Salomon
System przetwarzania wiedzy

Vision Scope, ver 1.2

Historia wersji

Data	Wersja	Opis	Autorzy
18-04-2005	1.0	Początkowa wersja Vision pierwszego etapu	Przemysław Misiuda, Dominik Seweryn
27-04-2005	1.1	Poprawiona wersja Vision	Przemysław Misiuda, Dominik Seweryn
07-10-2005	1.2	Auktualizacja dokumentu Vision, etap drugi	Dominik Seweryn

1 Cel

Celem projektu jest rozbudowa platformy Salomon o obsługę drzew decyzyjnych oraz implementacja algorytmów budowy i wizualizacji drzew decyzyjnych.

2 Ramy i założenia projektowe

Mamy do dyspozycji platformę Salomon służącą jako platforma do uruchamiania zadań. Platformę tą będziemy rozbudowywać o nową funkcjonalność - obsługę drzew decyzyjnych, polegającą na opracowaniu struktury w bazie danych przeznaczonej do przechowywania drzew decyzyjnych i obsłudze takich struktur. Implementacja algorytmów budowy drzew decyzyjnych oraz wstępnej wizualizacji będzie realizowana w postaci pluginów obsługiwanych przez tą platformę.

3 Opis użytkownika

W przypadku niniejszego projektu będziemy mieli do czynienia z dwoma rodzajami użytkowników. Pierwszy z nich to tzn. użytkownik końcowy - osoba pragnąca korzystać z funkcjonalności, jaką chcemy zaoferować w zakresie budowy drzew decyzyjnych - korzystania z zaimplementowanych przez nas algorytmów. Tu istotne będą funkcje związane bezpośrednio z przygotowaniem danych na potrzeby algorytmu, wyborem typu algorytmu, prezentacją uzyskanych wyników. Drugi rodzaj użytkownika to ktoś pragnący w przyszłości wykorzystać rozbudowaną w tym projekcie funkcjonalność platformy o reprezentację drzew decyzyjnych. W tym przypadku ważne będą informacje o interfejsach i opis ich funkcji.

4 Wymagania projektowe

4.1 Funkcjonalne

4.1.1 Wymagania funkcjonalne

Interfejs:

- ekran do wyboru źródła danych (baza, tabela)
- wskazanie kolumny decydującej o przydziale elementów do poszczególnych gałęzi w drzewie
- ustalanie które atrybuty (które kolumny tabeli) mają być użyte przy tworzeniu drzewa

- ustalanie miejsca składowania wyników działania algorytmu
- wybór algorytmu tworzenie drzewa
- uruchamianie
- wizualizowanie wyników obliczeń - graf ilustrujący stworzone zależności

4.2 Niefunkcjonalne

4.2.1 Systemowe

- Stworzenie kodu i api umożliwiającego łatwe dołączenie innych implementacji algorytmu tworzenia drzew decyzyjnych.
- Napisanie intuicyjnej dokumentacji użytkownika i dokumentacji do kodu.
- Stworzenie przykładów ilustrujących działanie pluginu.

4.2.2 Jakościowe

- Napisanie wydajnego algorytmu budowy drzew decyzyjnych
- Reprezentacja drzew decyzyjnych w sposób umożliwiających ich łatwą wizualizację

5 Architektura i narzędzia

System nasz będzie rozbudowywał platformę Salomon (o obsługę drzew decyzyjnych) oraz dostarczał trzy pluginy dla tej platformy - jeden dla określenia tabeli i kolumn, które posłużą do konstrukcji drzewa, drugi dla algorytmu do tworzenia drzew decyzyjnych, trzeci dla wizualizacji tych drzew. Większość procesu kodowania systemu będzie polegała na zaimplementowaniu zdefiniowanych przez twórców platformy Salomon interfejsów. Z tej przyczyny nie będą nas interesowały zagadnienia połączenia z bazą danych, warstwy sieciowej itp., będą one obsługiwane przez platformę Salomon. Konfigurowanie pluginów będzie dokonywane przez graficzny interfejs użytkownika ukazujący się po wybraniu odpowiedniej wtyczki. W procesie kodowania będziemy używać IDE Eclipse Dzięki możliwości eksportowania ustawień będziemy mogli korzystać z tych samych ustawień, co twórcy platformy Salomon.

6 Ryzyko

6.1 Niewykryte dotąd błędy w rozwijanych pluginach

6.1.1 Opis ryzyka

Istnieje zagrożenie, iż w kodzie pluginów nadal występują błędy, które mimo testów nie zostały wykryte. Dotyczyć to może zwłaszcza pluginu drugiego i algorytmu budowy drzew decyzyjnych.

6.1.2 Ranga ryzyka

Średnie.

6.1.3 Wpływ na projekt

Znaczący. Może spowodować opóźnienia w realizacji projektu o czas potrzebny na zlokalizowanie błędów i ich naprawienie. Im później błąd zostanie dostrzeżony tym trudniej bedzie ustalić jego źródło, gdyż może nie być jasne czy pochodzi on z nowo napisanego kodu wprowadzającego dodatkowe funkcjonalności, czy też z dotychczas istniejącej części.

6.1.4 Metody wykrywania

Tego typu zagrożenie może być trudne do wykrycia, jeżeli nie zostało dotychczas zidentyfikowane w procesie testów. Podejrzenia powinno budzić np. pojawianie się dziwnych wyników przy testowaniu kolejnego zaimplementowanego algorytmu, błąd algorytmu budowy drzew decyzyjnych podczas swej pracy, czy zachowanie sie pluginu w sposób odmienny niż było to ustalone za pomocą zmiennych. Nie musi to od razu oznaczać błędów tkwiących w początkowych wersjach tworzonych pluginów, ale trzeba i taką ewentualność brać pod uwagę.

6.1.5 Strategia redukcji zagrożenia

Dokładne testy na kilku zbiorach danych dotychczas istniejącej implementacji pluginów. Gruntowne testy klas, metod, interfejsów.

6.2 Wprowadzenie błędów do istniejącej funkcjonalności podczas rozbudowywania pluginów

6.2.1 Opis ryzyka

W trakcie rozbudowywania pluginów może wystąpić sytuacja, że pojawią się błędy w dotychczas dobrze działającej funkcjonalności.

6.2.2 Ranga ryzyka

Średnie.

6.2.3 Wpływ na projekt

Średni. Pojawienie się błędów w testach, które wykonywały się dotychczas bez problemów bedzie z dużym prawdopodobieństwem oznaczało, iż gdzieś w kodzie wprowadzony został błąd przy okazji dodawania nowej funkcjonalności. Błąd tego typu będzie powodował opóźnienie w realizacji projektu związane z czasem potrzebnym na jego identyfikacje i naprawienie, ewentualnie także modyfikacje tego co zostało dopisane jako nowa funkcjonalność.

6.2.4 Metody wykrywania

Testy regresyjne.

6.2.5 Strategia redukcji zagrożenia

Intensywne testy, protokół odbioru.