# CaliberRM

- **Starbase Corp.** `http://www.starbase.com/`
- **RM == ,,Requirements Management”**
- Wchodzi w skład oferowanego ,,**RDDP Solution**” (Requirements Driven Development Process)
  - CaliberRM
  - StarTeam:
    * version control system
    * Change Logs
    * Defect Tracking System
    * discussion forum
    * ...
  - TestDirector (Mercury Interactive)
  - TogetherSoft: CASE
- **CaliberRM Professional** - system wspomagania decyzji
- **CaliberRM SDK** - API do CaliberRM:
  - OLE
  - Java

# CaliberRM

- **3-tier application**
- **Umożliwia wspólną edycję wymagań oraz wzajemną komunikację między użytkownikami:**
  - kierownicy
  - programiści (developers)
  - analitycy
  - użytkownicy
  - ...
  - framework administrator
- **Przygotowanie projektu:**
  - Utworzenie projektu: nazwa, opis.
  - Powiązanie użytkowników i grup z projektem.
  - Ustalenie praw dostępu.
  - Dołączenie słowników (glossaries) - ujednolicenie słownictwa.
  - Zdefiniowanie typów wymagań poprzez grupowanie atrybutów.

# CaliberRM - opis programu

- **Okno programu:**
  - drzewo typów wymagań i wymagań po lewej stronie
  - panele po prawej stronie:
    * opis i właściwości wymagania
    * dyskusje
- **Typy wymagań** stanowią węzły głównego poziomu. Przykłady:
  - wymagania funkcjonalne
  - wymagania marketingowe
  - wymagania biznesowe
  - ...
- **Atrybuty:**
  - systemowe
    * priorytet
    * status
  - zdefiniowane przez użytkownika (custom)
    Definiowanie atrybutu:
    * nazwa (,,poziom ryzyka'', ,,fundusze zatwierdzone'')
    * typ (13 możliwych)
    * opcje (zwiększ wersję przy modyfikacji, wartość początkowa,...)

# CaliberRM

- **Korzystanie z programu:**
  - edycja wymagań: atrybuty, wersja, właściciel, zewnętrzne źródła danych
  - edycja powiązań między wymaganiami (macierz, diagram)
  - prowadzenie dyskusji
  - ,,baselines'' : tworzenie i porównywanie
- **Zalety RDDP z wykorzystaniem CaliberRM:**
  - ograniczenie ilości błędów i poprawę jakości produktów
  - polepszenie komunikacji między osobami związanymi z projektem
  - ograniczenie konieczności powtórnego tworzenia części wymagań
  - zwiększenie kontroli i zrozumienia projektu

# Wymagania - gromadzenie informacji

- **Zasada pięciu pytań:**
  - Co?
  - Kto?
  - Gdzie?
  - Kiedy?
  - Jak?
- **Analitycy nadmiernie skupiają się na pytaniu Co klient chce robić przy pomocy aplikacji.**
- **Podział wymagań:**

  **Co?** - Wymagania funkcjonalne (funkcje).

  **Kto? Gdzie? Kiedy? Jak?** - Wymagania niefunkcjonalne (ograniczenia).

  **Dlaczego?** - *Rationale* (uzasadnienia).
- **Odpowiedzialność za wymagania:**

  **Klient** - wymagania funkcjonalne

  **Analityk** - wymagania niefunkcjonalne

# Wymagania niefunkcjonalne

- **Kategorie wymagań niefunkcjonalnych:**

| *Pytanie* | *Rodzaj wymagań* |
|---|---|
| Kto? | bezpieczeństwa |
| Gdzie? | topograficzne |
| Kiedy? | czasowe |
| Jak wiele? | skalowalności |
| Jak często? | dotyczące częstości |
| Jak szybko? | wydajnościowe |
| Jak łatwo? | ,,używalności'' (usability) |

- **Identyfikacja wymagań niefunkcjonalnych:**
  - dla każdego wymagania funkcjonalnego zadać z każdej z powyższych kategorii pytania odpowiednim decydentom.
  - po uzyskaniu odpowiedzi zgrupować je w ,,zasady''
- **Każda zasad uzyskanych w ten sposób jest instancją wymagania niefunkcjonalnego. Należy je powiązać z wymaganiami funkcjonalnymi, do których się odnosi.**
- **Powiązania:**

  **zależności (dependancies)** - między wymaganiami niefunkcjonalnymi

  **ograniczenia (constraints)** - między wym. funkcjonalnym, a niefunkcjonalnym