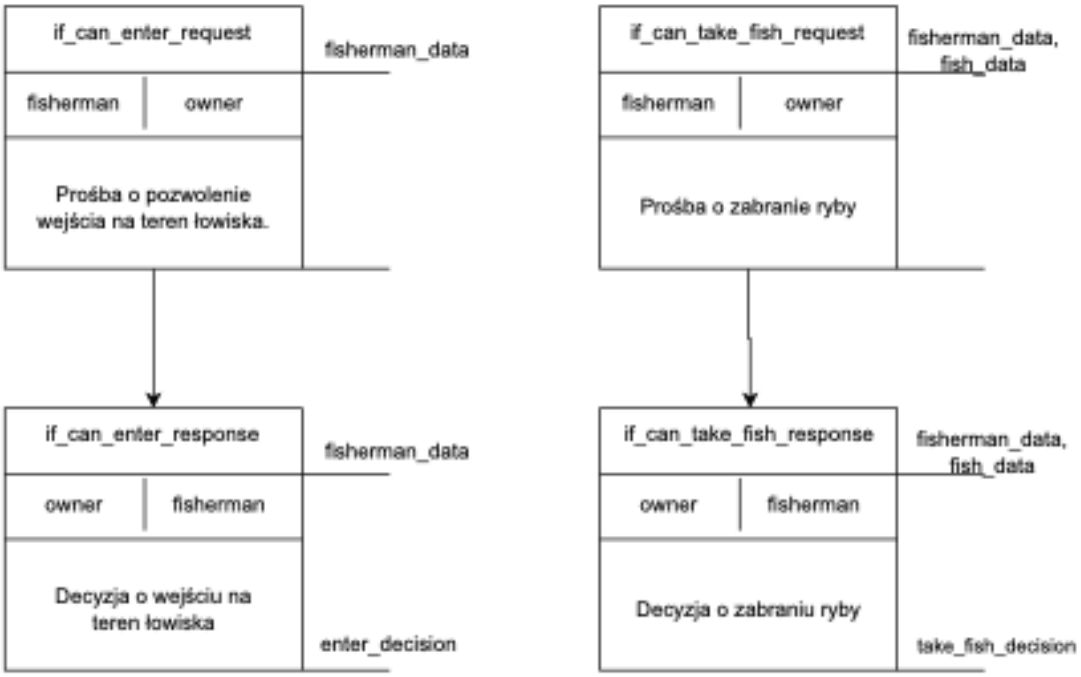
Model znajomości

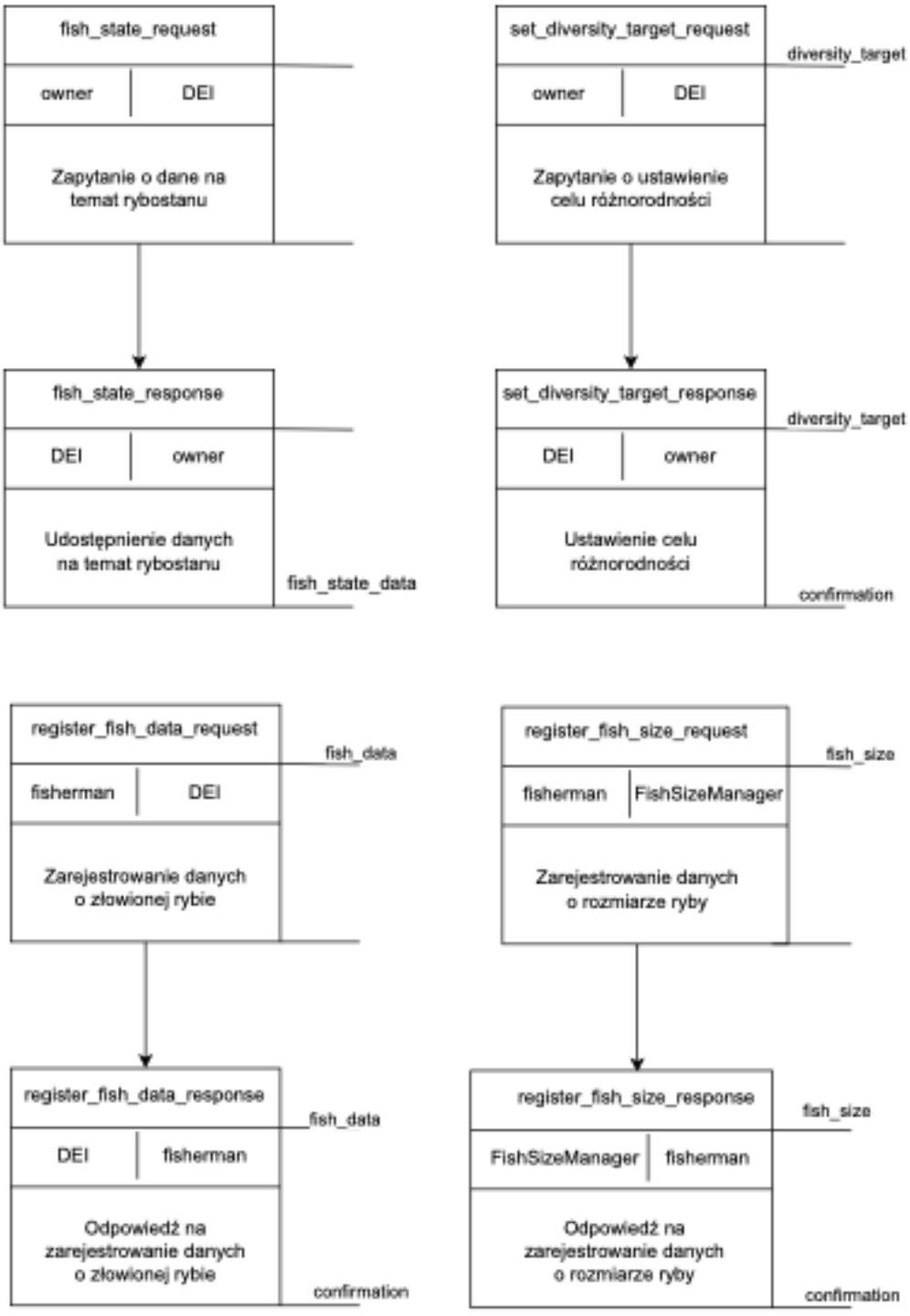
Poniższy diagram przedstawia schemat komunikacji agentów. Agent *Owner* komunikuje się ze wszystkimi agentami, ponadto agent *FishermanAgent* jednostronnie wysyła informacje do agenta *FishCaretakerAgent* (konkretnie jest to informacja o tym, jaką rybę agent zabiera ze sobą).

