Salomon

System przetwarzania wiedzy

Moduł:

Drzewa decyzyjne

Software Architect Document, ver 1.0

Historia wersji

Data	Wersja	Opis	Autorzy
18-04-2005	1.0	Pierwsza wersja dokumentu	Mateusz Nowakowski

1 Wprowadzenie (Introduction)

Celem dokumentu jest określenie architektury modułu Salomona obsługujących drzewa decyzyjne, założeń implementacyjnych jakie musi spełniać.

2 Architektura (Architectural Representation)

Architektura systemu została stworzona zgodnie z technologią RUP. Zawiera ona:

- model przypadków użycia
- model logiczny
- model danych

3 Założenia i cele architekturalne

3.1 Założenia , ograniczenia

W związku z tym, że moduł drzew decyzyjnych jest częścią Salomona zostały na niego nałożone ograniczenia m.in:

- Moduł musi korzystać z modelu danych Salomona. Konfiguracja, model drzew decyzyjnych muszą się znajdować w tej samej bazie, dostęp do jakichkolwiek źródeł danych musi zapewniać Salomon.
- Salomon narzuca architekturę pluginową. Core obsługi drzew decyzyjnych musi być doimplementowany do Salomona, natomiast cała obsługa drzew decyzyjnych musi znajdować się w pluginach.
- Układ pakietów.Interfejsy muszą znajdować się w salomon.platform.data.tree natomiast ich implementacja w salomon.engine.data.tree
- Model obsługi danych: data + manager data
- Interfejsy pluginów

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

- Pluginy odpalane są z poziomu rozwiązania. Uzyskując dostęp do dwóch baz danych:
  - Bazy definicyjnej - bazy przechowującej konfigurację Salomona. Wszelkie informacje dotyczące zbudowanych drzew decyzyjnych będą sie 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

3.2 Cele architekturalne

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

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

4 Przypadki użycia (Use-Case View)

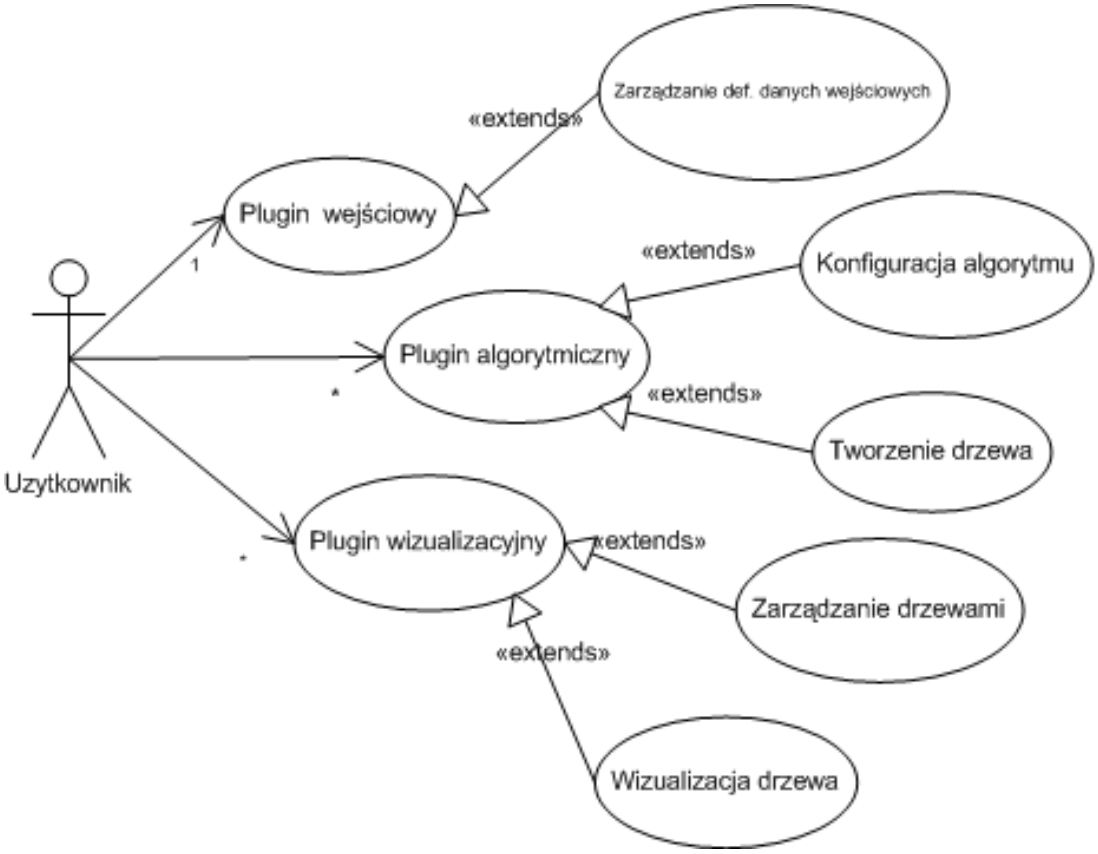
Poniższy opis zakłada że użytkownik uruchomił platformę Salomon oraz poprawnie zdefiniował solution oraz pragnie wykonać operacje na drzewach decyzyjnych.

