STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Łukasz WYCIŚLIK, Łukasz WARCHAŁ
Politechnika Śląska, Instytut Informatyki

# ORACLE ENTERPRISE MANAGER GRID CONTROL JAKO ROZSZERZALNE ŚRODOWISKO DO ZARZĄDZANIA ROZPROSZONĄ INFRASTRUKTURĄ PROGRAMOWO--SPRZĘTOWĄ, NA PRZYKŁADZIE IMPLEMENTACJI MODUŁU DO MONITOROWANIA WYNIKÓW AUDYTU BEZPIECZEŃSTWA

**Streszczenie**. Artykuł przedstawia możliwości rozwiązania Oracle Grid Control oraz jego zastosowania w obszarze zarządzania i administrowania podsystemami składowymi w złożonych, korporacyjnych systemach informatycznych. Tytułem wstępu autorzy kreślą krótki rys historyczny przedstawiający ewolucyjną drogę rozwoju systemów firmy Oracle do zarządzania infrastrukturą informatyczną. W kolejnym rozdziale przedstawiona zostaje obecna koncepcja firmy Oracle, w oparciu o którą zbudowane są wspomniane systemy oraz możliwości aplikacyjne rozwiązania Grid Control. W dalszej części artykułu autorzy przedstawiają możliwości rozszerzenia funkcjonalności rozwiązania Grid Control na przykładzie stworzonego modułu do monitorowania wyników audytu bezpieczeństwa. Artykuł kończą rozważania co do dalszych możliwych zastosowań rozszerzania funkcjonalności rozwiązania Grid Control.

**Słowa kluczowe**: Oracle, Grid Control, SZBD

# ORACLE ENTERPRISE MANAGER GRID CONTROL – AN EXTENSIBLE ENVIRONMENT FOR SOFTWARE-HARDWARE GRID INFRASTRUCTURE MANAGEMENT – ON BASIS OF A SECURITY AUDIT MONITORING PLUGIN

**Summary**. The article presents functionality and areas of application in complex, corporate computing systems management of Oracle Grid Control. As preliminaries, the authors describe an evolutionary way the Oracle management systems were developed. Next, on basis of Grid Control application, the preset approach for building management systems is presented. Finally the authors show the plugin mechanism that allows extending Grid Control functionality to administrating of custom systems.

On basis of a security audit monitoring a custom plugin development process was shown.

**Keywords**: Oracle, Grid Control, DBMS

## 1. Wstęp

Oracle Corporation to firma mająca od dawna ugruntowaną pozycję na rynku producentów oprogramowania. Choć obecnie jest trzecią firmą na świecie (po IBM i Microsoft) pod względem wielkości przychodów i posiada bardzo szeroką ofertę produktów, to początkowo była przedsiębiorstwem kojarzonym głównie z serwerem relacyjnej bazy danych. Był to i jest produkt przeznaczony do zastosowań profesjonalnych – oferujący dużą elastyczność konfiguracji pozwalającą efektywnie wykorzystywać zasoby pamięciowe i obliczeniowe środowisk, w których funkcjonuje. Duże możliwości konfiguracji, zarówno w trakcie instalacji, jak również w trakcie utrzymywania funkcjonującej już bazy danych, okupione zostały komplikacją czynności administracyjnych. Narzędzia aplikacyjne wspomagające proces administrowania bazą danych Oracle rozwijały się w naturalny sposób w rytm rozwoju możliwości funkcjonalnych samej bazy danych oraz technologii wytwarzania aplikacji, a w szczególności technologii rzutujących na możliwości budowy interfejsu graficznego tych aplikacji. Początkowo były to aplikacje konsolowe, dedykowane do poszczególnych obszarów administracyjnych (np. importy/eksporty danych, strojenie wydajności itp.). Ważnym kryterium technologicznym tworzenia tych aplikacji była ich dostępność dla różnych platform systemowych (Windows, Linux, Unix) stąd rozwój możliwości ich warstwy prezentacji możliwy był dopiero po okrzepnięciu technologii Java. Na początku technologia Java wykorzystywana była wyłącznie do implementacji warstwy prezentacji w technologii grubego klienta (ang. fat client), zaś funkcjonalność merytoryczna wywoływana była przez nie z aplikacji konsolowych. Z czasem narzędzia te (np. DBA Studio) ugruntowały swoją pozycję, ustanawiając nowe standardy i pozyskując wielu zwolenników wśród administratorów systemów baz danych.

