POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTROTECHNIKI I AUTOMATYKI

Imię i nazwisko autora rozprawy: Przemysław Spychalski
Dyscyplina naukowa: Automatyka i Robotyka

ROZPRAWA DOKTORSKA

Tytuł pracy w języku polskim: System wieloagentowy wspomagający projektowanie wybranych układów sterowania okrętowych podsystemów elektroenergetycznych

Tytuł pracy w języku angielskim: Multi-agent system for aided-design of selected control systems for ship power subsystems

Promotor
Dr hab. inż. Ryszard Arendt
Promotor pomocniczy
Dr inż. Andrzej Kopczyński

Gdańsk, rok 2019

SPIS TREŚCI

1. WSTĘP

Projektowanie układów sterowania statków wiąże się z doborem odpowiedniej struktury (topologii) oraz jej elementów składowych. Spełnienie wymagań statycznych przez elementy składowe nie gwarantuje jednak otrzymania optymalnych właściwości dynamicznych zaprojektowanego układu sterowania. Zadania projektowe są trudne do sformalizowania, dlatego do ich rozwiązywania wykorzystuje się systemy wspomagające projektowanie, coraz częściej bazujące na metodach sztucznej inteligencji. Trend ten zapoczątkowany został już w latach osiemdziesiątych XX wieku, jako odpowiedź na nowe możliwości związane z dynamicznym rozwojem technik komputerowych [1]. W tamtym okresie powstały pierwsze koncepcje inteligentnych systemów do komputerowego wspomagania projektowania. Od samego początku celem tej klasy systemów było częściowe zautomatyzowanie pewnych etapów procesów projektowych, takich jak wstępna analiza wykonalności, selekcja materiałów i komponentów, optymalizacja parametrów, planowanie produkcji oraz synteza gotowych produktów inżynierskich. Systemy te okazywały się bardzo przydatne, jednakże na ówczesnym etapie rozwoju technologicznego nie było możliwości wyeliminowania pewnych ich ograniczeń, wśród których można wskazać brak możliwości projektowania kreatywnego (systemy nie radziły sobie z tworzeniem nowych klas układów od podstaw, a jedynie bazowały na już istniejących rozwiązaniach). Kolejnym ograniczeniem było to, że systemy te skupiały się na bardzo wąskiej i specjalistycznej domenie (nie były wystarczająco generyczne), a także nie dawały możliwości integracji z już istniejącymi rozwiązaniami z dziedziny CAD (ang. Computer Aided Design) [1].

Wspomniane zagadnienia są problemami nadal aktualnymi. W ostatnich latach pojawiło się wiele publikacji opisujących coraz doskonalsze systemy do projektowania wspomaganego komputerowo. Jednym z takich przykładów jest zintegrowany system inteligentny do wspomagania projektowania mechanizmów przenoszenia napędu [2]. Kolejnym przykładem może być rozproszony system do kolaboracyjnego rozwijania produktów inżynierskich, umożliwiający równoczesne projektowanie oraz prowadzenie symulacji w jednym spójnym środowisku [3]. Następnym zaprezentowanym niedawno podejściem do zintegrowanego projektowania wspomaganego komputerowo jest system z bazą wiedzy do tworzenia prototypów elementów mechanicznych, takich jak przekładnie jednostopniowe [4]. Również w przemyśle widoczne jest silne ukierunkowanie prac na integrowanie metod sztucznej inteligencji z nowymi lub już istniejącymi środowiskami CAD. Firma SolidWorks dodała do swojego oprogramowania opcję, która przyspiesza proces wytwarzania zaprojektowanych w tym środowisku elementów poprzez automatyczne generowanie ścieżek narzędziowych dla sterowanych komputerowo maszyn produkcyjnych [5]. Firma ConcertalSystems złożyła patent, w którym zastrzegła system SDA (ang. System Design Automation) do automatyzacji projektowania systemów układów elektronicznych mieszanych sygnałów, w tym projekty oprogramowania wbudowanego [6]. Są to tylko wybrane przykłady tego typu systemów, ponieważ ilość badań zmierzających do implementacji środowisk wspomagających lub automatyzujących projektowanie jest ogromna.

Motywacją wybrania takiej tematyki rozprawy doktorskiej były obserwacje dynamicznego rozwoju dziedzin nauki i techniki zajmujących się zagadnieniami szeroko pojętej sztucznej inteligencji. Szczególnie w ostatnich latach pojawienie się wysokowydajnych jednostek obliczeniowych, takich jak GPU (ang. Graphics Processing Unit) oraz procesorów wielordzeniowych sprawiło, że używanie nawet

najbardziej skomplikowanych metod inteligencji obliczeniowej, działających w czasie rzeczywistym (przykładowo Deep Leaning) stało się faktem [7]. Coraz częściej dyskutowane są możliwości zastosowania algorytmów sztucznej inteligencji do wyręczania ludzi z wykonywania prac powtarzalnych i monotonnych [8]. Niemal każdy proces projektowy posiada pewne wspólne, podobne do siebie etapy, które można wspomagać lub automatyzować przez zastosowanie elementów sztucznej inteligencji, takich jak systemy ekspertowe, systemy wieloagentowe oraz algorytmy uczenia maszynowego [5]. Wdrożenie systemów wspomagających projektowanie przyczynia się w dużej mierze do skrócenia czasu otrzymania gotowych rozwiązań inżynierskich, ustandaryzowania procesów projektowych oraz osiągnięcia wysokiej efektywności ekonomicznej [8]. Czynniki te są szczególnie istotne w przemyśle stoczniowym podczas przygotowywania projektów ofertowych, dlatego zdecydowano o utworzeniu dedykowanego systemu właśnie dla tej gałęzi gospodarki.

Przedmiotem rozprawy doktorskiej jest implementacja rozproszonego systemu wieloagentowego wspomagającego projektowanie wybranych układów sterowania okrętowych podsystemów elektroenergetycznych. Utworzony system składać się będzie z autonomicznych agentów programowych, które poprzez realizację unikalnych algorytmów oraz wzajemną komunikację wspomagać będą rozwiązywanie zadań projektowych w zakresie doboru odpowiedniej struktury i elementów układów sterowania, jak również generować będą raport z analizy utworzonego projektu. W zaproponowanym systemie wieloagentowym proces projektowania rozpoczynać się będzie pobraniem od użytkownika wytycznych projektowych dotyczących m.in. informacji o okręcie oraz jego instalacji elektroenergetycznej, wybranej strukturze sterowania, ograniczeniach projektowych, parametrach symulacji itd. Kolejny etap polegać będzie na wczytaniu modeli matematycznych elementów układu, utworzeniu z nich gotowej struktury systemu sterowania oraz przeprowadzeniu badań symulacyjnych. Na podstawie analizy trajektorii sterowania i charakterystyki elementów modelu, jak również skonfrontowaniu tych danych z wytycznymi towarzystw klasyfikacyjnych dokonywana będzie ocena jakości otrzymanego rozwiązania projektowego (jako kryterium jakościowe przyjęto zgodność dynamiki zamodelowanego układu sterowania z wymaganiami towarzystw klasyfikacyjnych). Efektem działania systemu wieloagentowego będzie wygenerowanie raportu z analizy układu sterowania w postaci gotowego projektu ofertowego, odpowiadającego rzeczywistym projektom realizowanym przez stoczniowe biura projektowe (raport, podobnie jak pozostałe elementy systemu, zrealizowano w języku angielskim z uwagi na możliwość jego potencjalnego wdrożenia w przemyśle międzynarodowym).

Rozdział 1 zawiera zdefiniowaną tezę oraz cele rozprawy doktorskiej. Rozdział 2 zawiera przegląd literatury z zakresu systemów wspomagających projektowanie, takich jak systemy ekspertowe oraz przykłady zastosowania systemów wieloagentowych do tego typu zagadnień. Rozdział 3 zawiera koncepcję architektury systemu wieloagentowego oraz opis platformy UMAP. Rozdział 4 zawiera charakterystykę poszczególnych typów agentów zaimplementowanych w systemie. Rozdział 5 zawiera szczegóły dotyczące algorytmów szeregowania procesów i uczenia maszynowego zastosowanych w systemie wieloagentowym oraz porównanie ich z innymi rozwiązaniami. Rozdział 6 zawiera deskrypcje przebiegu procesu projektowania przykładowej struktury układu sterowania podsystemu elektroenergetycznego w celu zademonstrowania działania systemu na konkretnym przykładzie.

Rozdział 7 zawiera wyniki testów wydajnościowych systemu wieloagentowego. Rozdział 8 zawiera podsumowanie przeprowadzonych badań.

1.1. Teza rozprawy

Implementacja rozproszonego systemu wieloagentowego umożliwi zdecentralizowane oraz współbieżne prowadzenie procesów projektowych przy doborze odpowiedniej struktury i elementów składowych wybranych układów sterowania okrętowych podsystemów elektroenergetycznych, jak również pozwoli na ocenę zgodności dynamiki otrzymanych rozwiązań projektowych z wymaganiami towarzystw klasyfikacyjnych.

1.2. Cel główny rozprawy

Celem rozprawy doktorskiej jest implementacja systemu wieloagentowego wspomagającego projektowanie wybranych układów sterowania okrętowych podsystemów elektroenergetycznych. Działanie zaproponowanego systemu zademonstrowane zostanie na przykładzie procesu projektowania struktury układu sterowania podsystemu elektroenergetycznego z autotransformatorowym rozruchem silnika indukcyjnego. Przedstawione zostaną efekty działania systemu wieloagentowego, takie jak wygenerowane elementy projektu ofertowego (utworzonego na podstawie reguł zawartych w bazie wiedzy) oraz charakterystyki dynamiczne utworzonego modelu podsystemu elektroenergetycznego ze sterowaniem silnikiem indukcyjnym poprzez autotransformator.

1.3. Zakres dodatkowy rozprawy

Zakres dodatkowy rozprawy doktorskiej obejmuje zaimplementowanie dwóch algorytmów (algorytmu uczenia maszynowego i mechanizmu szeregowania procesów), mających na celu zwiększenie wydajności systemu wieloagentowego podczas wspomagania zadań projektowych. Poprawność implementacji obu mechanizmów zweryfikowana zostanie poprzez porównanie wydajności systemu wieloagentowego, podczas gdy algorytmy są aktywne oraz nieaktywne. Dokonane zostanie również porównanie wydajności zaimplementowanego systemu wieloagentowego ze scentralizowanym systemem ekspertowym (przykład z literatury) podczas wspomagania projektowania struktury układu sterowania podsystemu elektroenergetycznego ze sterowaniem silnikiem indukcyjnym poprzez autotransformator (jako kryterium przyjmuje się czas potrzebny do wykonania obliczeń przez oba systemy).

2. PRZEGLĄD SYSTEMÓW WSPOMAGAJĄCYCH PROJEKTOWANIE

2.1. Wprowadzenie

Wspomaganie projektowania polega na zastosowaniu oprogramowania komputerowego do zagadnień związanych z tworzeniem projektów technicznych. W zakres komputerowego wspomagania projektowania wchodzą między innymi następujące aspekty: cyfrowe odwzorowywanie konstrukcji, symulacje, wizualizacje, prototypowanie, operowanie bazami danych elementów, przygotowywanie prezentacji ofertowych. Komputerowe wspomaganie projektowania bardzo często wykorzystywane jest do optymalizacji i analizy układów, procesów i konstrukcji, a także do obliczeń wytrzymałościowych FEM (ang. Finite Element Method) lub chociażby do przeprowadzenia inżynierii odwrotnej (ang. Reverse Engineering). Stosuje się przy tym wspomniane metody sztucznej inteligencji, takie jak sztuczne sieci neuronowe oraz systemy z bazą wiedzy wspomagające podejmowanie decyzji oraz arkusze kalkulacyjne i edytory tekstowe do projektowania technicznego. Przykładowe elementy procesu projektowania wspomaganego komputerowo zawierają [9]:

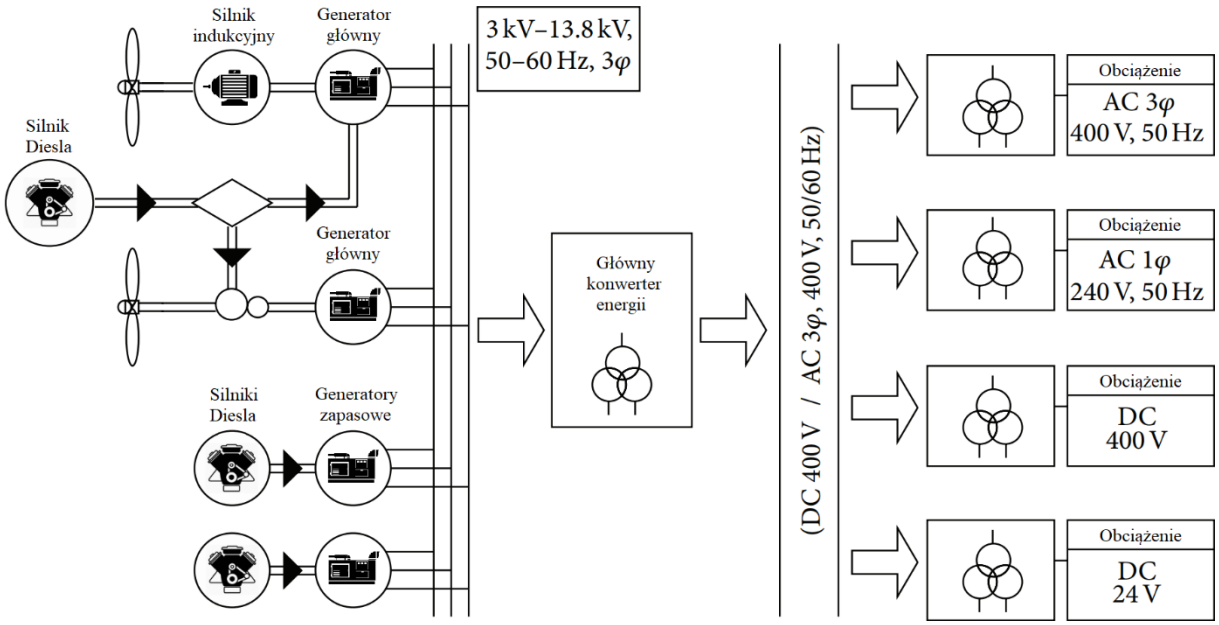
- zebranie wymagań dotyczących projektowanego produktu,
- utworzenie cyfrowego modelu (prototypu) wyrobu inżynierskiego,
- przeprowadzenie obliczeń (symulacji, analizy wytrzymałościowej),
- sprawdzenie poprawności funkcjonalnej na podstawie zebranych danych,
- naniesienie ewentualnych poprawek w projekcie i powtórzenie obliczeń,
- opracowanie rachunku kosztów i dokumentacji projektu wymaganej przepisami.

Komputerowe wspomaganie projektowania stosowane jest w wielu gałęziach przemysłu, jednakże w rozprawie doktorskiej skupiono się na wykorzystaniu tego typu systemów do projektowania układów sterowania podsystemów elektroenergetycznych statków. Układy te są zbiorami powiązanych ze sobą urządzeń, mających na celu wytwarzanie, przekształcanie i rozdział energii elektrycznej na statku. Okrętowe układy elektroenergetyczne składają się z dwóch głównych bloków, czyli części mechanicznej (w skład której wchodzą m.in. silniki wysokoprężne lub turbiny gazowe) oraz części elektrycznej (do której zalicza się m.in. generatory napięcia, prądnice synchroniczne, regulatory napięcia, sieć elektroenergetyczną oraz odbiorniki energii) [10].

Silnik wysokoprężny jest podstawową jednostką napędową, która może być używana do mechanicznego napędzania statku i generatora lub tylko generatora. W pierwszym przypadku silnik Diesla wprawia w ruch wał, który z kolei napędza generator (poprzez przekładnię) oraz śrubę napędową. W drugim wypadku silnik wysokoprężny napędza tylko generator, który z kolei wytwarza energię elektryczną zasilającą silnik indukcyjny wprawiający w ruch śrubę napędową. Druga metoda ma na celu zminimalizowanie stosowania mechanicznych przekładni i znana jest w literaturze jako IFEP (ang. Integrated Full Electric Propulsion) [11].

Oprócz generatora głównego na statku powinny znajdować się dodatkowe generatory, które dostarczą energię do okrętowej instalacji elektroenergetycznej w przypadku awarii zespołu prądotwórczego lub podczas zwiększonego zapotrzebowania na energię. W literaturze światowej pojawia się wymaganie na dwa dodatkowe generatory pomocnicze [12]. Polskie przepisy nakazują

natomiast zastosowanie przynajmniej jednego generatora dodatkowego, który umożliwi podstawowe zasilanie najważniejszych urządzeń na statku w razie awarii i jednocześnie zapewni minimalne warunki socjalno-bytowe dla załogi oraz pozwoli na ponowny rozruch głównego układu napędowego [11, 13]. Diagram podstawowego podsystemu elektroenergetycznego statku przedstawiono na rysunku Rys. 1.



Rys. 1. Diagram podstawowego podsystemu elektroenergetycznego statku [11]

Napięcie wyjściowe generatorów różni się w zależności od producenta i zawiera się w przedziale 3-13,8 kV przy częstotliwości 50-60 Hz. Energia wytworzona przez generatory dostarczana jest do głównej magistrali zasilającej, przez którą przekazywana jest dalej do odbiorników (obciążenia), wykorzystując przy tym przetworniki energii oraz transformatory. Magistrala może być zasilana prądem stałym lub przemiennym, przy czym obecnie bardziej popularne w okrętownictwie towarowym oraz pasażerskim jest zasilanie prądem przemiennym. Ostatnim elementem podsystemu elektroenergetycznego statku jest obciążenie (odbiorniki energii). Standardowo są to urządzenia zasilane jednofazowym napięciem przemiennym 230 V, 50 Hz oraz trójfazowym 400 V, 50 Hz np. trójfazowe silniki indukcyjne [11, 14].

Parametry instalacji elektrycznej statku oraz ich dopuszczalne zakresy określone są przez normy towarzystw klasyfikacyjnych w celu zapewnienia należytego bezpieczeństwa eksploatacji. Do utrzymania pożądanych parametrów stosowane są układy sterowania okrętowych podsystemów elektroenergetycznych. Nowoczesne systemy tego typu składają się z elementów takich jak: układ adaptacji mocy generatora, układ zabezpieczeń siłowni okrętowej, system zarządzania energią, system sterowania przetwornicą częstotliwości oraz automatyczny regulator napięcia [15, 16].

Układ adaptacji mocy generatora GPA (ang. Generator Power Adaptation) dynamicznie dostosowuje moc systemów napędowych do rzeczywistej mocy czynnej oraz biernej generatora. Jego zadaniem jest zapobieganie nagłym i niedopuszczalnym przeciążeniom zespołów prądotwórczych, jak również uniknięcie dzielenia obciążenia czynnego i biernego pomiędzy generatorami. Układ adaptacji mocy generatora to podstawowe zabezpieczenie przeciw przeciążeniom generatora i nagłym zanikom napięcia w siłowni okrętowej [15, 16].

Układ zabezpieczeń siłowni okrętowej zawiera wszystkie układy zabezpieczeniowe, które przewidziane są dla standardowych systemów zasilania, takie jak zabezpieczenia różnicowe, nadprądowe/nadnapięciowe oraz przed przepływem zwrotnym. Ponadto układ ten monitoruje podsystem elektroenergetyczny pod kątem wystąpienia np. nagłego, nierównomiernego dzielenia obciążenia biernego lub czynnego (krótkotrwałego oraz długotrwałego) mogącego wystąpić podczas awarii automatycznego regulatora napięcia generatora [15, 16].

System zarządzania energią pozwala na awaryjne odłączenie generatorów w przypadku wykrycia przeciążenia lub awarii. Układ ten pozwala również na załączenie generatora pomocniczego w celu przywrócenia nominalnych parametrów elektrowni okrętowej. System ten poprzez sterowanie przetwornicą częstotliwości zapewnia stabilne zasilanie oraz uniknięcie strat mocy i awarii silników Diesla w normalnych warunkach eksploatacyjnych [15, 16].

Automatyczny regulator napięcia AVR (ang. Automatic Voltage Regulator) steruje wzbudzeniem generatorów okrętowych. Sterownik AVR posiada w uproszczeniu charakterystykę podobną do regulatora PID, ze stacjonarnym ograniczeniem współczynnika całkującego, który wymusza spadek napięcia wprost proporcjonalny do obciążenia generatora. Spadek napięcia zapewnia równomierny rozkład mocy biernej w równoległym połączeniu generatorów i przyczynia się do spełnienia wymagań dotyczących odchyleń wartości prądu w sieci elektroenergetycznej [15, 16]. Wytyczne Polskiego Rejestru Statków określające dopuszczalne spadki napięć przedstawiono na rysunku Rys. 2.

Dopuszczalne odchylenia dla systemów zasilanych prądem przemiennym			
Parametry	Dopuszczalne odchylenia od wartości znamionowych		
	Długotrwałe	Krótkotrwałe	
		Wartość	Czas
Napięcie	+6%, –10%	±20%	1,5 sek
Częstotliwość	±5%	±10%	5 sek

Rys. 2. Wytyczne Polskiego Rejestru Statków określające dopuszczalne spadki napięć [13]

Z uwagi na dużą liczbę wymagań towarzystw klasyfikacyjnych stosowane są systemy z bazą wiedzy, które automatyzują proces projektowania układów sterowania okrętowych podsystemów elektroenergetycznych i weryfikują, czy pozwalają one spełnić przyjęte wymagania. Przykładem takich systemów z bazą wiedzy są:

- systemy ekspertowe,
- systemy wieloagentowe.

2.2. Systemy ekspertowe

System ekspertowy (ang. Expert System) jest to z reguły scentralizowany system komputerowy przeznaczony do rozwiązywania specjalistycznych zadań poprzez naśladowanie procesu wnioskowania odpowiedniego dla człowieka, eksperta w danej dziedzinie. Systemy ekspertowe zazwyczaj składają się z dwóch podstawowych elementów, czyli bazy wiedzy oraz mechanizmu wnioskowania. Baza wiedzy zawiera zbiór informacji opisujących dany problem, natomiast mechanizm wnioskowania poprzez aplikowanie reguł na bazie wiedzy znajduje rozwiązanie rozpatrywanego problemu. Implementację tego typu systemów od podstaw ułatwiają szkieletowe systemy ekspertowe, będące środowiskiem programistycznym z pustą bazą wiedzy. Inżynierowie opracowujący nowy system ekspertowy gromadzą wiedzę z określonej problematyki, reprezentują ją w odpowiedniej postaci oraz wprowadzają do bazy wiedzy systemu szkieletowego. Częstym sposobem reprezentacji wiedzy są proste reguły w postaci operacji logicznych IF..THEN..ELSE lub nieco bardziej rozbudowane struktury drzew decyzyjnych. Jeśli chodzi o metody wnioskowania, to używane jest wnioskowanie w tył (regresywne od tezy do aksjomatów), w przód (progresywne od aksjomatów do nowych reguł) oraz metoda mieszana. Interakcja użytkownika z systemem ekspertowym odbywa się najczęściej poprzez zadawanie pytań i kończy się wypracowaniem konkluzji przez system [17, 18].

Zastosowania systemów ekspertowych do zagadnień wspomagania projektowania statków są często spotykane w literaturze. Przykładem może być system ekspertowy do badania konceptów projektowych dla statków kontenerowych [19]. Kolejnym przykładem jest system ekspertowy ALDES, który został opracowany przy użyciu podejścia prototypowego i hierarchicznej dekompozycji procesu projektowego [20]. Aspekty zastosowania systemów z bazą wiedzy do wspomagania projektowania promów pasażerskich, ze szczególnym uwzględnieniem podziału na alternatywne przepisy probabilistyczne w celu zapewnienia ich bezpieczeństwa również są spotykane w literaturze [21]. Kompleksowy przegląd literatury związany z systemami ekspertowymi wspomagającymi projektowanie podsystemów statków został przedstawiony w [22]. Badania prowadzone w polskich jednostkach naukowych również przyczyniły się do rozwoju tego typu systemów. Przykładem jest system ekspertowy wspomagający projektowanie statków (układ sterów strumieniowych wraz z podsystemem elektroenergetycznym), który zaimplementowany został w środowisku Exsys i składa się m.in. z interfejsu użytkownika, bazy danych elementów składowych oraz aplikacji symulacyjnej [23-27]. Oprócz szeregu zalet systemy ekspertowe posiadają również pewne ograniczenia. Ocena krytyczna pozwala wyróżnić następujące wady systemów tego typu:

- systemy ekspertowe posiadają zazwyczaj scentralizowaną strukturę,
- centralizacja algorytmów prowadzi do nadmiernego obciążenia jednostki,
- z systemu ekspertowego może korzystać naraz tylko jeden użytkownik,
- aktualizacja bazy wiedzy okresowo blokuje możliwość pracy z systemem,
- sekwencyjne działanie, które uniemożliwia zrównoleglenie obliczeń,
- poszczególne procesy projektowe nie są realizowane współbieżnie,
- brak modułowości prowadzi do trudności w lokalizacji błędów.

Przedstawione ograniczenia scentralizowanych systemów ekspertowych mogą zostać rozwiązane poprzez zastosowanie architektury wieloagentowej.

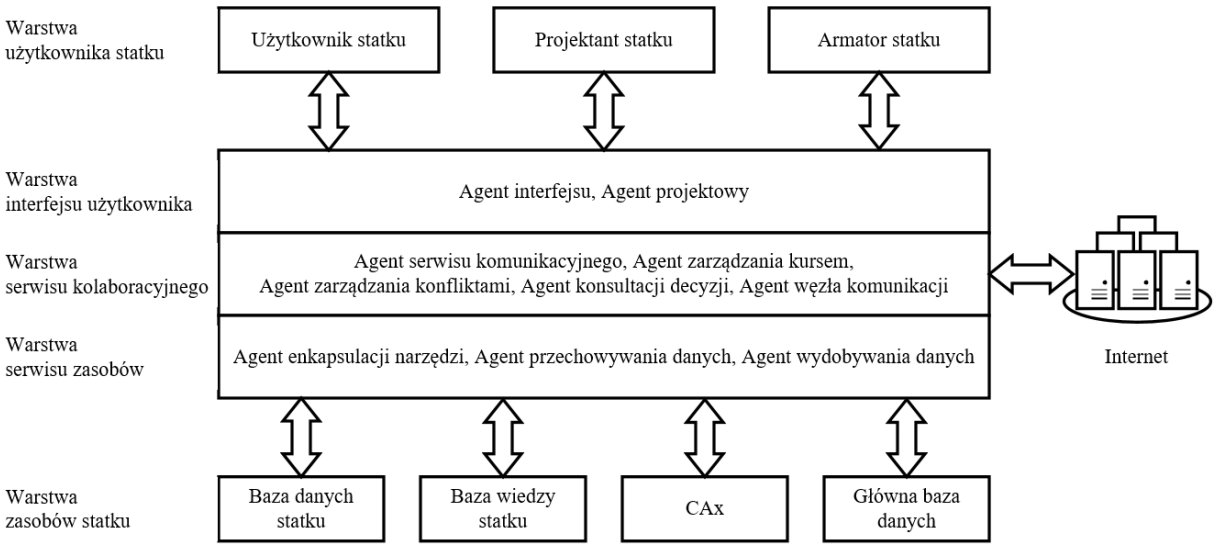
2.3. Systemy wieloagentowe

System wieloagentowy (ang. Multi-Agent System) jest to rozproszony system komputerowy złożony z co najmniej kilku współpracujących ze sobą jednostek (agentów) realizujących wspólny, globalny cel. Koncepcja oraz pierwsze implementacje systemów wieloagentowych pojawiły się już pod koniec lat osiemdziesiątych XX wieku, jednak podejście to jest nadal aktualne i rozwijane w środowiskach akademickich, jak również w zastosowaniach komercyjnych. Przykładem może być oprogramowanie antywirusowe bazujące na architekturze agentowej, aplikacje wspomagające kontrolę ruchu lotniczego oraz zarządzanie przedsiębiorstwem [28, 29]. Systemy wieloagentowe stosowane są wszędzie tam, gdzie zachodzi potrzeba rozwiązania problemu o naturze rozproszonej lub złożonej obliczeniowo. Programowanie agentowe cechuje poziom abstrakcji wyższy, niż np. w programowaniu obiektowym i strukturalnym. Przewagą takiego podejścia w stosunku do innych paradygmatów programowania jest zdecentralizowane przeprowadzanie obliczeń, dzięki czemu możliwe jest równoległe wykonywanie poszczególnych części algorytmów (również na odrębnych jednostkach sprzętowych) i znacznie szybsze opracowanie rozwiązania przez system [28-30].

Kolejnym aspektem przemawiającym na korzyść systemów wieloagentowych jest odporność na zakłócenia oraz błędy. Obliczenia wykonywane są zazwyczaj redundantnie, po czym dokonywana jest ocena jakości otrzymanych rozwiązań. Bazując na określonym kryterium, wybrany zostaje najlepszy z dostępnych rezultatów. Pozwala to uniknąć błędów, które mogą wystąpić w programowaniu obiektowym, przez to że każdy obiekt zakłada poprawność danych otrzymanych od innego obiektu. Dane w systemie wieloagentowym przesyłane są w sposób asynchroniczny i nadmiarowy. Uszkodzenie pewnego fragmentu nie skutkuje zanikiem funkcjonalności systemu jako całości, co jest bardzo istotne np. w środowisku sieciowym [28, 29]. Podstawową jednostką w systemie jest agent (ang. Software Agent). Nie istnieje spójna, jednoznaczna definicja agenta, jednak można stwierdzić, iż jest to autonomiczny program komputerowy, posiadający zdolność wykonywania określonego zakresu działań oraz komunikowania się z innymi agentami, w celu wspólnego rozwiązania danego problemu. Najczęściej pojedynczy agent nie posiada dostatecznej wiedzy oraz umiejętności do samodzielnego rozwiązania postawionego zadania. Dodatkowymi cechami agentów, często wymienianymi w literaturze są [28-30]:

- zdolność uczenia się,
- zdolność nawiązywania komunikacji,
- zdolność adaptacji do nowych warunków,
- zdolność reagowania w czasie rzeczywistym,
- zdolność posługiwania się językiem naturalnym.

Podobnie jak w przypadku systemów ekspertowych, podjęte zostały próby zastosowania systemów wieloagentowych do zagadnień wspomagania projektowania podsystemów statków. Zwięzły opis takiego systemu przedstawiono w [31], jednak autorzy w artykule skupili się bardziej na opracowaniu metody rozwiązywania konfliktów w procesie projektowania, bazując na wnioskowaniu opartym na przypadkach (ang. Case-based reasoning). Kolejny przykład systemu tego typu został wspomniany w [32], jednak i tym razem autorzy bardziej niż na opisie systemu wieloagentowego skoncentrowali się na aspekcie dystrybucji funkcji oceniającej na wiele komputerów w celu poprawy wydajności algorytmu optymalizacji. Podobnie w [33] autorzy zaimplementowali system do projektowania statków, jednak publikacja skupia się na zastosowaniu w tym systemie metodologii wieloagentowej, hybrydowej, koewolucyjnej optymalizacji rojem cząstek (HCPSO) do zagadnień wspomagania projektowania. W pełni udokumentowany system do kolaboracyjnego projektowania statków oparty na architekturze wieloagentowej i ontologii przedstawiono w [34]. System ten składa się z wielu warstw, w których ontologia jest używana do reprezentacji wiedzy o projektowaniu statków, natomiast język OWL używany jest do opisu tej wiedzy. Schemat systemu wieloagentowego do kolaboracyjnego projektowania statków przedstawiono na rysunku Rys. 3.



Rys. 3. Schemat systemu wieloagentowego do kolaboracyjnego projektowania statków (przykład z literatury) [34]

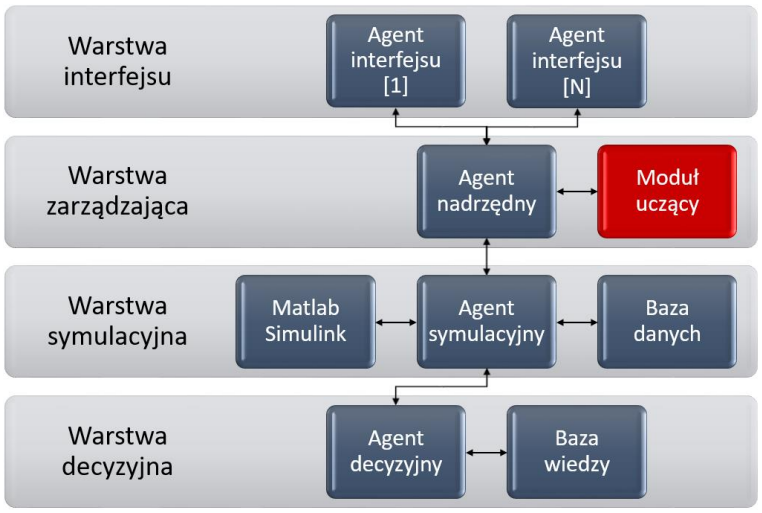
2.4. Podsumowanie

Przedstawiony system wieloagentowy służy jedynie jako narzędzie, które ułatwia wymianę informacji i dzielenie się wiedzą przez inżynierów projektantów oraz umożliwia integrację wielu narzędzi i środowisk CAD w jednym, spójnym systemie. W literaturze brakuje natomiast przykładów systemów wieloagentowych bezpośrednio wspomagających projektowanie, w tym sensie, że ich działanie kończy się wygenerowaniem gotowego projektu technicznego lub ofertowego (chodzi tutaj o kompleksowy projekt, który może zostać przedstawiony np. inwestorowi). W ramach rozprawy doktorskiej (część projektowa) postanowiono taki system utworzyć, bazując częściowo na wynikach badań systemów ekspertowych przedstawionych w [23-27]. Rozdział 3 zawiera koncepcję architektury zaimplementowanego systemu wieloagentowego.

