Salomon

System przetwarzania wiedzy

Moduł:
Drzewa decyzyjne

Software Architect Document, ver 1.1

Historia wersji

Data	Wersja	Opis	Autorzy
18-04-2005	1.0	Pierwsza wersja dokumentu	Mateusz Nowakowski
30-09-2005	1.1	Aktualizacja po zaimplementowaniu pierwszej wersji produktu	Mateusz Nowakowski
08-11-2005	1.2	Opis pluginu wnioskującego Aktualizacja większości rozdziałów	Mateusz Nowakowski
15-11-2005	1.3	Poprawa rozdziału 3. Przypadki użycia	Mateusz Nowakowski

Spis treści

Spis rysunków

1 Wprowadzenie

Celem dokumentu jest określenie architektury modułu Salomona obsługującego drzewa decyzyjne.

2 Założenia oraz cele architekturalne

2.1 Założenia, ograniczenia

Moduł służący do obsługi drzew decyzyjnych jest częścią platformy Salomon, w związku z tym zostały na niego nałożone następujące ograniczenia:

- Konfiguracja oraz model drzew decyzyjnych muszą się znajdować w tej samej bazie. Dostęp do jakichkolwiek źródeł danych musi zapewniać Salomon.
- Salomon narzuca architekturę pluginową. API obsługi drzew decyzyjnych musi być doimplementowany do Salomona, natomiast cała obsługa drzew decyzyjnych musi znajdować się w pluginach.
- Układ pakietów. Interfejsy drzew muszą znajdować się w *salomon.platform.data.tree*, natomiast ich implementacja w *salomon.engine.data.tree*
- Platforma Salomon zakłada możliwość obsługi zdalnych obiektów. W związku z tym pluginy nie tworzą ani nie zarządzają bezpośrednio danymi. Tworzą, modyfikują dane tylko i wyłącznie za pomocą metod wystawianych przez zestaw manager'ów. Pluginy operują również tylko za pomocą interfejsów, a nie konkretnych instancji klas.

Moduł drzew decyzyjnych korzysta również z realizowanych właśnie rozszerzeń Salomona o tzw. *rozwiązania* (*solution*). Ograniczenia i możliwości z tego płynące są następujące:

- Pluginy uruchamiane są z poziomu rozwiązania, co umożliwia dostęp do dwóch baz danych:
- Bazy definicyjnej - bazy przechowującej konfigurację Salomona. Wszelkie informacje dotyczące zbudowanych drzew decyzyjnych będą się znajdować w tej bazie
- Baza wejściowa - dodatkowa baza określona w rozwiązaniu. Z punktu widzenia modułu drzew decyzyjnych jest to baza przechowująca potencjalne dane, na podstawie których mogą być stworzone drzewa decyzyjne

2.2 Cele architekturalne

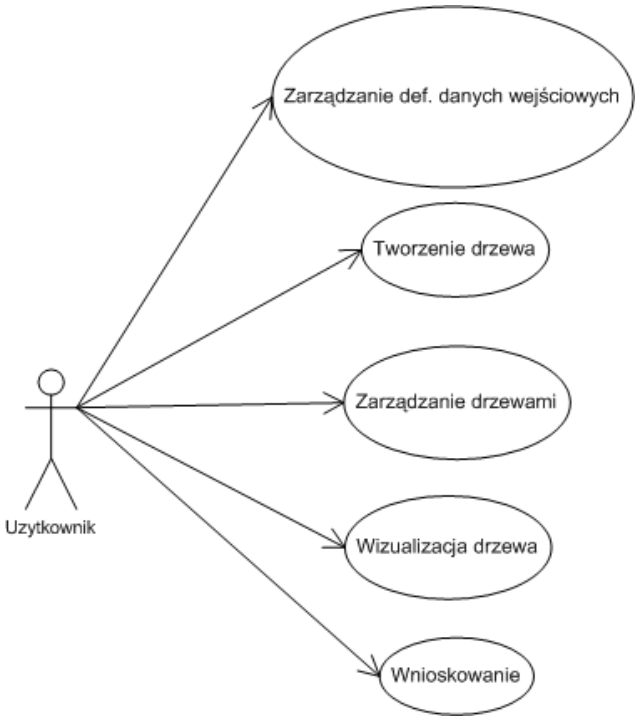
Cele architekturalne wynikające z powyższych ograniczeń oraz z założeń funkcjonalnych są następujące:

- Rozszerzyć core platformy Salomon o obsługę drzew decyzyjnych. Udostępnić API pluginom umożliwiające im tworzenie usuwanie, przeglądanie oraz usuwanie drzew decyzyjnych.
- Stworzyć zestaw pluginów obsługujących powyższe operacje na drzewach decyzyjnych.

