



PROJEKT INŻYNIERSKI

Edytor graficzny systemów rozmytych dla języka Python

ID projektu - 46300

Opiekun projektu - dr inż. Jerzy Dembski

Dokument nr 1: Organizacja i Infrastruktura Projektu

Streszczenie projektu:

Celem projektu jest tworzenie edytora graficznego systemu rozmytego z wykorzystaniem dowolnej biblioteki Pythona (np. pygame, opencv, opengl) pozwalającego na tworzenie i kształtowanie zbiorów rozmytych, definiowanie reguł rozmytych, wizualizację działania systemu dla zadanych wartości wejściowych oraz uczenie systemu na podstawie danych uczących metodą ANFIS wraz z przedstawieniem systemu w postaci wielowarstwowego modelu neuronowego do dalszego uczenia. System powinien pozwalać też na zapis i odczyt systemu rozmytego z pliku tekstowego, jak również przedstawienia go jako funkcji przetwarzającej dane wejściowe.

Streszczenie dokumentu:

Celem dokumentu jest zdefiniowanie organizacji i infrastruktury projektu, obejmujących organizację pracy zespołu, komunikację, spotkania, wymianę dokumentów i kodu, ze szczególnym uwzględnieniem narzędzi wspomagających.

| | |
|---------------|-------------------------------------|
| Wersja: | 1.5 |
| Data wydania: | 19.03.2025 |
| Redaktor: | Adam Zarzycki |
| Współautorzy: | Julian Kulikowski, Filip Wesołowski |
| Etap/zadanie: | 1 |
| Nazwa pliku: | OiIP_v1.5.pdf |
| Liczba stron: | 14 |



Historia zmian

| Wersja | Data | Opis zmiany |
|--------|------------|---|
| 1.0 | 12.03.2025 | Wstępne wypełnienie dokumentu |
| 1.1 | 13.03.2025 | Dodanie punktów 3. i 4. |
| 1.2 | 15.03.2025 | Wstępne dodanie punktów 1. i 2. |
| 1.3 | 17.03.2025 | Dokończenie punktów 1. i 2. |
| 1.4 | 18.03.2025 | Dodanie punktów 5. i 6. |
| 1.5 | 19.03.2025 | Sprawdzenie poprawności i spójności dokumentu |





SPIS TREŚCI

| | |
|--|-----------|
| 1. Wstęp | 4 |
| 1.1. Opis produktu | 4 |
| 1.2. Cel i zakres produktu | 4 |
| 1.3. Ograniczenia | 5 |
| 1.4. Główne etapy projektu | 5 |
| 1.5. Bibliografia | 5 |
| 2. Interesariusze i użytkownicy końcowi | 6 |
| 2.1. Interesariusze | 6 |
| 2.2. Użytkownicy końcowi | 7 |
| 3. Zespół | 9 |
| 3.1. Członkowie zespołu | 9 |
| 3.2. Umiejętności członków zespołu | 9 |
| 3.3. Obszary odpowiedzialności | 10 |
| 3.4. Tryb pracy zespołu | 10 |
| 4. Komunikacja w zespole i z interesariuszami | 11 |
| 4.1. Organizacja spotkań | 11 |
| 4.1.1. W zespole | 11 |
| 4.1.2. Z opiekunem projektu | 11 |
| 4.1.3. Z interesariuszami | 11 |
| 4.2. Środki i sposoby komunikacji | 11 |
| 4.2.1. W zespole | 11 |
| 4.2.2. Z opiekunem projektu | 11 |
| 4.2.3. Z interesariuszami | 12 |
| 5. Współdzielenie dokumentów i kodu | 13 |
| 5.1. Repozytorium i dostęp | 13 |
| 5.2. Konfiguracja repozytorium | 13 |
| 5.3. Porządek w dokumentacji | 13 |
| 5.4. Nazewnictwo plików | 13 |
| 5.5. Wersjonowanie dokumentacji | 13 |
| 6. Narzędzia | 14 |
| 6.1. Narzędzia wspomagające organizację projektu | 14 |
| 6.2. Narzędzia wspomagające dokumentację | 14 |
| 6.3. Narzędzia do wytwarzania i testowania | 14 |
| 6.4. Narzędzia do komunikacji i organizacji spotkań | 14 |