3. KONCEPCJA ARCHITEKTURY SYSTEMU WIELOAGENTOWEGO

3.1. Wprowadzenie

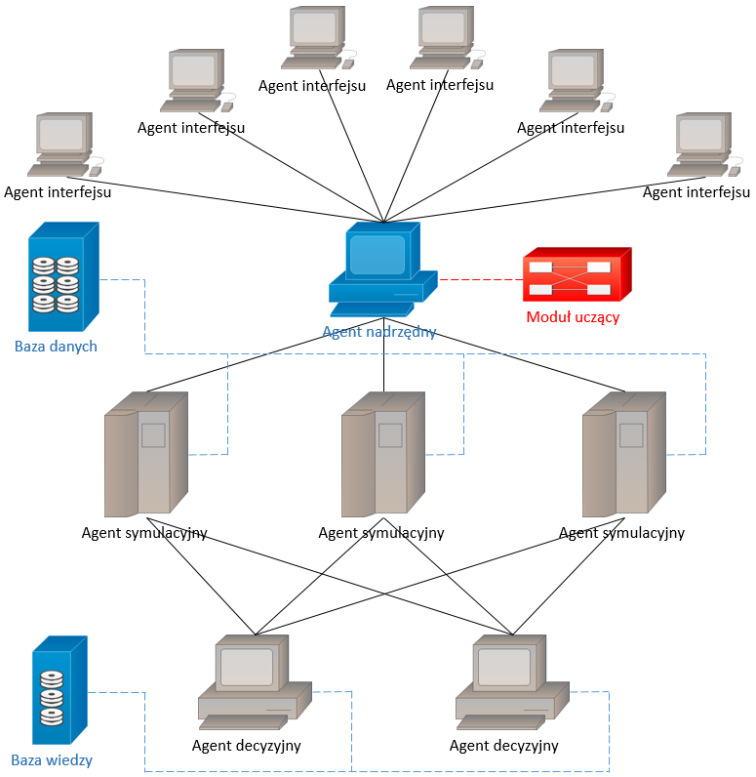
Zaimplementowano system wieloagentowy wspomagający projektowanie wybranych układów sterowania okrętowych podsystemów elektroenergetycznych, który składa się z agenta interfejsu, agenta symulacyjnego, agenta decyzyjnego/ekspertowego, agenta nadrzędnego (Master), jak również bazy danych, bazy wiedzy oraz modułu uczącego. Dodatkowo agent symulacyjny i agent interfejsu korzystają podczas swojego działania z zewnętrznych aplikacji, takich jak Matlab Simulink (środowisko służące do dynamicznej analizy systemów sterowania poprzez badania symulacyjne). Uruchomione instancje agentów oraz używane przez nich zewnętrzne aplikacje tworzą warstwy logiczne zaimplementowanego systemu wieloagentowego. Wyróżnić można warstwę interfejsu, warstwę zarządzającą, warstwę symulacyjną oraz warstwę decyzyjną. W każdej warstwie operują agenci tego samego typu, pomiędzy którymi nie zachodzą żadne interakcje (komunikują się oni jedynie z agentami operującymi w niższych lub wyższych warstwach systemu) [35]. Architekturę zaimplementowanego systemu wieloagentowego do wspomagania projektowania przedstawiono na rysunku Rys. 4.



Rys. 4. Architektura zaimplementowanego systemu wieloagentowego do wspomagania projektowania [35]

Zrealizowany projekt podobnie jak inne systemy rozproszone może zostać uruchomiony na wielu komputerach różniących się między sobą wydajnością i wersjami systemów operacyjnych. Komputery te powinny być połączone ze sobą siecią lokalną LAN (ang. Local Area Network), jednak możliwe jest użycie do tego celu globalnej sieci Internet poprzez zastosowanie tunelowania. Sieciowość systemu była wymaganiem podstawowym, ponieważ każdy system wieloagentowy powinien być skalowalny (to znaczy umożliwiać dodanie nowych komputerów do już istniejącej infrastruktury, nie wymagając przy tym restartu całego systemu). Dzięki zastosowaniu rozproszonej architektury możliwe było także zredukowanie kosztów sprzętu. Badania symulacyjne wymagają dużych mocy obliczeniowych, dlatego wspomniany wcześniej system ekspertowy może zostać uruchomiony tylko na bardzo wydajnych jednostkach (ponieważ zbieranie danych i symulacje odbywają się na tej samej maszynie) [30, 36, 37].

W przypadku systemu wieloagentowego użytkownicy podczas wprowadzania wytycznych projektowych korzystać mogą z właściwie dowolnego komputera, ponieważ agent interfejsu wymaga do działania jedynie platformy wieloagentowej (minimalne wymagania sprzętowe). Obliczenia statyczne i symulacyjne odbywać się mogą na zdalnych serwerach lub w chmurze obliczeniowej (nie muszą to być jednostki fizycznie dostępne tam, gdzie komputery z uruchomionymi agentami interfejsu). Baza danych również może znajdować się w lokalizacji sieciowej. Agent Master z modułem uczącym wymaga natomiast stabilnego serwera, ponieważ jest to dość newralgiczny punkt systemu (swego rodzaju węzeł sieci). Niezawodność systemu zwiększyć można poprzez uruchomienie kilku instancji agenta na wielu maszynach (warto wspomnieć, że na jednym komputerze może działać więcej, niż jeden typ agenta, więc nie musi to być dedykowana jednostka). System wieloagentowy cechuje zatem elastyczność konfiguracji, skalowalność i efektywność ekonomiczna [30, 36, 37]. Przykładową realizację warstwy sprzętowej zaimplementowanego systemu wieloagentowego przedstawiono na rysunku Rys. 5.



Rys. 5. Przykładowa realizacja warstwy sprzętowej zaimplementowanego systemu wieloagentowego [37]

System wieloagentowy do wspomagania projektowania wybranych układów sterowania zaimplementowany został z myślą o pracy wielu użytkowników w systemie jednocześnie. Pożądane jest zatem, aby przetwarzanie informacji w systemie odbywało się w jak najkrótszym czasie. Sumaryczny czas wygenerowania rozwiązania przez system wieloagentowy uzależniony jest od czasu trwania poszczególnych kroków procesu projektowania. Najbardziej czasochłonny jest etap badań symulacyjnych po zakończeniu budowy modelu układu sterowania. Mając na uwadze spełnienie założeń wydajnościowych oraz zachowanie spójności przy heterogenicznym środowisku sprzętowym niezbędny był wybór odpowiedniej platformy wieloagentowej do implementacji systemu [30, 35].

3.2. Platforma wieloagentowa

Do zarządzania agentami wchodzącymi w skład systemu służy platforma wieloagentowa. Jest to oprogramowanie stosowane do podglądu interakcji oraz komunikacji pomiędzy agentami, dodawania nowych instancji agentów do systemu, zawieszania działania agentów lub w razie konieczności usuwania agentów z systemu. Standardy działania platform (jak również całych systemów wieloagentowych) określa organizacja FIPA (ang. Foundation for Intelligent Physical Agents). Do głównych wymagań stawianych platformie wieloagentowej można zaliczyć [28-30]:

- stabilność (ang. Stability),
- krzepkość (ang. Robustness),
- użyteczność (ang. Usability),
- wydajność (ang. Performance),
- skalowalność (ang. Scalability),
- bezpieczeństwo (ang. Security),
- zgodność ze standardami (ang. Compatibility).

Podczas wyboru platformy wieloagentowej do implementacji systemu pod uwagę branych było kilka dostępnych rozwiązań, takich jak: JADE (ang. Java Agent Development Framework), SPADE (ang. Smart Python Multi-Agent Development Environment) i UMAP (ang. Universal Multi-Agent Platform). Porównanie cech każdej z wymienionych platform przedstawiono w tabeli Tab. 1.

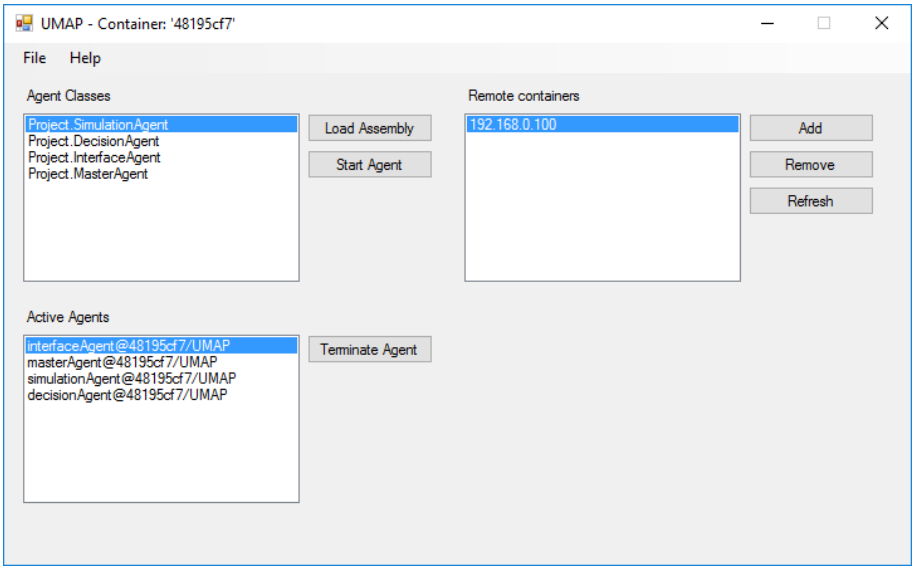
Tab. 1. Porównanie cech platform wieloagentowych JADE, SPADE, UMAP [38]

	JADE	**SPADE**	**UMAP**
Zgodność ze standardem FIPA	Tak	Tak	Tak
Język implementacji platformy	Java	Python	C#
Język implementacji agentów	Java	Python, Java, C++	C#, C++/CLI, VB.NET i inne zgodne z CLI (ang. Common Intermediate Language)
System operacyjny	Niezależna od systemu	Niezależna od systemu	Microsoft Windows, uniezależnienie od systemu poprzez platformę MONO
Dostęp do kodu źródłowego	Tak	Tak	Tak
Bezpośrednia zależność agentów od platformy	Tak	Nie	Tak
Bezpośrednia komunikacja pomiędzy agentami	Tak	Tak	Tak
Język komunikacji agentów	ACL	XMPP	ACL
Protokół transportu komunikatów	XML	XMPP, HTTP, SIMBA	XML

Komunikacja poprzez pamięć współdzieloną	Tak	Nie	Tak
Graficzny interfejs użytkownika	Tak	Tak	Tak

3.2.1. Uniwersalna platforma wieloagentowa UMAP

Uniwersalna platforma wieloagentowa UMAP daje możliwość uruchomienia w każdej warstwie zaimplementowanego systemu nieograniczonej ilości współpracujących ze sobą agentów. Pozwala to na równoczesne prowadzenie wielu procesów projektowych. Dzięki temu czas potrzebny do otrzymania rozwiązań kilku zadań projektowych jest tylko nieznacznie dłuższy w stosunku do czasu przetworzenia pojedynczego żądania przez system scentralizowany. Platforma UMAP napisana została w języku C# z wykorzystaniem .NET Framework zgodnie ze specyfikacją FIPA. Posiada ona graficzny interfejs użytkownika ułatwiający zarządzanie systemem wieloagentowym [30, 38]. Okno główne uniwersalnej platformy wieloagentowej UMAP przedstawiono na rysunku Rys. 6.



Rys. 6. Okno główne uniwersalnej platformy wieloagentowej UMAP [30]

Platforma UMAP dystrybuowana jest jako plik *.exe, natomiast jej instalacja na komputerze wymaga jedynie biblioteki .NET w wersji 3.0 lub nowszej. Po zainstalowaniu platformy dostępne są przykładowe projekty prostych systemów wieloagentowych służące do celów zapoznania ze środowiskiem. Implementacja systemu wieloagentowego w obrębie platformy rozpoczyna się od utworzenia nowego projektu, zdefiniowania nazw agentów oraz ich zachowań podczas zdarzeń takich jak: inicjalizacja, deinicjalizacja, otrzymanie wiadomości, jak również określenia cyklicznych akcji wykonywanych podczas okresu działania każdej instancji agenta. Nie trzeba definiować sposobu komunikacji pomiędzy agentami (platforma UMAP zapewnia to domyślnie). Komunikacja zrealizowana została w oparciu o komunikaty w formacie XML oraz obsługę gniazd sieciowych (ang. Socket) [30, 38].

Po zdefiniowaniu wszystkich typów agentów projekt należy skompilować. Wówczas wygenerowana zostaje biblioteka *.dll, która przechowuje logikę działania systemu wieloagentowego. Uruchomienie systemu wieloagentowego odbywa się w kilku etapach. Podczas inicjalizacji platformy UMAP na danym

komputerze utworzony zostaje lokalny kontener, któremu należy nadać nazwę (domyślnie jest to automatycznie generowany identyfikator) oraz podać numer portu używanego do komunikacji z innymi kontenerami. Następnie należy wczytać bibliotekę *.dll za pomocą przycisku „Load Assembly". W tym momencie w oknie „Agent Classess" powinny pojawić się zdefiniowane wcześniej typy agentów. Uruchomienie nowej instancji agenta na komputerze odbywa się poprzez naciśnięcie przycisku „Start Agent" i nadaniu mu określonej nazwy. Po inicjalizacji agent widoczny jest w oknie „Active Agents", opisany jako: `<nazwa_agenta>@<nazwa_kontenera>` [38, 39].

Na tym etapie widoczne są jedynie instancje agentów uruchomione w kontenerze lokalnym. Aby nowo dodani agenci mogli komunikować się z pozostałymi instancjami agentów w systemie, należy dodać kontener zdalny za pomocą przycisku „Add", umieszczonym po prawej stronie okna interfejsu. Podczas tej operacji administrator systemu wieloagentowego zostaje poproszony o podanie nazwy kontenera uruchomionego na komputerze zdalnym, jego adresu IP oraz numeru portu komunikacyjnego. Po dodaniu kontenera zdalnego zostaje on wyświetlony na liście w oknie „Remote containers". Uniwersalna platforma wieloagentowa UMAP składa się z trzech podstawowych elementów, które umożliwiają m.in. dodawania nowych jednostek do systemu [38, 39]:

- usługa katalogowania agentów DF (ang. Directory Facilitator),
- system zarządzania agentami AMS (ang. Agent Management System),
- system transportu komunikatów MTS (ang. Message Transport System).

Za pomocą platformy UMAP utworzyć można nieograniczoną ilość kontenerów, wewnątrz których uruchamiane będą procesy agentów. Każdy proces składa się domyślnie z kilku niezależnych wątków. Jeden wątek procesu agenta zarezerwowany jest do obsługi wiadomości w formacie ACL (ang. Agent Communication Language), natomiast pozostałe realizują współbieżnie zdefiniowane algorytmy projektowe. Aby ustandaryzować implementację agentów, platforma UMAP udostępnia jednolity, uniwersalny szkielet agenta będący klasą abstrakcyjną. Definicja wspomnianych wcześniej zachowań agentów odbywa się poprzez nadpisywanie czterech podstawowych metod szkieletu [30, 38, 39]:

- OnActivate(); – metoda wykonywana podczas inicjalizacji nowej instancji agenta,
- OnDeactivate(); – metoda wykonywana podczas deinicjalizacji instancji agenta,
- HandleMessage(Message message); – metoda wykonywana podczas otrzymania wiadomości,
- Run(); – metoda wykonywana przez cały cykl życia danej instancji agenta.

3.3. Podsumowanie

Utworzony system wieloagentowy posiada architekturę warstwową. W każdej warstwie systemu operują agenci tego samego typu, którzy komunikują się z agentami niższych i wyższych warstw. Wybór platformy wieloagentowej do implementacji systemu padł ostatecznie na uniwersalną platformę wieloagentową UMAP, powstałą na Politechnice Śląskiej w Gliwicach. Platforma UMAP zapewnia odpowiednią skalowalność systemu wieloagentowego oraz stosunkową łatwość implementacji agentów. Rozdział 4 zawiera charakterystykę poszczególnych typów agentów w systemie.

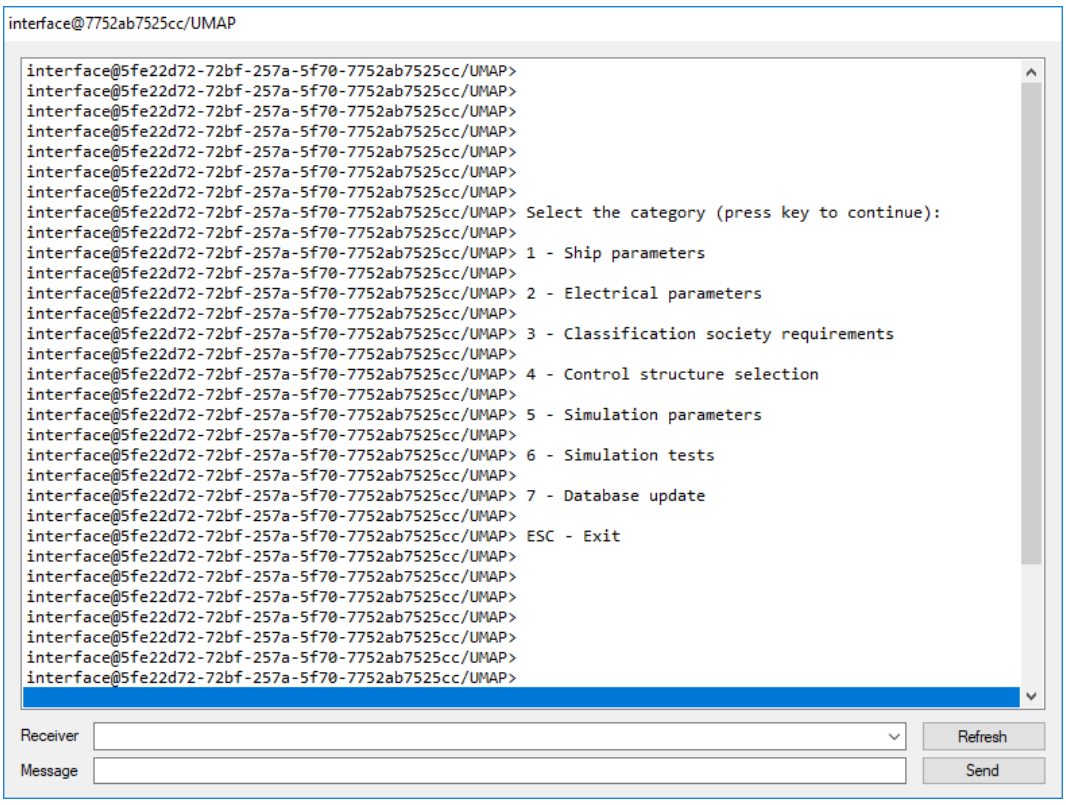
4. CHARAKTERYSTYKA POSZCZEGÓLNYCH AGENTÓW W SYSTEMIE

4.1. Wprowadzenie

System wieloagentowy składa się z czterech podstawowych typów agentów: agenta interfejsu (ang. Interface Agent), agenta symulacyjnego (ang. Simulation Agent), agenta decyzyjnego/ekspertowego (ang. Decision Agent), agenta nadrzędnego (ang. Master Agent). Kolejne podrozdziały opisują atrybuty poszczególnych typów agentów w zaimplementowanym systemie oraz ich rolę w procesie wspomagania projektowania układów sterowania okrętowych podsystemów elektroenergetycznych.

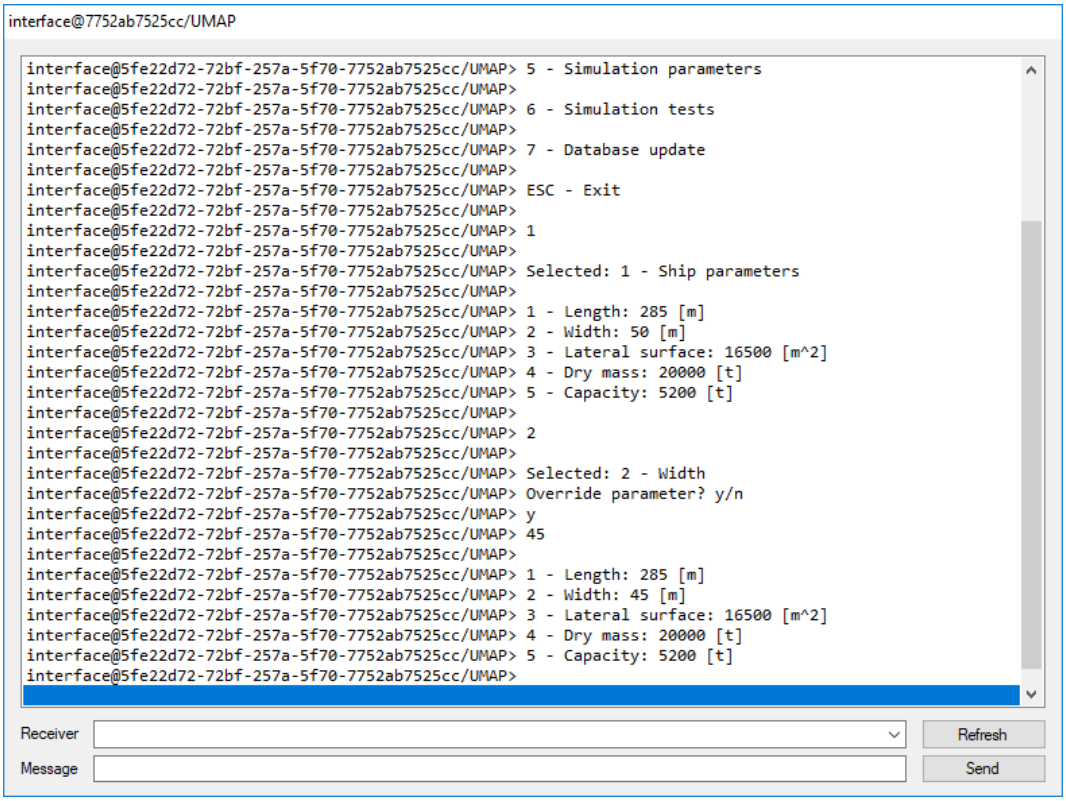
4.2. Agent interfejsu (Interface Agent)

Agent interfejsu ma na celu umożliwienie interakcji użytkowników z systemem wieloagentowym. Uruchomienie kilku instancji agenta tego typu pozwala na równoczesną pracę wielu inżynierów projektantów z systemem wieloagentowym, przy czym standardowo na każdym komputerze działa tylko jeden Interface Agent. Głównym zadaniem agenta interfejsu jest zbieranie od użytkowników wytycznych i ograniczeń odnoszących się do realizowanych projektów układów sterowania okrętowych podsystemów elektroenergetycznych. Aby proces akwizycji danych przebiegał w sposób przejrzysty i zrozumiały interfejs generuje pytania, na które odpowiadają użytkownicy. Rozpoczęcie zadawania pytań następuje po wybraniu odpowiedniej kategorii projektowej. Dzięki temu formułowane są nowe lub modyfikowane już istniejące (wcześniej wprowadzone) założenia do projektowanego układu sterowania [30, 35]. Menu główne interfejsu użytkownika systemu wieloagentowego, które udostępnia Interface Agent przedstawiono na rysunku Rys. 7.



Rys. 7. Menu główne interfejsu użytkownika systemu wieloagentowego [30]

Użytkownik ma możliwość wprowadzenia z menu informacji o okręcie, dla którego dany układ jest projektowany, cech jego instalacji elektrycznej i ograniczeń wynikających z wymagań towarzystwa klasyfikacyjnego, co w dalszej kolejności pozwala na wygenerowanie predefiniowanego raportu z analizy zamodelowanego systemu sterowania (projekt ofertowy tworzony przez agenta ekspertowego). Innymi bardzo ważnymi informacjami zbieranymi przez agenta interfejsu są parametry symulacji i wartości zmiennych dla modeli elementów układu sterowania aktualizowane w bazie danych (bez tego agent symulacyjny byłby w stanie przeprowadzić testy symulacyjne jedynie z domyślnymi parametrami). Dane od użytkownika wprowadzane są do systemu wieloagentowego z wykorzystaniem podstawowych elementów interfejsów graficznych, takich jak pola tekstowe (textbox). Interfejs posiada kilka poziomów ekranów dialogowych, po których nawiguje użytkownik, dokonując modyfikacji parametrów znajdujących się w określonych kategoriach projektowych [30, 35]. Strukturę komunikatów wyświetlanych na ekranie dialogowym w menu interfejsu użytkownika przedstawiono na rysunku Rys. 8.



Rys. 8. Struktura komunikatów wyświetlanych na ekranie dialogowym w menu interfejsu użytkownika [30]

4.2.1. Tabela danych

Informacje wprowadzone przez użytkownika do systemu wieloagentowego zapisywane są przez agenta interfejsu w postaci tabelarycznej. Tabela, w której gromadzone są dane zbierane za pośrednictwem ekranów dialogowych posiada dwie kolumny (element oraz wartość). Tabela składa się wyłącznie ze znaków w kodzie ASCII, ponieważ wtedy możliwa jest jej serializacja i przesyłanie w przystępnej formie pomiędzy agentami platformy UMAP (uniwersalna platforma wieloagentowa UMAP obsługuje wiadomości wyłącznie w postaci tekstowej). Przyjęta forma przesyłanych komunikatów

nie jest standardem, jednak zdecydowano się na zachowanie takiej konwencji (zaczerpniętej ze środowiska Exsys) ze względu na przejrzystą formę danych, która ułatwia debugowanie oraz analizę ciągów znaków za pomocą wbudowanych funkcji środowiska .NET C# (ang. String parsing). Tabela przechowywana jest w pliku „Parameters.frm", odczytywanym podczas inicjalizacji agenta interfejsu użytkownika [23-27, 35, 36]. Strukturę tabeli danych zebranych od użytkownika systemu wieloagentowego przez agenta interfejsu przedstawiono w tabeli Tab. 2.

Tab. 2. Struktura tabeli danych zebranych od użytkownika systemu przez agenta interfejsu [36]

Element	Value
SHIP	0
INSTALLATION	0
CLASSIFICATION	0
STRUCTURE	0
SIMULATION	0
DATABASE	0
RESEARCH	1
LENGTH	285
WIDTH	45
LATERAL_SURFACE	16500
DRY_MASS	20000
CAPACITY	5200
GENERATOR_AMOUNT	5
GENERATOR_P	2800
MOTOR_AMOUNT	4
MOTOR_U	440
MOTOR_I	100
POWER_LINE_LENGTH	15
POWER_LINE_CS	25
CONDUCTIVITY	58.6
FACTOR	0.0044
GENERATOR_OVERLOAD_S	1.5
GENERATOR_U_PICK_%	20
GENERATOR_U_DROP_%	20
POWER_LINE_U_DROP_S	25
CONTROL_STRUCTURE	5
SIMULATION_TIME	100
GENERATOR1_ON_TIME	30
GENERATOR2_ON_TIME	30
MOTOR1_ON_TIME	40
MOTOR2_ON_TIME	41
PROPELLER1A_ON_TIME	60
PROPELLER1A_HP	0.5
PROPELLER1B_ON_TIME	70
PROPELLER1B_HP	1
PROPELLER1_OFF_TIME	80
PROPELLER2A_ON_TIME	65
PROPELLER2A_HP	0.5
PROPELLER2B_ON_TIME	75
PROPELLER2B_HP	1
PROPELLER2_OFF_TIME	85
INCLINATION_ANGLE_HP	60
DEBUG_MODE	0

Pierwsza sekcja pod nagłówkiem tabeli jest sekcją kontrolną, która niesie informację dla agenta interfejsu dotyczącą postępów użytkownika w dostarczaniu danych do innych sekcji. W momencie otwarcia pliku *.frm pola tej sekcji są zerowane. Każde pole odpowiada jednej kategorii projektowej (np. pole „SHIP" odpowiada kategorii „Ship parameters"). W momencie uzupełnienia przez użytkownika danej kategorii w interfejsie, odpowiadające jej pole przyjmuje niezerową wartość. Akwizycja danych zostaje zakończona wówczas, gdy wszystkie pola oprócz ostatniego są różne od zera lub kiedy użytkownik wymusi badania symulacyjne z domyślnymi parametrami (wybór kategorii „Simulation tests" i ustawienie pola „RESEARCH" na wartość różną od zera). Agent interfejsu skanuje plik *.frm z określonym interwałem czasowym (metoda spinlock). Jeżeli podczas skanowania pliku *.frm status pól sekcji kontrolnej wskazuje na zakończenie akwizycji danych, wówczas zawartość pliku (tabela danych) przesyłana jest dalej do odpowiedniego agenta systemu i pola znów zostają wyzerowane.

Kolejne dwie sekcje przechowują dane dotyczące parametrów statku oraz cech jego instalacji elektrycznej. Użytkownik systemu wieloagentowego definiuje parametry, takie jak długość i szerokość całkowita oraz powierzchnia boczna statku, dla którego dany układ sterowania jest projektowany, jak również jego masę oraz maksymalną ładowność statku (im większa inercja obiektu, tym większą moc

podsystem elektroenergetyczny będzie musiał wygenerować, aby zasilić układ napędowy). Jeżeli chodzi o parametry instalacji elektrycznej, to przechowywane zmienne określają ilość generatorów, jaka ma być zainstalowana na statku i ich moc, jak również parametry (napięcie i prąd znamionowy) silników indukcyjnych, będących głównymi odbiornikami energii elektrycznej oraz właściwości linii kablowych (długość linii, przekroje kabli, konduktywność) [23-27, 30, 35].

Ostatnie sekcje zawierają odpowiednio: ograniczenia towarzystwa klasyfikacyjnego, rodzaj struktury układu sterowania, parametry symulacji oraz pole do uruchomienia trybu debugowego. Ograniczenia dotyczą wyłącznie charakterystyk instalacji elektrycznej i zespołu prądotwórczego. Jest to na przykład dopuszczalny czas przeciążenia generatora, maksymalny wzrost napięcia na każdym z generatorów, jak również dopuszczalne spadki napięć na generatorach lub liniach kablowych. Użytkownik w większości przypadków nie musi dokonywać zmian wartości domyślnych (wartości te są stałe), jednak w przypadku zmiany dopuszczalnych charakterystyk przez towarzystwo klasyfikacyjne możliwa jest aktualizacja tych atrybutów. Jeżeli chodzi o parametry symulacji, to w sekcji tabeli danych zapisywane są zmienne definiujące scenariusz symulacji, takie jak czas załączenia generatorów i silników indukcyjnych, czas załączenia i wyłączenia skoku śrub nastawnych oraz wartość i kąt nachylenia krzywej tego skoku, jak również całkowity czas symulacji [23-27, 30, 35].

Dodatkowo agent interfejsu odpowiada za wstępną weryfikację informacji od użytkownika. Weryfikacja ta odbywa się na etapie wprowadzania danych do pól tekstowych. Sprawdzany jest typ danych (m.in. czy są to wartości liczbowe), czy dane te mieszczą się w spodziewanym zakresie (m.in. czy są to wartości większe lub równe zero), czy wprowadzone dane nie zawierają białych znaków oraz czy wszystkie pola zostały wypełnione. Do wstępnej walidacji wprowadzonych danych użyte zostały wyrażenia regularne (ang. Regular expressions, w skrócie regex) Po zebraniu wszystkich niezbędnych informacji wysyłane są one dalej do agenta nadrzędnego [30, 35].

4.3. Agent symulacyjny (Simulation Agent)

Agent symulacyjny odpowiada za badania symulacyjne, które pozwalają na określenie charakterystyki dynamicznej utworzonych struktur układów sterowania. Podczas swojego działania korzysta on z zewnętrznego środowiska bazującego na oprogramowaniu Matlab Simulink oraz Microsoft Access. Ze względu na prowadzenie złożonych obliczeń dynamicznych agent ten potrzebuje stosunkowo dużych mocy obliczeniowych. Minimalne wymagania sprzętowe jednostki, na której może zostać uruchomiony agent symulacyjny to co najmniej dwurdzeniowy procesor z taktowaniem 2.5 GHz, 4 GB pamięci DRAM oraz 20 GB wolnej przestrzeni dyskowej. Po otrzymaniu danych wejściowych, zawierających wytyczne dotyczące projektowanej struktury (pochodzących od użytkownika systemu wieloagentowego), następuje wywołanie środowiska Matlab Simulink oraz zestawienie połączenia z bazą modeli elementów poprzez polecenie: load_system('model_library'). Baza modeli jest to biblioteka utworzona w programie Simulink, z której importowane są do przestrzeni roboczej komponenty niezbędne do utworzenia modelu układu sterowania okrętowych podsystemów elektroenergetycznych, np. model generatora lub silnika indukcyjnego [30, 35].

Wczytywanie modeli elementów do przestrzeni roboczej środowiska symulacyjnego odbywa się trójetapowo, rozpoczynając od identyfikacji, które elementy będą potrzebne do przeprowadzenia badań

i odnalezienia ich w bazie modeli. Informacji tej dostarcza sekcja „STRUCTURE", która zawiera pole „CONTROL_STRUCTURE" w tabeli danych. Wartość tego pola określa rodzaj struktury układu sterowania podsystemu elektroenergetycznego. Na podstawie numeru każdego typu struktury wiadomo jakie elementy powinny zostać wczytane, natomiast sekcja „INSTALLATION" określa ile elementów danego typu wskazał użytkownik. Dla przykładu wartość 5 oznacza strukturę sterowania poprzez autotransformator, w której skład wchodzą: silniki diesla (lub turbiny gazowe), generatory synchroniczne, autotransformatory, silniki indukcyjne, regulator napięcia oraz części mechaniczne odbiornika energii (wał napędowy i śruba o skoku nastawnym). Kolejnym etapem jest określenie pozycji bazowej elementów względem innych bloków modeli w przestrzeni roboczej, tak aby bloki nie nachodziły na siebie. Ostatnim krokiem jest wczytanie elementów z bazy modeli poprzez polecenie add_block, np. wczytanie jednego bloku modelu autotransformatora odbywa się po wydaniu instrukcji:

```
add_block('model_library/Autotransformer','model/Autotransformer_1','position',pos)
```

Jeżeli wszystkie modele elementów zostały poprawnie wczytane z bazy, wówczas następuje automatyczne generowanie połączeń między wejściami i wyjściami bloków modeli. Do automatycznego zestawiania połączeń zastosowano funkcję add_line, np. połączenie wyjścia bloku modelu generatora synchronicznego z wejściem bloku autotransformatora odbywa się po wydaniu instrukcji:

```
add_line('model','Generator_1/2','Autotransformer_1/2','autorouting','on')
```

Warto nadmienić, że instrukcja add_line standardowo wyrysowuje jak najprostsze połączenia pomiędzy wejściami/wyjściami bloków modeli wskazanymi w parametrach tej funkcji. Oznacza to, że linie będą poprowadzone wyłącznie równolegle do siebie i mogą przecinać inne bloki modeli, które znalazły się na ścieżce przejścia tych linii. Aby temu zapobiec każde wywołanie funkcji add_line zawiera parametr „autorouting" ustawiony na „on". Pozwala to na osiągnięcie lepszej czytelności wygenerowanego automatycznie modelu układu. Po zakończeniu budowy kompletnego modelu systemu sterowania uruchamiany jest algorytm weryfikujący poprawność utworzonej struktury. Wszystkie wejścia i wyjścia bloków modeli matematycznych muszą być ze sobą sprzężone, aby uniknąć błędów podczas uruchomienia symulacji w Matlab Simulink [23-27, 30, 35].