3 Przypadki użycia

Poniższy opis zakłada że użytkownik uruchomił platformę Salomon oraz poprawnie zdefiniował *solution* oraz pragnie wykonać operacje na drzewach decyzyjnych.

Poniższy diagram przedstawia punkt widzenia użytkownika.



Rysunek 1: Diagram przypadków użycia

Aby zdefiniować jakiekolwiek drzewo decyzyjne użytkownik musi określić na jakich danych pragnie je stworzyć. Tworzenie drzewa polega na wyborze algorytmu budującego oraz odpowiednie skonfigurowanie jego parametrów. Użytkownik musi również

mieć możliwość zarządzania istniejącymi drzewami oraz móc je odpowiednio wizualizować. Drzewa decyzyjne tworzy się na pewnym małym zbiorze danych by następnie zweryfikować ich poprawność na wiekszym zbiorze danych. Użytkownik zatem powinien mieć możliwośc wykorzystania drzew w praktyce.

Powyższe przypadki użycia oraz założenia platfomy Salomon prowadza do następujących operacji użytkownika w ramach platformy.

Użytkownik wybiera najpierw tzw. *plugin wejściowy*. Następnie użytkownik wybiera zdefiniowane wcześniej dla obecnego *solution*'a dane wejściowe lub nowe poprzez zdefiniowanie nazwy danych wejściowych (dla późniejszej identyfikacji), wybiera tabelę a następnie kolumnę reprezentującą wynik decyzji – *kolumna decyzyjna* oraz listę kolumn reprezentujące przesłanki decyzji – *kolumny decydujące*. Następnie musi określić, na których wierszach będzie budowane drzewo decyzyjne. Użytkownik może wybrać więcej niż jedno źródło danych, z których potem będą generowane drzewa decyzyjne.

Następnie użytkownik wybiera pluginy algorytmiczne, które utworzą drzewa decyzyjne na podstawie wcześniej zdefiniowanych danych wejściowych. Konfiguracją pluginów algorytmicznych opiera się (jeżeli jakakolwiek występuje) na zdefiniowaniu parametrów algorytmu, ktorego używa plugin.

Jeśli użytkownik sobie życzy może wybrać również plugin wizualizacyjny, który przedstawi stworzone wcześniej drzewa. Plugin ten ma możliwość działać również samodzielnie. Użytkownik konfigurując plugin może sam określić jakie istniejące drzewa mają zostać zwizualizowane. Może również usunąć wybrane drzewa decyzyjne.

Każde drzewo decyzyjne można zweryfikować. Samodzielny plugin wnioskujący umożliwia przeprowadzenie wnioskowania za pomocą wybranego drzewa na innym zbiorze danych. Użytkownik wybiera drzewo oraz zbiór danych, następnię w oknie wyników przedstawione są wnioski np. procent poprawności, które wiersze zawierają inny wynik niż spodziewany itp.

3.1 Developer Use-Case View

Osoby chcące rozwijać moduł drzew decyzyjnych mogą w zasadzie rozwijać go tylko o dodatkowe pluginy algorytmiczne, gdyż plugin wejściowy oraz plugin wizualizacyjny są wspólne. Deweloper będzie musiał zaimplementować tylko określony interfejs lub klasę abstrakcyjną (decyzja na poziomie implementacji).