1. Wstęp

1.1. Opis produktu

Produktem końcowym projektu jest wytworzenie programu pozwalającego użytkownikowi końcowemu tworzenie, kształtowanie oraz wizualizację zbiorów rozmytych. Najważniejszą funkcją programu jest oferowanie przystępnego i interaktywnego interfejsu graficznego, za pomocą którego użytkownik jest w stanie wchodzić w interakcje ze zbiorami rozmytymi. Program oprócz tego również udostępnia możliwość zapisu i odczytu systemu rozmytego z pliku tekstowego, definiowanie reguł rozmytych oraz uczenie systemu na podstawie danych uczących. Całość projektu jest realizowana w języku Python.

Logika rozmyta jest formą logiki wielowartościowej. W logice rozumianej w sposób klasyczny wartości przyjmują tylko jeden z dwóch stanów "prawda" lub "fałsz", logika rozmyta, będąca jej uogólnieniem pozwala na przyjmowanie wartości pośrednich. Oznaczają one tylko częściową przynależność, a także określają jej stopień, im bliżej wartości "prawda" tym większa przynależność. Wykorzystuje się ją często w przypadku uczenia maszynowego, gdzie tradycyjna logika nie byłaby wystarczająca i prowadziłaby do sprzeczności lub nieścisłości systemu. Zbiory rozmyte są obiektami matematycznymi ze zdefiniowaną "funkcją przynależności". Elementy tych zbiorów mają swój stopień przynależności określany za pomocą liczby rzeczywistej z przedziału $[0, 1]$.

1.2. Cel i zakres produktu

Głównym celem projektu jest utworzenie programu, który swoją funkcjonalnością będzie odpowiadał aplikacji Fuzzy Logic Designer dostępnej w języku programowania MATLAB. Środowisko MATLAB nie jest inicjatywą darmową, ani Open Source, aby móc z niego korzystać należy wykupić kosztowną licencję na ograniczoną ilość czasu. Z kolei współczesne biblioteki, które służą do obsługi zbiorów rozmytych w języku Python, nie są wyposażone w interfejsy graficzne i wykorzystuje się je z poziomu wiersza poleceń. Nie posiadają także wizualizacji na bieżąco. Co za tym idzie, jeżeli przeciętny użytkownik potrzebuje skorzystać z logiki rozmytej jest on zmuszony kupić kosztowną licencję, której może nie potrzebować lub borykać się z ograniczeniami aktualnych implementacji.

Nasz program ma za zadanie umożliwiać pracę z logiką rozmytą, która łączy w sobie przystępność interakcji implementacji z MATLABA, ale bez potrzeby zakupu licencji. Innymi słowy pragniemy rozwiązać dylemat wyboru między łatwą pracą przez interfejs graficzny, a kosztem pieniężnym poprzez zaoferowanie darmowej alternatywy.

Pod kątem języka programowania Python ma on na celu zaoferowanie alternatywnego sposobu korzystania z aktualnie istniejących bibliotek obsługujących zbiory rozmyte. Zamiast wywoływać je z poziomu wiersza poleceń udostępnia on możliwość bardziej kontrolowanej interakcji oraz wizualizacji krok po kroku.

Potencjalnym celem końcowym produktu jest jego zastosowanie w celach dydaktycznych. Pozwala on na uczenie logiki rozmytej w interfejsie graficznym, bez potrzeby kupna licencji.



1.3. Ograniczenia

Największym ograniczeniem nałożonym na projekt jest potencjalna różniaca się implementacja logiki rozmytej w środowisku MATLAB, a w dostępnych bibliotekach języka Python. Mogą one powodować niewielkie różnice w zbiorach wynikowych. Nie powinno to jednak mieć szczególnego wpływu na projekt.