Ciągły wzrost popytu na tanią moc obliczeniową i niezawodność systemów komputerowych spowodował rozwinięcie koncepcji architektur gridowych (ang. grid computing), gdzie obliczenia i przetwarzanie danych mogą być rozpraszane na węzłach o różnych architekturach będących niejednokrotnie pod kontrolą heterogenicznych systemów operacyjnych. Taka architektura niesie ze sobą szczególne wyzwania jeśli chodzi o administrację i zarządzanie. Idealne narzędzie administratora powinno pozwalać zarządzać każdym rodzajem aplikacji i elementem infrastruktury z jednego miejsca w spójny sposób – oznacza to, że podobne operacje dotyczące różnych aplikacji czy systemów powinny być realizowane w podobny sposób

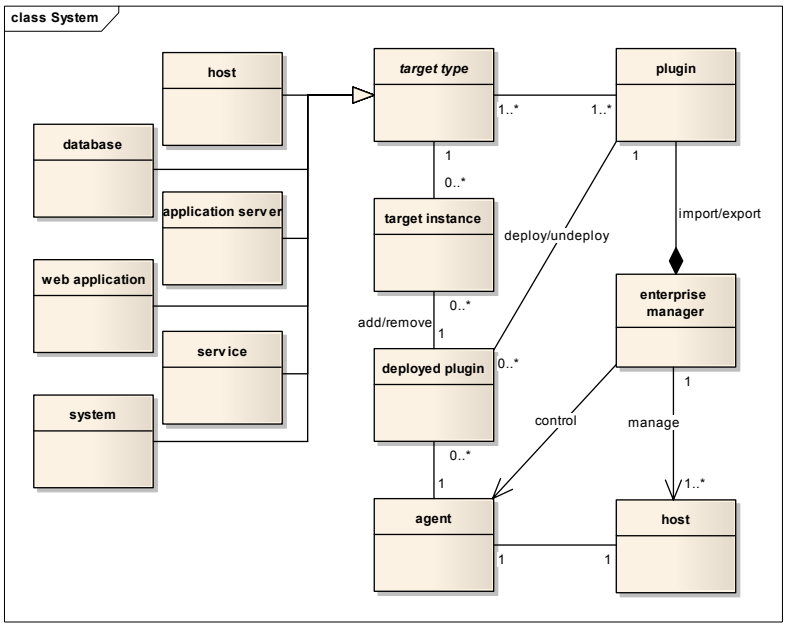
z wykorzystaniem jednolitego środowiska komunikacji z użytkownikiem. Odpowiedzią firmy Oracle na takie zapotrzebowanie jest system Enterprise Manager Grid Control, gdzie dzięki zastosowaniu trójwarstwowej architektury, aplikacja może być uruchamiana w dowolnej przeglądarce, zaś mechanizm modułów umożliwia włączanie nowych funkcjonalności w ramach rozbudowywanej infrastruktury informatycznej. Opis koncepcji i architektury tego rozwiązania zamieszczono w kolejnym rozdziale.

## 2. Koncepcja i architektura

W niniejszym rozdziale przedstawiona zostanie koncepcja (model pojęciowy) dziedziny przedmiotowej zobrazowana diagramem klas oraz propozycja rozwiązania postawionego problemu poprzez zaprezentowanie kluczowych elementów implementacji. Uzyskane efekty zobrazowane zostaną zrzutem ekranu pracującego systemu.

### 2.1. Model pojęciowy i koncepcja rozwiązania

Koncepcja Enterprise Manager Grid Control bazuje na założeniu scentralizowania mechanizmów zarządzania i administrowania całością infrastruktury programowo-sprzętowej. Dzięki zastosowaniu architektury trójwarstwowej możliwe jest uruchomienie aplikacji na każdym, uprawnionym do tego, komputerze wyposażonym w przeglądarkę internetową. Model pojęć dotyczących omawianej tematyki został przedstawiony na diagramie klas (rys. 1).