Modele komponentów systemu sterowania wczytane z bazy modeli do przestrzeni roboczej Matlab Simulink początkowo mają domyślne parametry. Aby nastąpiło odpowiednie dostosowanie parametrów modeli do konkretnego scenariusza symulacyjnego, niezbędne jest wczytanie wytycznych użytkownika zapisanych do bazy danych parametrów modeli utworzonej w Microsoft Access. W tym celu zastosowano następującą instrukcję, służącą do obsługi bazy danych z poziomu Matlab Simulink:

```
connection=database.ODBCConnection('Database','','');
```

Parametry modeli modyfikowane są w bazie danych przez użytkownika systemu wieloagentowego za pomocą ekranu dialogowego „Database update" z menu głównego agenta interfejsu. W przypadku braku niezbędnych informacji od użytkownika pobierana jest z bazy danych początkowa wartość zaczerpnięta z dokumentacji technicznej lub danych katalogowych producentów poszczególnych elementów systemów sterowania. Powodem zastosowania bazy danych Microsoft Access do wczytywania parametrów modeli, zamiast umożliwienia użytkownikowi ustawiania ich bezpośrednio w bazie modeli jest fakt, iż agent interfejsu i agent symulacyjny nie zawsze będą działać fizycznie na tych samych komputerach [23-27, 30, 35].

Wprowadzanie parametrów modeli elementów przez użytkowników systemu wieloagentowego bezpośrednio do bazy modeli oznaczałoby potrzebę instalacji oprogramowania Matlab Simulink na każdym komputerze, na którym działa agent interfejsu. Byłoby to nieefektywne ekonomicznie, ponieważ wymuszałoby stosowanie jednostek o znacznie wyższych parametrach sprzętowych do uruchamiania agentów interfejsu (agent symulacyjny i Matlab Simulink potrzebują wydajnych jednostek sprzętowych, natomiast agent interfejsu i zastosowana baza Microsoft Access mogą być uruchomione w zasadzie na każdym komputerze z systemem Windows). W opracowanym podejściu modele matematyczne są generyczne i dopiero po połączeniu z bazą danych oraz wczytaniu parametrów wprowadzonych przez użytkownika następuje ich specjalizacja do konkretnego zadania [23-27, 30, 35]. Wczytanie z bazy danych przykładowego parametru regulatora napięcia generatora (parametr Ui_max reprezentujący maksymalne dopuszczalne napięcie generatora) odbywa się po wydaniu instrukcji:

```
disp(['Automatic Voltage Regulator: ' num2str(AVR_generator)])
curs=exec(conn,'select * from AVR_generator where Ui_max');
curs=fetch(curs);
AVR_generator_array=cellfun(@mean,curs.Data);
Uimax=AVR_generator_array(AVR_generator,7);
```

Po wczytaniu parametrów modeli z bazy danych Microsoft Access następuje uruchomienie symulacji w oprogramowaniu Matlab Simulink. Wyniki badań symulacyjnych dostępne są w postaci charakterystyk przedstawiających np. spadek napięcia na generatorze po załączeniu silnika indukcyjnego, maksymalny prąd rozruchowy silnika indukcyjnego, spadek napięcia wynikający z impedancji przewodów oraz wiele innych. Matlab Simulink po zakończonej symulacji najpierw zapisuje wszystkie wyniki pomiarów do zbiorczego pliku „Results.dat", a następnie eksportuje z przestrzeni roboczej do plików *.mat wartości zmiennych symulacyjnych [23-27, 30, 35]. Na podstawie danych z plików *.mat wykreślić można przebiegi tych zmiennych. Na przykład wykres zmiennej ScopeData_dUg, reprezentującej spadek napięcia generatora uzyskuje się po wydaniu instrukcji:

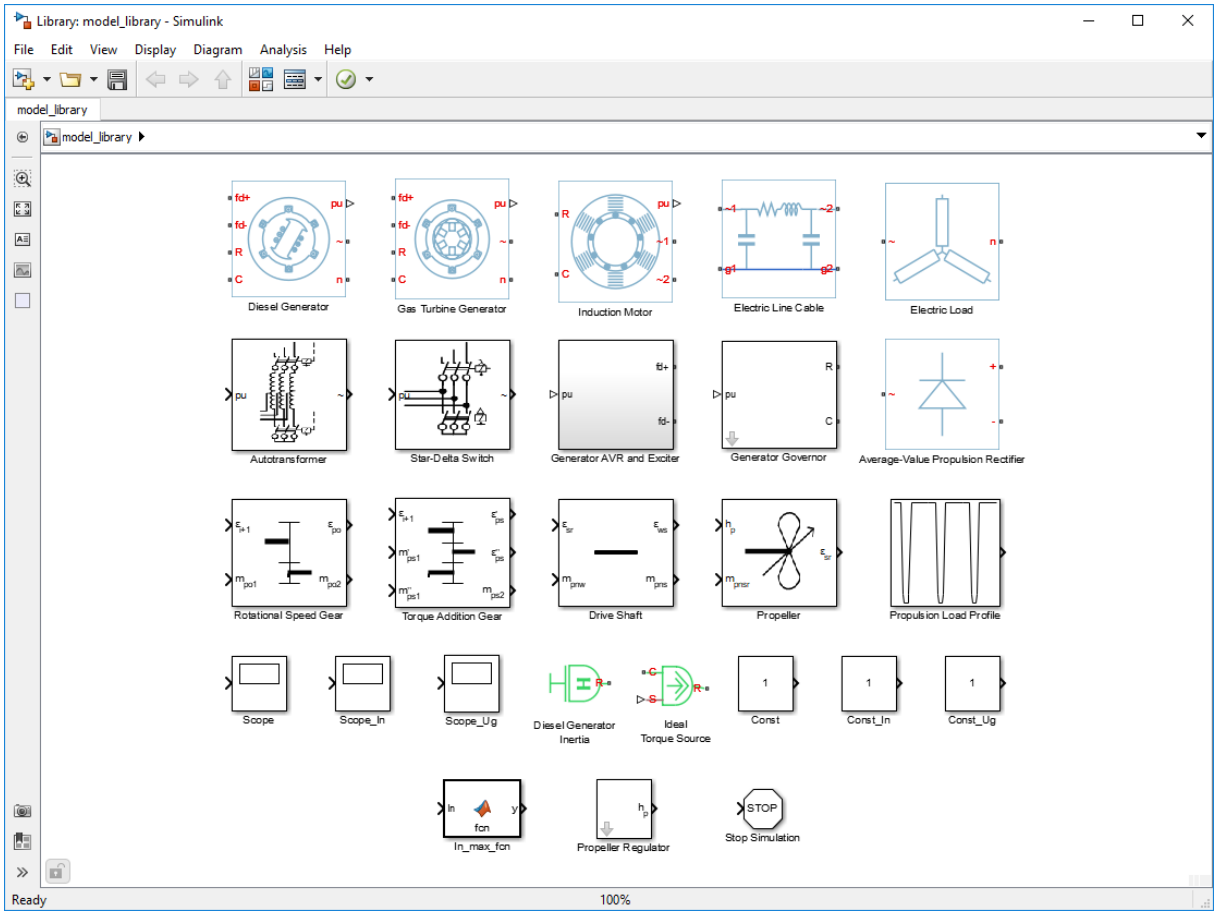
```
exist ScopeData_dUg;
if ans == 1
H=figure(10);
plot(ScopeData_dUg.time,ScopeData_dUg.signals.values,'black','LineWidth',1.5);
grid on;
set(gca,'fontsize',10);
A=get(0,'ScreenSize');
set(H,'Position',[50, A(4)*0.2-200, A(4)*0.20*4/3,A(4)*0.2]);
xlabel('t [s]');
ylabel({'Generator voltage drop';'DeltaU_g [%]'}););
end
```

Wyniki symulacji z pliku „Results.dat" dopisywane są do tabeli danych jako nowa sekcja. Agent symulacyjny, podobnie jak agent interfejsu wykorzystuje metodę „spinlock" do skanowania pliku *.dat z określonym interwałem czasowym w celu wykrycia zmian, które wskazują na zakończenie wykonywania operacji przez zewnętrzne oprogramowanie (w przypadku agenta symulacyjnego jest to Matlab Simulink). Jest to stosunkowo prosty sposób, aby zaimplementować współpracę dwóch środowisk programistycznych, które nie mogą się bezpośrednio ze sobą komunikować. W dalszej kolejności kompletna, zaktualizowana o nową sekcję tabela danych przesłana zostaje do agenta decyzyjnego. Poniżej przedstawiono pola dodane do tabeli danych przez agenta symulacyjnego wraz z ich opisem, natomiast kolejne podrozdziały zawierają szczegółowe opisy bazy modeli elementów systemów sterowania i bazy danych parametrów tych modeli [23-27, 30, 35].

```
+--------------------+------------+
|GENERATOR_U         |0.993022    |   //Napięcie generatora bez obciążenia
|GENERATOR_U_DROP    |2.567860    |   //Spadek napięcia generatora przy załączeniu silnika
|GENERATOR_U_DROP_A  |0.984192    |   //Spadek napięcia generatora przy załączeniu obc. A
|GENERATOR_U_DROP_B  |0.984192    |   //Spadek napięcia generatora przy załączeniu obc. B
|MOTOR_I             |4.764120    |   //Maksymalny prąd rozruchowy silnika indukcyjnego
|MOTOR_RPM_DROP_A    |-0.63290    |   //Spadek obrotów silnika po załączeniu obciążenia A
|MOTOR_RPM_DROP_B    |-0.54250    |   //Spadek obrotów silnika po załączeniu obciążenia B
|MOTOR_RPM           |0.049338    |   //Moment silnika bez obciążenia
|MOTOR_RPM_A         |0.108482    |   //Moment silnika z obciążeniem A
|MOTOR_RPM_B         |0.255518    |   //Moment silnika z obciążeniem B
|MOTOR_ON_TIME       |11.84010    |   //Czas rozruchu silnika indukcyjnego
|POWER_LINE_U_DROP   |0.181281    |   //Spadek napięcia w linii kablowej
+--------------------+------------+
```

4.3.1. Baza modeli: Simulink Models Library

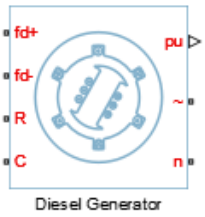
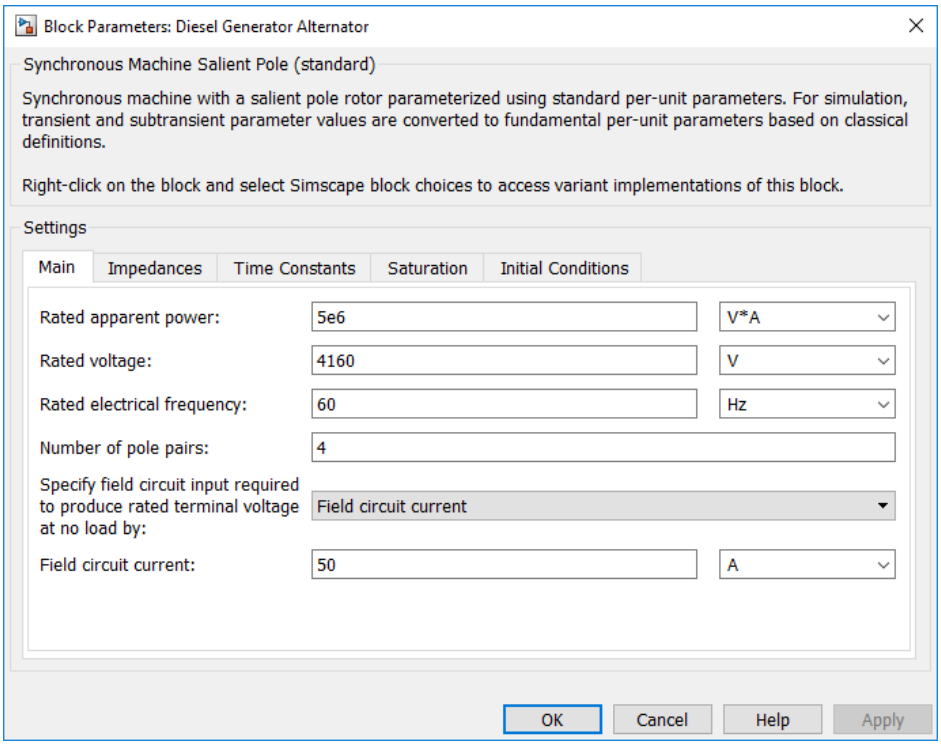
Baza modeli utworzona w programie Matlab Simulink przechowywana jest w pliku „model_library.mdl”, z którego agent symulacyjny wczytuje do przestrzeni roboczej wszystkie modele elementów systemu sterowania wskazane przez użytkownika w wytycznych projektowych. Baza modeli zawiera zarówno elementy elektroenergetyczne okrętów, jak i mechaniczne (model silnika wysokoprężnego, komponenty napędowe). Oprócz tego znajdują się tam regulatory niezbędne do budowy układów sterowania (niektóre z nich dostępne są jako osobne bloki, aczkolwiek część z nich zintegrowana została z modelami urządzeń elektroenergetycznych) [23-27, 30, 35, 40]. Bazę modeli (Simulink Models Library) przedstawiono na rysunku Rys. 9.



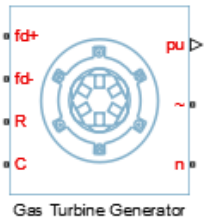
Rys. 9. Baza modeli (Simulink Models Library) [40]

Charakterystykę wybranych modeli elementów systemów sterowania z bazy modeli (najważniejsze elementy elektroenergetyczne, wraz z ich opisem, symbolem i subsystemem) przedstawiono w tabeli Tab. 3.

Tab. 3. Wybrane elementy systemów sterowania z bazy modeli (Simulink Models Library) [23-27, 30, 35, 40]

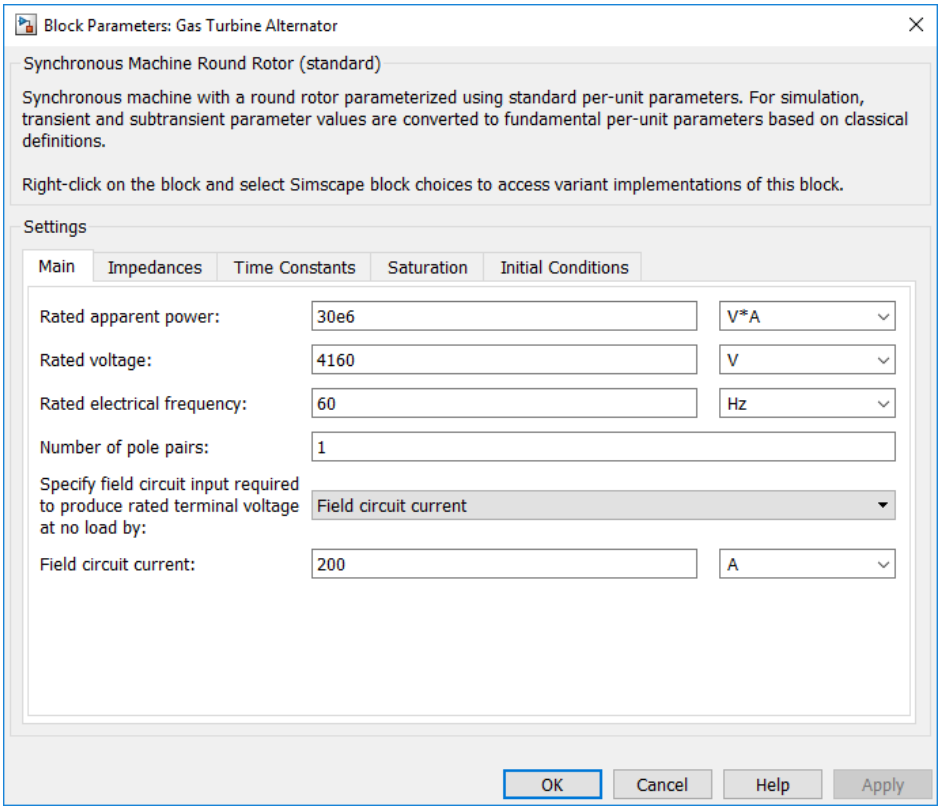
Nazwa i symbol elementu	Parametry i opis elementu
Generator napędzany silnikiem wysokoprężnym 	Generator jest to urządzenie służące do wytwarzania energii elektrycznej, które zapewnia podstawowe zasilanie okrętowych odbiorników elektroenergetycznych. Generator okrętowy napędzany jest przez wysokoprężny silnik spalinowy lub turbinę gazową [41]. Domyślne parametry modelu generatora synchronicznego napędzanego silnikiem wysokoprężnym [40]:  Wejścia i wyjścia bloku modelu generatora synchronicznego [40]: fd+ – Port elektryczny związany z dodatnim zaciskiem uzwojenia wzbudzenia, fd- – Port elektryczny związany z ujemnym zaciskiem uzwojenia wzbudzenia, R – Port mechaniczny związany z prędkością obrotową wirnika, C – Port mechaniczny związany z parametrami stojana, pu – Port elektryczny związany z napięciem generatora, ~ – Port elektryczny związany z prądem uzwojenia stojana, n – Port elektryczny związany z fazą neutralną.

Generator napędzany turbiną gazową



Generator jest to urządzenie służące do wytwarzania energii elektrycznej, które zapewnia podstawowe zasilanie okrętowych odbiorników elektroenergetycznych. Generator okrętowy napędzany jest przez wysokoprężny silnik spalinowy lub turbinę gazową [41].

Domyślne parametry modelu generatora synchronicznego napędzanego turbiną gazową [40]:



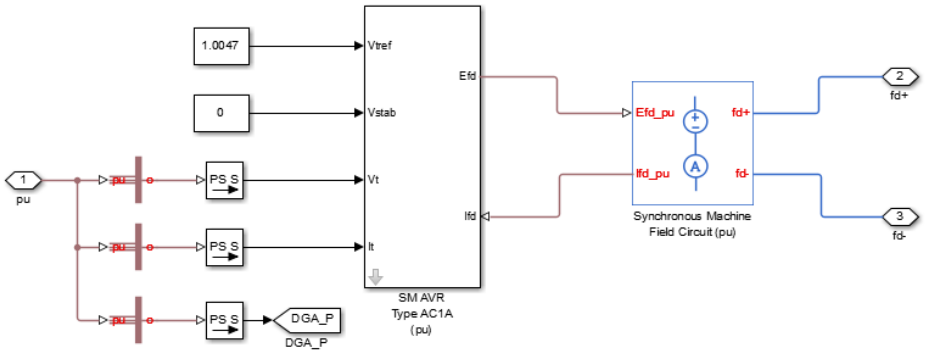
Wejścia i wyjścia bloku modelu generatora synchronicznego [40]:

fd+ – Port elektryczny związany z dodatnim zaciskiem uzwojenia wzbudzenia,
fd- – Port elektryczny związany z ujemnym zaciskiem uzwojenia wzbudzenia,
R – Port mechaniczny związany z prędkością obrotową wirnika,
C – Port mechaniczny związany z parametrami stojana,

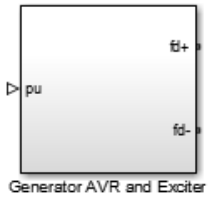
pu – Port elektryczny związany z napięciem generatora,
~ – Port elektryczny związany z prądem uzwojenia stojana,
n – Port elektryczny związany z fazą neutralną.

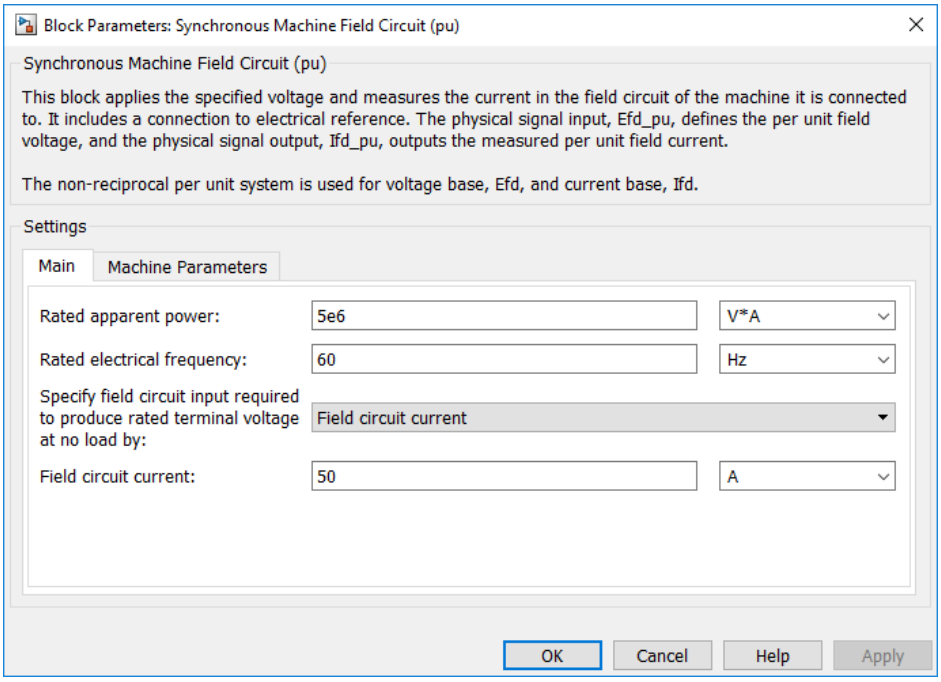
Układ AVR i wzbudzenia generatora służy do kompensacji mocy biernej w systemie elektroenergetycznym oraz steruje generatorem w taki sposób, aby utrzymać wartość napięcia w systemie na stałym poziomie (układ ten zawiera wbudowany regulator napięcia generatora). Rozpatrywany układ wzbudzenia oparty został o tyrystory półprzewodnikowe [41].

Domyślne parametry modelu układu AVR i wzbudzenia generatora [40]:



Układ AVR i wzbudzenia generatora

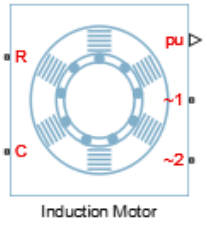
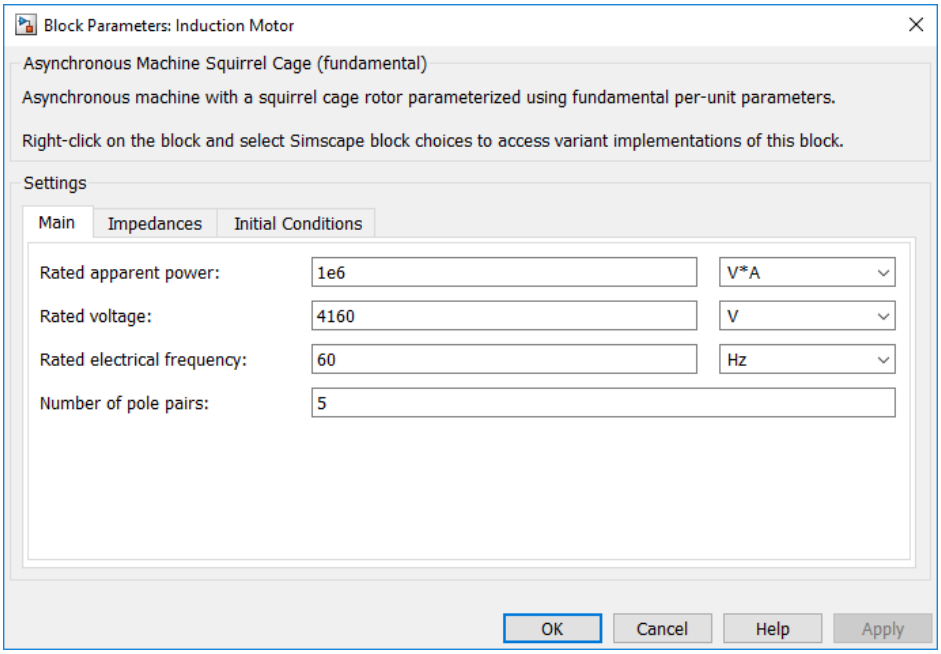
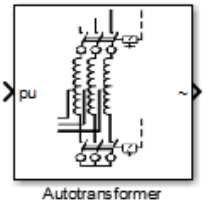


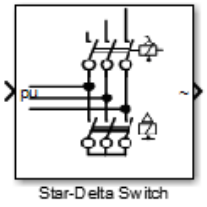
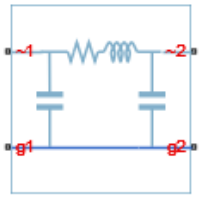
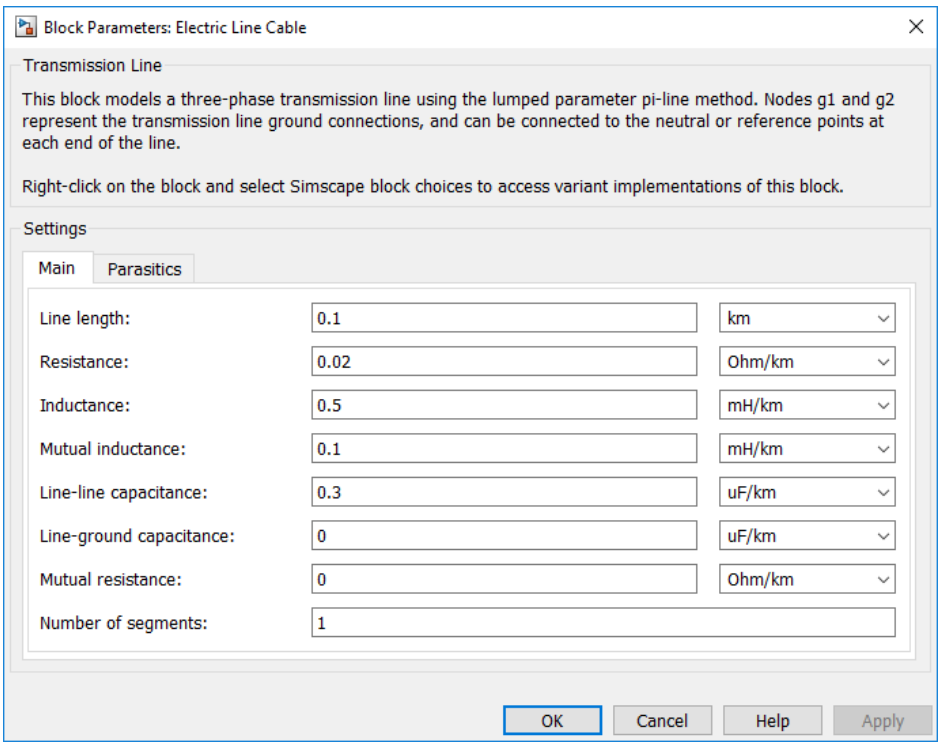


Wejścia i wyjścia bloku modelu układu AVR i wzbudzenia generatora [40]:

pu – Port elektryczny związany z napięciem generatora,

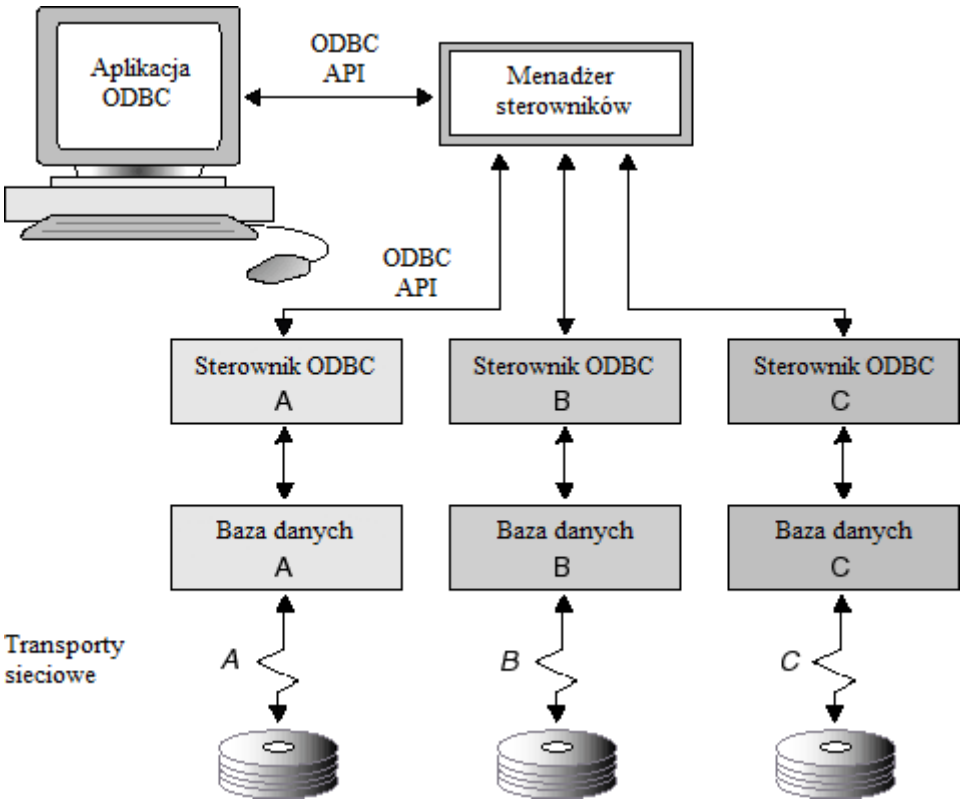
fd+ – Port elektryczny związany z dodatnim zaciskiem uzwojenia wzbudzenia,
fd- – Port elektryczny związany z ujemnym zaciskiem uzwojenia wzbudzenia.

Silnik indukcyjny  Induction Motor	Silnik indukcyjny jest to urządzenie do wytwarzania energii mechanicznej z dostarczonej energii elektrycznej. Silnik składa się z nieruchomego stojana oraz obracającego się wewnątrz wirnika. Dostarczony do silnika prąd przemienny powoduje indukowanie się zmiennego pola magnetycznego, które prowadzi do wytworzenia momentu obrotowego [41]. Domyślne parametry modelu silnika indukcyjnego [40]:  Wejścia i wyjścia bloku modelu silnika indukcyjnego [40]: R – Port mechaniczny związany z prędkością obrotową wirnika, C – Port mechaniczny związany z parametrami stojana, pu – Port elektryczny związany z napięciem silnika indukcyjnego, ~1 – Port elektryczny związany z wartością skuteczną prądu pierwszej fazy, ~2 – Port elektryczny związany z wartością skuteczną prądu drugiej fazy.
Autotransformator  Autotransformer	Autotransformator jest to rodzaj transformatora, w którym pomiędzy obwodami pierwotnym i wtórnym nie występuje separacja galwaniczna. Służy on do sterowania rozruchem silnika indukcyjnego poprzez zmianę wartości napięcia. Dzięki połączeniu obu obwodów posiada on dodatkowo lepszą sprawność w porównaniu do standardowego transformatora, osiągniętą poprzez redukcję masy własnej [42]. Wejścia i wyjścia bloku modelu autotransformatora [23-27, 30, 35]: pu – Port elektryczny związany z napięciem wejściowym, ~ – Port elektryczny związany z napięciem wyjściowym.

Przełącznik gwiazda-trójkąt  	Przełącznik gwiazda-trójkąt służy do sterowania rozruchem silnika indukcyjnego poprzez zmianę konfiguracji połączenia uzwojeń. Na początku rozruchu moment silnika jest trzykrotnie mniejszy, ponieważ dostarczany prąd ma również trzykrotnie niższą wartość (przy gwieździe). Po zmianie konfiguracji w trójkąt silnik osiąga prędkość nominalną [43] Wejścia i wyjścia bloku modelu przełącznika gwiazda-trójkąt [23-27, 30, 35]: pu – Port elektryczny związany z napięciem wejściowym, ~ – Port elektryczny związany z napięciem wyjściowym.
Sieć elektroenergetyczna  	Model sieci elektroenergetycznej służy do uwzględniania spadku napięcia na przewodach oraz impedancji i reaktancji sieci [41]. Domyślne parametry modelu sieci zasilającej [40]:  Wejścia i wyjścia bloku modelu sieci elektroenergetycznej [40]: ~1 – Port elektryczny związany z napięciem wejściowym, g1 – Port elektryczny związany z uziemieniem, ~2 – Port elektryczny związany z napięciem wyjściowym, g1 – Port elektryczny związany z uziemieniem.

4.3.2. Baza danych: Microsoft Access Database

Baza danych parametrów utworzona w programie Microsoft Access przechowywana jest w pliku „database.mdb", z którego agent symulacyjny wczytuje parametry modeli elementów układu sterowania zapisane przez użytkownika podczas sesji ustanowionej przez agenta interfejsu. Zestawienie połączenia z bazą danych parametrów modeli z poziomu agenta interfejsu odbywa się za pośrednictwem ODBC (ang. Open Database Connectivity). ODBC jest standardem, który udostępnia API (ang. Application Programming Interface) do obsługi bazy danych niezależnie od używanej aplikacji, języka programowania oraz rodzaju bazy danych. ODBC dostępny jest standardowo w niemal każdym systemie operacyjnym i nie wymaga instalacji dodatkowego oprogramowania. Jedynym wymaganiem do zastosowania ODBC jest instalacja sterownika (ODBC driver) w odpowiedniej wersji bitowej, kompatybilnej ze środowiskiem symulacyjnym [23-27, 35, 44]. Schemat komunikacji aplikacji z bazą danych z wykorzystaniem standardu ODBC przedstawiono na rysunku Rys. 10.