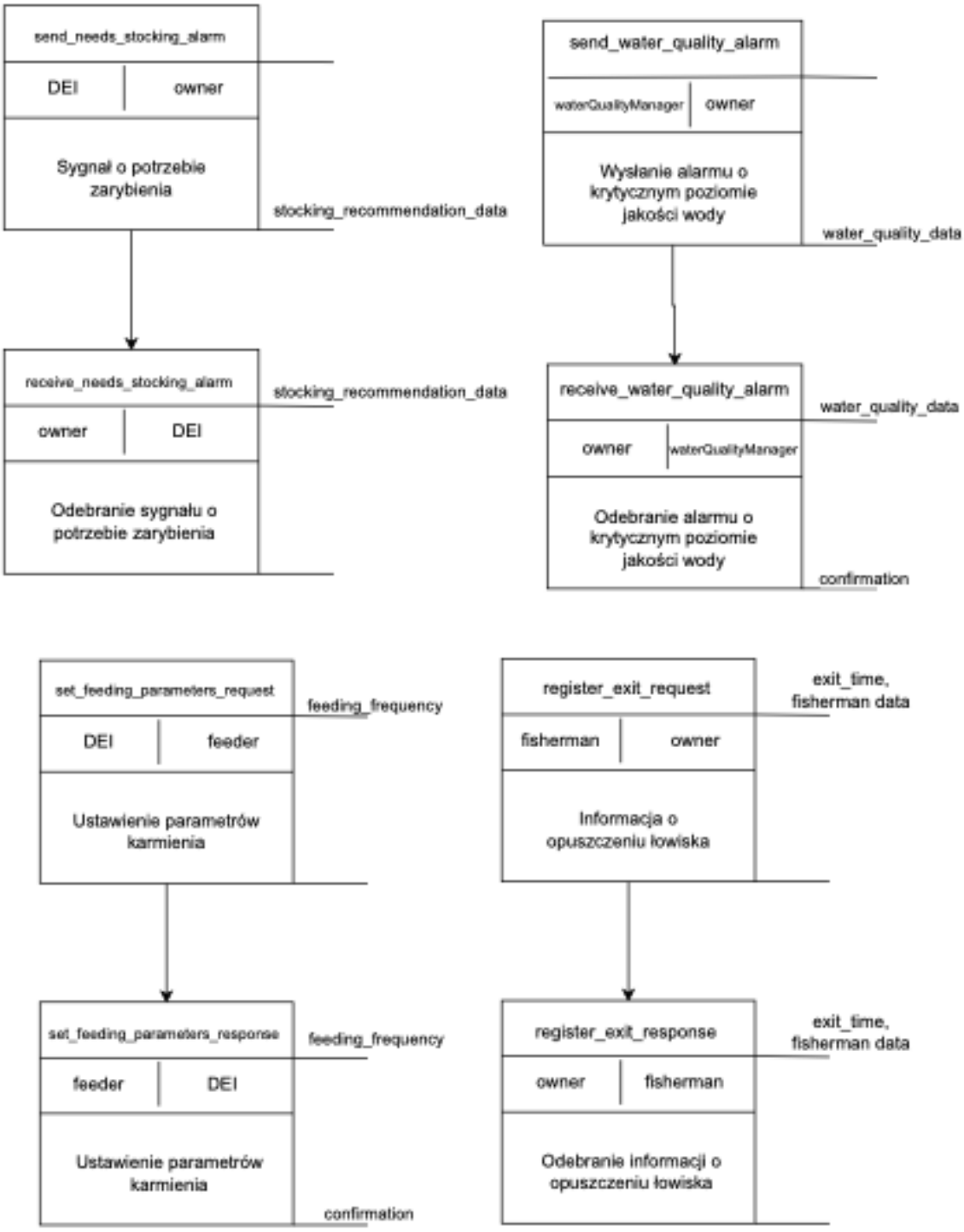


Model Interakcji

Poniższe diagramy przedstawia protokoły wstępujące w systemie, które tworzą interakcje pomiędzy rolami.




fish_state_request
owner
DEI
Zapytanie o dane na temat rybostanu
fish_state_response
DEI
owner
Udostępnienie danych na temat rybostanu
fish_state_data
set_diversity_target_request
diversity_target
owner
DEI
Zapytanie o ustawienie celu różnorodności
set_diversity_target_response
diversity_target
DEI
owner
Ustawienie celu różnorodności
confirmation
register_fish_data_request
fish_data
fisherman
DEI
Zarejestrowanie danych o złowionej rybie
register_fish_data_response
fish_data
DEI
fisherman
Odpowiedź na zarejestrowanie danych o złowionej rybie
confirmation
register_fish_size_request
fish_size
fisherman
FishSizeManager
Zarejestrowanie danych o rozmiarze ryby
register_fish_size_response
fish_size
FishSizeManager
fisherman
Odpowiedź na zarejestrowanie danych o rozmiarze ryby
confirmation


send_needs_stocking_alarm
DEI
owner
Sygnał o potrzebie zarybienia
stocking_recommendation_data
receive_needs_stocking_alarm
stocking_recommendation_data
owner
DEI
Odebranie sygnału o potrzebie zarybienia
send_water_quality_alarm
waterQualityManager
owner
Wysłanie alarmu o krytycznym poziomie jakości wody
water_quality_data
receive_water_quality_alarm
water_quality_data
owner
waterQualityManager
Odebranie alarmu o krytycznym poziomie jakości wody
confirmation
set_feeding_parameters_request
feeding_frequency
DEI
feeder
Ustawienie parametrów karmienia
set_feeding_parameters_response
feeding_frequency
feeder
DEI
Ustawienie parametrów karmienia
confirmation
register_exit_request
exit_time, fisherman data
fisherman
owner
Informacja o opuszczeniu łowiska
register_exit_response
exit_time, fisherman data
owner
fisherman
Odebranie informacji o opuszczeniu łowiska