Większym, poważniejszym ograniczeniem programu jest licencja MATLAB oraz zamknięta natura implementacji aplikacji Fuzzy Logic Designer. Bez wglądu do kodu źródłowego aplikacji odwzorowywanie jej będzie w głównej mierze polegać na obserwowaniu działania, a następnie inżynierii wstecznej. Jest to dużo trudniejsze w realizacji, niż bezpośredni wgląd w kod źródłowy.

Środowisko MATLAB, w przeciwieństwie do języka Python, zostało zoptymalizowane pod kątem dużej ilości obliczeń na obszernych zbiorach danych. Oznacza to, że odwzorowana aplikacja będzie działać mniej wydajnie niż jej pierwowzór.

1.4. Główne etapy projektu

| Kod etapu | Krótki Opis Etapu | Planowany przedział czasowy |
|-----------|---|-----------------------------|
| Etap A | Zbieranie wymagań projektu. | 12.03.2025 - 31.03.2025 |
| Etap B | Inżynieria wsteczna aplikacji. | 1.04.2025 - 15.04.2025 |
| Etap C | Pierwsza implementacja aplikacji własnej. | 16.04.2025 - 31.05.2025 |
| Etap D | Spotkanie kontrolne z interesariuszami. | 01.06.2025 |
| Etap E | Wprowadzenie poprawek. Dokończenie implementacji | 02.06.2025 - 01.08.2025 |
| Etap F | Utworzenie dokumentacji produktu oraz innych związanych dokumentów. | 02.08.2025 - 30.10.2025 |
| Etap G | Prezentacja finalnego produktu interesariuszom. | 01.11.2025 |
| Etap H | Wdrożenie finalnego produktu. | 01.11.2025 - 01.12.2025 |

1.5. Bibliografia

- https://en.wikipedia.org/wiki/Fuzzy_logic,
- https://en.wikipedia.org/wiki/Fuzzy_set,
- https://mfiles.pl/pl/index.php/Fuzzy_logic,
- L. A. Zadeh (1965) "[Fuzzy sets](#)",



2. Interesariusze i użytkownicy końcowi

2.1. Interesariusze

| Interesariusz | Opis |
|---|---|
| Zleceniodawca/opiekun: dr inż. Jerzy Dembski | Oczekuje wiernego odtworzenia aplikacji Fuzzy Logic Designer. Potrzebuje, aby oferowała ona te same funkcjonalności oraz potrafiła przeprowadzić te same operacje matematyczne. Najważniejszym oczekiwaniem interesariusza jest implementacja interfejsu graficznego, który przedstawia wyniki obliczeń. |
| Pracownicy Katedry Inteligentnych Systemów Interaktywnych | Od dłuższego czasu borykają się z brakiem odpowiedniego narzędzia dydaktycznego do nauczania zbiorów rozmytych. Oczekują od aplikacji przystępności i możliwości pokazania kolejnych kroków podczas tworzenia reguł rozmytych. |
| Studenci Politechniki Gdańskiej | Studentom, podobnie jak i pracownikom przede wszystkim zależy na przystępności aplikacji. Potrzebują możliwości zrozumienia co robią krok po kroku. Najważniejszy jest dla nich przyjazny oraz intuicyjny interfejs. Oczekują także, że aplikacja będzie ogólnodostępna. |
| PyTorch | PyTorch jest popularną i powszechnie używaną biblioteką obsługującą różne funkcje wykorzystywane w uczeniu maszynowym. Zbiory rozmyte Sugeno są często wykorzystywane jako w celu tworzenia danych wejściowych w uczeniu maszynowym. Kompatybilność projektu usprawniłaby w dużym stopniu działanie tej biblioteki. |
| Użytkownicy zewnętrzni | Użytkownicy, którzy nie są związani z Politechniką Gdańską, ale jednocześnie również potrzebują aplikacji do realizacji logiki rozmytej. Najważniejszym dla nich aspektem jest posiadanie alternatywy dla płatnego oprogramowania MATLAB. |