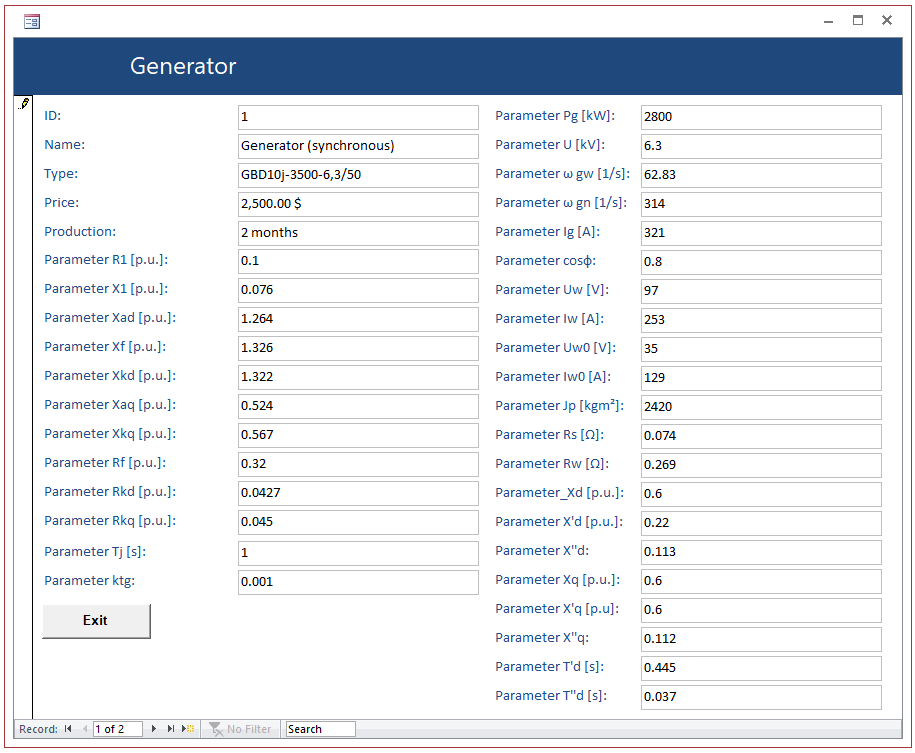


Rys. 10. Schemat komunikacji aplikacji z bazą danych z wykorzystaniem standardu ODBC [44]

Charakterystykę wybranych modeli elementów systemów sterowania z bazy danych (najważniejsze elementy elektroenergetyczne, wraz z opisem ich podstawowych parametrów elektrycznych) przedstawiono w tabeli Tab. 4.

Tab. 4. Wybrane elementy systemów sterowania z bazy danych (Microsoft Access Database) [23-27, 35, 37]

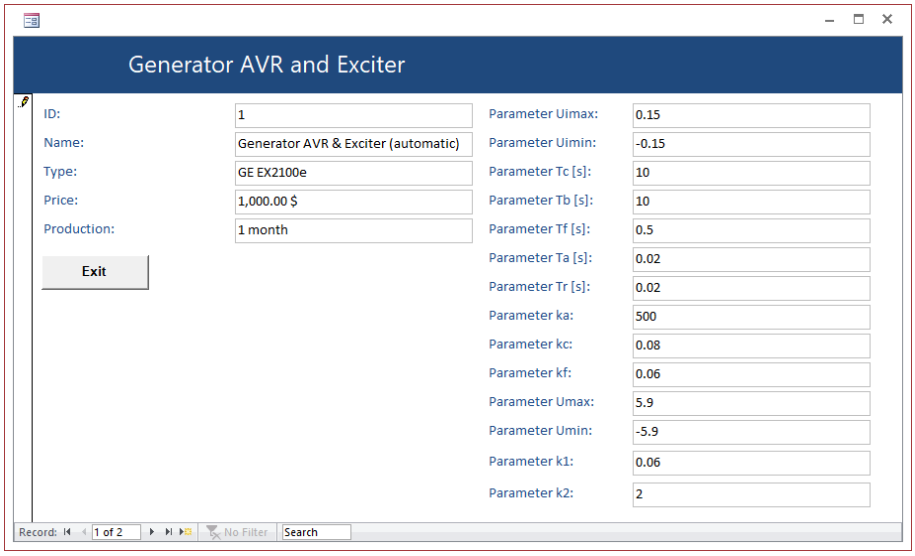
Formularz bazy danych umożliwiający edytowanie parametrów modelu generatora synchronicznego:



Najważniejsze parametry z bazy danych dla modelu generatora synchronicznego [23-27, 35, 37]:

Parametr I_g – Wartość skuteczna prądu generatora,
Parametr R_{kd} – Wartość rezystancji zastępczej obwodu tłumiącego w osi d generatora,
Parametr X_{kd} – Wartość reaktancji zastępczej obwodu tłumiącego w osi d generatora,
Parametr R_{kq} – Wartość rezystancji zastępczej obwodu tłumiącego w osi q generatora,
Parametr X_{kq} – Wartość reaktancji zastępczej obwodu tłumiącego w osi q generatora,
Parametr X_d – Wartość reaktancji synchronicznej w osi d generatora,
Parametr X_q – Wartość reaktancji synchronicznej w osi q generatora,
Parametr X_{ad} – Wartość reaktancji podłużnej twornika generatora,
Parametr X_{aq} – Wartość reaktancji poprzecznej twornika generatora.

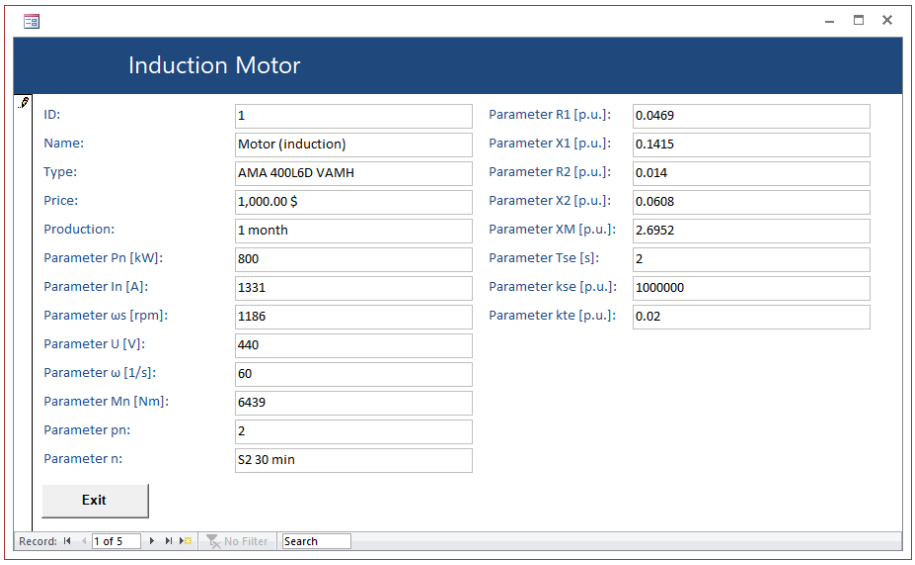
Formularz bazy danych umożliwiający edytowanie parametrów modelu układu AVR i wzbudzenia generatora:



Najważniejsze parametry z bazy danych dla modelu układu AVR i wzbudzenia generatora [23-27, 35, 37]:

Parametr $U_{i\,max}$ – Wartość maksymalnego uchybu sterowania napięciem,
Parametr $U_{i\,min}$ – Wartość minimalnego uchybu sterowania napięciem,
Parametr T_a – Wartość stałej czasowej regulatora napięcia,
Parametr T_b – Wartość stałej opóźnienia fazy sprzężenia zwrotnego (korekcja szeregowa),
Parametr T_c – Wartość stałej przyspieszenia fazy sprzężenia zwrotnego (korekcja szeregowa),
Parametr T_f – Wartość stałej czasowej członu sprzężenia zwrotnego,
Parametr T_r – Wartość stałej czasowej układu pomiarowego napięcia,
Parametr k_a – Wartość wzmocnienia członu regulatora napięcia,
Parametr k_f – Wartość wzmocnienia członu sprzężenia zwrotnego.

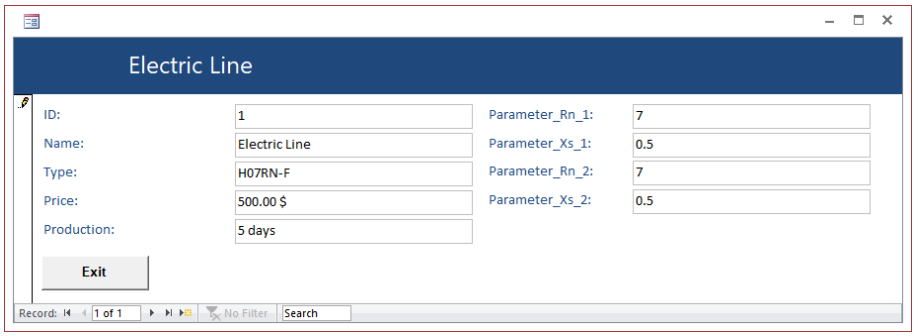
Formularz bazy danych umożliwiający edytowanie parametrów modelu silnika indukcyjnego:



Najważniejsze parametry z bazy danych dla modelu silnika indukcyjnego [23-27, 35, 37]:

Parametr I_n – Wartość skuteczna prądu obciążenia,
Parametr U – Wartość skuteczna napięcia zasilającego,
Parametr R_1 – Wartość rezystancji stojana,
Parametr X_1 – Wartość reaktancji stojana,
Parametr R_2 – Wartość rezystancji obwodu klatki,
Parametr X_2 – Wartość reaktancji sprzężenia wirnika i obwodu klatki,
Parametr M_2 – Wartość reaktancji rozproszenia,
Parametr T_{se}, k_{se}, k_{te} – parametry mechaniczne wału silnika.

Formularz bazy danych umożliwiający edytowanie parametrów modelu sieci elektroenergetycznej:



Najważniejsze parametry z bazy danych dla modelu sieci elektroenergetycznej [23-27, 35, 37]:

Parametr R_{n_1} – Wartość rezystancji linii elektrycznej,
Parametr X_{s_1} – Wartość reaktancji linii elektrycznej.

4.4. Agent ekspertowy/decyzyjny (Decision Agent)

Agent decyzyjny zaimplementowany został w celu oceny zgodności dynamiki otrzymanego rozwiązania projektowego z wymaganiami towarzystwa klasyfikacyjnego, wskazanymi przez użytkownika w wytycznych projektowych (okrętowe podsystemy elektroenergetyczne podlegają bowiem regulacjom ustanowionym przez towarzystwa klasyfikacyjne, które zapewniać mają bezpieczeństwo jednostek pływających). Agent decyzyjny na podstawie reguł zawartych w bazie wiedzy konfrontuje otrzymane charakterystyki z ograniczeniami dotyczącymi przykładowo dopuszczalnej wartości spadku napięcia na generatorze podczas załączenia obciążenia i na tej podstawie generuje raport z analizy zamodelowanego układu sterowania okrętowych podsystemów elektroenergetycznych. Agent decyzyjny dokonuje niezbyt złożonych operacji, a zatem do działania wymaga o wiele mniejszych zasobów obliczeniowych, porównując z agentem symulacyjnym. Decision agent przeprowadza statyczne obliczenia, które są mniej czasochłonne od dynamicznych symulacji, dlatego w omawianym systemie wieloagentowym stosunek liczby agentów decyzyjnych do symulacyjnych jest znacznie niższy (generalnie na jednego agenta decyzyjnego przypada kilku agentów symulacyjnych) [30, 35].

W przytoczonym przykładzie systemu ekspertowego do wspomagania projektowania podsystemu sterów strumieniowych [23-27], środowisko Exsys operowało na tych samych zmiennych przy zbieraniu danych od użytkownika, jak również podczas późniejszych operacjach logicznych (wnioskowanie). O ile w podejściu scentralizowanym było to możliwe, ponieważ w momencie inicjalizacji zmiennych alokowany był obszar pamięci jednej i tej samej maszyny fizycznej (na którym działało środowisko Exsys), tak w przypadku systemu wieloagentowego niezbędna jest ponowna inicjalizacja zmiennych procesowych na zdalnej maszynie (agent decyzyjny operuje zazwyczaj na innym komputerze, niż agent symulacyjny). Reinicjalizacja zmiennych odbywa się na podstawie rozszerzonej tabeli danych otrzymanej od agenta symulacyjnego. Podobne operacje wykonuje też agent symulacyjny, jednak jest to tylko kilka wybranych zmiennych, a oprócz tego ma on do dyspozycji środowisko Matlab, które pozwala w prosty sposób operować na zmiennych macierzowych. W przypadku agenta decyzyjnego, zaimplementowanego w technologii .NET Framework w języku C# wymagane jest operowanie jednocześnie na wszystkich zmiennych, które otrzymywane zostały bezpośrednio od innych agentów. Dla porównania wczytanie przez agenta symulacyjnego (Matlab) oraz decyzyjnego (.NET C#) zmiennej określającej ilość generatorów zainstalowanych na statku uzyskuje się za pomocą instrukcji:

```
//Simulation Agent (Matlab)
structure = dlmread('Parameters.frm', '|', 3, 2);
GENERATOR_AMOUNT = structure(32:32);
```

```
//Decision Agent (.NET C#)
protected override void HandleMessage(Message message)
{
    string[] content = message.content.Split('\n');
    if (message.sender.agentName.Contains("simulation"))
    {
        int GENERATOR_AMOUNT = 0;
        foreach (string line in content)
        {
            if (line.Contains("GENERATOR_AMOUNT"))
                GENERATOR = Convert.ToInt16(line.Substring(22, 15).Trim());
        }
    }
}
```

Agent decyzyjny dzieli otrzymaną tabelę danych na pojedyncze linie zawierające znaki w kodzie ASCII. Następnie sprawdza on, czy konkretna linia zawiera oczekiwaną zmienną. Jeżeli tak, wówczas z ciągu znaków odczytywane jest pole, zawierające wartość tej zmiennej. W dalszej kolejności z podciągu usuwane są białe znaki z wykorzystaniem metody Trim() i następuje konwersja otrzymanej wartości na zmienną całkowitoliczbową. Pozostałe zadania agenta decyzyjnego poza procesem odtwarzania zmiennych na zdalnej maszynie były stosunkowo proste do zaimplementowania dzięki wbudowanym funkcją środowiska .NET Framework. Kolejne podrozdziały opisują reguły wnioskowania oraz proces generowania raportu (projektu ofertowego) przez agenta decyzyjnego.

4.4.1. Baza wiedzy i reguły wnioskowania

Agent decyzyjny podczas oceny korzysta z informacji zawartych w bazie wiedzy. Baza wiedzy jest to zbiór danych zebranych od ekspertów w danych dziedzinach, przechowywanych w postaci cyfrowej. Sformułowanie reguł logicznych operujących na bazie wiedzy pozwala na utworzenie prostego systemu ekspertowego. W przypadku systemu wieloagentowego reguły te zapisane są w postaci IF...THEN...ELSE (najbardziej typowy wzorzec zaczerpnięty z systemów ekspertowych). Akwizycja danych do bazy wiedzy i formułowanie reguł wnioskowania jest to proces długotrwały, więc w tym przypadku skorzystano z dostępnych rozwiązań [23-27]. W zaimplementowanym systemie występują dokładnie 72 reguły wnioskowania, z czego 12 jest bardziej rozbudowanych poprzez wielopoziomowe zastosowanie instrukcji ELSE. Poniżej przedstawiono przykładowe reguły:

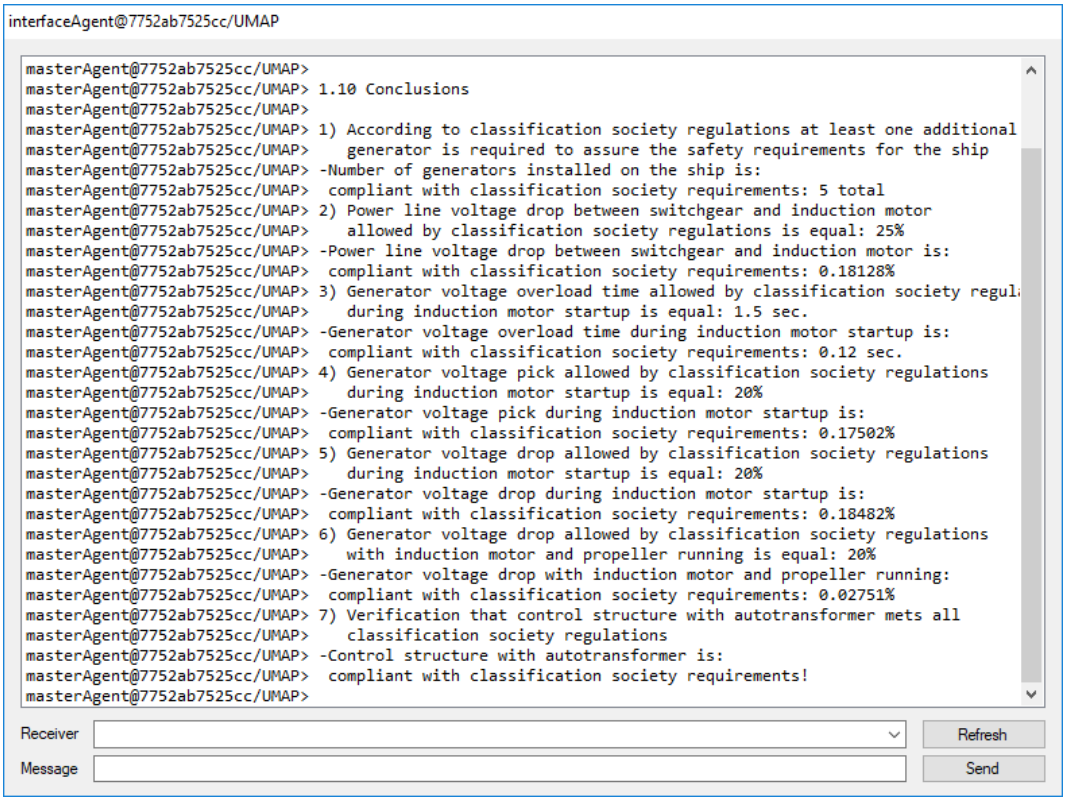
```
//Rule for induction motor selection
if (THRUST_SIMULATED < 57)
{
    THRUST = 57;
    FREQUENCY = 60;
    MOTOR_P_MAX = 430;
    MOTOR_TYPE = "CPT 1,25";
}
```

```
//Rule for control structure selection
if (GENERATOR == 1 && DIESEL == 1 && AUTOTRANSFORMER == 1 && MOTOR == 1 && VALVE == 1 &&
PROPELLER == 1)
{
    STRUCTURE = "Structure with autotransformer";
}
```

```
//Rule for classification society compliance verification
INFERENCE += "\t 5) Generator voltage drop allowed by classification society regulations
during induction motor startup is equal: " + GENERATOR_U_DROP_% + "%\n";
                if (GENERATOR_U_DROP < GENERATOR_U_DROP_% * GENERATOR_U)
                {
                    INFERENCE += "\t Generator voltage drop during induction motor startup
is: compliant with classification society requirements: " + GENERATOR_U_DROP +
"%\n";
                }
                else
                {
                    INFERENCE += "\t Generator voltage drop during induction motor startup
is: not compliant with classification society requirements. Please readjust AVR
parameters\n";
                }
```

4.4.2. Generowanie projektu ofertowego

Agent decyzyjny generuje raport, będący jednocześnie gotowym projektem ofertowym odnoszącym się do okrętowych podsystemów elektroenergetycznych. Raport ten oprócz oceny dynamiki utworzonego systemu sterowania zawiera także informacje na temat jego struktury, elementów, wyniki symulacji oraz wytyczne dla użytkownika systemu wieloagentowego (np. propozycje zmian nastaw regulatorów lub modyfikacja struktury układu w celu poprawy trajektorii sterowania, tak aby spełnione zostały wyznaczone kryteria towarzystwa klasyfikacyjnego). W raporcie dostarczane są również informacje bardziej ogólne, takie jak np. charakterystyki komponentów systemu sterowania pobranych od użytkownika lub z danych katalogowych producentów. Następnie gotowy raport odsyłany jest do agenta nadrzędnego, który przekazuje go dalej do odpowiedniego agenta interfejsu, aby właściwy użytkownik mógł zapoznać się z wynikami analizy dokonanej przez system wieloagentowy. Raport, podobnie jak pozostałe elementy systemu, zrealizowano w języku angielskim, aby umożliwić potencjalne wdrożenie aplikacji w przemyśle międzynarodowym [30, 35, 37]. Fragment przykładowego projektu ofertowego wygenerowanego przez system wieloagentowy przedstawiono na rysunku Rys. 11.



Rys. 11. Fragment przykładowego projektu ofertowego wygenerowanego przez system wieloagentowy [37]

4.5. Agent nadrzędny (Master Agent)

Agent nadrzędny odpowiedzialny jest za nadzorowanie działania systemu wieloagentowego oraz przydzielanie procesów projektowych pozostałym agentom. Pośredniczy on w wymianie informacji pomiędzy agentami interfejsu oraz agentami obliczeniowymi w celu zwiększenia wydajności systemu wieloagentowego. Master otrzymuje informacje (tabele danych) od wszystkich agentów interfejsu, po czym inicjalizuje algorytmy uczenia maszynowego, dokonuje priorytetyzacji zadań dzięki zaimplementowanemu mechanizmowi szeregowania procesów oraz przydziela zadania konkretnym agentom symulacyjnym [30, 35].

Jeśli agent nadrzędny posiada dużą ilość zakolejkowanych wytycznych projektowych, a agenci obliczeniowi są niedostępni (sytuacja nadmiernego obciążenia systemu), wówczas Master powołać może kolejną instancję agenta symulacyjnego w celu przywrócenia pożądanej wydajności systemu (tylko jeżeli dostępne są wymagane zasoby sprzętowe). Dzięki temu nie jest konieczne stosowanie mechanizmu wywłaszczania zasobów systemowych, ponieważ żądania przetwarzane są równolegle przez kilku agentów obliczeniowych. Master posiada również możliwość zredukowania liczby agentów w systemie, gdy do danego agenta nie przyszło żądanie wykonania zadania przez odpowiednio długi czas [30, 35].

Potrzeba implementacji agenta nadrzędnego została dostrzeżona podczas analizy działania systemu wieloagentowego, w którym operowała znaczna liczba agentów interfejsu, symulacyjnych oraz decyzyjnych (scenariusz testowy zakładał 100 uruchomionych instancji agentów danego typu). Przy tak zdefiniowanym wielkoskalowym systemie zdarzały się sytuacje, które odbijały się niekorzystnie na efektywności jego działania. Agent nadrzędny posiada zaimplementowane algorytmy szeregowania procesów oraz uczenia maszynowego, które rozwiązują wspomniane problemy (ich opis został przedstawiony w rozdziale 5) [30, 35, 45].

4.6. Podsumowanie

Ze względu na złożoność procesu projektowania układów sterowania okrętowych podsystemów elektroenergetycznych, dokonano dekompozycji poszczególnych zadań projektowych pomiędzy cztery podstawowe typy agentów działających w systemie. Agent interfejsu ma na celu umożliwienie interakcji użytkowników z systemem wieloagentowym. Agent symulacyjny odpowiada za badania symulacyjne, które pozwalają na określenie charakterystyki dynamicznej utworzonych struktur układów sterowania. Agent decyzyjny zaimplementowany został w celu oceny zgodności dynamiki otrzymanego rozwiązania projektowego z wymaganiami towarzystwa klasyfikacyjnego. Agent nadrzędny odpowiedzialny jest za nadzorowanie działania systemu wieloagentowego, jak również wykonywanie algorytmów szeregowania procesów oraz uczenia maszynowego. Rozdział 5 zawiera szczegóły dotyczące zaimplementowanych mechanizmów.

5. ALGORYTMY SZEREGOWANIA PROCESÓW I UCZENIA MASZYNOWEGO

5.1. Wprowadzenie

Algorytmy szeregowania procesów i uczenia maszynowego mają na celu zwiększenie wydajności systemu wieloagentowego podczas realizacji zadań projektowych. W podrozdziale 4.5 wspomniane zostały scenariusze testowe, w których system wieloagentowy działał w sposób mało efektywny. Pierwszym z nich był niewłaściwy rozkład obciążenia na dostępne zasoby sprzętowe podczas obliczeń symulacyjnych (nadmierne obciążanie danej jednostki, podczas gdy inne nie miały przydzielonego zadania). Drugim, nieco rzadziej występującym scenariuszem było ponowne prowadzenie obliczeń symulacyjnych dla wcześniej przetworzonych zapytań (wytyczne projektowe od kilku agentów były takie same, jednak proces budowy modelu systemu sterowania i obliczenia symulacyjne dla tej struktury były prowadzone od początku). Zaimplementowane algorytmy mają na celu rozwiązanie obu problemów [35, 45].

5.2. Algorytm szeregowania procesów

Zaobserwowano, że metoda odpowiedzialna za listowanie dostępnych agentów platformy UMAP opiera się głównie na kolejności ich rejestracji w DF (Dictionary Facilitator). Z tego powodu kilka procesów obliczeniowych może zostać przydzielonych do pojedynczego agenta symulacyjnego (który kilkukrotnie znalazł się pierwszy na liście), podczas gdy inni pozostają w stanie bezczynności. Z tego względu w systemie wieloagentowym wspomagającym projektowanie układów sterowania dodany został mechanizm szeregowania procesów (symulacji). Szeregowanie procesów (alternatywnie nazywane planowaniem, ang. Scheduling) jest to strategia udostępniania zasobów, takich jak czas procesora, obszar pamięci operacyjnej dla określonego procesu [45].

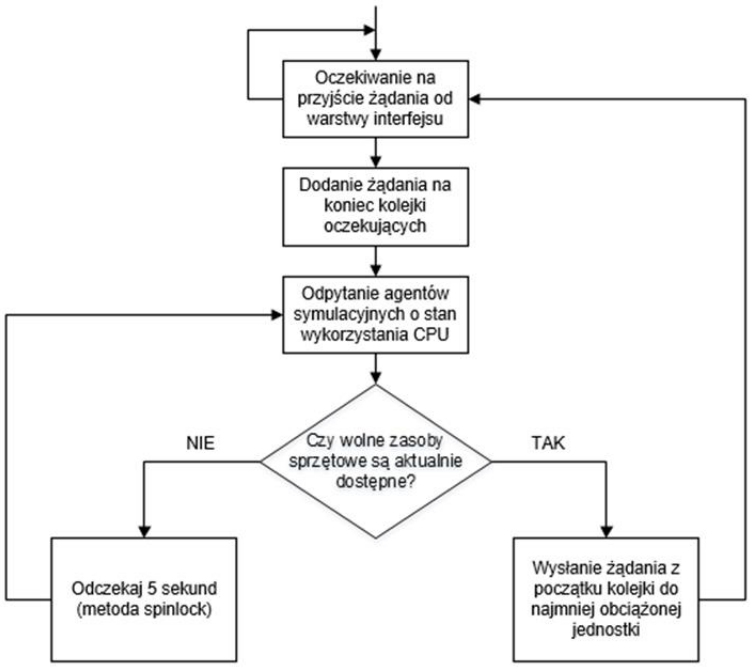
Znanych jest wiele algorytmów, które odpowiadają za schemat przypisywania procesów do zasobów sprzętowych. Najprostszym z nich jest FCFS (ang. First-Come, First-Served), który polega na wykonywaniu procesów w takiej kolejności, jak zostały przysłane do jednostki obliczeniowej. Kolejnym mechanizmem jest SJN (ang. Shortest-Job-Next), czyli uruchamianie w pierwszej kolejności procesów, które mają najkrótszy czas wykonywania. Ostatnim z podstawowych algorytmów szeregowania jest planowanie rotacyjne (ang. Round Robin). W przeciwieństwie do wcześniej przedstawionych technik używany jest tutaj mechanizm wywłaszczania, w wyniku czego każdy proces cyklicznie otrzymuje określony kwant czasu dostępu do procesora, co z kolei pozwala na quasi-równoległe przetwarzanie wszystkich procesów [46].

Szeregowanie procesów w systemach wieloagentowych jest bardziej skomplikowane ze względu na rozproszenie systemu oraz liczbę jednostek biorących udział w procesie planowania. Wyróżnia się trzy podstawowe kryteria podziału tego zagadnienia, biorąc pod uwagę samodzielność, komunikację i kooperację agentów w procesie szeregowania zadań [47]. Aby uprościć schemat przydzielania zadań agentom symulacyjnym zastosowano scentralizowane podejście szeregowania procesów. Za wykonywanie zaimplementowanego algorytmu odpowiedzialny jest agent Master, który operuje w warstwie zarządzającej, znajdującej się pomiędzy warstwą interfejsu i warstwą symulacyjną. W momencie, kiedy od warstwy interfejsu odbierane są kolejne żądania wymagające obliczeń

symulacyjnych, wówczas Master szereguje je i przydziela w taki sposób, aby czas przetworzenia wszystkich zapytań był jak najkrótszy. Zostało to osiągnięte poprzez odpytywanie agentów symulacyjnych o poziom wykorzystania procesora w danej chwili i przydzielenie zadania jednostce najmniej obciążonej [45]. Podstawowy opis działania algorytmu szeregowania procesów (symulacji) w systemie wieloagentowym znajduje się w sekcji Alg. 1., natomiast jego schemat blokowy przedstawiono na rysunku Rys. 12.

Alg. 1. Podstawowy opis działania algorytmu szeregowania procesów (symulacji) w systemie wieloagentowym [45]

```
1) Odbierz żądanie od agenta interfejsu.
2) Dodaj żądanie na koniec kolejki oczekujących.
3) Odpytaj agentów symulacyjnych o procentowy poziom wykorzystania procesora w danej chwili
   z zastosowaniem instrukcji:
   c:\> wmic cpu get loadpercentage
4) Jeżeli najmniejszy procentowy wskaźnik wykorzystania CPU wszystkich jednostek jest
   większy niż 90% (brak dostępnych zasobów), wówczas odczekaj 5 sekund i idź do punktu 3.
5) Wyślij żądanie znajdujące się na początku kolejki do najmniej obciążonej jednostki
   (metoda FIFO), następnie idź do punktu 1.
```



Rys. 12. Schemat blokowy algorytmu szeregowania procesów (symulacji) w systemie wieloagentowym [45]

Opracowany algorytm stanowi alternatywę do powszechnie znanych algorytmów szeregowania. Zastosowanie mechanizmu szeregowania procesów przyczyniło się do zwiększenia wydajności systemu wieloagentowego (wyniki testów porównujących wydajność systemu wieloagentowego z aktywnym i nieaktywnym mechanizmem szeregowania procesów przedstawiono w rozdziale 7).

5.3. Algorytm uczenia maszynowego

Uczenie maszynowe (ang. Machine learning) jest zagadnieniem, które dotyczy m.in. tworzenia systemów doskonalących swoje działanie na podstawie doświadczeń z przeszłości. Uczenie maszynowe jest poddziedziną inteligencji obliczeniowej, która wywodzi się z badań nad komputerowym

rozpoznawaniem wzorców [48]. W systemach wieloagentowych aspekt uczenia maszynowego jest bardziej skomplikowany ze względu na ich rozproszoną architekturę oraz mnogość jednostek (agentów) biorących udział w procesie uczenia. W literaturze spotkać można różne klasyfikacje uczenia maszynowego, biorąc pod uwagę kryteria i kierunki badań z ostatnich lat. Na uwagę zasługuje klasyfikacja sposobu uczenia w systemach wieloagentowych ze względu na stopień rozproszenia (uczenie może przebiegać w sposób scentralizowany lub rozproszony) [49]. Ważnym aspektem jest również liczba agentów odnoszących korzyść z procesu uczenia (korzyść może osiągnąć tylko jeden agent lub kilka agentów tworzących pewną współpracującą ze sobą grupę) [49].

Kolejnym kryterium klasyfikacji jest sposób zachowania agentów w procesie uczenia. Agenci mogą kooperować ze sobą podczas nauki lub zdobywać wiedzę samodzielnie [50]. Uczenie kooperacyjne dzieli się na dwie główne podkategorie: uczenie zespołowe i uczenie przez współzawodnictwo [51]. W uczeniu zespołowym występuje przeważnie co najmniej jeden agent, który pełni rolę mentora, podnoszącego potencjał całego zespołu. W uczeniu poprzez współzawodnictwo każdy agent wykonuje algorytmy uczenia maszynowego w celu udoskonalenia tylko własnych możliwości. Niemniej jednak oba typy uczenia kooperacyjnego przyczyniają się do podniesienia użyteczność grupy agentów jako całości. Uczenie w systemach wieloagentowych klasyfikować można również według kryterium określającego, czy agenci w systemie są świadomi o własnym (lub zespołowym) procesie uczenia. Ważna jest również jednorodność (lub różnorodność) metod i algorytmów uczenia maszynowego w systemie wieloagentowym [52]. Omawiane zagadnienie podzielić można też ze względu na to, czy w procesie uczenia wykorzystywane były modele. Użycie modeli może dać agentowi wiedze „a priori" o systemie, w którym będzie działać, jak również o dynamice pozostałych agentów [53]. W ostatnim czasie rozpatrywane były także aspekty uczenia interaktywnego i uczenia on-line w zaszumionym, często zmieniającym się środowisku [54, 55].

Wiele kompleksowych przeglądów naukowych dotyczących uczenia maszynowego zostało wydanych w ciągu ostatnich dwóch dekad [51, 56, 57]. Zawierają one informacje dotyczące zastosowania metod uczenia w systemach wieloagentowych. Wyróżnić można trzy podstawowe typy uczenia: uczenie z nauczycielem, uczenie bez nauczyciela i uczenie ze wzmacnianiem. Uczenie z nauczycielem jest najchętniej stosowane i polega na generalizowaniu problemu oraz budowaniu bardziej ogólnej hipotezy na podstawie danych trenujących [58]. Wypracowana hipoteza podlega ocenie zewnętrznego źródła, czyli nauczyciela, który posiada wiedzę na temat rozwiązywanego problemu klasyfikacji. W tym rodzaju uczenia występuje sprzężenie zwrotne, dzięki czemu cel jest znany, a jego efekty mierzalne. Efektem uczenia z nauczycielem jest poprawna klasyfikacja przyszłych danych wejściowych na podstawie wcześniej wyuczonych wzorców. Uczenie z nauczycielem stosowane jest do sprawnego wyszukiwania zależności pomiędzy danymi wejściowymi oraz wyjściowymi. Dzięki temu poprawia ono wydajność systemów wieloagentowych podczas obsługi żądań poprzez samoczynną adaptację agentów [59].