Poniższy diagram przedstawia punkt widzenia użytkownika:

Użytkownik wybiera najpierw tzw. *plugin wejściowy*, określa jego konfigurację. 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, nazwijmy ją *kolumną decyzyjną* oraz listę kolumn reprezentujące przesłanki decyzji, nazwijmy je *kolumnami decydującymi*. 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 jest zależna od nich samych.

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



Rysunek 1: Use-Case View

4.1 Developer Use-Case View

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

5 Model logiczny (Logical View)

5.1 Ogólne założenia

Moduł drzew decyzyjnych składa się z dwóch części:

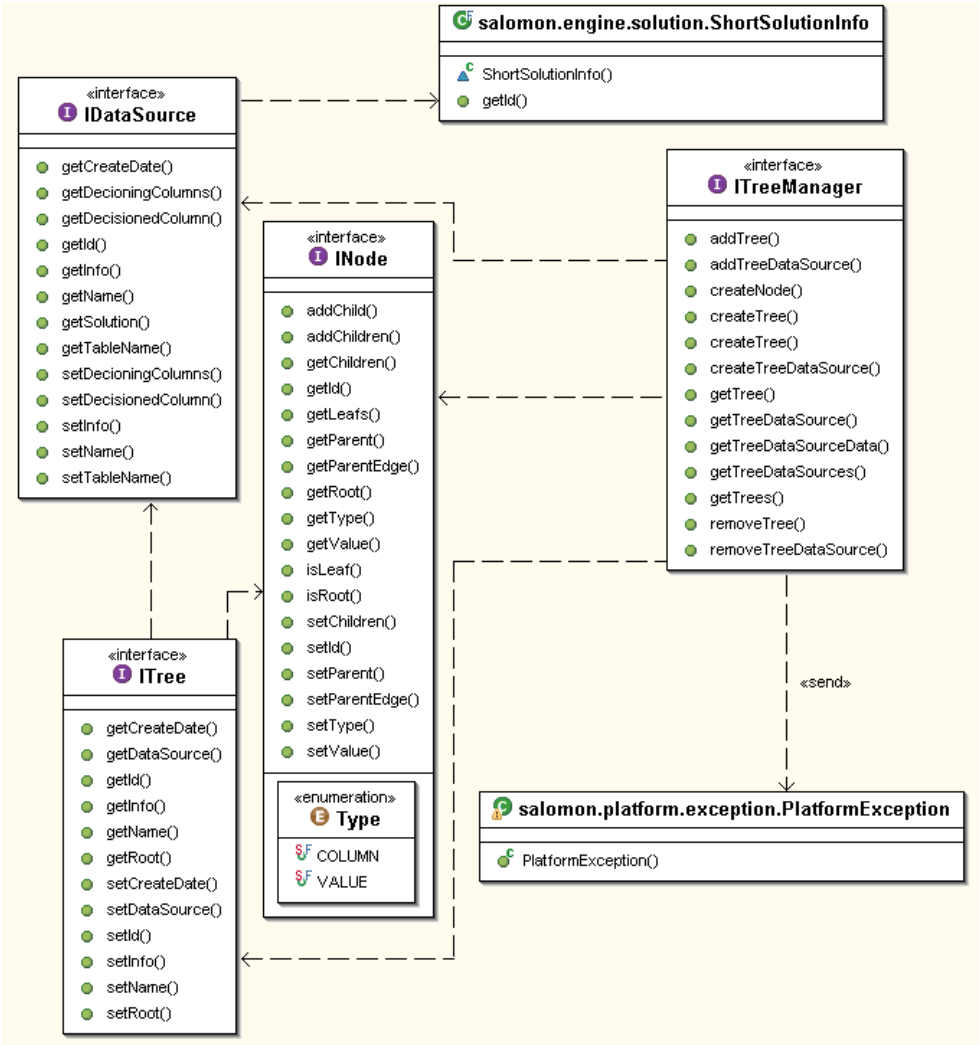
- Rozszerzenia jądra platformy Salomon o podstawową obsługę drzew decyzyjnych

- Zestawu 3 typów pluginów: pluginu definiującego dane wejściowe, pluginu algorytmicznego, pluginu wizualizującego, zarządzającego drzewami

5.2 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. Reszte implementują pluginy.