2.2. Użytkownicy końcowi

| Użytkownik | Specyfikacja | Opis Specyfikacji |
|---|---|---|
| Studenci Politechniki Gdańskiej | Profil | Należy założyć, że studenci nigdy wcześniej nie mieli kontaktu z logiką rozmytą. Powinna być mu zaoferowana intuicyjna obsługa systemu, możliwość zobaczenia konsekwencji działań krok po kroku, a także możliwość cofnięcia połączonych działań. |
| | Warunki wykorzystania systemu | Zakłada się, iż system będzie przede wszystkim wykorzystywany w warunkach laboratoryjnych. System musi być kompatybilny z oprogramowaniem występującym na komputerach znajdujących się w laboratoriach. |
| | Wymagania względem interfejsu użytkownika | Jako, iż zakłada się, że studenci nie mieli wcześniej styczności z logiką rozmytą najważniejszym wymaganiem jest przystępność. Interfejs powinien naprowadzać użytkownika na odpowiednie kolejne kroki oraz jasno przekazywać swój stan. |
| Pracownicy Katedry Inteligentnych Systemów Interaktywnych | Profil | Zakłada się, iż są to osoby, które miały znaczny kontakt z systemami rozmytymi. Korzystali już z aplikacji w języku MATLAB i są zaznajomieni z jej funkcjonalnościami. Zakłada się też, że część z nich oprócz wykorzystania aplikacji w projektach prywatnych będzie z niej korzystać w celach dydaktycznych podczas zajęć. |
| | Warunki wykorzystania systemu | System będzie wykorzystywany zarówno w warunkach laboratoryjnych (główne założenia), a także do projektów prywatnych. Co za tym idzie jego implementacja powinna wspierać systemy przez nich wykorzystywane prywatnie. Zdecydowanie potrzebne jest wsparcie dla systemów Windows, MacOS i popularniejszych dystrybucji Linux. |



| | | |
|------------------------|---|--|
| | Wymagania względem interfejsu użytkownika | Zakłada się, że pracownicy politechniki mają doświadczenie z logiką rozmytą, a także z podobnymi aplikacjami służącymi do jej obsługi. Co za tym idzie interfejs użytkownika powinien im udostępniać możliwość szybkiego i efektywnego pracy z programem za pomocą skrótów klawiszowych. Najważniejsze funkcje powinny być możliwe do realizacji za pomocą klawiatury. |
| Użytkownicy zewnętrzni | Profil | Do tych użytkowników, zaliczają się osoby, które nie są związane z Politechniką Gdańską, ale także zyskałyby na darmowej aplikacji wizualizującej zbiory rozmyte. Jest to bardzo szeroki podzbiór osób, który zawiera wszystkich od użytkowników z zerowym doświadczeniem, po tych którzy pracują z logiką rozmytą na co dzień. |
| | Warunki wykorzystania systemu | Należy tu założyć, że z tak szerokim gronem potencjalnych użytkowników końcowych aplikacja będzie uruchamiana na każdym możliwym sprzęcie wspierającym język Python. Próba wsparcia tak różnorodnego zbioru maszyn w jednakowym stopniu jest nierealistyczna. Należy zidentyfikować najczęściej wykorzystywane maszyny oraz systemy operacyjne i skupić się na nich. |
| | Wymagania względem interfejsu użytkownika | Tutaj także należy założyć jak najbardziej zróżnicowane doświadczenie, a co za tym idzie wymagania. Interfejs powinien naprowadzać użytkowników niedoświadczonych, a jednocześnie pozwalać doświadczonym na szybką i produktywną pracę. |



3. Zespół

3.1. Członkowie zespołu

| Imię i nazwisko | Dane kontaktowe |
|-------------------|---------------------------|
| Adam Zarzycki | s193243@student.pg.edu.pl |
| Filip Wesołowski | s193486@student.pg.edu.pl |
| Julian Kulikowski | s188898@student.pg.edu.pl |