Poza środowiskiem akademickim algorytmy uczenia maszynowego rozwijane są w ośrodkach badawczych firm głównie z branży informatycznej. Dzięki rozwojowi tej dziedziny wiedzy stała się możliwa budowa między innymi autonomicznych pojazdów, które poruszają się w ruchu ulicznym bez (lub z ograniczonym udziałem) kierowcy, interfejsów sterowanych głosem naturalnym oraz aplikacji do

rozpoznawania obrazu [56]. Co więcej, w niektórych zastosowaniach programy, które posiadają umiejętność uczenia przewyższają zdolnościami ludzi, ekspertów w danej dziedzinie. Przykładem może być przegrana wielokrotnego mistrza w grę „GO" ze sztuczną inteligencją stworzoną przez Google [60]. Możliwości implementacji algorytmów oraz metod uczenia maszynowego jest bardzo wiele. Niektóre z nich bazują na obserwacjach natury (np. sztuczne sieci neuronowe, uczenie oparte o algorytmy ewolucyjne), inne na teorii wspomagania decyzji (np. drzewa decyzyjne, algorytm random forests), jeszcze inne na równaniach probabilistycznych (np. sieci Bayesa, klasyfikator Bayesa). W zależności od potrzeb stosowane są różne rozwiązania, niemniej jednak bardzo często należące do tych trzech głównych grup [35].

Sztuczne sieci neuronowe (ang. Artificial neural networks) są to narzędzia obliczeniowe przeznaczone do modelowania i rozwiązywania wielkoskalowych, złożonych problemów [61]. Metoda ta bazuje na założeniach występujących w sieci połączeń komórek nerwowych występujących w mózgu istot żywych. Podejście to wykorzystuje mechanizmy znane z biologicznej sieci neuronowej do rozwiązywania zadań klasyfikacji, ale nie naśladuje bezpośrednio operacji zachodzących pomiędzy neuronami. Analogicznie jednak tworzone są połączenia między sztucznymi neuronami w poszczególnych warstwach. Co więcej, każde takie połączenie posiada odpowiednią wagę (porównywalnie z synapsami). Dodatkowo próg aktywacji sztucznego neuronu odpowiada tranzycji biologicznego ciała komórki nerwowej do stanu aktywnego. Sztuczne sieci neuronowe składają się zazwyczaj z kilku warstw neuronów: warstwy wejściowej, warstwy wyjściowej oraz opcjonalnie warstwy ukrytej (która umożliwia głębokie uczenie, ang. Deep learning). Sztuczne sieci neuronowe posiadają wiele zalet, wśród których można wyróżnić paralelizm obliczeniowy, adaptowalność oraz nieliniowość pozwalającą lepiej estymować rzeczywiste zagadnienia. Ostatnie badania nad zastosowaniem sztucznych sieci neuronowych w systemach wieloagentowych skupiają się na rozwiązywaniu problemu konsensusu i aproksymacji nieliniowej dynamiki w warunkach niepewności [62, 63].

Sieci Bayesa (ang. Bayesian networks) są probabilistycznymi reprezentacjami zbioru zmiennych losowych i ich rozkładu prawdopodobieństwa przedstawionymi za pomocą skierowanego grafu acyklicznego (ang. Directed acyclic graph) [64]. Sieci Bayesa w czytelny sposób prezentują rozkład prawdopodobieństwa w każdym węźle grafu. Zmienne w tych węzłach są niezależne od ich następników (wartości w niższej gałęzi grafu). W przeciwieństwie do drzew decyzyjnych nie jest możliwe znalezienie ścieżki od jednej zmiennej do drugiej, ponieważ węzły nie są ze sobą bezpośrednio połączone. Z sieciami Bayesa powiązany jest bardzo popularny w uczeniu maszynowym klasyfikator Bayesa (ang. Naive Bayes classifier). Klasyfikator ten jest specjalnym rodzajem sieci Bayesowskiej zorientowanej na niezależność, w której poszczególne zmienne nie mają swoich poprzedników (wartości w wyższej gałęzi grafu). Główną zaletą sieci Bayesa jest to, że są one bezpośrednimi reprezentacjami zależności modelowanego problemu, a nie modelem procesu wnioskowania [65]. Zastosowania tych sieci obejmują wspomaganie decyzji w warunkach niepewności, modelowanie systemów, kontrola i planowanie produkcji. Badania nad rozproszonym uczeniem wieloagentowym opartym na sieciach Bayesa zostały niedawno opublikowane w [66].

Sztuczne sieci neuronowe i sieci Bayesa wymagają wstępnego treningu na znacznej ilości danych uczących, aby możliwa była poprawna klasyfikacja wzorców [61]. Z tego powodu w omawianym

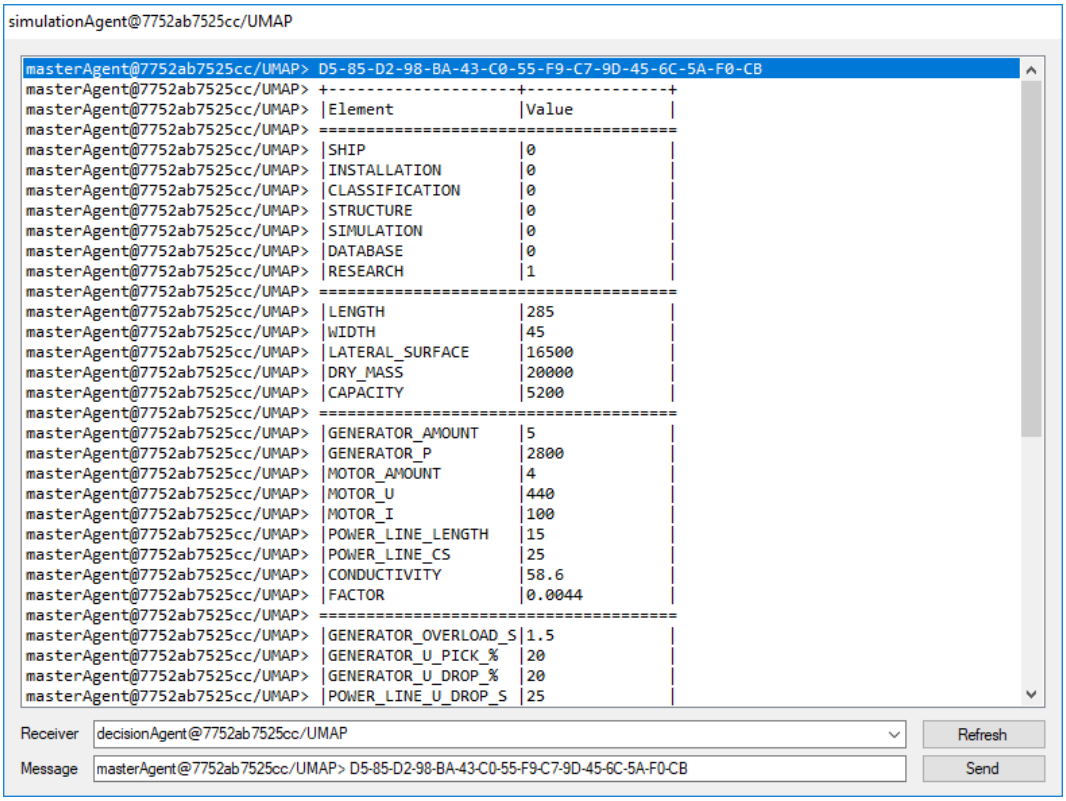
przypadku obydwa algorytmy uczenia byłyby nieaktywne przez długi okres od uruchomienia systemu wieloagentowego (do czasu zebrania odpowiedniej ilości danych uczących). Dodatkowo nie jest możliwe w prosty sposób przeprowadzenie ekstrakcji danych (wiedzy) z wcześniej wyuczonej sieci [67]. Pewne badania wskazują również, że sieci Bayesa są mało wydajne, kiedy sieć została wytrenowana w standardowy sposób oraz koszt obliczeniowy treningu sieci Bayesa jest relatywnie wysoki [64]. Kolejnym wymaganiem, którego nie są w stanie spełnić zarówno sztuczne sieci neuronowe i sieci Bayesa jest możliwość bezpośredniego operowania na typie danych wejściowych używanym domyślnie w systemie wieloagentowym, czyli na łańcuchach znaków. Oba wspomniane algorytmy przyjmują liczby zmiennoprzecinkowe jako wartości wejściowe, tymczasem agenci w systemie przesyłają informacje w postaci znormalizowanych komunikatów tekstowych w języku FIPA-ACL. Dlatego też dostosowanie tych algorytmów do pracy w systemie wieloagentowym wymaga użycia zaawansowanych technik, takich jak „bag-of-words model" [35].

Podstawowymi wymaganiami implementacji algorytmu uczenia maszynowego w systemie wieloagentowym była efektywność, prostota działania (która przekładała się na niski koszt obliczeniowy) oraz stabilność i niezawodność. Dodatkowo algorytm musiał być w pełni autonomiczny (czyli nie potrzebować wcześniejszego treningu na danych uczących) oraz nie wymagać ingerencji użytkowników systemu wieloagentowego w proces uczenia (użytkownicy mogą nie mieć dostępu do komputera, na którym operuje agent nadrzędny i gdzie odbywa się proces uczenia). Algorytm uczenia maszynowego dedykowany dla systemów wieloagetowych powinien także domyślnie operować na łańcuchach znaków oraz umożliwiać uczenie on-line, w którym nowa wiedza gromadzona jest w czasie rzeczywistym podczas działania systemu. Zaimplementowany algorytm musiał także w znaczący sposób przyczyniać się do zwiększenia wydajności działania systemu wieloagentowego oraz wykazywać większą efektywność obliczeniową podczas ekstrakcji danych porównując z rozwiązaniami znanymi z literatury (powinien szybciej rozpoznawać wzorce, niż robią to sztuczne sieci neuronowe i sieci Bayesa) [35].

Biorąc pod uwagę wspomniane ograniczenia dostępnych rozwiązań i przyjęte założenia, zaimplementowano nowy algorytm uczenia maszynowego oparty o tablice asocjacyjne, dedykowany dla systemów wieloagentowych oraz innych systemów rozproszonych. Dzięki temu algorytmowi agent nadrzędny przechowuje założenia projektowe, wraz z ich rozwiązaniami wygenerowanymi przez system wieloagentowy (uczenie skojarzeniowe). Celem uczenia jest wczytanie gotowego raportu z analizy systemu sterowania po rozpoznaniu identycznych wytycznych projektowych, które w przeszłości przetwarzane były przez system (bez konieczności budowy modelu i prowadzeniu badań symulacyjnych od początku). Dane historyczne przechowywane są w formie tablic asocjacyjnych w pliku *.dat tworzonym podczas inicjalizacji kolejnej instancji agenta typu Master [30, 35]. Tablica asocjacyjna jest to struktura danych, która przechowuje pary zmiennych: unikalny klucz oraz zdefiniowaną wartość, do której możliwy jest dostęp po podaniu klucza. Tablice asocjacyjne służą do przechowywania dużej ilości danych, do których wymagany jest bardzo szybki dostęp, dlatego znalazły one zastosowanie np. w serwerach baz danych, czy protokole Kademlia [35, 68].

Postać klucza w tablicy asocjacyjnej może być dowolna, rozpoczynając od liczb porządkowych, poprzez łańcuchy znaków, kończąc na tzw. hashach (najważniejsze jest, aby była to wartość unikalna). W zaimplementowanym algorytmie zastosowano sumę hashującą (wartość otrzymana po zastosowaniu

funkcji skrótu). Zdecydowano się na takie rozwiązanie ze względu na duży rozmiar danych wejściowych (łańcuch zawierający dane wejściowe składa się z ponad 3000 znaków, natomiast hash z niespełna 32). Do obliczenia klucza zastosowano popularny algorytm kryptograficzny MD5 (ang. Message-Digest Algorithm 5). Funkcja skrótu MD5 pozwala z ciągu danych o dowolnej długości wygenerować 128-bitowy skrót. Algorytm ten opracowany został w 1992 roku przez Rona Rivesta (współtwórcę RSA). Skróty 128-bitowe MD5 mają mniejszą złożoność w porównaniu np. z SHA-1 (160-bitowy), dzięki czemu ich obliczanie, serializacja, przesyłanie i zapisywanie w pliku jest nieco prostsze [30, 35]. Wytyczne projektowe (tabelę danych) oraz funkcję skrótu MD5 dla tego ciągu znaków (klucz) przedstawiono na rysunku Rys. 13.

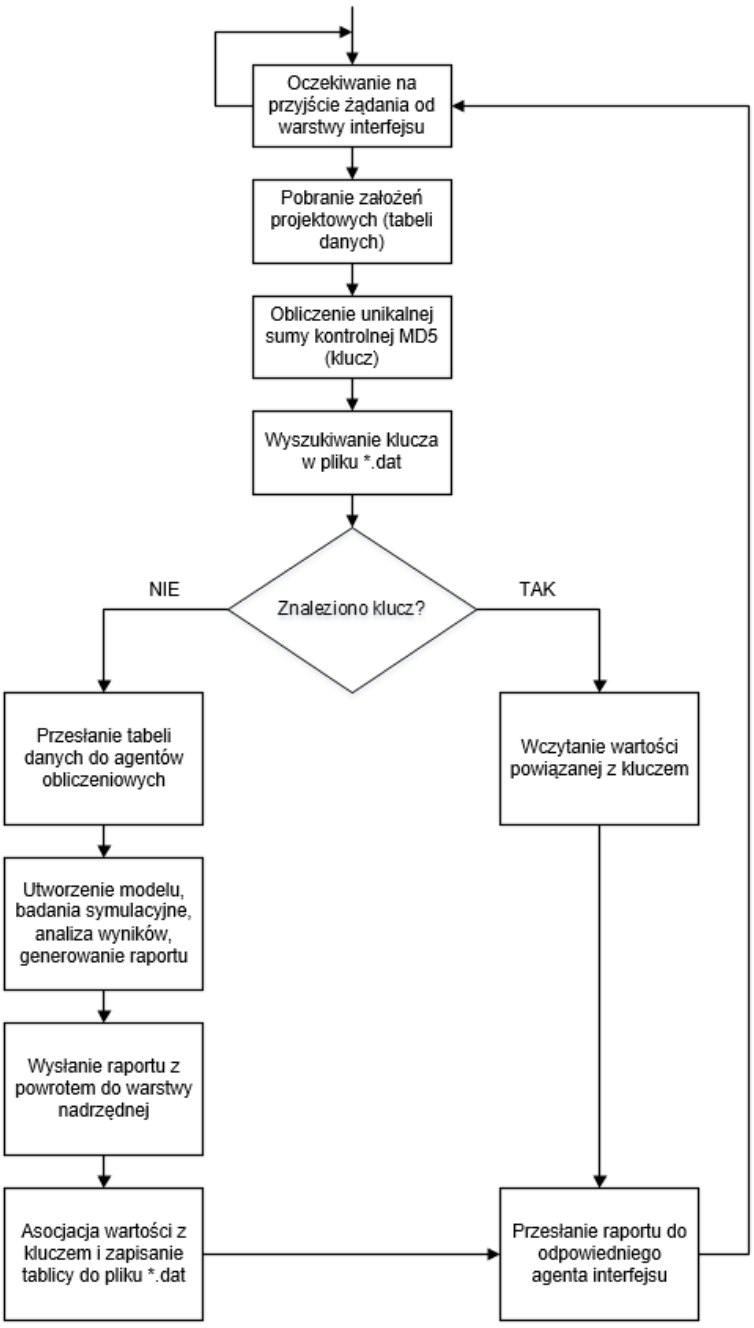


Rys. 13. Wytyczne projektowe (tabela danych) oraz funkcja skrótu MD5 dla tego ciągu znaków (klucz) [35]

Moduł programowy wykonujący algorytm uczenia maszynowego oparty o tablice asocjacyjne został umieszczony w strukturze systemu wieloagentowego w warstwie zarządzającej. Powodem takiego podejścia jest fakt, że przez warstwę zarządzającą przechodzą żądania zarówno od warstwy interfejsu zawierające wytyczne projektowe użytkowników systemu, jak i od warstw obliczeniowych (symulacyjnej i decyzyjnej) generujących rozwiązania projektowe. Dzięki temu możliwe jest również nadzorowanie modułu uczącego bezpośrednio przez agenta nadrzędnego, wyręczając użytkownika z tej konieczności. Plik modułu uczącego tworzony jest zawsze na tym samym komputerze, na którym uruchomiony zostaje agent Master. Oczywiście istnieje możliwość utworzenia tego pliku modułu uczącego w lokalizacji sieciowej, jednak podczas dostępu i tak zostanie on zbuforowany na maszynę lokalną. Dlatego takie rozwiązanie powodowałoby niepotrzebne opóźnienia podczas dostępu do pliku modułu uczącego przez agenta nadrzędnego [30, 35]. Podstawowy opis działania algorytmu uczenia maszynowego znajduje się w sekcji Alg. 2., natomiast jego schemat blokowy przedstawiono na rysunku Rys. 14.

Alg. 2. Podstawowy opis działania algorytmu uczenia maszynowego w systemie wieloagentowym [35]

```
1)  Oczekiwanie na przyjście żądania od warstwy interfejsu.
2)  Pobranie założeń projektowych (tabeli danych) w postaci łańcucha znaków kodu ASCII.
3)  Obliczenie unikalnej sumy kontrolnej MD5 (klucz) z pobranego ciągu znaków kodu ASCII.
4)  Przeszukanie pliku *.dat zawierającego dane historyczne w celu odnalezienia klucza.
5)  Weryfikacja, czy plik zawiera poszukiwaną sygnaturę klucza.
    * Klucz został znaleziony:
6)  Wczytanie wartości powiązanej z kluczem, będącej wcześniej wygenerowanym rozwiązaniem.
7)  Przesłanie raportu do odpowiedniego agenta interfejsu bez przeprowadzania symulacji.
    * Klucz nie został znaleziony:
6)  Przesłanie założeń projektowych do najmniej obciążonego agenta symulacyjnego, symulacje.
7)  Wygenerowanie raportu z analizy oraz wysłanie go z powrotem do warstwy zarządzającej.
8)  Asocjacja klucza (hash MD5 z tabeli danych) z wartością (raport z analizy systemu).
9)  Zapisanie utworzonej tablicy asocjacyjnej do pliku *.dat jako nowa linia.
10) Przesłanie raportu z analizy systemu do odpowiedniego agenta interfejsu.
```



Rys. 14. Schemat blokowy algorytmu uczenia maszynowego w systemie wieloagentowym [35]

Jak wcześniej wspomniano, na urządzeniu pamięci masowej komputera wraz z inicjalizacją każdej instancji agenta nadrzędnego tworzony jest nowy plik z rozszerzeniem *.dat w celu przechowywania tablic asocjacyjnych utworzonych w procesie uczenia. Plik *.dat tworzony jest w metodzie OnActivate() wywoływanej podczas uruchamiania agenta. Podczas deinicjalizacji agenta wywoływana jest natomiast metoda OnDeactivate() zawierająca kod odpowiedzialny za usunięcie pliku sprzężonego z daną instancją agenta nadrzędnego. Czas życia pliku jest więc zbieżny z okresem aktywności agenta Master. Na jednym komputerze może działać więcej, niż jeden agent zarządzający, dlatego aby uniknąć błędów wynikających ze współbieżności dostępu (np. gdy jedna instancja czyta plik *.dat, natomiast druga chce dokonać zapisu), każdy agent Master tworzy osobny plik [35]. Pliki zapisywane są w tej samej lokalizacji, więc aby można było je rozróżnić posiadają inne nazwy, będące automatycznie wygenerowanym GUID (ang. Globally Unique Identifier), na przykład:

```
8867ed64-0ecd-4587-af08-bf54523245d7.dat
```

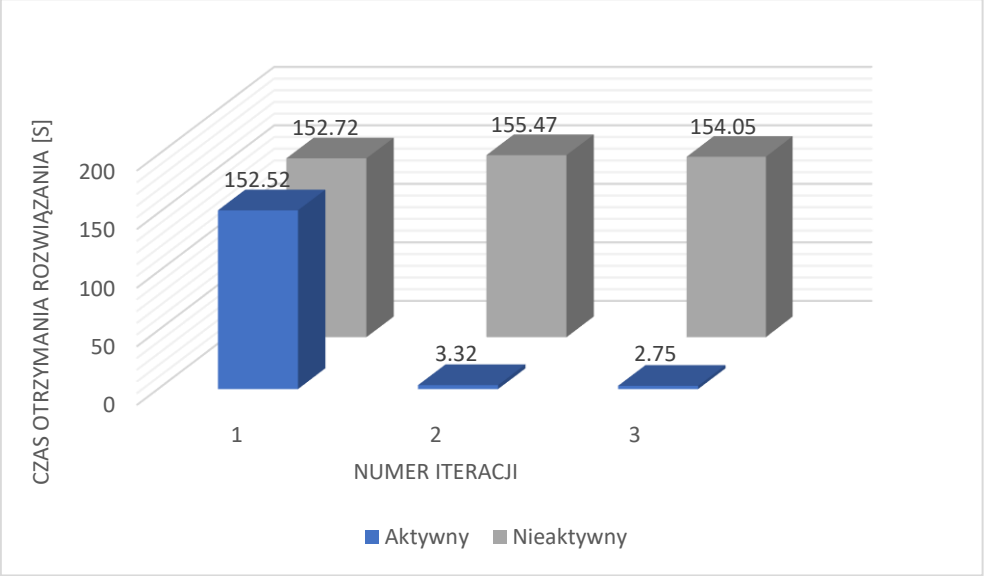
Wydajność algorytmu uczącego determinuje w dużej mierze czas przeszukiwania pliku *.dat zawierającego zapisane tablice asocjacyjne. Do tego celu użyta została klasa StreamReader z mscorlib.dll, która umożliwia odczytywanie znaków ze strumienia bitowego w określonym kodowaniu (standardowo jest to UTF-8). Przy deklarowaniu nowego obiektu klasy StreamReader następuje wskazanie pliku modułu uczącego jako docelowego strumienia (hashFile). W dalszej kolejności plik przeszukiwany jest linia po linii w celu znalezienia klucza (hashKey). Jeżeli w danej linii znajduje się szukany klucz, wówczas wczytana linia pliku dzielona jest na dwie części (klucz oraz wartość). Jako znak separatora użyta została nierozłączna spacja, która w kodzie ASCII oznaczona jest jako \u00A0. Następnie po dekonkatenacji formułowana jest wiadomość zwrotna dla agenta interfejsu. Jako zawartość wiadomości wysyłana jest tylko druga część dekonkatenowanego ciągu (raport). Na tym przeszukiwanie pliku zostaje zakończone i strumień jest zamykany [35]. Mechanizm poszukiwania klucza i tworzenia odpowiedzi uruchamiany jest po wywołaniu następujących instrukcji:

```
System.IO.StreamReader sr = new System.IO.StreamReader(hashFile)
while ((line = sr.ReadLine()) != null)
{
    if (line.Contains(hashMessage))
    {
        string[] report = line.Split('\u00A0');
        Message reply = message.CreateResponse();
        reply.performative = Performative.inform;
        reply.content = report[1].ToString();
        this.SendMessage(reply);
        sr.Close();
        return;
    }
    counter++;
}
sr.Close();
```

5.4. Wydajność algorytmu uczenia maszynowego

W celu weryfikacji działania zaproponowanego algorytmu uczenia maszynowego i jego wpływu na efektywność systemu wieloagentowego przeprowadzona została seria badań wydajnościowych. W pierwszym etapie analizy zmierzono czas wygenerowania rozwiązania projektowego przez system wieloagentowy, kiedy algorytm uczenia był nieaktywny. Następnie sprawdzono jak zmieni się wydajność systemu po włączeniu algorytmu. W tym celu wysłano trzy bezpośrednio następujące po sobie żądania

zawierające takie same założenia projektowe i zmierzono czas otrzymania odpowiedzi od warstwy nadrzędnej systemu wieloagentowego. W dalszej kolejności uaktywniono algorytm uczenia maszynowego i powtórzono identyczną procedurę pomiarową [35]. Porównanie wydajności systemu wieloagentowego (obejmujące czas potrzebny do otrzymania rozwiązania mierzony w sekundach) z aktywnym (bliższe kolumny na wykresie) i nieaktywnym (dalsze kolumny na wykresie) algorytmem uczenia maszynowego przedstawiono na rysunku Rys. 15.

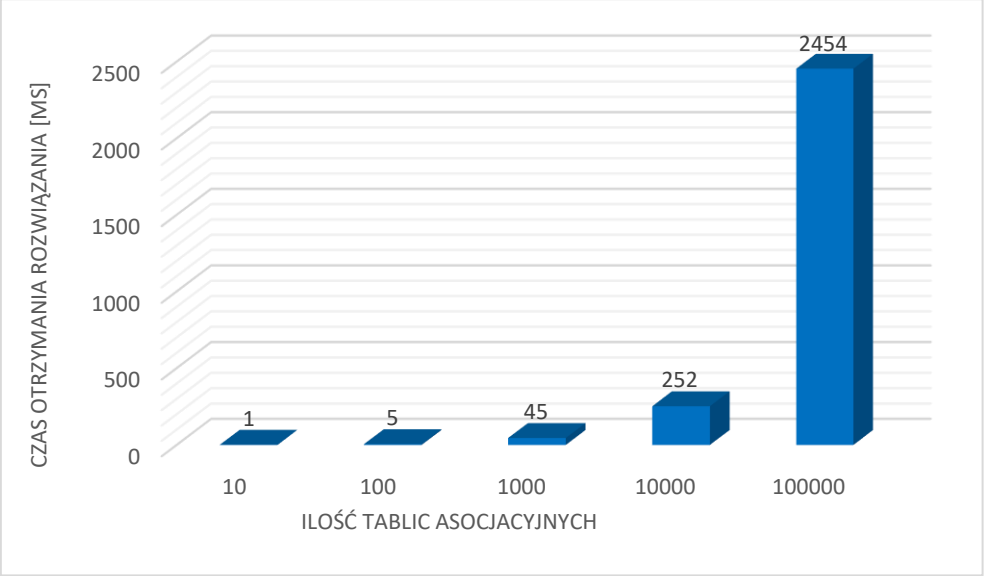


Rys. 15. Porównanie wydajności systemu wieloagentowego z aktywnym i nieaktywnym algorytmem uczenia maszynowego (niższy wynik – lepsza wydajność) [35]

Zaobserwowano, że czas przetwarzania pierwszej pary żądań był bardzo zbliżony zarówno dla przypadku kiedy algorytm uczenia maszynowego był aktywny, jak i nieaktywny. Wynika to z faktu, iż algorytm nie posiadał wówczas jeszcze danych historycznych, na których mógłby nastąpić proces uczenia (dane te zostały dostarczone właśnie w pierwszej iteracji). Dla drugiej oraz każdej kolejnej pary żądań zaobserwowano znaczący wzrost wydajności rozproszonego systemu wieloagentowego. Czas generowania raportu z analizy układu sterowania zmalał do niespełna kilku sekund niezbędnych do serializacji wiadomości do formatu XML oraz przesyłania ich pomiędzy komputerami, na których uruchomione były instancje agentów. Udowodniono doświadczalnie, że implementacja algorytmu uczenia maszynowego przekłada się na wzrost wydajności systemu wieloagentowego na poziomie odpowiednio: 4682,83%; 5564,72% (w kolejnych iteracjach nieujętych na wykresie było to odpowiednio 4964,74%; 6209,31%) [35].

Kolejny etap badań wydajności algorytmu uczenia maszynowego obejmował sprawdzenie czasu przeszukiwania określonej ilości tablic asocjacyjnych zapisanych w pliku modułu uczącego. Pomiar obejmował ilość czasu potrzebnego na wydobycie z pliku określonej wartości powiązanej z kluczem oraz przesłanie jej do agenta interfejsu. W celu zapewnienia identycznych warunków początkowych pomiaru, zbiory danych historycznych zostały sztucznie wygenerowane. W przykładzie zakładany był najgorszy scenariusz, w którym poszukiwana tablica asocjacyjna znajdowała się na końcu pliku, ponieważ kolejność zapisanych danych miała znaczenie podczas procesu przeszukiwania pliku *.dat. Istotny wpływ na wynik pomiaru miały również parametry pamięci masowej jednostki, na której działał

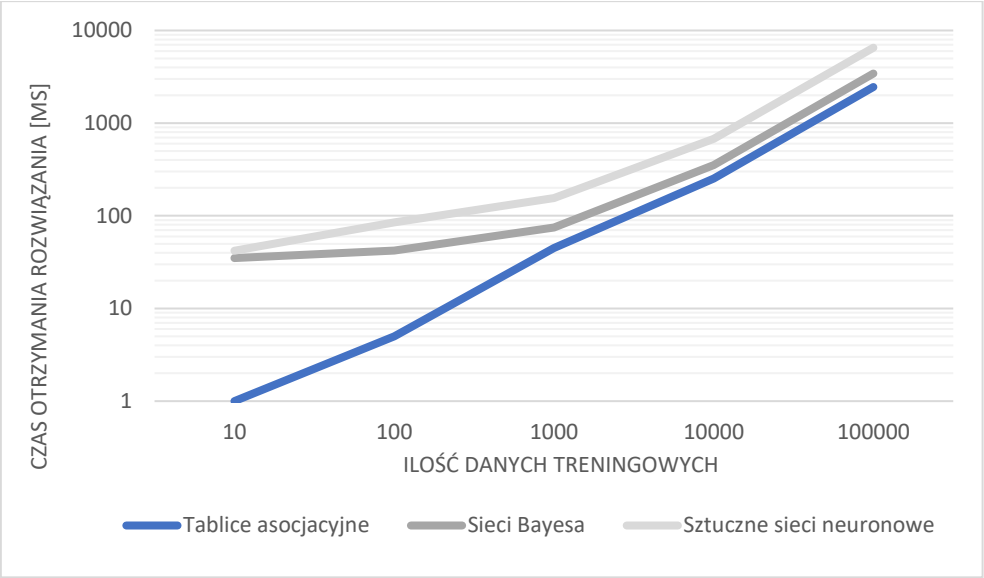
algorytm (Intel SSDSCKJF180A5) [35]. Porównanie wydajności algorytmu uczenia maszynowego (obejmujące czas potrzebny do otrzymania rozwiązania mierzony w milisekundach) podczas przeszukiwania pliku zawierającego zdefiniowaną ilość tablic asocjacyjnych przedstawiono na rysunku Rys. 16.



Rys. 16. Porównanie wydajności algorytmu uczenia maszynowego podczas przeszukiwania pliku zawierającego zdefiniowaną ilość tablic asocjacyjnych [35]

Finalny zestaw badań wydajnościowych obejmował porównanie zaproponowanego algorytmu uczenia maszynowego opartego o tablice asocjacyjne ze sztucznymi sieciami neuronowymi i sieciami Bayesa. Aby dodać obsługę tych algorytmów w module uczącym zastosowano gotowe rozwiązania. W przypadku sztucznych sieci neuronowych wykorzystano bibliotekę Encog3 [69], natomiast dla sieci Bayesa zmodyfikowano udostępniony na stronie magazynu firmy Microsoft przykład wykorzystania klasyfikatora Naive Bayes do rozwiązania podobnego problemu klasyfikacji [70].

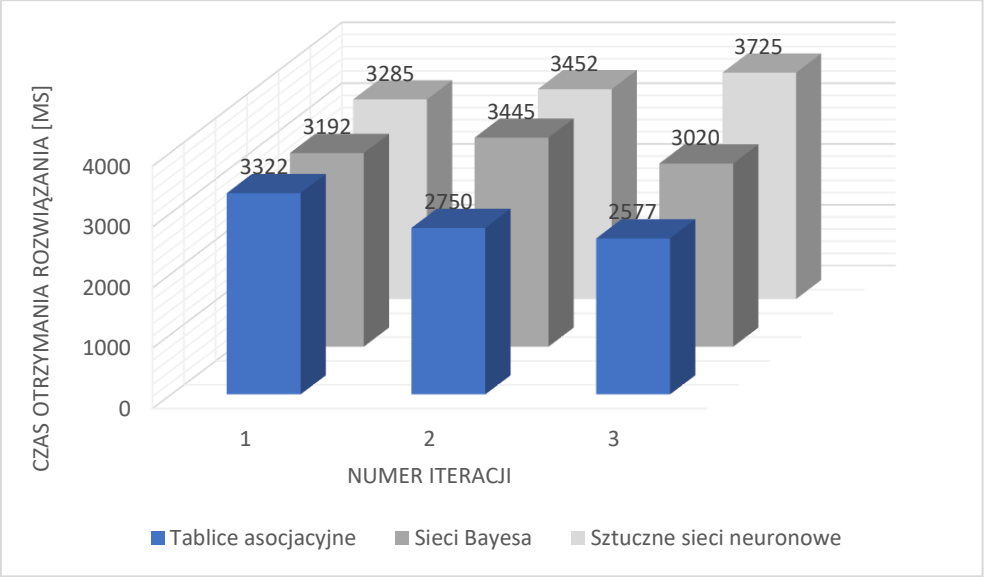
Domyślnie w obu przykładach dane uczące były losowo generowane po uruchomieniu aplikacji, dlatego kod źródłowy został zmodyfikowany w taki sposób, aby zestawy danych uczący pobierane były z pliku *.dat. Dodatkowo biblioteka Encog3 operowała na danych numerycznych, stąd też niezbędna była implementacja mechanizmu konwersji, która umożliwiła podanie łańcuchów znaków na wejście sieci neuronowej (w tym celu wykorzystano „bag-of-words model") [35]. Porównanie wydajności różnych algorytmów uczenia maszynowego (obejmujące czas potrzebny do otrzymania rozwiązania mierzony w milisekundach) podczas przeszukiwania pliku zawierającego zdefiniowaną ilość danych treningowych przedstawiono na rysunku Rys. 17.



Rys. 17. Porównanie wydajności różnych algorytmów uczenia maszynowego podczas przeszukiwania pliku zawierającego zdefiniowaną ilość danych treningowych (niższy wynik – lepsza wydajność) [35]

Wraz ze wzrostem ilości danych zaobserwowano rosnące dysproporcje w wydajności zastosowanych algorytmów uczenia maszynowego. Sztuczne sieci neuronowe okazały się najmniej wydajnym algorytmem w przeprowadzonym porównaniu. Sieci bayesowskie były bardziej wydajne, jednak przez konieczność wcześniejszego treningu sieci na zestawach danych uczących algorytm ten osiągnął gorsze wyniki niż zaproponowane rozwiązanie. W przypadku dużych ilości danych treningowych różnice w wydajności algorytmów były mniej zauważalne [35].