Rys. 1. Model dziedziny pojęciowej
Fig. 1. Domain model

Aplikacja Enterprise Manager Grid Control zarządza (ang. manage) składowymi elementami struktury grid (ang. host). Jest to możliwe dzięki istnieniu oprogramowania agenta na każdym elemencie składowym, za pomocą którego realizowana jest kontrola (ang. control) przez Enterprise Managera [1]. Diagram ten, aby zwiększyć czytelność, przedstawia najczęstszy przypadek, wyłączając sytuację, kiedy na jednej maszynie funkcjonuje wiele partycji logicznych bądź wirtualnych maszyn – wtedy oprogramowanie agenta powinno zostać zainstalowane odpowiednio na każdej partycji bądź wirtualnej maszynie. Administrowaniu i zarządzaniu podlegać mogą różne typy zasobów: serwery aplikacyjne, instancje bazy danych, systemy operacyjne, usługi itp. W celu rozszerzenia funkcjonalności systemu zarządczego należy zaimportować (ang. import) odpowiedni moduł (ang. plugin) – dostarczony przez zewnętrznego dostawcę jako gotowy produkt lub też stworzony na własne potrzeby. Jeden moduł może implementować funkcjonalność zarządzania jednym bądź kilkoma typami zasobów. Zaimportowane moduły należy zainstalować (ang. deploy) na agentach maszyn, na których istnieje potrzeba zarządzania zasobami danego typu. Na każdej maszynie należy następnie dodać instancję zarządzanego typu (ang. target instance). Nic nie stoi na przeszkodzie, aby na jednej maszynie powołać kilka instancji zarządzanego typu (z różnymi ustawieniami) – np. w przypadku potrzeby zarządzania wieloma instancjami bazy danych.

Oprócz funkcjonalności wbudowanych w aplikację Enterprise Manager istnieje możliwość jej rozszerzania o szereg dostępnych typów zasobów poprzez instalację modułów. Moduły te dostarcza zarówno sam Oracle (np. wsparcie dla JBOSS, DB2, Tomcat itd.), jak również dostawcy niezależni (np. wparcie dla MySQL, Citrix Presentation Server itd.).

Oracle dostarcza również narzędzia do tworzenia modułów obsługujących własne typy zasobów. W kolejnym rozdziale przedstawiono sposób tworzenia i wdrożenia modułu na przykładzie typu zasobu, jakim jest dziedzinowy system autentykacji.

### 2.2. Implementacja przykładowego modułu

Proces tworzenia nowego modułu, pozwalającego monitorować w aplikacji Enterprise Manager nowy typ zasobów, składa się z kilku etapów. W pierwszej kolejności wykonywane są prace koncepcyjne. Określa się zakres zbieranych danych, na podstawie których tworzone są obserwowane wskaźniki, ustala się częstotliwość ich zbierania oraz wartości graniczne, decydujące o tym, w której grupie powiadomień znajdzie się informacja o wartości danego wskaźnika (np. *Alert*, *Warning* etc.).

W omawianym rozwiązaniu projektowany moduł powinien pobierać (co 10 minut) z dziedzinowego systemu bezpieczeństwa (ACS – *Access Control System*) informacje o nieudanych próbach autentykacji. Jeśli ich liczba przekracza 5, zgłaszane jest ostrzeżenie (ang. *Warning*), gdy przekracza 10 – alarm (ang. *Alert*).

Kolejny etap tworzenia własnego modułu polega na opisaniu definicji nowego typu zasobu, przy użyciu języka XML. Powstały plik (*Target Type Metadata File*) specyfikuje zbiory danych (ang. *Metrics*), na podstawie których budowane są wskaźniki, oraz wskazuje parametry (wartości zmienne), które mają zostać określone w momencie tworzenia instancji ww. typu (np. parametry połączenia z dziedzinowym systemem). Niezbędnym elementem każdej wtyczki jest także plik XML (*Target Type Default Collections File*) określający, jak często gromadzić ww. zbiory danych.