Diagramy BPMN

Diagram konwersacji

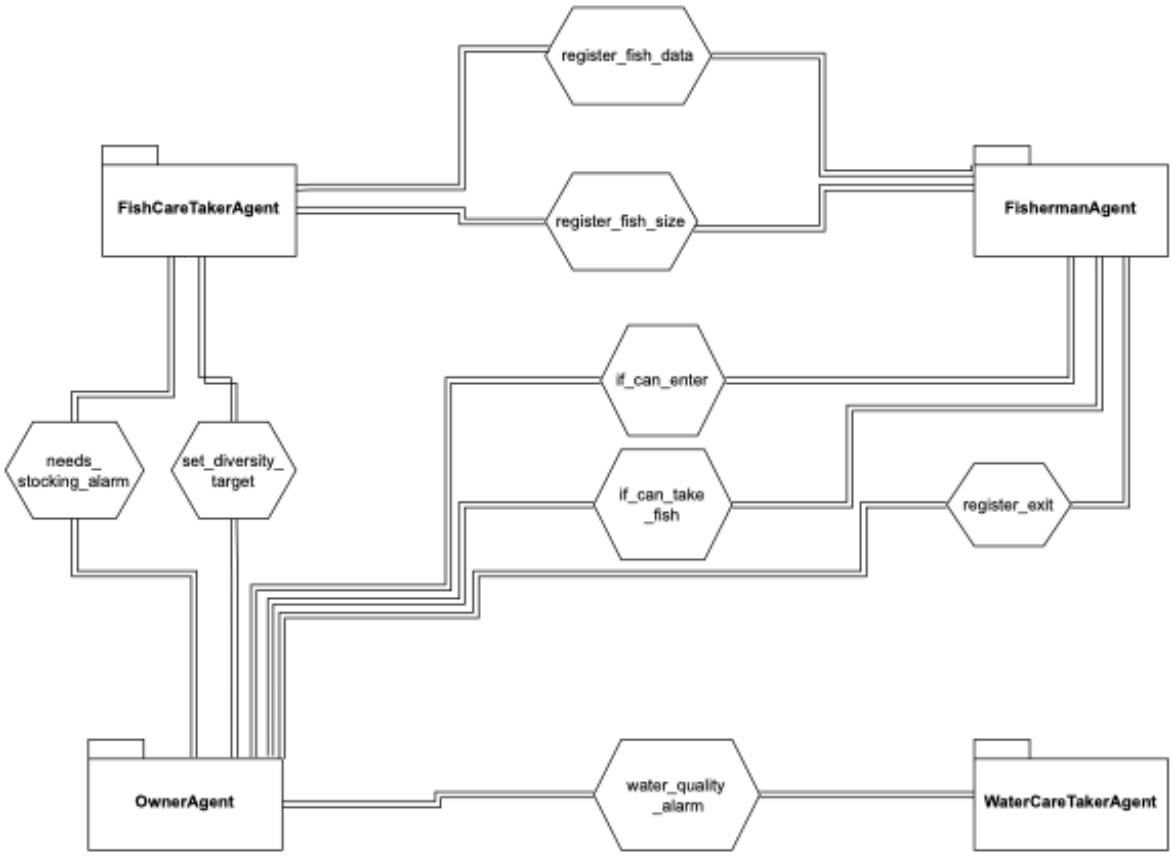
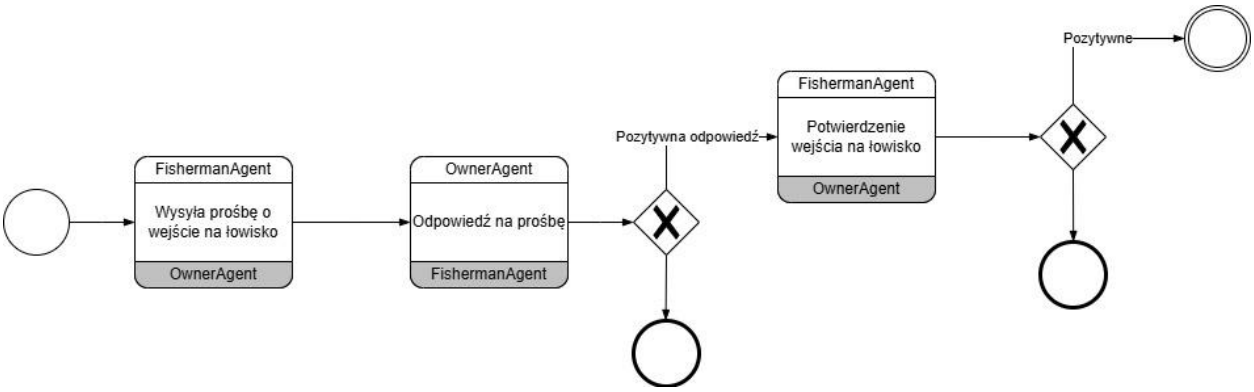
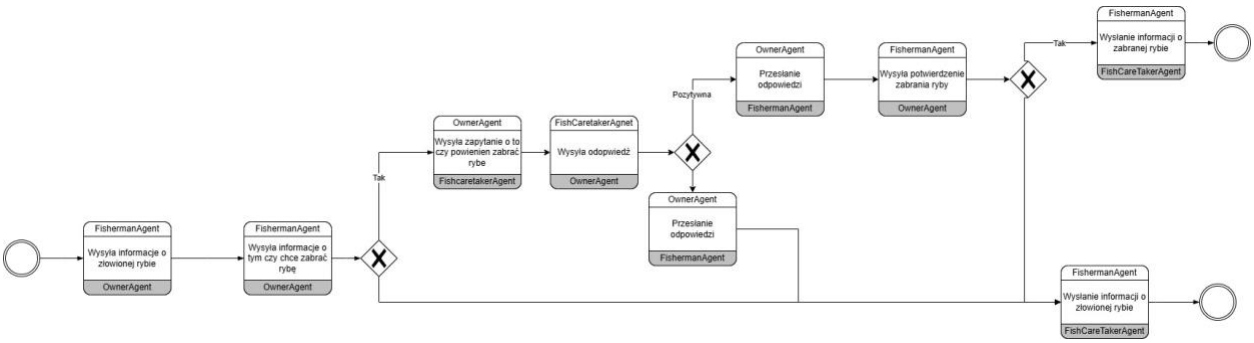