Niemniej jednak, dla każdej testowanej ilości zestawów danych, najbardziej efektywny był algorytm uczenia maszynowego oparty o tablice asocjacyjne. Podczas wyszukiwania rozwiązania w obrębie 10 wpisów, czas ekstrakcji danych z tablic asocjacyjnych, sieci Bayesa i sztucznej sieci neuronowej wynosił odpowiednio: 1 ms, 35 ms, 42 ms [35]. Porównanie wydajności systemu wieloagentowego (obejmujące czas potrzebny do otrzymania rozwiązania mierzony w milisekundach), podczas używania różnych algorytmów uczenia maszynowego przedstawiono na Rys. 18.



Rys. 18. Porównanie wydajności systemu wieloagentowego podczas używania różnych algorytmów uczenia maszynowego (niższy wynik – lepsza wydajność) [35]

5.5. Podsumowanie

Zaproponowany algorytm uczenia maszynowego oparty o tablice asocjacyjne okazał się mniej złożonym obliczeniowo i bardziej wydajnym substytutem sztucznych sieci neuronowych i sieci Bayesa. Nie wymagał on wcześniejszego treningu na danych uczących, dlatego działał w trybie on-line i szybciej dostarczał rozwiązania po przedstawieniu przykładowego wzorca. Implementacja algorytmu uczenia maszynowego w systemie wieloagentowym do wspomagania projektowania wybranych układów sterowania pozwoliła osiągnąć poprawę wydajności w zakresie od 4682,83% do 5564,72% poprzez skrócenie czasu przetwarzania żądań na poziomie 150 sekund. Ponadto proces wyodrębniania rozwiązania z modułu uczącego nie wprowadzał istotnych opóźnień w działaniu systemu wieloagentowego, nawet w przypadku stosunkowo dużej liczby tablic asocjacyjnych [35]. Rozdział 6 zawiera opis przebiegu procesu doboru przykładowej struktury układu sterowania z uwzględnieniem działania opisanych algorytmów szeregowania procesów i uczenia maszynowego.

6. PRZEBIEG PROCESU PROJEKTOWANIA

6.1. Wprowadzenie

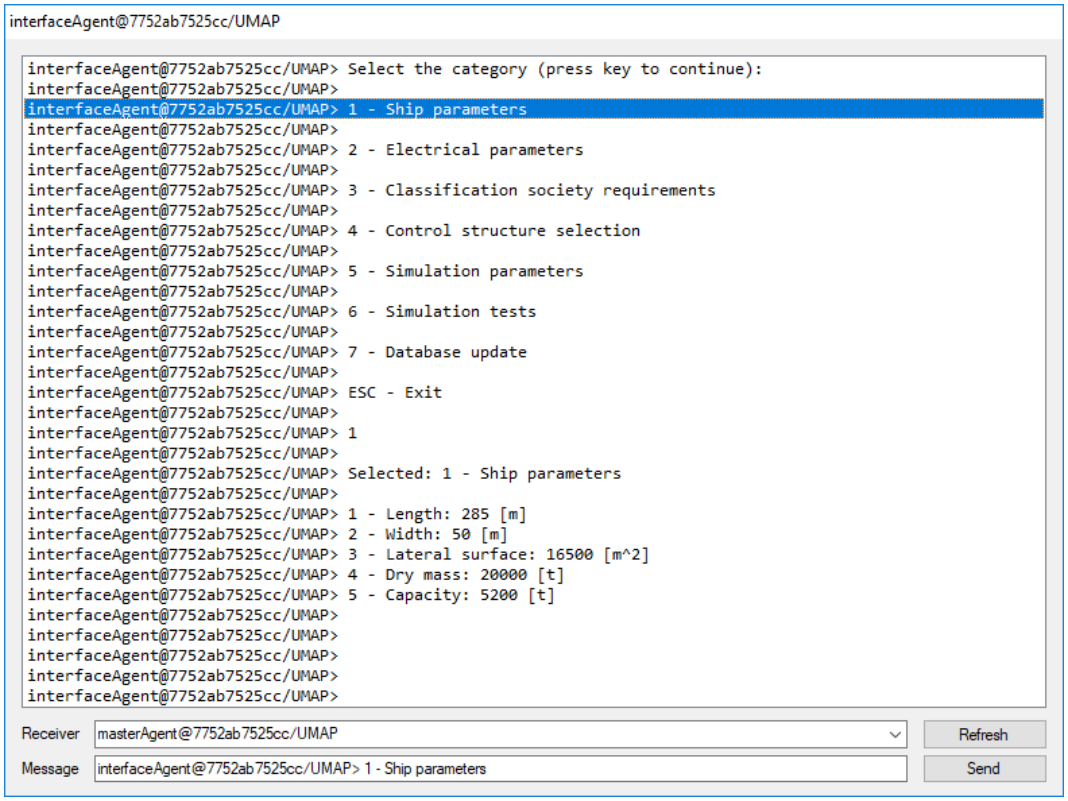
Zademonstrowanie działania systemu wieloagentowego odbędzie się na przykładzie procesu projektowania struktury układu sterowania okrętowego podsystemu elektroenergetycznego z rozruchem silnika indukcyjnego poprzez autotransformator. Struktura z autotransformatorem zapewnia dynamiczne sterowanie momentem silnika indukcyjnego poprzez zmianę przekładni autotransformatora. Przy rozruchu autotransformatorowym występuje około dwukrotne przekroczenie prądu nominalnego silnika indukcyjnego, natomiast śruba ustawiona jest w pozycji jałowej. Elementami struktury sterowanie z rozruchem autotransformatorowym są: okrętowy silnik wysokoprężny, generator synchroniczny, układ AVR i wzbudzenia generatora, autotransformator, silnik indukcyjny, wał napędowy oraz śruba o skoku nastawnym [23-27, 30, 37].

6.2. Projektowanie układu sterowania podsystemu elektroenergetycznego

Rozpoczęcie procesu projektowania rozpoczyna się standardowo od uruchomienia nowej instancji agenta interfejsu i wprowadzenia parametrów dla projektowanej struktury. Okna agenta interfejsu zawierające najważniejsze założenia projektowe dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono w tabeli Tab. 5.

Tab. 5. Okna agenta interfejsu zawierające najważniejsze założenia projektowe dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 37]

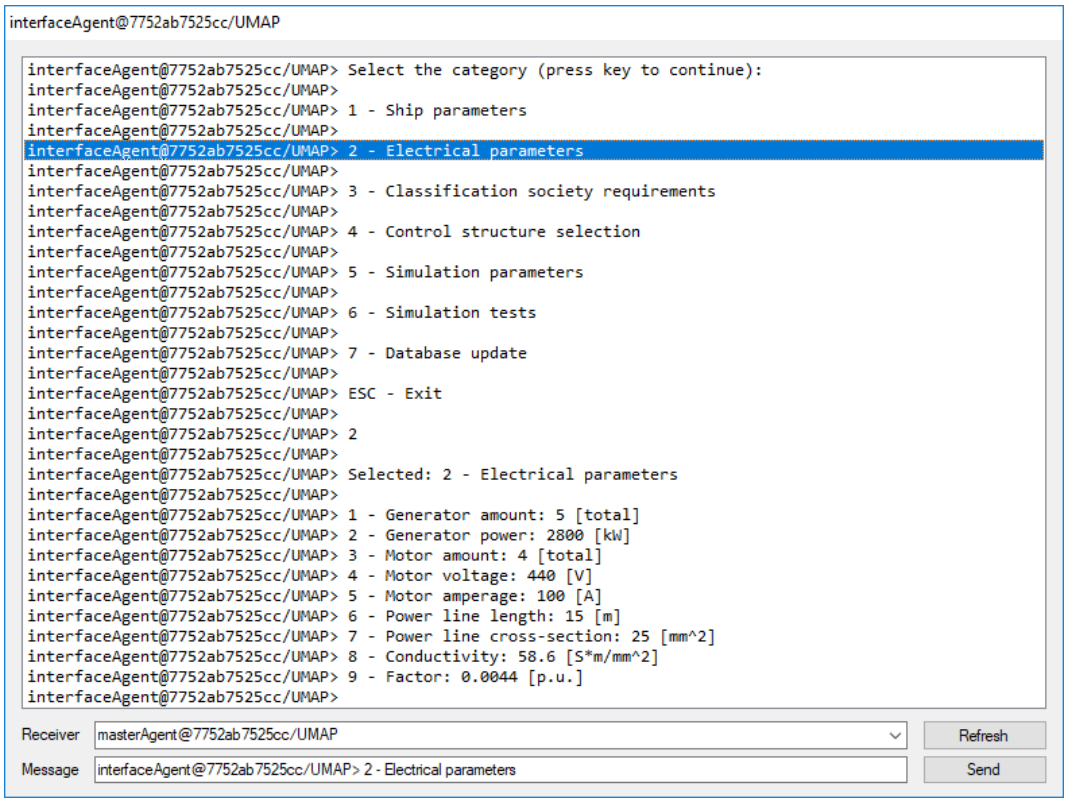
Okno agenta interfejsu umożliwiające edytowanie założeń projektowych dotyczących parametrów statku:



Najważniejsze parametry wprowadzone przez użytkownika w sekcji parametrów statku [23-27, 30, 37]:

Długość statku (ang. Length) – 285 m,
Szerokość statku (ang. Width) – 50 m,
Powierzchnia boczna statku (ang. Lateral surface) – 16500 m^2,
Masa własna statku (ang. Dry mass) – 20000 t,
Ładowność statku (ang. Capacity) – 5200 t.

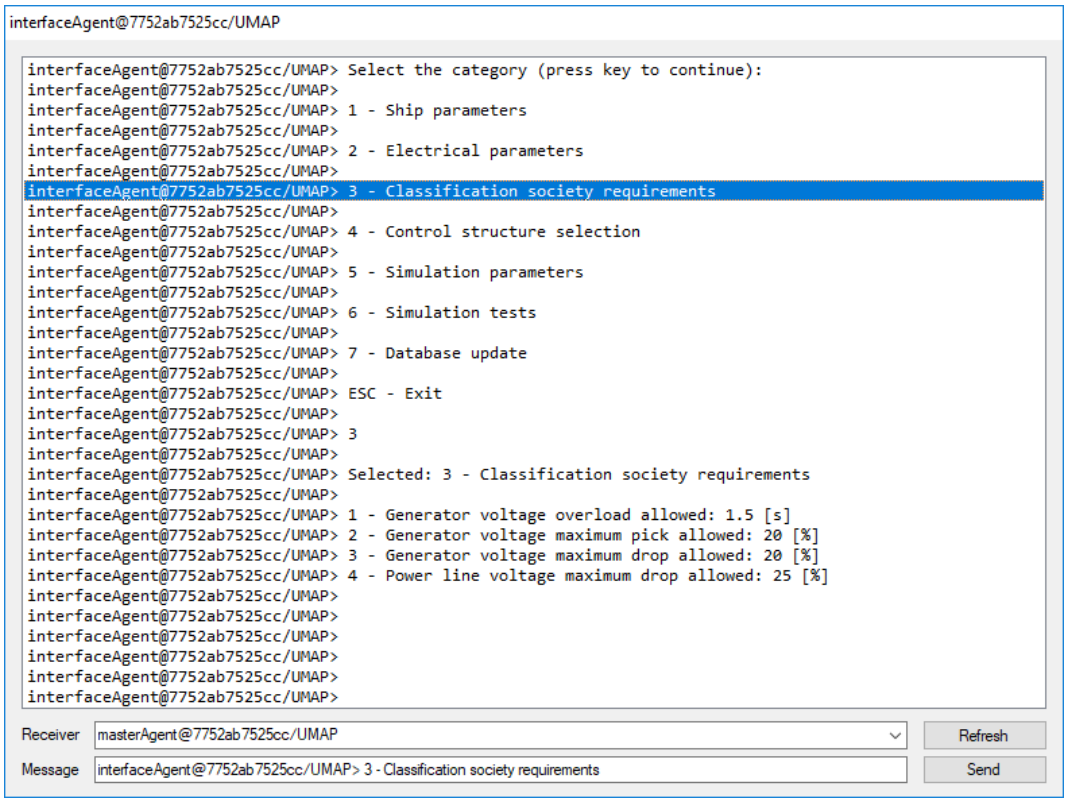
Okno agenta interfejsu umożliwiające edytowanie założeń projektowych dotyczących parametrów elektrycznych:



Najważniejsze parametry wprowadzone przez użytkownika w sekcji parametrów elektrycznych [23-27, 30, 37]:

Liczba generatorów (ang. Generator amount) – 5 szt.,
Moc generatorów (ang. Generator power) – 2800 kW,
Liczba silników indukcyjnych (ang. Motor amount) – 4 szt.,
Napięcie silnika indukcyjnego (ang. Motor voltage) – 440 V,
Prąd silnika indukcyjnego (ang. Motor amperage) – 100 A,
Długość linii elektroenergetycznej (ang. Power line length) – 15 m,
Przekrój poprzeczny linii (ang. Power line cross-section) – 25 mm^2,
Konduktywność linii (ang. Conductivity) – 58,6 $S*m/mm^2$,
Współczynnik przeliczania (ang. Factor) – 0,0044 j.

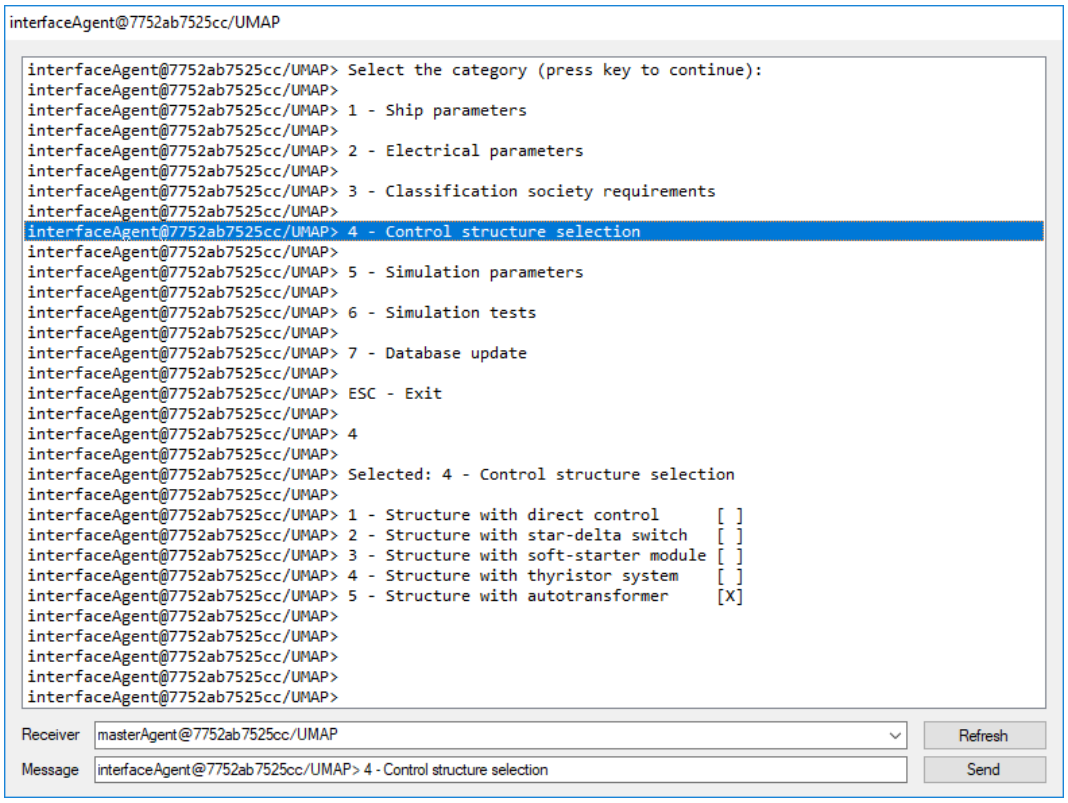
Okno agenta interfejsu umożliwiające edytowanie założeń projektowych dotyczących wymagań towarzystwa klasyfikacyjnego:



Najważniejsze parametry wprowadzone przez użytkownika w sekcji wymagań towarzystwa klasyfikacyjnego [23-27, 30, 37]:

Dopuszczalny czas przekroczenia nominalnego napięcia generatora (ang. Generator voltage overload allowed) – 1,5 s,
Dopuszczalna wartość przekroczenia nominalnego napięcia generatora (ang. Generator voltage maximum pick allowed) – 20%,
Dopuszczalna wartość spadku nominalnego napięcia generatora (ang. Generator voltage maximum drop allowed) – 20%,
Dopuszczalna wartość spadku nominalnego napięcia linii elektroenergetycznej (ang. Power line voltage maximum drop allowed) – 25%.

Okno agenta interfejsu umożliwiające edytowanie założeń projektowych dotyczących struktury sterowania:



Najważniejsze parametry wprowadzone przez użytkownika w sekcji struktury sterowania [23-27, 30, 37]:

Struktura ze sterowaniem bezpośrednim rozruchu silnika indukcyjnego (ang. Structure with direct control) – nie wybrano,

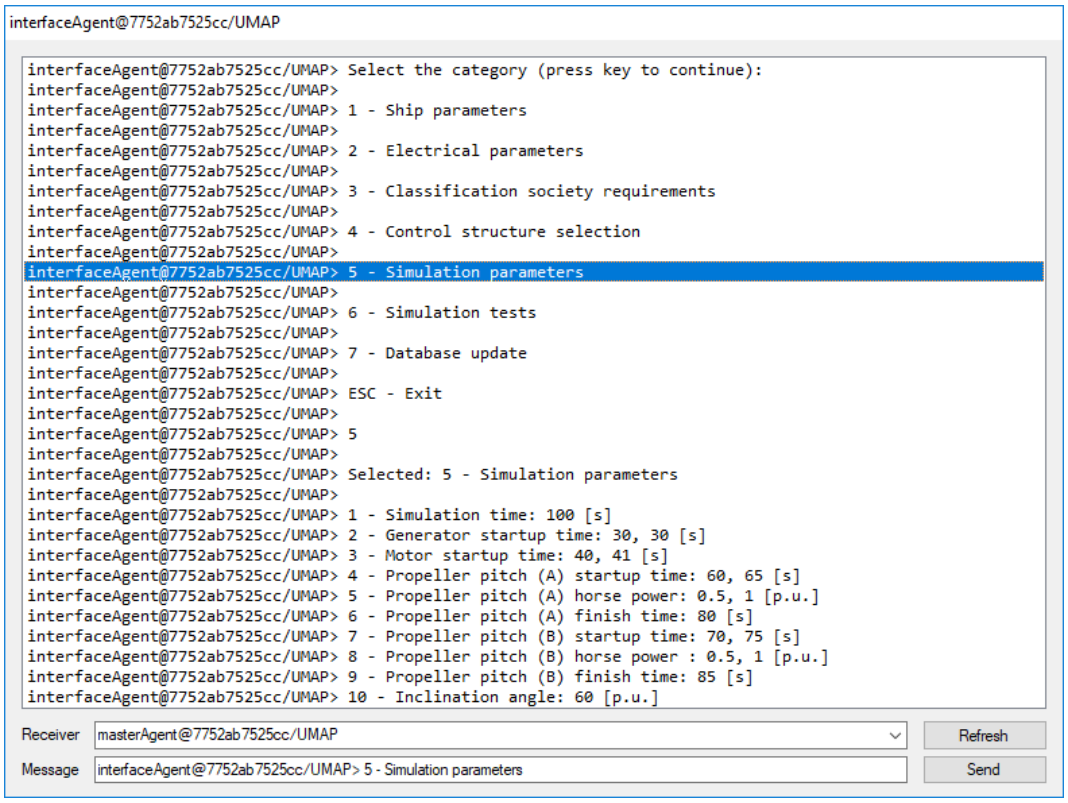
Struktura ze sterowaniem rozruchu silnika indukcyjnego poprzez przełącznik gwiazda-trójkąt (ang. Structure with star-delta switch) – nie wybrano,

Struktura ze sterowaniem rozruchu silnika indukcyjnego poprzez moduł soft-start (ang. Structure with soft-starter module) – nie wybrano,

Struktura ze sterowaniem rozruchu silnika indukcyjnego poprzez system tyrystorowy (ang. Structure with thyristor system) – nie wybrano,

Struktura ze sterowaniem rozruchu silnika indukcyjnego poprzez autotransformator (ang. Structure with autotransformer) – wybrano.

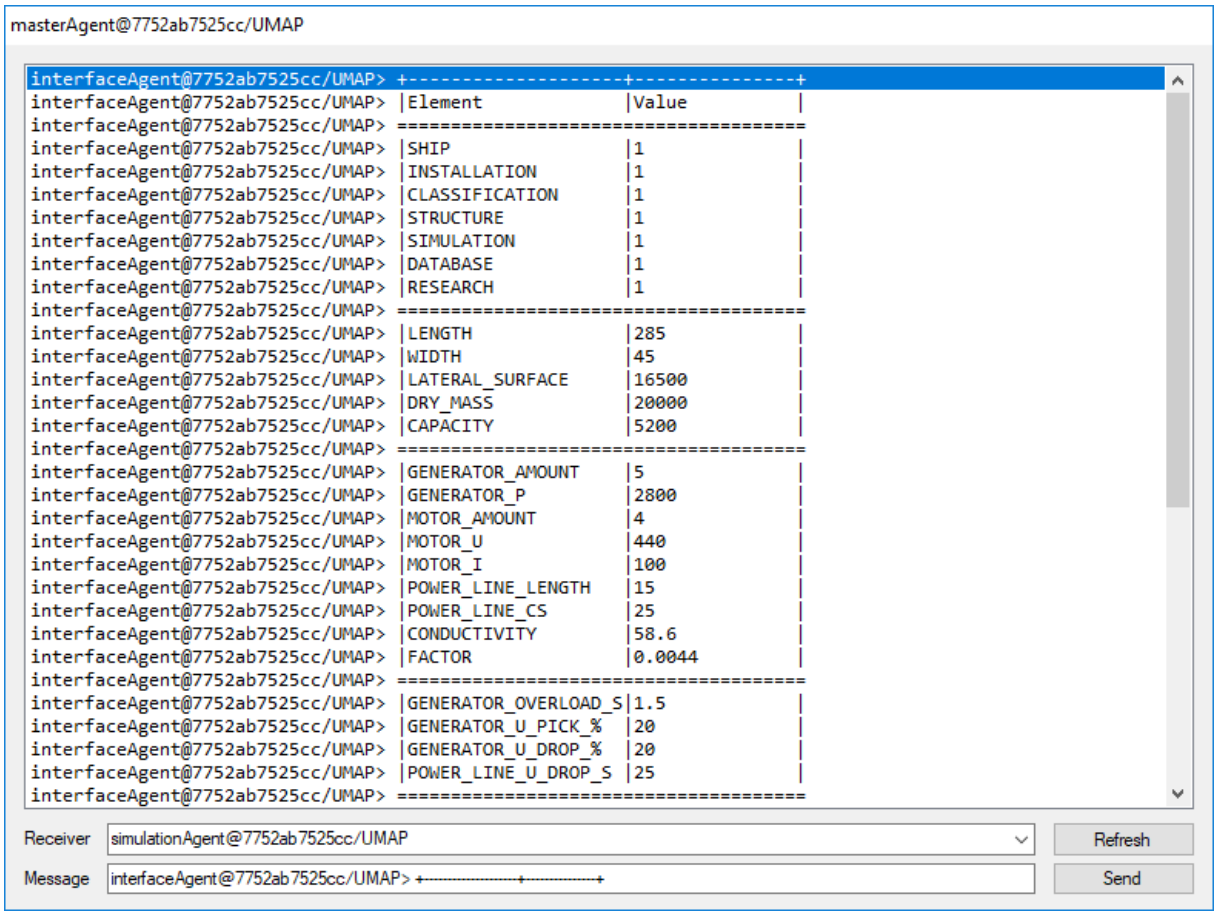
Okno agenta interfejsu umożliwiające edytowanie założeń projektowych dotyczących parametrów symulacji:



Najważniejsze parametry wprowadzone przez użytkownika w sekcji parametrów symulacji [23-27, 30, 37]:

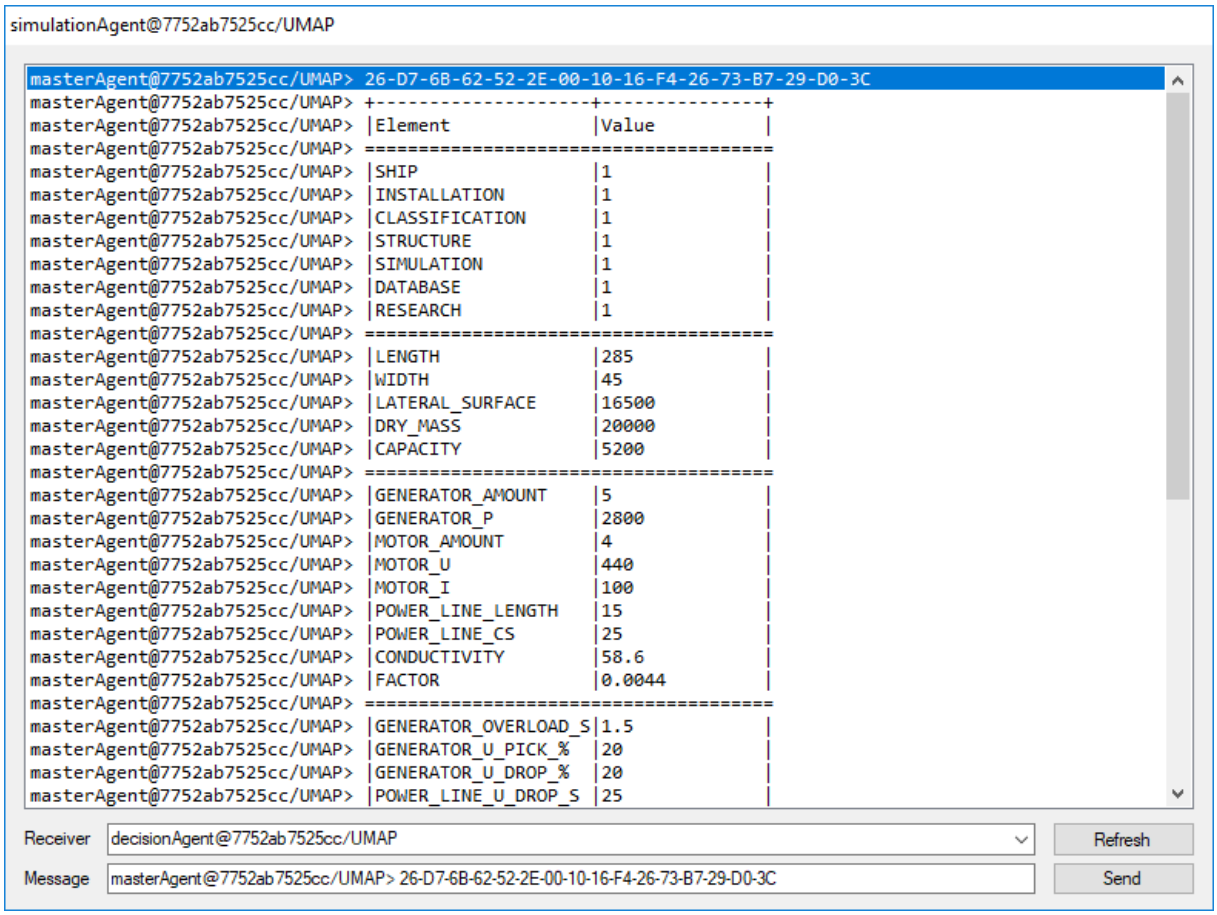
Czas trwania symulacji (ang. Simulation time) – 100 s,
Czas rozruchu generatora (ang. Generator startup time) – 30 s (początek), 30 s (koniec),
Czas rozruchu silnika indukcyjnego (ang. Motor startup time) – 40 s (początek), 41 s (koniec),
Czas rozruchu śruby nastawnej (A) (ang. Propeller pitch (A) startup time) – 60 s (początek), 65 s (koniec),
Moc rozruchowa śruby nastawnej (A) (ang. Propeller pitch (A) horse power) – 0,5 j (na początku), 1 j (na końcu),
Czas zatrzymania śruby nastawnej (A) (ang. Propeller pitch (A) finish time) – 80 s (koniec),
Czas rozruchu śruby nastawnej (B) (ang. Propeller pitch (A) startup time) – 70 s (początek), 75 s (koniec),
Moc rozruchowa śruby nastawnej (B) (ang. Propeller pitch (A) horse power) – 0,5 j (na początku), 1 j (na końcu),
Czas zatrzymania śruby nastawnej (B) (ang. Propeller pitch (A) finish time) – 85 s (koniec),
Kąt nachylenia skoku śruby nastawnej (ang. Inclination angle) – 60°.

Wszystkie informacje wprowadzone przez użytkownika systemu wieloagentowego aktualizowane są w pliku „Parameters.frm”, a po zakończeniu akwizycji danych zawartość pliku (tabela danych) zostaje wczytana przez agenta interfejsu i przesłana jest do agenta nadrzędnego [23-27, 35, 36]. Okno agenta nadrzędnego zawierające tabelę danych dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono na rysunku Rys. 19.



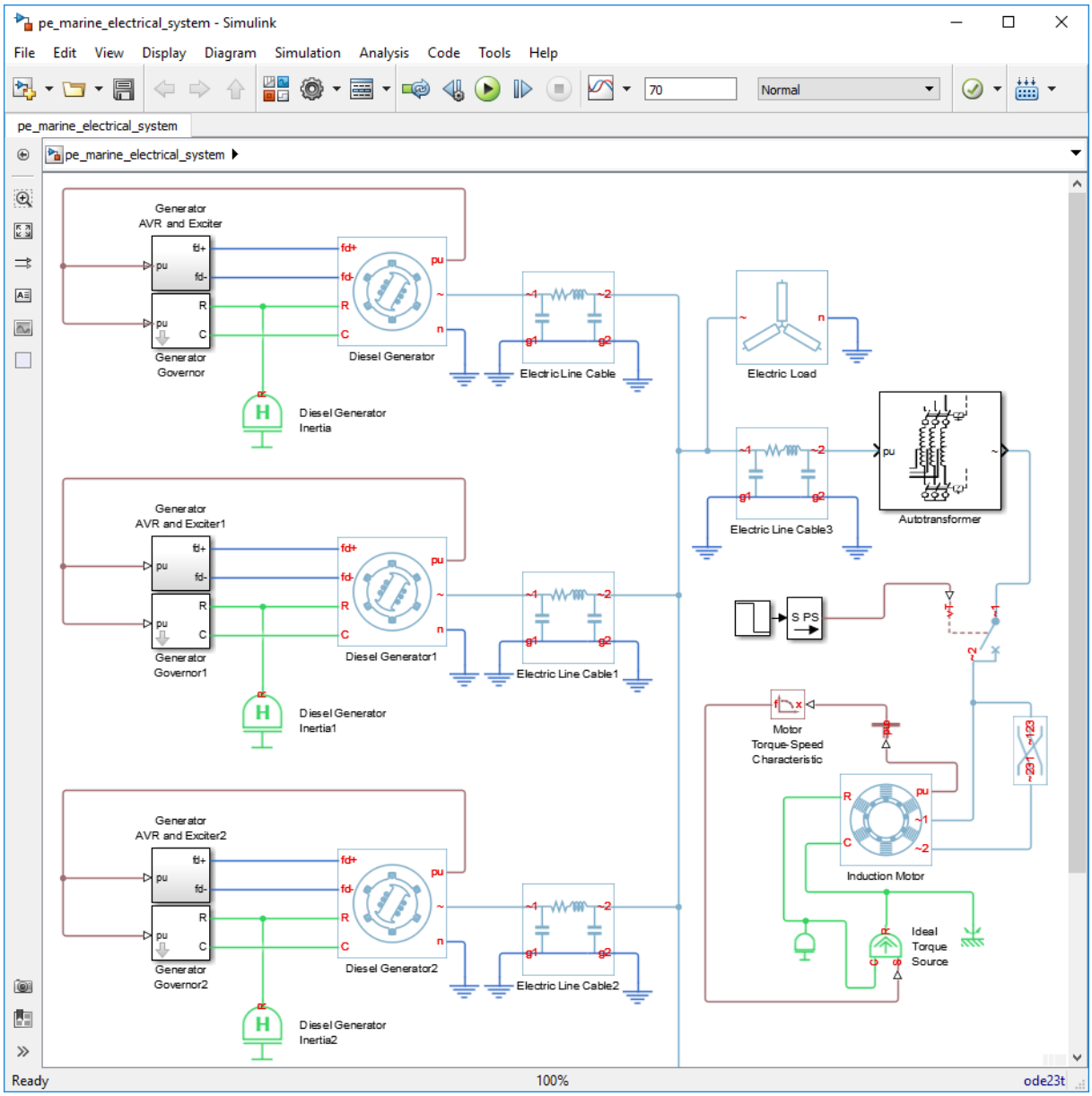
Rys. 19. Okno agenta nadrzędnego zawierające tabelę danych dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 35, 36]

Agent nadrzędny (Master) oblicza sumę kontrolną MD5 dla otrzymanego ciągu znaków (tabeli danych), aby rozpocząć wykonywanie algorytmu uczenia maszynowego. Master pobiera następnie listę wszystkich aktywnych agentów symulacyjnych i odpytuje ich o gotowość wykonania obliczeń w celu zainicjalizowania algorytmu szeregowania procesów. Hash MD5 zostaje dodany do nagłówka tabeli danych i wytyczne projektowe wysłane zostają do najmniej obciążonego agenta symulacyjnego pod względem wykorzystania CPU [23-27, 35, 36, 45]. Okno agenta symulacyjnego zawierające tabelę danych z nagłówkiem kontrolnym w postaci sumy MD5 dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono na rysunku Rys. 20.