W omawianym module plik *Target Type Metadata File* ma postać[1]:

```
<TargetMetadata META_VER="1.0" TYPE="user_audit_2" TARGET_VERSION="1.2">
  <Display>
    <Label NLSID="user_audit_label">Audyt nieudanych prob logowania systemu
ACS</Label>
  </Display>
  <InstanceProperties>
    <!-- wartości zmienne -->
  </InstanceProperties>

  <Metric NAME="Response" TYPE="TABLE">
    <!-- metryka określająca dostępność systemu -->
  </Metric>

  <Metric NAME="Users" TYPE="TABLE">
    <Display>
      <Label NLSID="user_audit_users">Uzytkownicy</Label>
    </Display>
    <TableDescriptor>
     <ColumnDescriptor NAME="UName" TYPE="STRING" IS_KEY="TRUE">
          <Display>
            <Label NLSID="metric_users_username">Uzytkownik</Label>
          </Display>
     </ColumnDescriptor>
     <ColumnDescriptor NAME="LOGINFAILED" TYPE="NUMBER" IS_KEY="FALSE">
          <Display>
            <Label NLSID="metric_users_logfailed">Nieudane logowania</Label>
          </Display>
     </ColumnDescriptor>
    </TableDescriptor>
    <QueryDescriptor FETCHLET_ID="SQL">
       <Property NAME="MachineName" SCOPE="INSTANCE">host</Property>
       <Property NAME="Port" SCOPE="INSTANCE">port</Property>
       <Property NAME="ServiceName" SCOPE="INSTANCE">instance</Property>
       <Property NAME="UserName" SCOPE="INSTANCE">username</Property>
       <Property NAME="password" SCOPE="INSTANCE">password</Property>
     <Property NAME="STATEMENT" SCOPE="GLOBAL">select UNAME, LOGINFAILED from
acsadm.useraudit</Property>
     <Property NAME="NUMROWS" SCOPE="GLOBAL">30</Property>
    </QueryDescriptor>
  </Metric>
</TargetMetadata>
```

Najważniejszym fragmentem ww. pliku jest element <*Metric*>. Opisuje on kolekcję danych (*ang. Metric*) – w postaci tabeli z kolumnami określonych typów, reprezentującą monitorowany wskaźnik (liczba nieudanych autentykacji do aplikacji biznesowych). Element <*QueryDescriptor*> specyfikuje sposób, w jaki ww. kolekcja jest budowana – wartość atry-

[1] Mniej istotne elementy zostały pominięte w celu zachowania czytelności.

butu *FETCHLET_ID* ustawiona na *SQL* informuje, że dane uzyskiwane są z zapytania SQL (lista dozwolonych wartości ww. atrybutu dostępna jest w [1]).

Plik Target Type Default Collections File ma następującą postać:

```
<TargetCollection TYPE="user_audit_2">
 <CollectionItem NAME="Response">
   <Schedule>
     <IntervalSchedule INTERVAL="1" TIME_UNIT="Min" />
   </Schedule>
   <Condition COLUMN_NAME="Status" CRITICAL="0" OPERATOR="EQ" />
</CollectionItem>

 <CollectionItem NAME="Users">
   <Schedule>
     <IntervalSchedule INTERVAL="10" TIME_UNIT="Min" />
   </Schedule>
<Condition COLUMN_NAME="LOGINFAILED" WARNING="5" CRITICAL="10" OPERATOR="GE"
/>
 </CollectionItem>
</TargetCollection>
```

W elemencie <*Schedule*> specyfikuje się odstępy między kolejnymi pobraniami danych. Element <*Condition*> definiuje wartości graniczne, na podstawie których następuje przydział do odpowiednich grup powiadomień.

Do zapewnienia poprawności omówionych plików XML, zarówno pod względem syntaktycznym, jak i semantycznym, można wykorzystać konsolowe narzędzie ILINT. Oprócz weryfikacji zgodności plików z definicjami typu dokumentu[2] (ang. DTD), przeprowadza ono ich heurystyczną analizę pod względem poprawności zdefiniowanych kolekcji danych, częstotliwości ich zbierania itp.