4 Model logiczny

4.1 Ogólne założenia

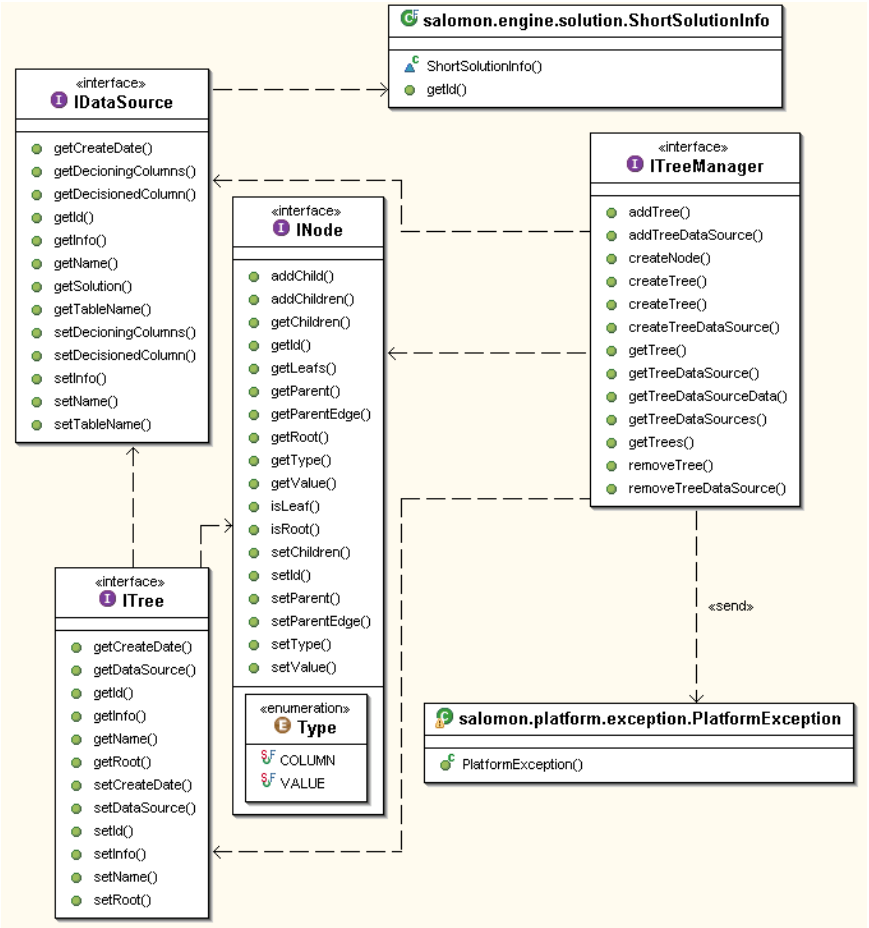
Moduł drzew decyzyjnych składa się z dwóch części:

- Rozszerzenia jądra platformy Salomon o podstawową obsługę drzew decyzyjnych
- Zestawu 4 typów pluginów: pluginu definiującego dane wejściowe, pluginu algorytmicznego, pluginu wizualizującego, zarządzającego drzewami oraz pluginu wnioskującego

4.2 Model rozszerzeń jądra Salomona

Rozszerzenia jądra Salomona polega na dostarczeniu interfejsów oraz ich implementacji definiujących i zarządzających drzewami decyzyjnymi. Jądro Salomona będzie dostarczać mechanizmów zapisu nowych oraz odczytu informacji o istniejących drzewach decyzyjnych i nic więcej. Resztę implementują pluginy.

Poniższy diagram przedstawia układ interfejsów do zaimplementowania:



Rysunek 2: Model rozszerzeń

4.3 Architektura pluginów

Jak wcześniej wspomniano będą 4 rodzaje pluginów:

- Plugin wejściowy - definiujący dane na podstawie których będą tworzone drzewa decyzyjne.
- Plugin algorytmiczny - tworzący drzewa na podstawie określonych danych wejściowych
- Plugin wizualizacyjny, zarządzający drzewami istniejącymi drzewami.
- Plugin wnioskujący - przeprowadzajacy testy drzew na zbiorach danych.

Pluginy są zgodne a architekturą Salomona i sposób ich budowania jest przez nią określony.

4.3.1 Plugin wejściowy

Plugin wejściowy zarządza istniejącymi definicjami danych wejściowych (dodaje, usuwa) jak również określa, na podstawie jakich danych wejściowych późniejsze w kolejce pluginy algorytmiczne mają tworzyć drzewa.