Diagram Choreografii

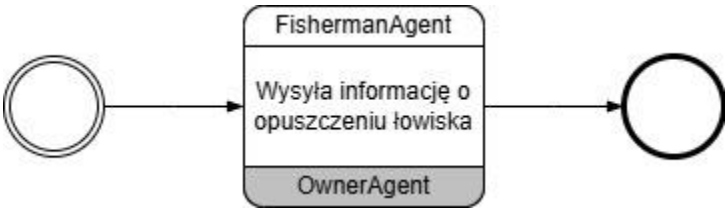
Wejście na łowisko



Złowienie/Zabranie ryby



Informowanie o opuszczeniu łowiska

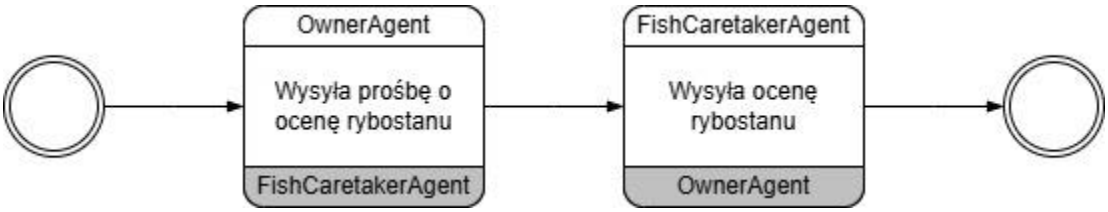


Zmiana ustawień różnorodności

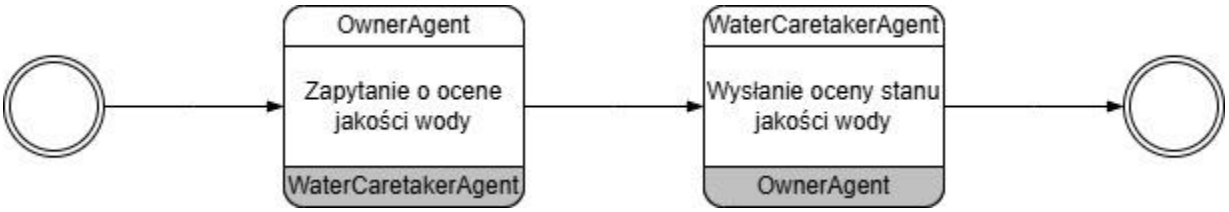
Informowanie o przeprowadzonym zarybieniu