Ostatnim etapem tworzenia wtyczki jest umieszczenie jej w przenośnym archiwum (ang. *MPA – Management Plugin Archive*), dzięki czemu łatwo można ją przenieść ze środowiska wytwórczego do środowisk produkcyjnych (nawet w innych instalacjach EM). Archiwum może zawierać więcej niż jeden moduł, co sprawia, że w prosty sposób da się zbudować zestaw narzędzi monitorujących wybrane fragmenty infrastruktury programowo-sprzętowej.

Narzędzie Enterprise Manager Command Line Interface (EM CLI) umożliwia stworzenie archiwum.

Poniżej znajduje się przykład użycia tego narzędzia:

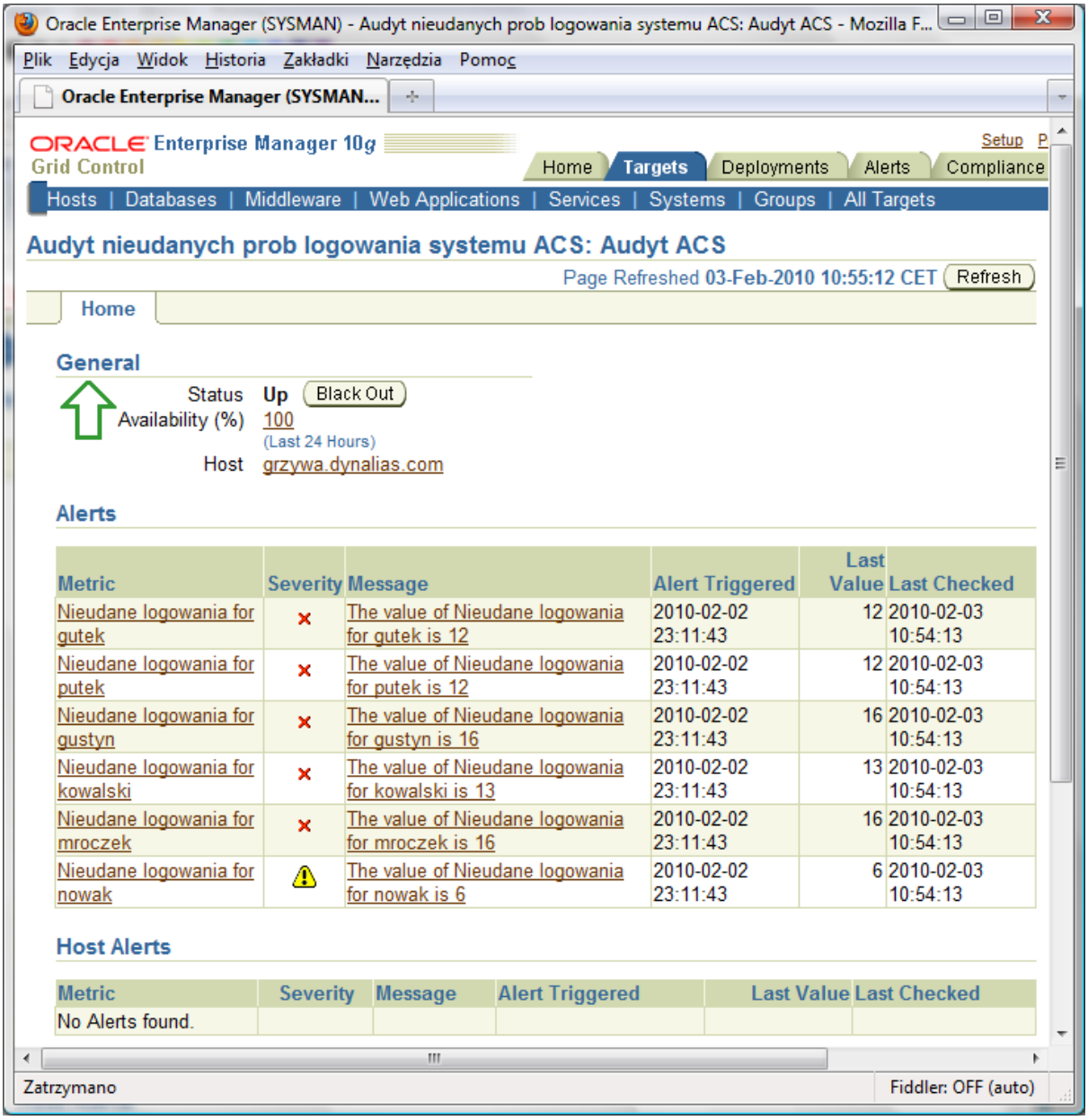
```
C:\user_audit>emcli add_mp_to_mpa -mpa="user_audit_2.jar" -mp_version="1.2"
-ttd="ttd\user_audit_2.xml" -dc="dc\user_audit_2.xml"
Management Plug-in "user_audit_2" version 1.2 requiring minimum OMS version
10.2.0.1 added successfully to MPA "user_audit_2.jar"
```