Pluginy komunikują się poprzez środowisko ustawiając w nich zmienne. Plugin, po wyborze zestawu danych wejściowych ustawia w środowisku listę identyfikatorów danych wejściowych i kończy pracę.

4.3.2 Plugin algorytmiczny

Plugin algorytmiczny ma niezależną konfigurację oraz działanie (zależne od algorytmu). Natomiast komunikacja z sąsiednimi pluginami jest ściśle określona. Każdy plugin algorytmiczny w kolejce sprawdza zawartość zmiennej środowiskowej Salomona z listą identyfikatorów danych wejściowych, pobiera z jądra Salomona szczegółowe informacje o nich i tworzy drzewo decyzyjne, zapisuje je do bazy oraz umieszcza identyfikator w zmiennej w środowisku.

Plugin algorytmiczny będzie dziedziczył z abstrakcyjnej klasy TreeAlgorithmPlugin, która wykona za dewelopera kwestię komunikacji z sąsiednimi pluginami (decyzja na poziomie implementacji).

4.3.3 Plugin wizualizacyjny, zarządzający

Plugin ten podobnie jak plugin wejściowy pracuje w dwóch trybach. Jako samodzielny plugin służy do wizualizacji istniejących w bazie drzew decyzyjnych oraz umożliwia usuwanie drzew wygenerowanych drzew decyzyjnych.

Jeżeli przed tym pluginem miał miejsce plugin algorytmiczny i odpowiednia zmienna z listą stworzonych identyfikatorów drzew znajduje się w środowisku, wówczas plugin ten wizualizuje każde drzewo na tej liście.

Panel z ustawieniami będzie się składał z trzech elementów. Listy rozwijanej z której będzie można wybrać drzewo do wizualizacji, gdy Plugin będzie działał w trybie niezależnym. Będzie posiadał checkbox do wyboru czy chcemy żeby plugin wizualizował drzewo z listy, czy też ma działać z innymi pluginami i wizualizować drzewo stworzone przez nie. Trzecim elementem panelu z ustawieniami będzie przycisk Usuń drzewo, który będzie pozwalał nam skasować drzewo wybrane na liście. Natomiast panel z wynikiem będzie wyświetlał nam albo komunikat o tym co należy zrobić aby wizualizować drzewo, jeśli zrobimy coś nie tak, albo też będzie nam wyświetlał stworzone przez nas drzewo w postaci JTree.

4.3.4 Plugin wnioskujący

Plugin wnioskujący może również pracować w dwóch trybach. Jednak zalecana jest praca samodzielna. W przypadku gdy w środowisku znajduję się lista identyfikatorów drzew wówczas plugin uruchamia wnioskowanie dla każdego drzewa z tej listy. Danymi wejściowymi są tabele, z których drzewa powstały pomijając wiersze, z których przeprowadzane było budowanie drzewa. Plugin w oknie rezultatów przedstawia wnioski z testów dla każdego drzewa w postaci ogólnych informacji o drzewie (jego nazwa, zbiór danych, z którego powstał), informacja na jakich wierszach było przeprowadzone testowanie oraz wynik testów:

- Procent zgodności - stosunek liczby wierszy z poprawnym wynikiem testu do ogólnej liczby testowanych wierszy
- Lista wierszy (przedziałów) dla których wynik testu był nieprawidłowy

W przypadku pracy samodzielnej (tzn. w sytuacji kiedy plugin nie wykryje obecności listy drzew w środowisku) użytkownik wybiera drzewo, które chce testować, oraz określa zbiór danych do testowania. Zbiór danych musi należeć do tej samej tabeli z której drzewo powstało, ale musi zawierać inne wiersze. Wynik wnioskowania jest identyczny jak w przypadku pracy niesamodzielnej.