Alert o krytycznym rybostanie

Ocena rybostanu



Ocena jakości wody



Alert o złym stanie wody

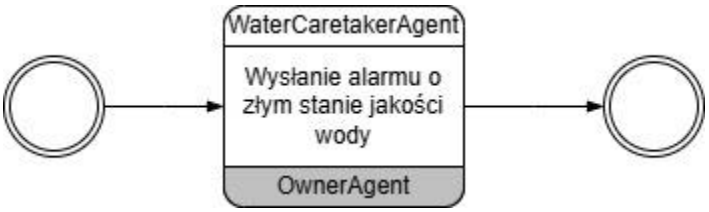
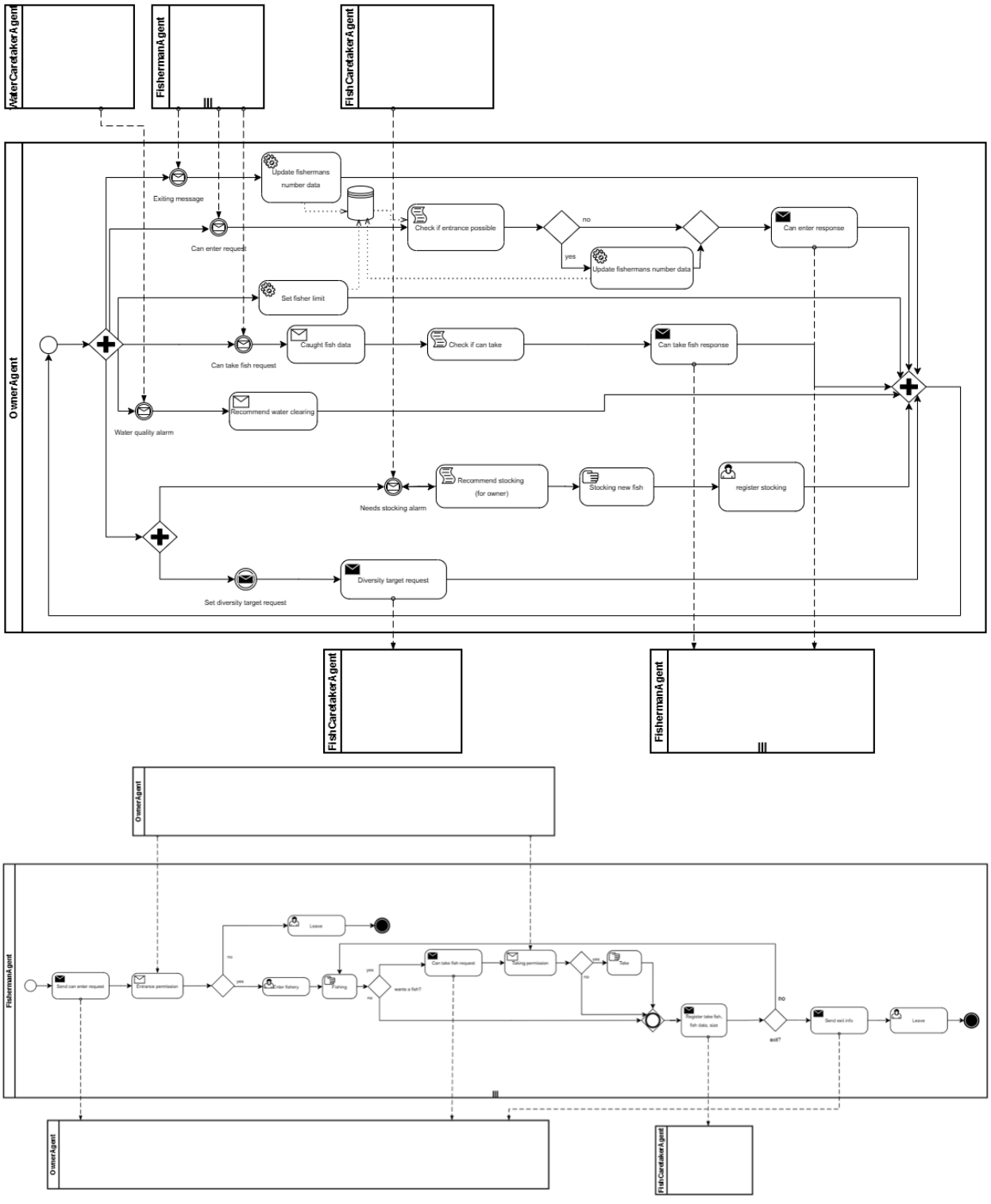
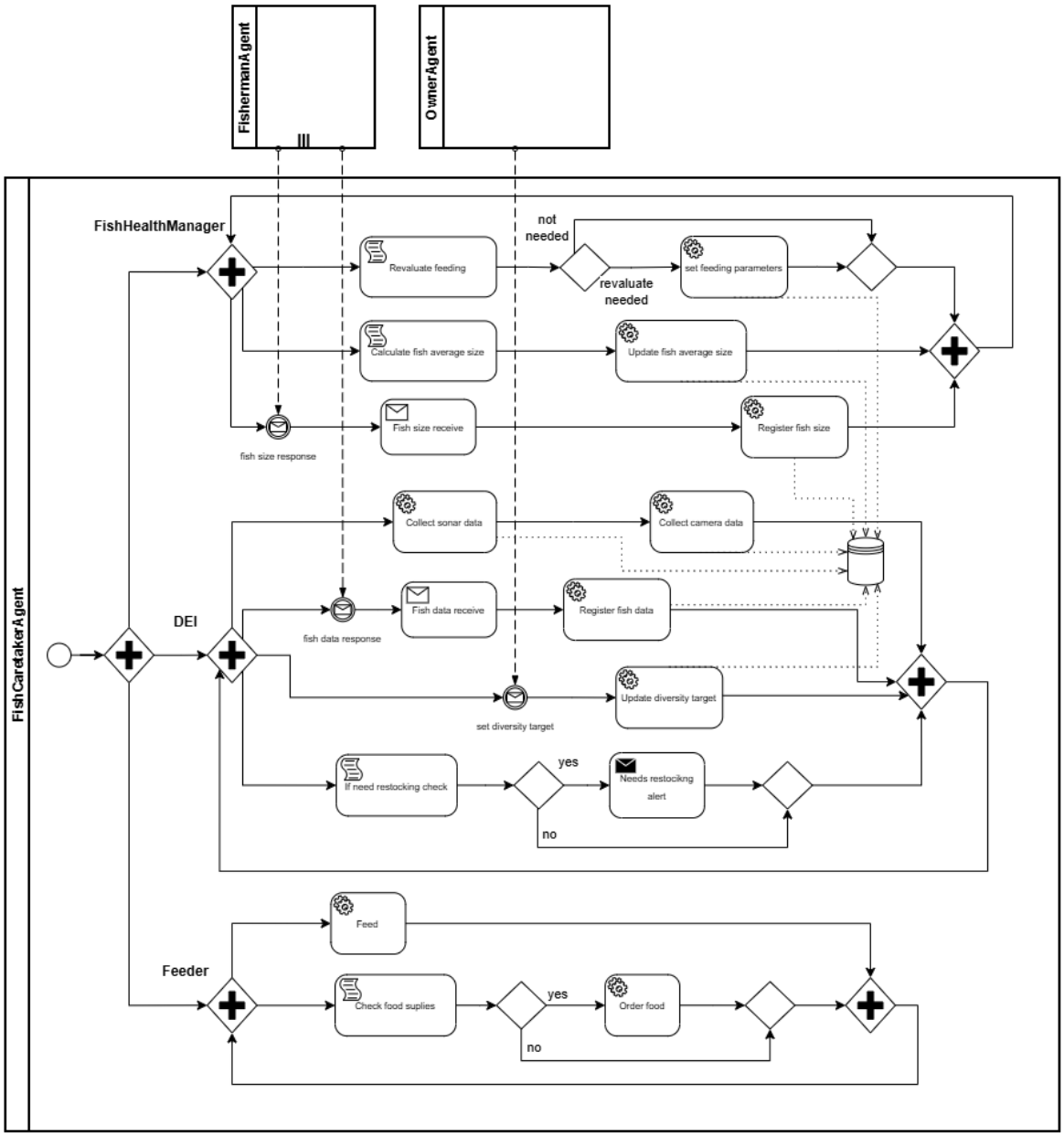
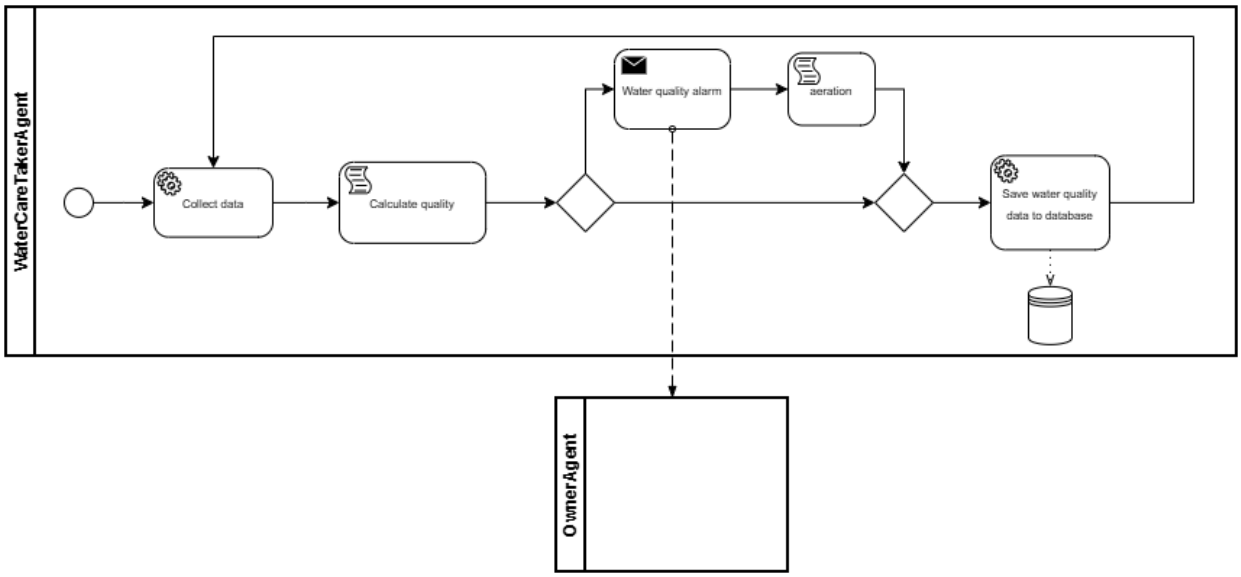