Archiwum zawierające wytworzone moduły należy następnie zaimportować w aplikacji EM, a potem zainstalować na wybranych agentach maszyn. Po utworzeniu instancji zarzą-

[2] Pliki DTD zlokalizowane są w: $AGENT_HOME/sysman/admin/dtds

dzanych typów (zdefiniowanych we wtyczce) można obserwować interesujące nas wskaźniki.

Na rysunku 2 pokazano ekran aplikacji EM przedstawiający wynik funkcjonowania opisanego wyżej modułu.




Rys. 2. Przykład wytworzonego modułu
Fig. 2. Example of developed plugin

## 3. Podsumowanie

W artykule przedstawiono możliwości rozwiązania Oracle Grid Control oraz jego zastosowania w obszarze zarządzania i administrowania podsystemami składowymi w złożonych,

korporacyjnych systemach informatycznych. Autorzy przedstawili możliwości rozszerzenia funkcjonalności rozwiązania Grid Control na prostym przykładzie stworzonego modułu do monitorowania wyników audytu bezpieczeństwa. Jednak możliwości rozszerzenia tym mechanizmem ograniczone są tylko możliwościami pozyskiwania informacji z systemów zewnętrznych. Najbardziej celowe wydaje się implementowanie modułów do zarządzania podsystemami stanowiącymi warstwy usługowe dla całości rozwiązań korporacyjnych. Są to więc najczęściej serwery bazodanowe oraz serwery aplikacyjne. Jako dalszy kierunek badań autorzy rozważają możliwość implementacji modułu do zarządzania podsystemem monitorowania realizacji przypadków użycia opisanym w [3].

## BIBLIOGRAFIA

1. Oracle Enterprise Manager – Extensibility Guide, March 2009.
2. Oracle Enterprise Manager – Connectors Integration Guide, May 2009.
3. Adaptable Graphical User Interfaces for Player-Based Applications. – Man-Machine Interactions, Springer Berlin/Heidelberg, Volume 59/2009.

Recenzenci: Dr inż. Bożena Małysiak-Mrozek
Dr hab. inż. Mirosław Zaborowski

Wpłynęło do Redakcji 31 stycznia 2010 r.

## Abstract

The article presents functionality and areas of application Oracle Grid Control in complex, corporate computing systems management. As preliminaries, the authors describe an evolutionary way the Oracle management systems were developed. Next, on basis of Grid Control application, the present approach for building management systems is presented. On the class diagram primary definitions related to remote administration in Grid Control were explained. Finally authors show the plugin mechanism that allows extending Grid Control functionality to administrating of custom systems. On basis of a security audit monitoring a custom plugin development process was shown. This process consists of development, validation, importing, deploying and adding to an agent a target instance. At the end authors suggested others possible applications of the custom plugin mechanism.

**Adresy**

Łukasz Wyciślik: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, l.wycislik@polsl.pl .

Łukasz Warchał: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, l.warchal@polsl.pl .