3.2. Umiejętności członków zespołu

| Imię i nazwisko | Umiejętności |
|-------------------|---|
| Adam Zarzycki | <ul style="list-style-type: none">• wysoki poziom zaawansowania w języku Python,• doświadczenie w pracy z bibliotekami graficznymi, w tym również w języku Python,• łatwość samoorganizacji pracy,• zdolności kierownicze,• zmysł analityczny. |
| Filip Wesołowski | <ul style="list-style-type: none">• wysoki poziom zaawansowania w języku Python,• doświadczenie w pracy z bibliotekami graficznymi, w tym również w języku Python,• łatwość wyszukiwania potrzebnych informacji w źródłach,• dobre wyczucie estetyki. |
| Julian Kulikowski | <ul style="list-style-type: none">• wysoki poziom zaawansowania w języku Python,• dobra znajomość programu MATLAB,• doświadczenie w projektowaniu i implementacji logiki aplikacji graficznych,• łatwość podejmowania decyzji,• wysoka skrupulatność. |



3.3. Obszary odpowiedzialności

| Imię i nazwisko | Obszary odpowiedzialności |
|-------------------|--|
| Adam Zarzycki | <ul style="list-style-type: none">• zaprojektowanie i implementacja interfejsu graficznego aplikacji,• opracowanie projektu systemu,• przeprowadzenie testów systemowych i akceptacyjnych aplikacji,• zarządzanie i organizacja prac zespołu,• komunikacja z interesariuszami,• opracowanie dokumentacji. |
| Filip Wesołowski | <ul style="list-style-type: none">• zaprojektowanie i implementacja interfejsu graficznego aplikacji,• opracowanie projektu systemu,• wyszukiwanie informacji z obszaru problemowego, potrzebnych do realizacji projektu,• przeprowadzenie testów systemowych i akceptacyjnych aplikacji,• opracowanie dokumentacji. |
| Julian Kulikowski | <ul style="list-style-type: none">• opracowanie projektu systemu,• implementacja logiki biznesowej aplikacji,• przeprowadzenie testów jednostkowych i integracyjnych aplikacji,• opracowanie dokumentacji. |

3.4. Tryb pracy zespołu

Zespół przez większość czasu trwania projektu inżynierskiego będzie pracować w rozproszeniu, z ewentualnymi wyjątkami pod postacią spotkań stacjonarnych z promotorem lub innymi interesariuszami (patrz punkt 4.).



4. Komunikacja w zespole i z interesariuszami

4.1. Organizacja spotkań

4.1.1. W zespole

Spotkania zespołu organizowane będą co tydzień, w celu ewaluacji postępów prac poszczególnych jego członków i wymiany posiadanej wiedzy i spostrzeżeń. Ewentualnie dopuszcza się możliwość przeprowadzenia spotkania w terminie krótszym niż tydzień od poprzedniego, w związku z wystąpieniem zdarzeń nieplanowanych (zmiana wymagań interesariuszy, skrócenie terminu realizacji projektu) i/lub losowych.

Dokładna data spotkania ustalana będzie przez członków zespołu za pomocą środków opisanych w punkcie 4.2.1.

4.1.2. Z opiekunem projektu

Spotkania ewaluacyjne z opiekunem projektu będą się odbywać co dwa tygodnie, w celu podsumowania wykonanych od ostatniego spotkania prac i ustalenia zadań priorytetowych dla dalszego rozwoju projektu.

4.1.3. Z interesariuszami

Nie przewiduje się regularnych spotkań z interesariuszami (poza opiekunem projektu).

4.2. Środki i sposoby komunikacji

4.2.1. W zespole

Członkowie zespołu będą się ze sobą komunikować poprzez prywatny kanał założony na platformie Discord. Spotkania będą się odbywać zdalnie, również z wykorzystaniem tej platformy.