5 Model danych

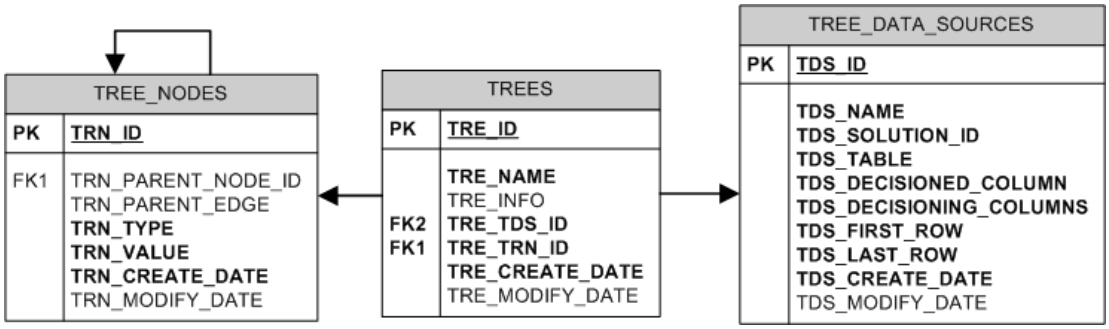
Moduł drzew decyzyjnych, zgodnie z logiką rozwiązań w Salomonie, korzysta z dwóch baz danych. Jedna z nich – *baza definicyjna* przechowuje informację o budowie już istniejących drzew oraz wszystkie niezbędne informacje potrzebne do zbudowania drzewa. Druga zewnętrzna – *baza wejściowa*, jest skojarzona z aktualnym rozwiązaniem i przechowuje dane wejściowe drzew decyzyjnych.

5.1 Baza wejściowa

Baza wejściowa musi zawierać tabelki lub widoki zawierające dane. W celu określenia danych wejściowych użytkownik musi wybrać tabelę a następnie kolumnę reprezentującą wynik decyzji, nazwijmy ją kolumną decyzyjną oraz listę kolumn reprezentujące przesłanki decyzji, nazwijmy je kolumnami decydującymi. Następnie użytkownik definiuję podzbiór tabeli lub widoku, na których będzie budowane drzewo. Innymi słowy format danych wejściowych jest zwykłą tabelką. Wybrana tabela i kolumny wraz z definicją bazy danych zostaje zapisana w bazie definicyjnej do późniejszego użycia przez pluginy algorytmiczne.

5.2 Baza definicyjna

Baza definicyjna ma za zadanie przechować strukturę drzewa oraz parametry z nim związane. Do opisu drzew zdefiniowane są 3 tabele przedstawione na poniższym diagramie ERD:



Rysunek 3: Diagram ERD

Diagram nie przedstawia powiązań między powyższymi tabelami a pozostałymi tabelami definicyjnymi Salomona.

Dane przechowywane w poszczególnych tabelach są następujące:

- TREE_NODES - tabela przechowuje węzły drzew. Każdy węzeł zawiera informację o poprzedzającym go węźle (null jeśli takowego nie posiada, czyli jest

korzeniem drzewa), o poprzedzającej krawędzi oraz o wartości zapisanej w węźle. Wartość w węźle może być zarówno nazwą kolumny lub wartością (zbiorem wartości) kolumny decyzyjnej (wówczas jest to liść drzewa)

- TREE_DATA_SOURCES - tabela reprezentujące wybrane przez użytkownika dane wejściowe potrzebne do zbudowania drzewa. Zawiera m.in.: wskaźnik do rozwiązania (solution) oraz nazwy wybranej wraz z wybranymi kolumnami: kolumnę decyzyjnej i kolumny decydujące, numer pierwszego i ostatniego wiersza. Informacja o rozwiązaniu potrzebna jest w celu zidentyfikowania parametrów bazy wejściowej oraz w celu umożliwienia korzystania z danych wejściowych przez wszystkie projekty w rozwiązaniu.

- TREES - tabela opisująca zbiorczo informację o drzewie. Zawiera m.in.:nazwę drzewa, wskaźnik do taska pluginu algorytmicznego, wskaźnik do węzła drzewa reprezentujący korzeń oraz wskaźnik do tabeli opisującej dane wejściowe.

Plugin wejściowy między innymi ma za zadanie wypełnić tabelę TREE_DATA_SOURCES. Natomiast plugin algorytmiczny pozostałe tabele.