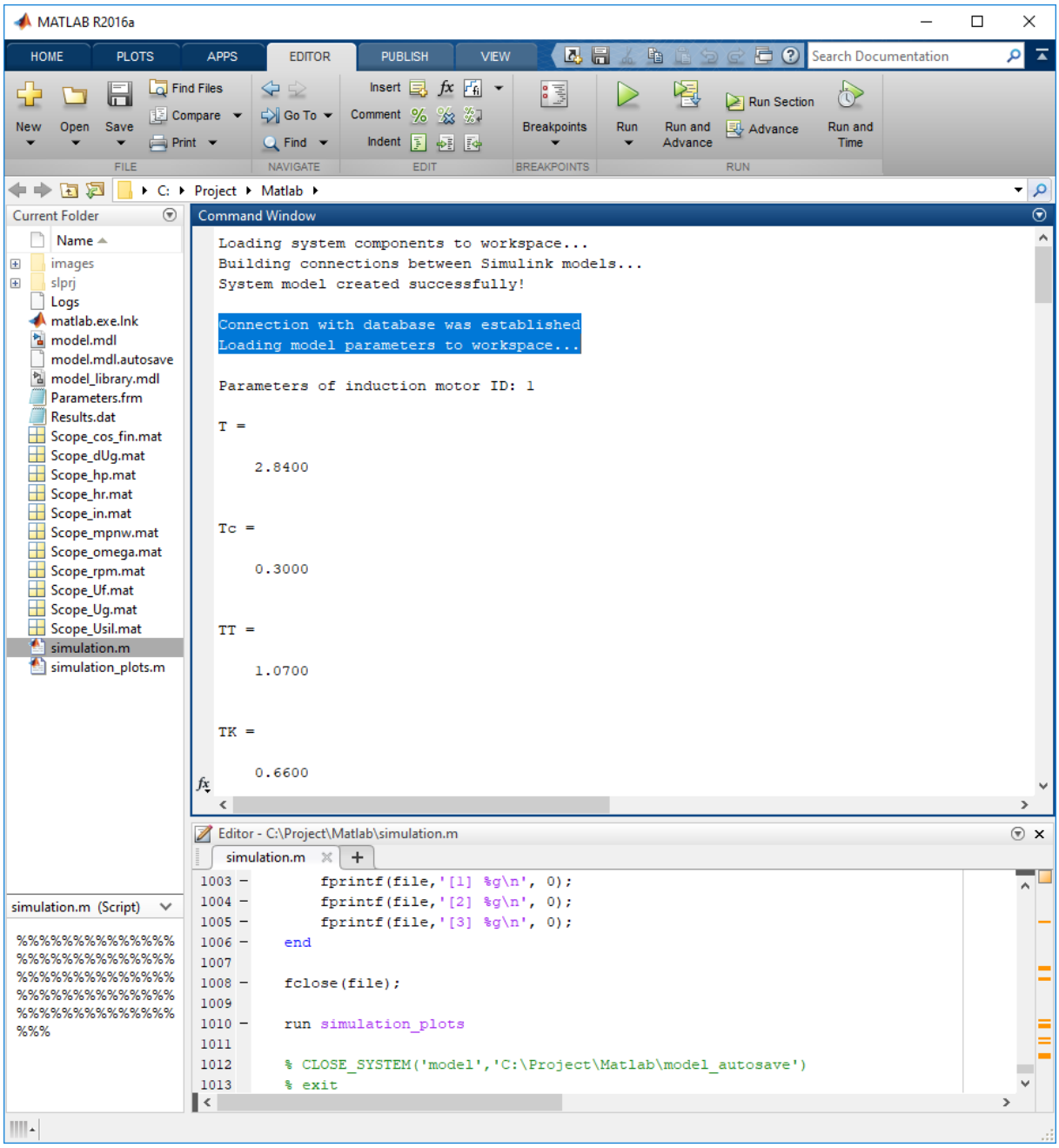
Rys. 20. Okno agenta symulacyjnego zawierające tabelę danych z nagłówkiem kontrolnym w postaci sumy MD5 dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 35, 36]

Agent symulacyjny na podstawie numeru wybranej przez użytkownika struktury sterowania podsystemu elektroenergetycznego wczytuje z bazy modeli niezbędne elementy do budowy układu (struktura numer 5 oznacza sterowanie rozruchem silnika indukcyjnego poprzez autotransformator). Po wczytaniu modeli matematycznych agent symulacyjny automatycznie generuje połączenia między blokami [23-27, 30, 35, 40]. Okno aplikacji symulacyjnej zawierające kompletny model układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono na rysunku Rys. 21.



Rys. 21. Okno aplikacji symulacyjnej zawierające kompletny model układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 35, 40]

Agent symulacyjny wczytuje następnie parametry modeli matematycznych z bazy danych. Połączenie z bazą danych ustanowione zostaje poprzez protokół ODBC. Za pomocą zapytań SQL z bazy danych pobrane zostają parametry, które aktualizowane są w przestrzeni roboczej środowiska Matlab Simulink. Po wczytaniu wszystkich parametrów następuje automatyczne uruchomienie symulacji [23-27, 30, 35, 40]. Okno aplikacji symulacyjnej zawierające wczytane parametry modeli elementów układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono na rysunku Rys. 22.



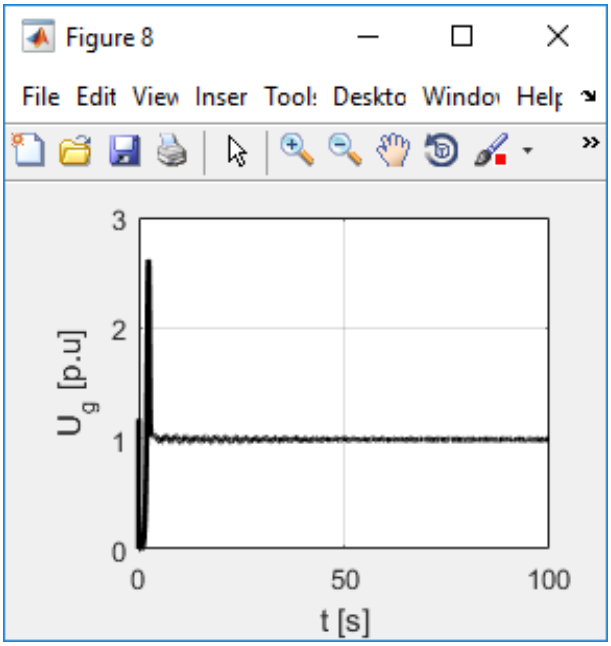
Rys. 22. Okno aplikacji symulacyjnej zawierające wczytane parametry modeli elementów układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 35, 40]

6.2.1. Wyniki badań symulacyjnych dla utworzonej struktury

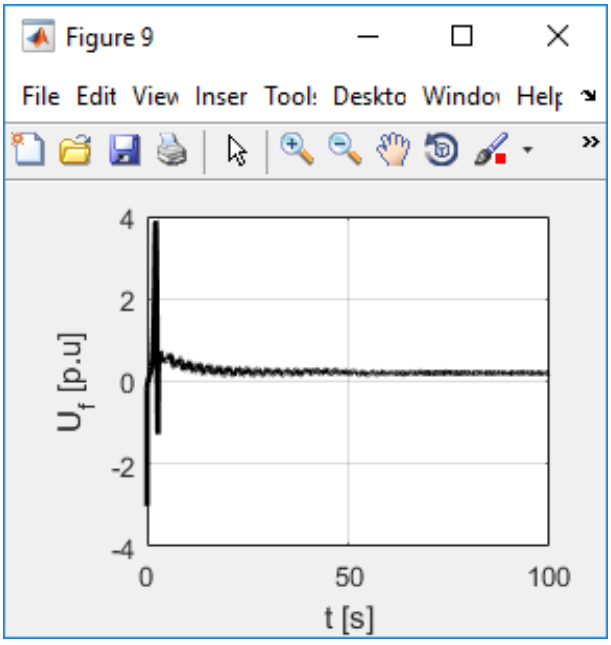
Po zakończeniu symulacji wyniki zostają wyświetlone na ekranie w postaci przebiegów zmiennych. Przebiegi te zostają zapisanie również lokalnie do plików *.mat, natomiast najbardziej kluczowe wyniki badań symulacyjnych zostają dodane jako kolejna sekcja do tabeli danych i przesłane dalej do agenta decyzyjnego [23-27, 30, 40]. Okna aplikacji symulacyjnej zawierające wygenerowane przebiegi zmiennych dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono w tabeli Tab. 6., natomiast okno agenta decyzyjnego zawierające syntetyczne wyniki symulacji (dodane jako nowa sekcja tabeli danych) przedstawiono na rysunku Rys. 23.

Tab. 6. Okna aplikacji symulacyjnej zawierające wygenerowane przebiegi zmiennych dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 40]

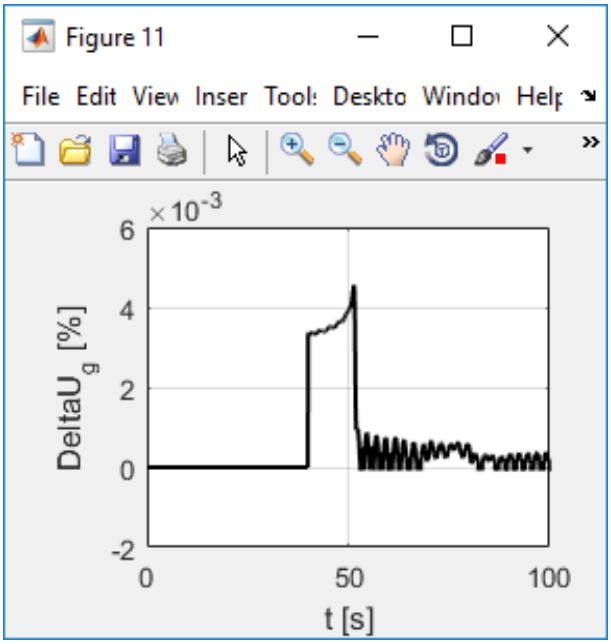
Okno aplikacji symulacyjnej zawierające wygenerowane przebiegi napięcia generatora:



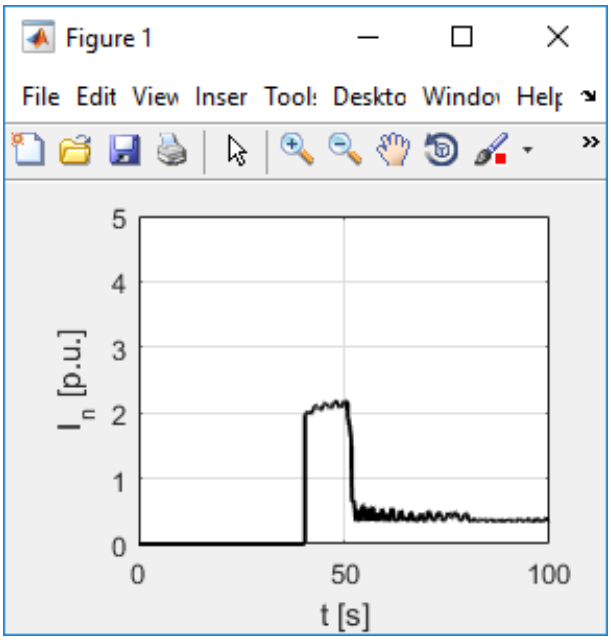
Okno aplikacji symulacyjnej zawierające wygenerowane przebiegi napięcia wzbudzenia generatora:



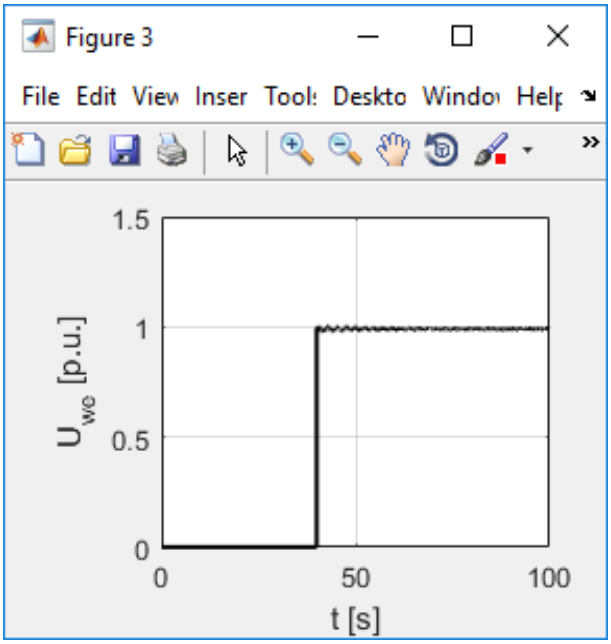
Okno aplikacji symulacyjnej zawierające wygenerowane przebiegi spadków napięcia generatora:

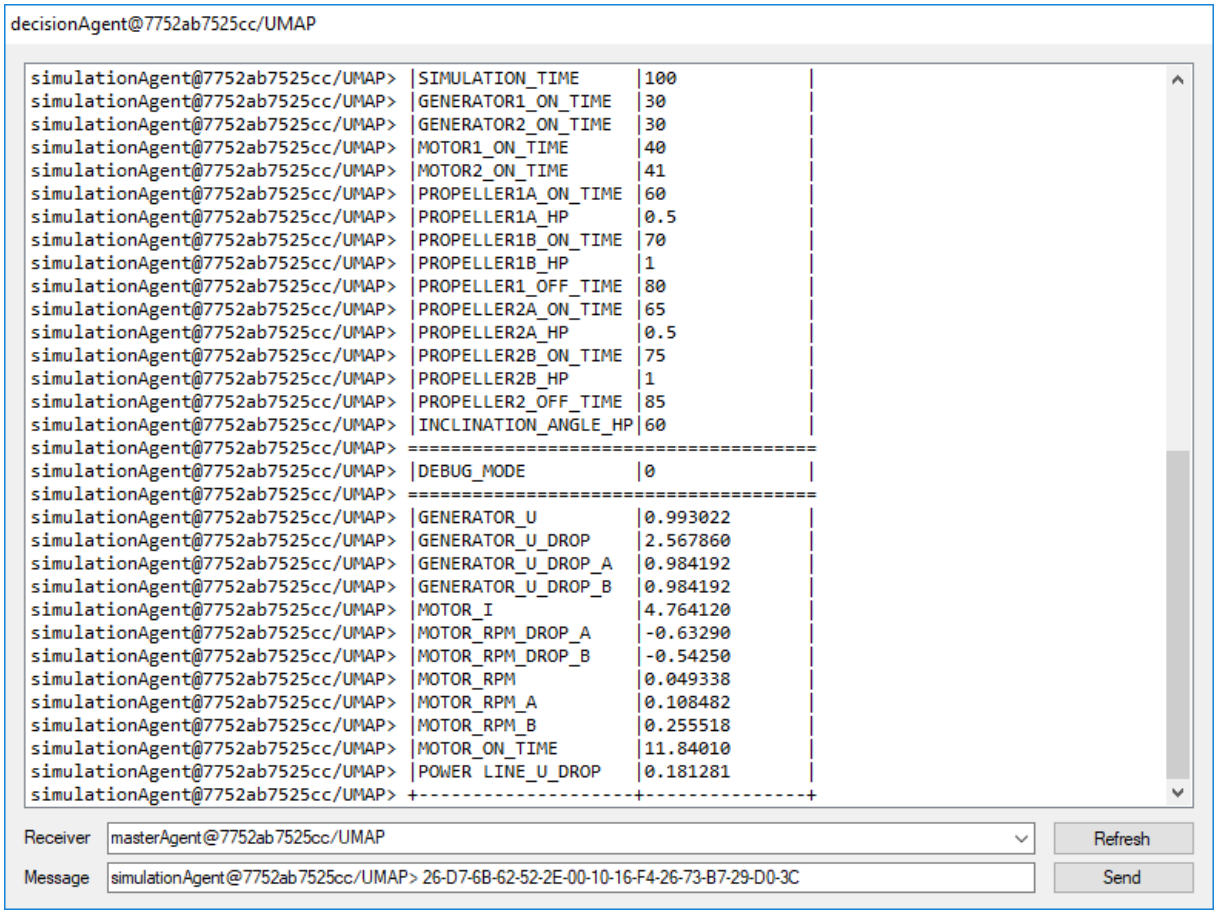


Okno aplikacji symulacyjnej zawierające wygenerowane przebiegi prądu silnika indukcyjnego:



Okno aplikacji symulacyjnej zawierające wygenerowane przebiegi napięcia silnika indukcyjnego:





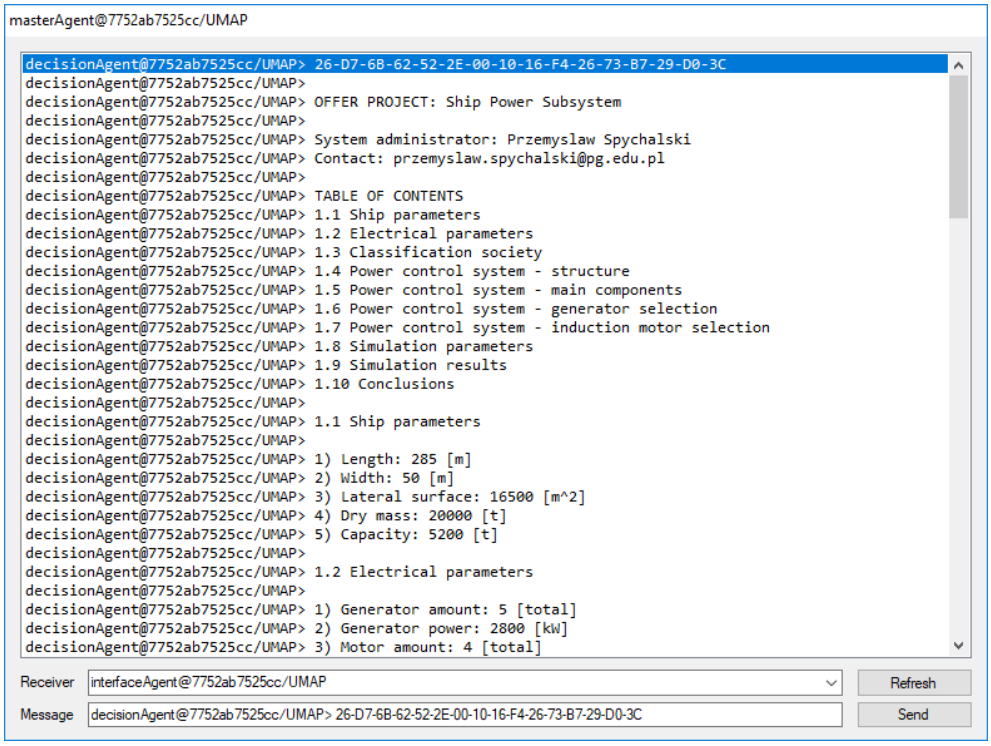
Rys. 23. Okno agenta decyzyjnego zawierające syntetyczne wyniki symulacji dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 40]

6.2.2. Wygenerowane elementy projektu ofertowego

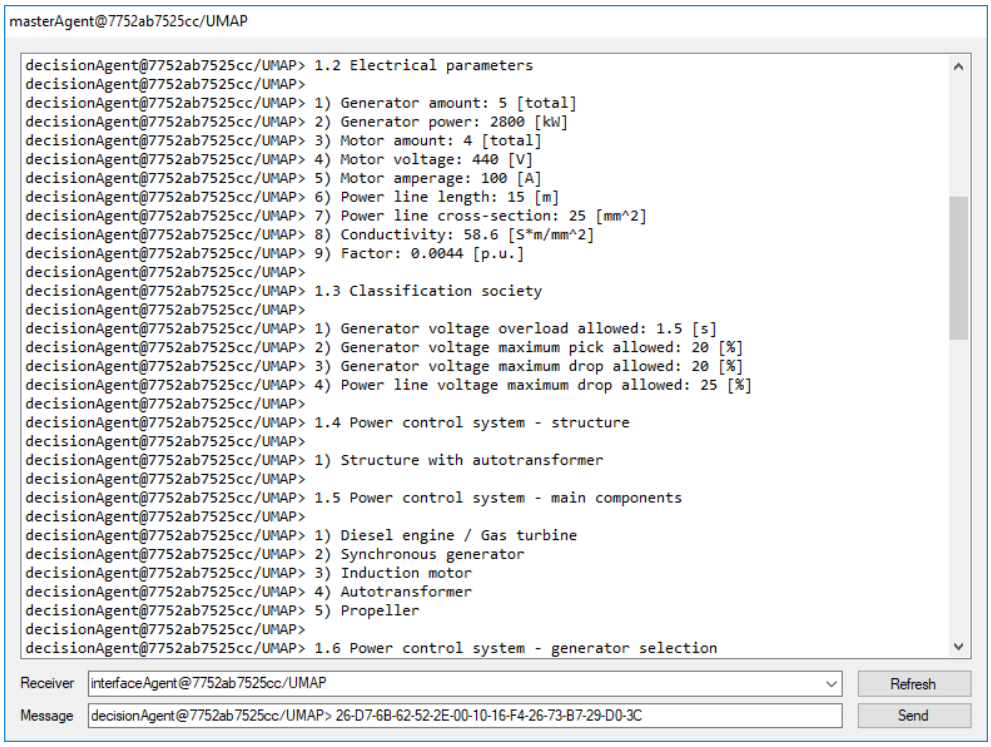
Agent decyzyjny korzystając z danych zawartych we wszystkich sekcjach tabeli danych (szczególnie tej zawierającej wyniki badań symulacyjnych struktury sterowania) generuje raport końcowy dla użytkownika systemu wieloagentowego. Raport ten ma formę projektu ofertowego, zawierającego podstawowe informacje o parametrach okrętowej instalacji elektroenergetycznej, dobranych elementach układu sterowania oraz podsumowanie, w którym oceniona zostaje zgodność dynamiki utworzonego układu z ograniczeniami towarzystwa klasyfikacyjnego [23-27, 30, 35, 36, 37]. Okna agenta nadrzędnego zawierające wygenerowane elementy projektu ofertowego dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono w tabeli Tab. 7.

Tab. 7. Okna agenta nadrzędnego zawierające wygenerowane elementy projektu ofertowego dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [23-27, 30, 35, 36, 37]

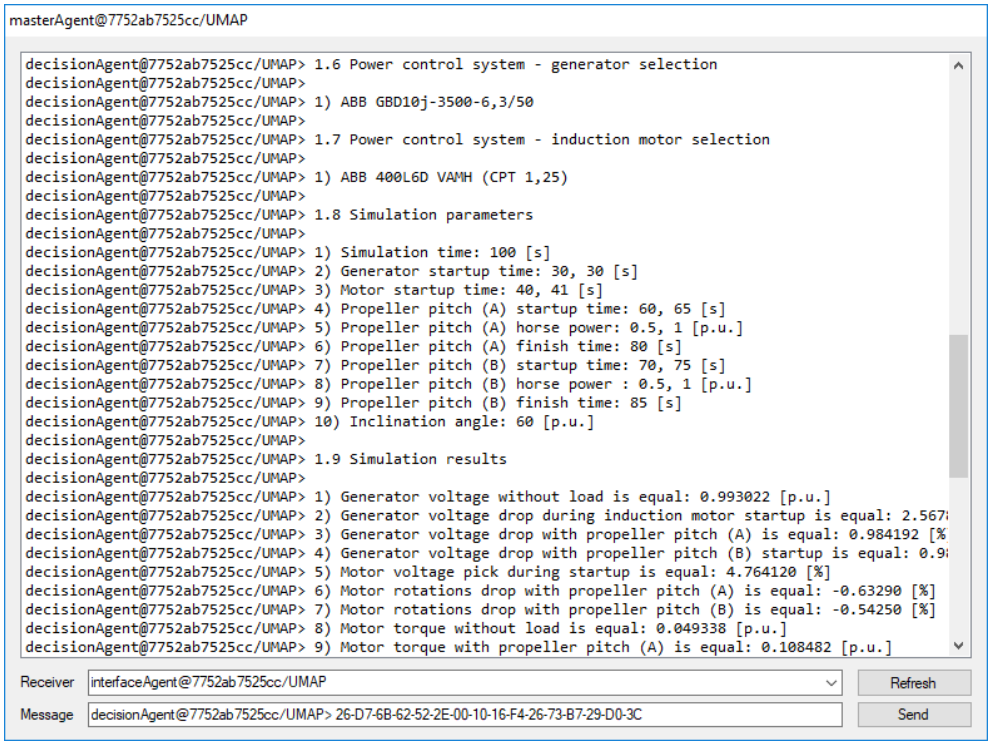
Okno agenta nadrzędnego zawierające wygenerowane sekcje projektu ofertowego dotyczące informacji kontaktowych, spisu treści i parametrów statku:



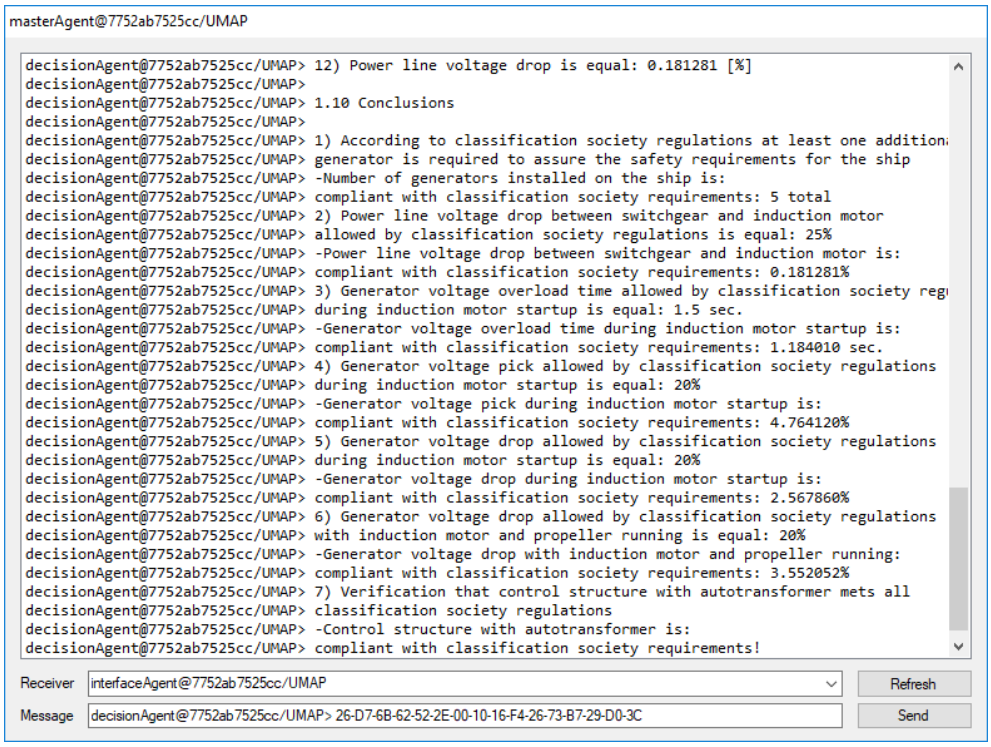
Okno agenta nadrzędnego zawierające wygenerowane sekcje projektu ofertowego dotyczące parametrów elektrycznych, wymagań towarzystwa klasyfikacyjnego, struktury sterowania i jej komponentów:



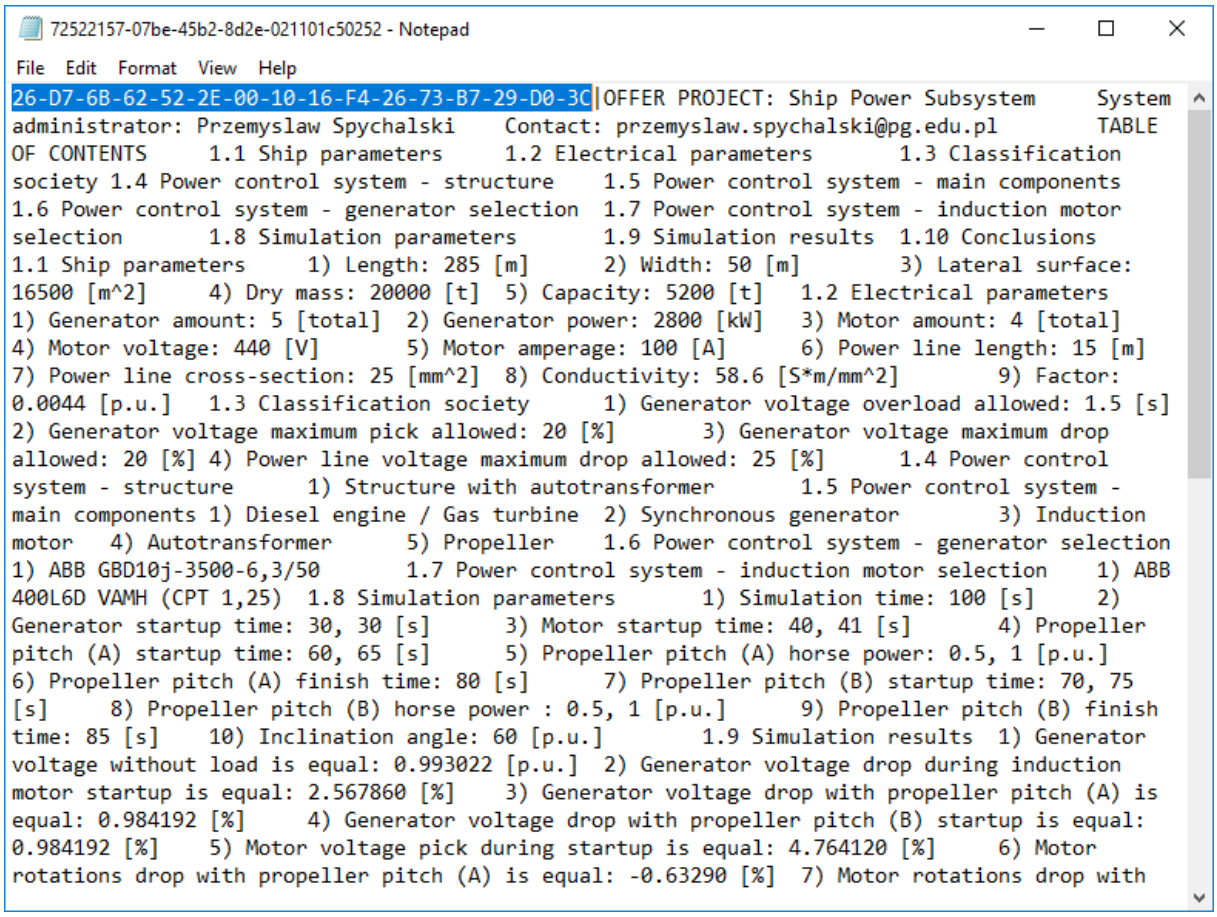
Okno agenta nadrzędnego zawierające wygenerowane sekcje projektu ofertowego dotyczące doboru generatora oraz silnika indukcyjnego z bazy danych, parametrów symulacji i jej wyników:



Okno agenta nadrzędnego zawierające wygenerowane sekcje projektu ofertowego dotyczące konkluzji systemu wieloagentowego i oceny zgodności dynamiki układu sterowania z wymaganiami towarzystwa klasyfikacyjnego:

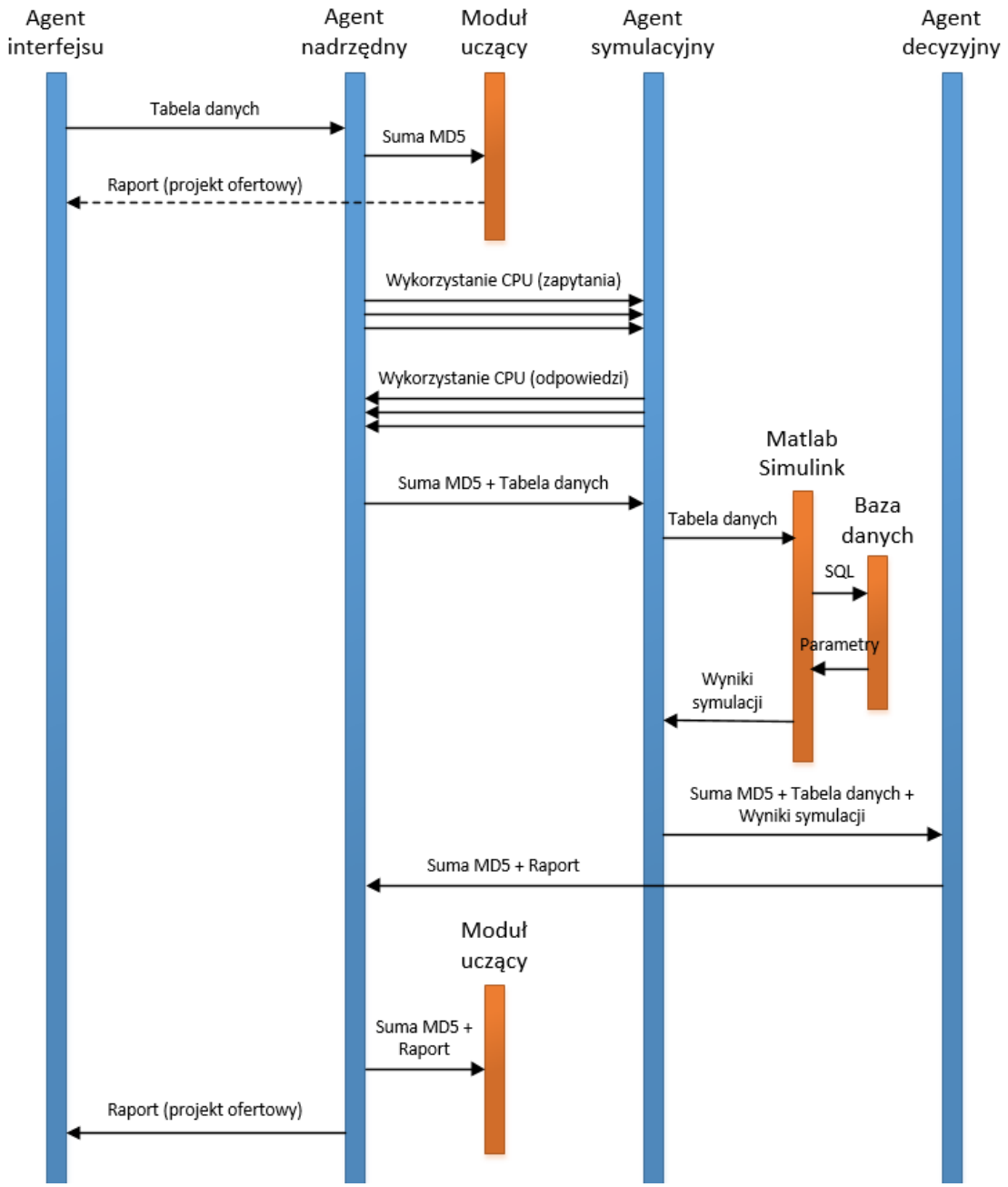


Zanim jednak wygenerowany raport trafi do odpowiedniego agenta interfejsu zostaje on przesłany do agenta nadrzędnego, który finalizuje proces uczenia maszynowego dla określonej struktury układu sterowania. Agent nadrzędny do wcześniej obliczonej sumy kontrolnej MD5 dodaje wygenerowany przez agenta decyzyjnego raport, oddzielony znakiem separatora. Powstała w ten sposób tablica asocjacyjna zapisana zostaje w pliku *.dat (moduł uczący) [30, 35, 37]. Okno pliku modułu uczącego zawierającego utworzoną tablicę asocjacyjną dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem przedstawiono na rysunku Rys. 24.