4.2.2. Z opiekunem projektu

Spotkania z opiekunem projektu będą się odbywać stacjonarnie, w starym budynku wydziału Elektroniki, Telekomunikacji i Informatyki na Politechnice Gdańskiej, w sali EA422. W wyjątkowych sytuacjach braku możliwości fizycznego stawienia się członków zespołu na wydziale, dopuszczalne jest przeprowadzenie spotkania online poprzez platformę Teams.

Ustalanie konkretnych terminów spotkań, jak również cała dodatkowa komunikacja z opiekunem projektu, przeprowadzana będzie za pomocą serwisu pocztowego Microsoft Outlook.



4.2.3. Z interesariuszami

W celu pozyskania informacji o pożądanых przez interesariuszy funkcjonalnościach, jak również opinii na temat prototypowych wersji projektu, przeprowadzane będą ankiety i kwestionariusze, przygotowane za pomocą platformy Google Forms. Ogłaszane one będą z użyciem serwisu pocztowego Microsoft Outlook.





5. Współdzielenie dokumentów i kodu

5.1. Repozytorium i dostęp

Wszystkie pliki projektowe, w tym dokumentacja i kod, będą przechowywane w repozytorium Git, w naszym przypadku w repozytorium GitHub. Każdy członek zespołu będzie miał dostęp do repozytorium za pomocą linku i swojego konta, co umożliwia ścisłą kontrolę nad wersjami plików i łatwą wymianę informacji.

- **Adres repozytorium:** <https://github.com/IkeaSzark/RPI>,
- **Sposób dostępu:** Repozytorium jest dostępne publicznie z uprawnieniami do zapisu dla członków zespołu.

5.2. Konfiguracja repozytorium

Filip Wesołowski będzie odpowiedzialny za konfigurację repozytorium, zarządzanie gałęziami i aktualizację repozytorium na GitHubie, a także zapewni przestrzeganie standardów w zakresie struktury plików i commitów.

5.3. Porządek w dokumentacji

Adam Zarzycki będzie odpowiedzialny za organizację dokumentacji projektu, zarówno pod względem wersjonowania, jak i jakości dokumentów. Będzie dbał o to, by wszystkie zmiany w dokumentacji były odpowiednio opisane w commitach, a wersje dokumentów były spójne z wersjami kodu.

5.4. Nazewnictwo plików

Nazewnictwo plików będzie zgodne z następującymi zasadami:

- **Pliki dokumentacji:** [nazwa_pliku]_[wersja].pdf (np. OilP_v1.0.pdf),
- **Pliki kodu:** [nazwa_modułu].[rozszerzenie] (np. interfejs.py).

5.5. Wersjonowanie dokumentacji

Dokumentacja będzie wersjonowana za pomocą repozytorium Git. Każda zmiana w dokumencie (np. dodanie sekcji, zmiana treści) będzie rejestrowana jako nowy commit z odpowiednim opisem, co pozwoli na śledzenie historii dokumentu.



6. Narzędzia

6.1. Narzędzia wspomagające organizację projektu

- **Git i GitHub:** Narzędzia do zarządzania kodem źródłowym oraz śledzenia wersji, umożliwiające współpracę w zespole,
- **Trello:** Używane do zarządzania zadaniami i organizacji pracy zespołu.

6.2. Narzędzia wspomagające dokumentację

- **Google Docs:** Współdzielona przestrzeń do wstępnych wersji dokumentacji,
- **Overleaf LaTeX:** Tworzenie i formatowanie profesjonalnych dokumentów.

6.3. Narzędzia do wytwarzania i testowania

- **Python:** Główny język programowania do realizacji aplikacji. Użycie bibliotek jak **Pygame** i **OpenCV**,
- **PyTest:** Narzędzie do testowania jednostkowego,
- **Selenium:** Automatyzacja testów aplikacji.

6.4. Narzędzia do komunikacji i organizacji spotkań

- **Discord** będzie używany jako główny środek komunikacji między zespołem.