Diagram Kolaboracji


WaterCaretakerAgent
FishermanAgent
FishCaretakerAgent
OwnerAgent
Update fishermans number data
Exiting message
Can enter request
Check if entrance possible
no
yes
Update fishermans number data
Can enter response
Set fisher limit
Caught fish data
Can take fish request
Check if can take
Can take fish response
Recommend water clearing
Water quality alarm
Recommend stocking (for owner)
Needs stocking alarm
Stocking new fish
register stocking
Diversity target request
Set diversity target request
FishCaretakerAgent
FishermanAgent
OwnerAgent
FishermanAgent
Send can enter request
Entrance permission
no
yes
Enter fishery
Fishing
yes
wants a fish?
no
Can take fish request
Taking permission
yes
no
Take
Leave
Register take fish, fish data, size
no
exit?
Send exit info
Leave
OwnerAgent
FishCaretakerAgent


FishermanAgent
OwnerAgent
FishCaretakerAgent
FishHealthManager
not needed
Revaluate feeding
revaluate needed
set feeding parameters
Calculate fish average size
Update fish average size
Fish size receive
fish size response
Register fish size
Collect sonar data
Collect camera data
DEI
Fish data receive
fish data response
Register fish data
Update diversity target
set diversity target
If need restocking check
yes
Needs restocikng alert
no
Feeder
Feed
Check food suplies
yes
Order food
no



Mapowanie interakcji na akty komunikacyjne FIPA

Poniższa tabela zawiera interakcje, które zostały zdefiniowane w modelu interakcji. Interakcje składają się z par protokołów: request i response. Każdy z protokołów został zmapowany na performatywę FIPA-ACL. W przypadku protokołów, których zapytanie jest performatywą `inform`, odpowiedź `inform` będzie potwierdzeniem odebrania wiadomości.

Interakcja	**Peformatywa FIPA-ACL**	
	Request	**Response**
prośba o pozwolenie na wejście	Request if	inform/disconfirm
prośba o pozwolenie na zabranie ryby	Request if	inform/disconfirm
zapytanie o status rybostanu	query	inform
ustawienie celu różnorodności	request	agree
zapytanie o jakość wody	query	inform
alarm o krytycznym poziomie jakości wody	request	agree
zarejestrowanie złowionej ryby	request	agree
zarejestrowanie wielkości złowionej ryby	request	agree
ustawienie parametrów karmienia	request	agree
informacja o opuszczeniu łowiska	inform	inform
sygnał o potrzebie zarybienia	request	agree

Feedback

- proszę żeby kolejne raporty były inkrementalne, tzn. zawierały wszystkie poprzednie części

- wymagania funkcjonalne:
    - ~~FR-B2: czy powiadomienie zawiera szczegóły dot. tego, który parametr został przekroczony?~~
    - ~~FR-D2: czy blokada zabrania ryby będzie ogólna (czyli po przekroczenie liczby ryb spowoduje zablokowanie), czy też per wędkarz?~~
    - ~~FR-E3: nie do końca rozumiem o co tutaj chodzi, proszę doprecyzować~~
- Role:
    - ~~lista aktywności i protokołów powinna wynikać z opisu~~
    - ~~niespójność nazw krotek w pozwoleniach i obowiązkach bezpieczeństwa, np: Pozwolenie na zabranie ryby i take_fish_permission - do poprawy~~
    - ~~niezgodność nazw protokołów z tymi, które tworzą obowiązki żywotne, np: register_exit vs register_exit_request~~
    - ~~brak wyróżnienia (podkreślenia) dla aktywności - do poprawy~~
    - ~~występują aktywności/protokoły, które nie są wymienione w obowiązkach żywotnych, np: "if_needs_stocking_request", którego nie ma w obowiązkach żyw. - do poprawy~~
- ~~Model interakcji - w porządku, ale mieszacie nazwy po polsku (model interakcji) i po angielsku (model ról)~~
- Diagram choreografii - złowienie/zabranie ryby: nie rozumiem ostatniej bramki
- ~~Diagramy kolaboracji - powinny być zgodne z obowią zkami żywotnymi. Dużo błędów, np:~~
    - ~~nie rozumiem FishermanAgent, górny przebieg (zapytanie o pozwolenie na wejście na łowisko) - po wejściu na łowisko diagram kończy się eventem rozpoczynającym?~~

Pytania na konsultacje:

- Jak zapisać obowiązek register_stocking w Owner?

-

Raport C – Implementacja

Implementacji podlegać będzie część wymagań funkcjonalnych systemu. Skupimy się na analityce wielkości rybostanu, zaleceniach dla właściciela łowiska, automatyzowaniu karmienia i limitach obowiązujących wędkarzy.

Pominiemy zatem rolę wodnika, elementy różnorodności gatunkowej w roli DEI oraz rolę managera wielkości osobników.

TODO:

Zadanie	Osoba	Status