Rys. 24. Okno pliku modułu uczącego zawierającego utworzoną tablicę asocjacyjną dla układu sterowania podsystemu elektroenergetycznego z autotransformatorem [30, 35, 37]

Następnie, również na podstawie sumy kontrolnej, sprawdzana jest nazwa agenta interfejsu, odpowiedzialnego za wysłanie tabeli danych, z której obliczona została ta suma. Z racji tego, że suma MD5 to unikalna wartość, określony zostaje odpowiedni agent interfejsu, który powinien otrzymać wygenerowany raport. Problem może pojawić się, wówczas gdy więcej niż jeden agent interfejsu posiada taką samą nazwę (nazwę definiuje się jako: `<nazwa_agenta>@<nazwa_kontenera>`). Jednak, aby do tego doszło na tej samej maszynie fizycznej musi działać więcej niż jeden Interface Agent, co nie jest zalecane. W rozpatrywanym przypadku raport z analizy otrzymaliby wszyscy agenci interfejsu, posiadający taką samą nazwę jak ten, od którego pochodziło pierwotne żądanie (tabela danych z wytycznymi dotyczącymi struktury z autotransformatorem). Po przesłaniu przez agenta nadrzędnego raportu do odpowiedniego agenta interfejsu proces projektowania zostaje zakończony [30, 35]. Schemat komunikacji pomiędzy poszczególnymi agentami i modułami używanymi przez tych agentów podczas procesu projektowania przedstawiono na rysunku Rys. 25.



Rys. 25. Schemat komunikacji pomiędzy poszczególnymi agentami i modułami używanymi przez tych agentów podczas procesu projektowania [30, 35]

6.3. Podsumowanie

Przedstawiona struktura układu sterowania podsystemu elektroenergetycznego z autotransformatorem stanowi oryginalną pracę autora i nie była wcześniej publikowana. Została ona następnie zaimplementowana we wspomnianym systemie ekspertowym [23-27], w którym wcześniej była niedostępna. Celem dodania nowej struktury w systemie ekspertowym było porównanie wydajności obu systemów (scentralizowanego ekspertowego i rozproszonego wieloagentowego) podczas procesu projektowania tego samego układu sterowania podsystemem elektroenergetycznym poprzez autotransformator. Rozdział 7 zawiera wyniki badań wydajnościowych systemu wieloagentowego.

7. WYDAJNOŚĆ SYSTEMU WIELOAGENTOWEGO

7.1. Wprowadzenie

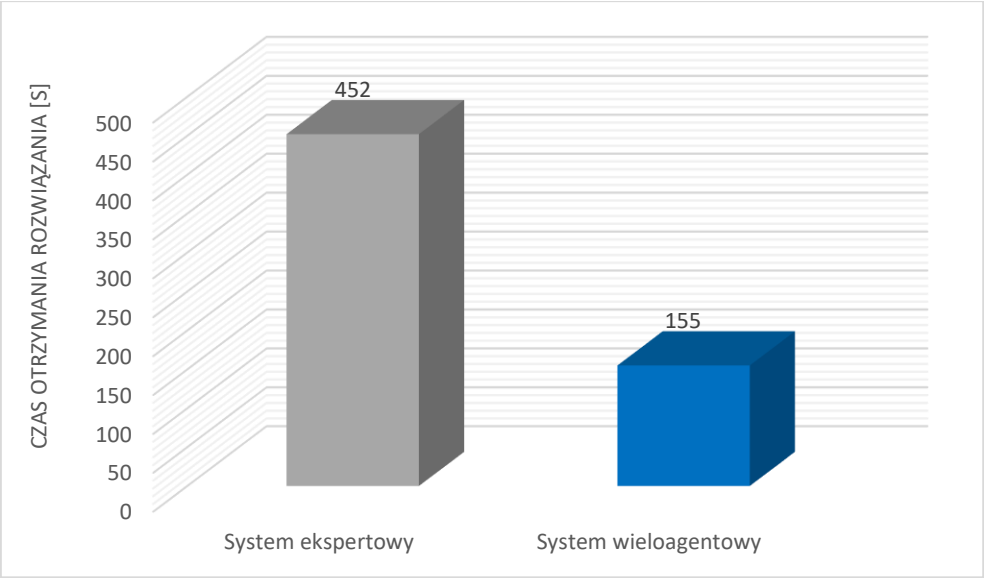
Jednym z wyznaczników jakości zaimplementowanego oprogramowania jest jego wydajność. Poprzez wydajność oprogramowania rozumie się ilość operacji obliczeniowych, jaką dany program jest w stanie wykonać w określonej jednostce czasu. Aby zbadać wydajność zaimplementowanego systemu wieloagentowego i porównać ją z wydajnością rozwiązań z literatury (system ekspertowy), zlecono wykonanie serii takich samych zadań obliczeniowych i zmierzono czas potrzebny obojgu systemom do ich zakończenia. W tym celu wysłano trzy następujące po sobie żądania wykonania symulacji dla przykładowej struktury układu sterowania, a następnie zmierzono czas wygenerowania rozwiązania przez system wieloagentowy i ekspertowy [35, 37, 45].

Pomiarów dokonano z wykorzystaniem narzędzi i metod takich jak Windows Performance Recorder (program do rejestrowania m.in. aktywności procesów działających w systemie operacyjnym wraz ze stanem zasobów jakie wykorzystują), punktów przerwań (ang. Breakpoint, funkcja umożliwiająca przerwanie wykonywania programu m.in. po wystąpieniu określonego zdarzenia i podglądu stanu zmiennych oraz czasu wykonywania instrukcji) i stopera (stoper stosowano jedynie wówczas gdy mierzono czas wykonywania operacji rozproszonych pomiędzy wiele komputerów i nie było możliwości bezpośredniego użycia obu poprzednich metod) [35, 37, 45].

Testy wydajnościowe przeprowadzone zostały w środowisku sprzętowym składającym się z 10 komputerów o następującej specyfikacji sprzętowej: Intel Core i7-6820HQ, DDRAM 2×DIMM 8 GB, Intel SSDSCKJF180A5. W przypadku systemu wieloagentowego środowisko zostało zainicjalizowane w następujący sposób: 3×Interface Agent, 1×Master Agent, 3×Simulation Agent, 3×Decision Agent. System ekspertowy nie miał możliwości pracy sieciowej, więc korzystał tylko z pojedynczego komputera. Kolejne podrozdziały zawierają pomiary wydajności systemu wieloagentowego oraz analizę wpływu zaimplementowanych algorytmów na efektywność działania systemu, jak również porównanie tych danych z dostępnymi rozwiązaniami [35, 37, 45].

7.2. Porównanie wydajności systemu wieloagentowego i ekspertowego

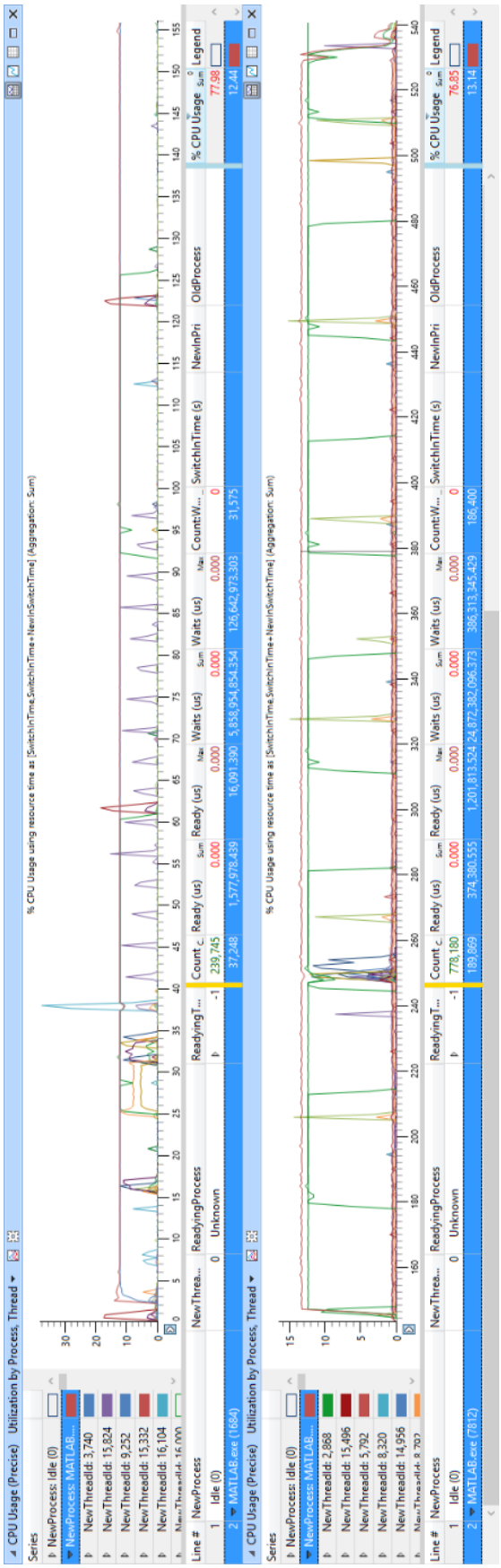
Pierwszym eksperymentem było porównanie wydajności systemu wieloagentowego z systemem ekspertowym, przedstawionym w podrozdziale 2.2 podczas rozwiązywania zadania projektowego, które było identyczne jak to sformułowane w podrozdziale 6.2 (biorąc pod uwagę niedeterministyczne czasy prowadzenia symulacji, przeprowadzono trzy iteracje badań i podano sumaryczny czas otrzymania rozwiązań dla wszystkich procesów projektowych) [35, 37, 45]. Porównanie wydajności systemu wieloagentowego i ekspertowego podczas rozwiązywania trzech iteracji zadania projektowego przedstawiono na rysunku Rys. 26.



Rys. 26. Porównanie wydajności systemu wieloagentowego i ekspertowego podczas rozwiązywania trzech iteracji zadania projektowego (niższy wynik – lepsza wydajność) [35, 37, 45]

Porównania wydajności obu systemów dokonano przy założeniu, iż system wieloagentowy korzysta z zaimplementowanego algorytmu szeregowania procesów obliczeniowych i algorytmu uczenia maszynowego. Dla przypomnienia mechanizm szeregowania oparty został na wymaganiu, że przychodzące żądania z warstwy interfejsu powinny zostać uszeregowane i wykonane przez najmniej obciążoną jednostkę (pod względem wykorzystania CPU) w warstwie symulacyjnej. Jeżeli natomiast wszystkie jednostki są zajęte, wówczas żądania powinny zostać dodane do kolejki i czekać na zwolnienie zasobów. W celu demonstracji działania opracowanego rozwiązania przeprowadzone zostały testy wydajnościowe systemu wieloagentowego z wyłączonym mechanizmem szeregowania (aby porównać jak zmieni się wydajność systemu wieloagentowego po uruchomieniu algorytmu).

Wyłączenie algorytmów dodatkowych odbywało się poprzez ustawienie pola „DEBUG_MODE” w tabeli danych na wartość niezerową. Metodologia pomiarów również w tym przypadku zakładała wysłanie trzech żądań obliczeń symulacyjnych z warstwy interfejsu i zmierzenie czasu otrzymania odpowiedzi od warstwy obliczeniowej (składającej się z trzech agentów). Monitorowano przy tym stan wykorzystania procesora pierwszej jednostki, na której uruchomiony został agent symulacyjny z zastosowaniem narzędzia Windows Performance Recorder [35, 37, 45]. Porównanie wydajności systemu wieloagentowego z aktywnym (górny wykres) i nieaktywnym (dolny wykres) algorytmem szeregowania procesów przedstawiono na rysunku Rys. 27.



Rys. 27. Porównanie wydajności systemu wieloagentowego z aktywnym i nieaktywnym algorytmem szeregowania procesów [45]

7.3. Podsumowanie

Porównanie wydajności systemu wieloagentowego i ekspertowego podczas rozwiązywania trzech iteracji zadania projektowania struktury układu sterowania okrętowego podsystemu elektroenergetycznego z rozruchem silnika indukcyjnego poprzez autotransformator wykazało, że system wieloagentowy posiada wydajność na poziomie około 190% większą od systemu ekspertowego, wyrażoną krótszym czasem przetwarzania identycznych żądań (155 sekund w przypadku systemu wieloagentowego, 452 w przypadku systemu ekspertowego). Zaobserwowano również wzrost wydajności systemu wieloagentowego po uruchomieniu algorytmu szeregowania procesów, wyrażony zredukowaniem czasu przetwarzania żądań z 542 do 155 sekund. Spowodowane było to uszeregowaniem procesów tak, że na każdego agenta przypadł tylko jeden proces do przetworzenia (kiedy algorytm był nieaktywny wszystkie trzy procesy zostały przydzielone pojedynczemu agentowi). Zastosowanie mechanizmu szeregowania procesów przyczyniło się do zwiększenia wydajności systemu wieloagentowego na poziomie 250% poprzez skrócenie czasu wykonywania symulacji. Uzyskane wyniki potwierdzają słuszność implementacji wspomnianego mechanizmu szeregowania oraz jego poprawne działanie [35, 37, 45].

8. ZAKOŃCZENIE

Celem rozprawy doktorskiej była implementacja rozproszonego systemu wieloagentowego, który umożliwia zdecentralizowane oraz współbieżne prowadzenie procesów projektowych przy doborze odpowiedniej struktury i elementów składowych wybranych układów sterowania okrętowych podsystemów elektroenergetycznych, jak również pozwala na ocenę zgodności dynamiki otrzymanych rozwiązań projektowych z wymaganiami towarzystw klasyfikacyjnych. Zadanie to zostało w pełni zrealizowane, co udowodniono poprzez zademonstrowanie działania systemu wieloagentowego na przykładzie procesu projektowania układu sterowania z rozruchem autotransformatorowym opisanym w rozdziale 6.

Wygenerowane elementy projektu ofertowego oraz charakterystyki dynamiczne utworzonego przez system wieloagentowy modelu układu elektroenergetycznego ze sterowaniem poprzez autotransformator potwierdziły, że możliwe jest częściowe zautomatyzowanie procesu doboru struktur systemów sterowania oraz ocena zgodności ich dynamiki z ograniczeniami towarzystw klasyfikacyjnych. Oryginalne rozwiązanie problemu naukowego, jakim jest wspomaganie projektowania okrętowych podsystemów elektroenergetycznych, osiągnięte zostało poprzez realizację następujących zadań w ramach rozprawy doktorskiej:

- przegląd literatury z zakresu inteligentnych systemów wspomagających projektowanie okrętowych układów sterowania, takich jak np. systemy ekspertowe i wieloagentowe,
- zidentyfikowanie poszczególnych etapów procesów projektowania oraz dekompozycja zadań projektowych na wiele komputerów w celu równoległego przetwarzania obliczeń,
- wybór odpowiednich narzędzi tj. platformy wieloagentowej oraz środowiska symulacyjnego do utworzenia systemu wieloagentowego wspomagającego projektowanie,
- implementacja czterech typów agentów działających w systemie wieloagentowym: agent interfejsu, agent nadrzędny, agent symulacyjny, agent decyzyjny,
- zdefiniowanie algorytmów wykonywanych przez poszczególne typy agentów oraz opracowanie modelu komunikacji, który umożliwiał przesyłanie pomiędzy agentami rezultatów ich obliczeń, aż do otrzymania rozwiązania końcowego (wygenerowania projektu ofertowego),
- opracowanie metod interakcji pomiędzy elementami systemu wieloagentowego, zaimplementowanymi w różnych środowiskach programistycznych, np. agentami platformy ze środowiskiem symulacyjnym i bazą danych,
- wprowadzenie licznych zabezpieczeń umożliwiających stabilną pracę wielu użytkowników w systemie, jak chociażby mechanizm szeregowania procesów i synchronizację wątków,

- zademonstrowanie działania systemu wieloagentowego na przykładzie projektowania struktury układu sterowania podsystemu elektroenergetycznego z autotransformatorem,

- porównanie wydajności systemu wieloagentowego z systemem ekspertowym (zaczerpniętym z literatury) podczas projektowania identycznej struktury systemu sterowania przez oba systemy.

Jako narzędzia do implementacji systemu wieloagentowego użyto uniwersalnej platformy wieloagentowej UMAP opracowanej na Politechnice Śląskiej w Gliwicach, którą autorzy udostępnili do darmowego użytku wraz z kodem źródłowym [38, 39]. Do symulacji zagadnień projektowych przez poszczególne typy agentów częściowo wykorzystane zostały gotowe komponenty, opracowane w ramach publikacji, prac dyplomowych oraz grantów realizowanych głównie w Katedrze Automatyki Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej oraz elementy przykładowego modelu z biblioteki „Marine Full Electric Propulsion Power System" środowiska Matlab Simulink [23-27, 40]. Ze względu na specyfikę architektury systemów wieloagentowych niezbędne było wprowadzenie w użytych modelach i innych komponentach szeregu znaczących modyfikacji, aby umożliwić ich wykorzystanie ich przez system rozproszony.

Podczas realizacji części projektowej rozprawy doktorskiej zidentyfikowano potencjalne usprawnienia, których implementacja przyczyniła się do zwiększenia wydajności systemu wieloagentowego. Był to wspomniany wcześniej mechanizm szeregowania procesów, odpowiadający za kolejkowanie i przydzielanie zasobów sprzętowych dla określonych procesów projektowych. Oprócz tego dodano możliwość rozpoznawania określonych wzorców przez system wieloagentowy, dzięki opracowaniu dedykowanego algorytmu uczenia maszynowego, bazującego na tablicach asocjacyjnych. Umożliwiło to natychmiastowe otrzymywanie rozwiązania zadania projektowego, w przypadku gdy podobne zagadnienie było już wcześniej przetwarzane przez system wieloagentowy. Ponadto, porównanie zaproponowanej nowej metody uczenia maszynowego z innymi rozwiązaniami (takimi jak sztuczne sieci neuronowe oraz sieci Bayesa) wykazały dobrą jakość implementacji i odpowiedni dobór metodologii do rozwiązania problemu naukowego [35, 45].

Oprócz potwierdzenia poprawności funkcjonalnej utworzonego systemu wieloagentowego (poprzez sprawdzenie jego działania przy projektowaniu struktury układu sterowania okrętowego podsystemu elektroenergetycznego z rozruchem autotransformatorowym), zweryfikowane zostały również jego parametry wydajnościowe. Poprzez wydajność oprogramowania rozumiana jest ilość operacji obliczeniowych, jaką dany program/system jest w stanie wykonać w określonej jednostce czasu. W przypadku systemu wieloagentowego, z którego korzysta jednocześnie wielu użytkowników, pragnących uzyskać rozwiązanie w jak najkrótszym czasie, aspekt ten był kluczowy. Dokonano porównania wydajności zaimplementowanego systemu wieloagentowego ze scentralizowanym systemem ekspertowym podczas wspomagania projektowania tej samej struktury układu sterowania (wymagane było przy tym dodanie struktury z rozruchem autotransformatorowym do systemu ekspertowego). Czas potrzebny do wykonania trzech iteracji obliczeń przez system wieloagentowy był o 65,7% krótszy, niż w przypadku systemu ekspertowego (155 sekund vs. 452 sekund) [35, 37].

Pozostałe wyniki badań przedstawione w rozdziale 7 wskazują jednoznacznie, że utworzony system oraz jego algorytmy pomocnicze prezentują lepszą wydajność, niż rozwiązania zaproponowane w literaturze, co przyczynia się do poszerzenia obecnego stanu wiedzy z zakresu systemów wieloagentowych wspomagających projektowanie [35, 37, 45]. Następujące rezultaty pracy badawczej, związanej z przygotowywaniem rozprawy doktorskiej zostały opublikowane przez autora w recenzowanych wydawnictwach naukowych:

- koncepcja wdrożenia systemów wieloagentowych do zagadnień projektowania podsystemów elektroenergetycznych statków [36],
- architektura systemu wieloagentowego wspomagającego projektowanie wybranych układów sterowania podsystemów okrętowych [30],
- porównanie działania systemów scentralizowanych i rozproszonych podczas rozwiązywania zadań projektowych dotyczących układów automatyki statków [37],
- koncepcja, implementacja i badania symulacyjne mechanizmu szeregowania procesów bazującego na pomiarach wykorzystania CPU, który dedykowany jest dla systemów wieloagentowych [45],
- koncepcja, implementacja i badania wydajnościowe algorytmu uczenia maszynowego opartego o tablice asocjacyjne, który utworzony został na potrzeby wdrożenia do systemu wieloagentowego [35].

Kierunek dalszych badań skupiać się będzie na uzupełnieniu bazy wiedzy i bazy modeli w celu dodania nowych struktur, takich jak np. układ sterowania rozruchem za pomocą rezystancji włączonej w obwód stojana silnika indukcyjnego. Zbadane zostaną również możliwości dalszego rozwoju algorytmu uczenia maszynowego, który bezpośrednio przyczynia się do zwiększenia wydajności systemu wieloagentowego. W pierwszej kolejności poprawiona zostanie jego złożoność czasowa podczas przeszukiwania tablic asocjacyjnych. Początkowo uzyskana złożoność liniowa O(n) przedstawiona na rysunku Rys. 16. zredukowana może zostać do stałej O(1) poprzez zastosowanie, chociażby kolekcji słownikowej (ang. Dictionary) i przechowywaniu całej struktury danych w pamięci operacyjnej. Wyeliminowano by wówczas narzut związany z przeszukiwaniem pliku *.dat, który znajduje się fizycznie na dysku. Wymagane byłoby wtedy tylko jednokrotne wczytanie/zapisanie zawartości kolekcji do pliku *.dat podczas inicjalizacji/deinicjalizacji agenta Master. Rozważyć można również synchronizowanie tablic asocjacyjnych, jeżeli w systemie działa kilka instancji agentów nadrzędnych (synchronizacja mogłaby zostać przeprowadzona np. podczas stanu bezczynności systemu). Pozwoliłoby na bardzo szybkie uzyskanie nowych danych uczących, ponieważ agenci nadrzędni wymienialiby się zebranymi przez siebie tablicami asocjacyjnymi [35]. Niewątpliwym sukcesem byłoby także komercyjne wdrożenie zaimplementowanego systemu wieloagentowego lub udostępnienie go na użytek przemysłowy na zasadzie licencji open source.

BIBLIOGRAFIA

1. Tomiyama T.: *Intelligent computer-aided design systems: Past 20 years and future 20 years*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 21, 2007, s. 27-29
2. Saric I., Muminovic A., Colic M., Rahimic S.: *Development of integrated intelligent computer-aided design system for mechanical power-transmitting mechanism design*, Advances in Mechanical Engineering 9, 2017, s. 1-16
3. Wang H., Zhang H.: *A distributed and interactive system to integrated design and simulation for collaborative product development*, Robotics and Computer-Integrated Manufacturing 26, 2010, s. 778-789
4. Karayel D., Ozkan S., Vatansever F.: *Integrated knowledge-based system for machine design*, Advances in Mechanical Engineering 5, 2013
5. Thilmany J.: *Putting artificial intelligence to work in CAD design*, Technology for Design and Engineering, 2017
6. Ledzius R. C., Stoddard R. J., Hupp G. A., Sahni S. B.: *Methods and systems for system design automation (sda) of mixed signal electronic circuitry including embedded software designs*, US Patent US20180218102A1, 2017
7. Lemley J., Bazrafkan S., CorcoranP.: *Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision*, IEEE Consumer Electronics 6, 2017, s. 48-56
8. Wisskirchen G., Biacabe B. T., Bormann U., Muntz A., Niehaus G., Soler G. J., Brauchitsch B.: *Artificial intelligence and robotics and their impact on the workplace*, IBA Global Employment Institute, 2017
9. Sydor M.: *Wprowadzenie do CAD. Podstawy komputerowo wspomaganego projektowania*, Wydawnictwo Naukowe PWN, Warszawa, 2009
10. Guerman-Gałkin S., Hrynkiewicz J., Jendrzejczak T.: *Zastosowanie technologii komputerowych do badania układów okrętowych systemów elektroenergetycznych*, Zeszyty Naukowe Akademii Morskiej w Szczecinie 1, 2004
11. Kobougias I., Tatakis E.C., Prousalidis J.M.: *PV systems installed in marine vessels: Technologies and specifications*, Advances in Power Enectronics, 2013
12. Adnanes A.K.: *Electrical installations and diesel electric propulsion*, ABB Technical Publication, 2003
13. Polski Rejestr Statków: *Przepisy klasyfikacji i budowy statków morskich, część VIII: Instalacje elektryczne i systemy sterowania*, Gdańsk, 2018
14. Weiming M.: *Development of vessel integrated power system*, Proceedings of the International Conference on Electrical Machines and Systems, 2011, s. 1-12
15. Siemens: *Improved dynamic behavior and protection for propulsion systems with SISHIPCIS drive*, https://w3.siemens.no/home/no/no/sector/industry/marine/documents/improved%20dynamic%20behavior%20and%20protection%20for%20propulsion%20systems%20with%20siship%20drive%20lv.pdf, (data dostępu 01.01.2019 r.)

16. Sørensen A.J.: *Marine control systems: Propulsion and motion control of ships and ocean structures*, Department of Marine Technology NTNU, 2012
17. Hayes-Roth F., Waterman D.A., Lenat D.B.: *Building expert systems*, Addison-Wesley Longman Publishing Co., 1983
18. Kowalski Z., Zieliński S., Dziworski J., Atendt R., Meler-Kapcia M., Olejnik B., Piotrowski J.: *System ekspercki do wspomagania projektowania układów automatyki siłowni statków*, Budownictwo Okrętowe i Gospodarka Morska 9, 1999, s. 24-28
19. Welsh M., Buxton I.L., Hills W.: *The application of an expert system to ship concept design investigations*, British Maritime Technology, 1995, s. 99-123
20. Helvacioglu S., Insel M.: *A reasoning method for a ship design expert system*, Expert Systems 22, 2005, s. 53-88
21. Sen P. Gerigk M.K.: *Some aspects of a knowledge-based expert system for preliminary ship subdivision design for safety*, 5th International Symposium on the Practical Design of Ships and Mobile Units, Newcastle, United Kingdom, 1992
22. Park J.H., Storch R.L.: *Overview of ship-design expert systems*, Expert Systems 19, 2002, s. 136-141
23. Kowalski Z., Arendt R., Meler-Kapcia M., Zieliński S.: *An expert system for aided design of ship systems automation*, Expert Systems with Applications 20, 2001, s. 261-266
24. Arendt R., van Uden E.: *A decision-making module for aiding ship system automation design: A knowledge-based approach*, Expert Systems with Applications 38, 2011, s. 410-416
25. Arendt R., Kopczyński A.: *Wykorzystanie środowiska ekspertowego do projektowania wybranych podsystemów energetycznych statków*, Pomiary Automatyka Robotyka 2, 2007
26. Arendt R., Kopczyński A.: *Zastosowanie modeli matematycznych przy projektowaniu podsystemów energetycznych statków*, Pomiary Automatyka Robotyka 12, 2008
27. Kopczyński A.: *Analiza i projektowanie układów sterowania sterami strumieniowymi statków z zastosowaniem systemu z bazą wiedzy*, Rozprawa doktorska WEIA PG, 2015
28. Weiss G.: *Multiagent systems: A modern approach to distributed artificial intelligence*, The MIT Press, 1999
29. Wooldridge M.: *An introduction to multiagent systems second edition*, John Wiley & Sons, 2002
30. Spychalski P., Arendt R.: *Rozproszony system wieloagentowy wspomagający projektowanie wybranych układów sterowania*, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej 48, 2016, s. 181-184
31. Lee K.H., Lee K.Y.: *Case-based conflict resolution in multi-agent ship design system*, Advances in Artificial Intelligence, 2005, s. 826-829
32. Türkmen B., Turan O.: *An application study of multi-agent systems in multi-criteria ship design optimisation*, Proceedings of the 3rd International Conference on Computer and IT Applications in Maritime Industries, 2004, s. 340-354
33. Cui H., Turan O.: *Application of a new multi-agent hybrid co-evolution based particle swarm optimisation methodology in ship design*, Computer-Aided Design 42, 2010, s. 1013-1027

34. Feng X.: *Ship collaborative design based on multi-agent and ontology*, Cooperative Design, Visualization, and Engineering, 2008, s. 249-252
35. Spychalski P., Arendt R.: *Machine learning in multi-agent systems using associative arrays*, Parallel Computing 75, 2018, s. 88-99
36. Arendt R., Spychalski P.: *An application of multi-agent system for ship's power systems design*, Proceedings of the 20th International Conference Transport Means, 2016, s. 380-384
37. Arendt R., Kopczyński A., Spychalski P.: *Centralized and distributed structures of intelligent systems for aided design of ship automation*, Proceedings of the 38th International Conference on Information Systems Architecture and Technology, 2017
38. Mrozek, D., Malysiak-Mrozek B., Waligora I.: *UMAP - A universal multi-agent patform for .NET developers*, Beyond Databases, Architectures, and Structures: Communications in Computer and Information Science 424, 2014, s. 300-311
39. Waligóra, I. Małysiak-Mrozek, B. Mrozek, D.: *Uniwersalna platforma wieloagentowa UMAP*, Studia Informatica 32, 2011, s. 85-100
40. Matlab: *Marine full electric propulsion power system toolbox*, https://www.mathworks.com/help/physmod/sps/examples.html, (data dostępu 01.01.2019 r.)
41. Akademia Morska w Gdyni: *Laboratorium automatyzacji systemów energetycznych statku*, http://atol.am.gdynia.pl/ase/skrypt/09_elektrownia_okretowa.pdf, (data dostępu 01.01.2019 r.)
42. Elhand: *Autotransformatory*, http://www.elhand.pl/autotransformatory, (data dostępu 01.01.2019 r.)
43. Praca zbiorowa: *Słownik terminologiczny elektryki: Aparaty elektryczne wchodzące w skład urządzeń rozdzielczych,* Część 1, Instytut Elektrotechniki, Warszawa, 1989
44. Oracle: *Database development guide: 24 using the Oracle ODBC driver*, https://docs.oracle.com/en/database/oracle/oracle-database/12.2/adfns/odbc-driver.html, (data dostępu 01.01.2019 r.).
45. Spychalski P., Arendt R.: *Mechanizm szeregowania procesów w systemie wieloagentowym wspomagającym projektowanie układów sterowania*, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej 57, 2017, s.135-136
46. Goel N., Garg R.: *A comparative study of CPU scheduling algorithms*, International Journal of Graphics & Image Processing, 2012
47. Weerdt M., Clement B.: *Introduction to planning in multiagent systems*, Multiagent and Grid Systems, 2009
48. Smola A., Vishwanathan S.V.N.: *Introduction to machine learning*, Cambridge University Press, 2008
49. Weiss G.: *Multiagent systems: A modern approach to distributed artificial intelligence*, MIT Press, 1999
50. Tan M.: *Multi-agent reinforcement learning: Independent vs. cooperative agents*, Proceedings of the 5th International Conference on Machine Learning, 1993, s. 330-337
51. Panait L., Luke S.: *Cooperative multi-agent learning: The state of the art*, Autonomous Agents and Multi-agent Systems 11, 2005, s. 387-434

52. Busoniu L., Babuška R., De Schutter B.: *A comprehensive survey of multi-agent reinforcement learning*, IEEE Transactions on Systems, Man, and Cybernetics 38, 2008, s. 156-172
53. Yang E., Gu D.: *Multi-robot systems with agent-based reinforcement learning: evolution, opportunities and challenges*, International Journal on Modelling, Identification and Control 6, 2009, s. 271-286
54. Khalil K.M., Abdel-Aziz M., Nazmy T., Salem A-B.: *MLIMAS: A framework for machine learning in interactive multi-agent systems*, Procedia Computer Science 65, 2015, s. 827-835
55. Xu J., Tekin C., Zhang S., Shaar M.: *Distributed multi-agent online learning based on global feedback*, IEEE Transactions on Signal Processing 63, 2015, s. 2225-2238
56. Tuyls K., Weiss G.: *Multiagent learning: basics, challenges, and prospects*, AI Magazine 33, 2012, s. 43-52
57. Stone P., Veloso M.: *Multi-agent systems: a survey from a machine learning perspective*, Autonomous Robotics 8, 2000, s. 345-383
58. Kotsiantis S.B.: *Supervised machine learning: A review of classification techniques*, Informatica 31, 2007, s. 249-268
59. Śnieżyński B.: *Agent-based adaptation system for service-oriented architectures using supervised learning*, Procedia Computer Science 29, 2014, s. 1057-1067
60. Silver D., Huang A., Maddison C.J., Guez A., Sifre L. et al.: *Mastering the game of GO with deep neural networks and tree search*, Nature 529, 2016, s. 484-489
61. Basheer I.A., Hajmeer M.: *Artificial neural networks: Fundamentals, computing, design, and application*, Journal of Microbiological Methods 43, 2000, s. 3-31
62. Zhao L., Jia Y.: *Neural network-based adaptive consensus tracking control for multi-agent systems under actuator faults*, International Journal of Systems Science 47, 2014, s. 1931-1942
63. Wang D., Ma H., Liu D.: *Distributed control algorithm for bipartite consensus of the nonlinear time-delayed multi-agent systems with neural networks*, Neurocomputing 174, 2016, s. 928-936
64. Grossman D., Domingos P.: *Learning Bayesian network classifiers by maximizing conditional likelihood*, Proceedings of the 21st International Conference on Machine Learning, 2004, s. 46-54
65. Pearl J., Stuart R.: *Bayesian networks*, UCLA Cognitive Systems Laboratory, 2000
66. Djuric P.M., Wang Y.: *Distributed Bayesian learning in multi-agent systems*, IEEE Signal Processing Magazine 29, 2012, s. 65-76
67. Andrews R., Diederich J., Tickle A.B.: *Survey and critique of techniques for extracting rules from trained artificial neural networks*, Knowledge-Based Systems 8, 1995, s. 373-389
68. Maymounkov P., Mazieres D.: *Kademlia: A peer-to-peer information system based on the XOR metric*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems, 2002, s. 53-65
69. Heaton J.: *Programming neural networks with Encog3 in C#,* Heaton Research, 2011
70. McCaffrey J.: *Test run - Naive Bayes classification with C#*, MSDN Magazine 28, 2013

SPIS RYSUNKÓW

SPIS TABEL