Poniższy diagram przedstawia układ interfejsów do zaimplementowania:



Rysunek 2: Model rozszerzeń

5.3 Architektura pluginów

Jak wcześniej wspomniano będą 3 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. Pluginy są zgodne a architekturą Salomona i sposób ich budowania jest przez nią określony.

5.3.1 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.

5.3.2 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 zawartośc 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 TreeAlgoritmPlugin, która sprowadzi wykona za dewolopera kwestię komunikacji z sąsiednimi pluginami. (decyzja na poziomie implementacji).

5.3.3 Plugin wizualizacyjny, zarządzający

Plugin ten podobnie jak plugin wejściowy pracuje w dwóch trybach. Jako samodzielny plugin służy do wizualizacji istniejących w bazie drzew decyzyjnych wraz z możliwością usunięcia. Jeżeli przed tym pluginem mial miejsce plugin algorytmiczny i

odpowiednia zmienna z listą stworzonych identyfikatorów drzew znajduje się w środowisku, wówczas plugin ten wizualizuje każde drzewo na tej liście.

6 Model danych (Data View)

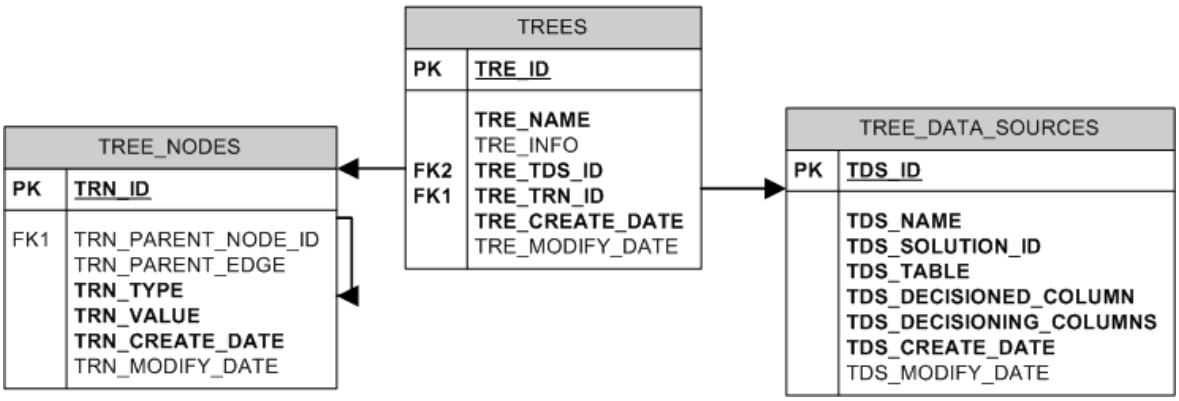
Moduł drzew decyzyjnych, zgodnie z logiką rozwiązań (sollution) w Salomonie, korzysta z dwóch baz danych. Jedna z nich, nazwijmy ją *bazą definicyjną*, przechowuje informację o budowie już istniejących drzew oraz wszystkie niezbędne informacje potrzebne do zbudowania drzewa. Druga, nazwijmy ją *bazą wejściową*, jest skojarzona z aktualnym rozwiązaniem i przechowuje dane wejściowe drzew decyzyjnych.

6.1 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. 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.

6.2 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. Informacja o logowaniu 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.

7 Założenia jakościowe (Quality)

- dopisać moduł jak najbardziej zgodny z salomonem (architektura pluginowa)
- korzystać z mechanizmow salomona jeśli to możliwe. jesli jakis mechanizm działa nie wystarczająco (np. zapis do bazy) to albo go samemu ulepszyć albo zlecić wykonanie drugiej grupie.
- zdefiniować API by można było jak najłatwiej tworzyć pluginy algorytmiczne