Analysis

for

Ruby ACL

Version 1.2

SirlJan

16.11.2011

Zadání

Navrhněte, implementujte a otestujte knihovnu/modul v programovacím jazyku Ruby realizující správu řízení přístupu pomocí ACL (access control list). Zaměřte se především na specifikaci rozhraní knihovny a na příklady jejího použití. Výsledkem bude nejen samotná realizace knihovny ale i podrobná programátorská dokumentace.

Úvod

Ruby ACL je knihovna spravující přístupová práva pro databáze. Byla vytvořena pro objektovou databázi Cellstore. Hlavním cílem projektu je vytvoření jemně nastavitelných („fine-grained“) řízení přístupu.

Ruby ACL je napsaná v jazyku Ruby. Ruby ACL je vydána ve formě RubyGem.

Specifikace

Níže uvedená tabulka popisuje specifikace projektu, které budou sbírány, analyzovány a specifikovány téměř po celou dobu vývoje aplikace.

SRS id	SRS description	SRS priority
FUNCTIONAL REQUIREMENTS		
0.0	Ruby ACL is library/module for Ruby programming language. Ruby ACL will manage control of access by ACL (Access Control List).	Must have
1.0	Ruby ACL will allow define access permission for objects	Must have
1.1	Ruby ACL will allow row-level and column level access control	???
2.0	Interface of Ruby ACL	Must have
3.0	Lists of accesses is protected from change	Must have
4.0	Ruby ACL will handle identity check	Won’t have
5.1	Ruby ACL will allow to work with ACLs	Must have
5.2	Ruby ACL will allow to create ACLs	Must have
5.3	Ruby ACL will allow to save ACLs	Must have
5.4	Ruby ACL will allow to load ACLs	Must have
6.0	Ruby ACL will stand for Default-Deny policy (At the beginning nobody cannot access to anywhere)	Must have
7.0	Editing and setting new custom privilege	Nice to have
NON-FUNCTIONAL REQUIREMETNS		
1.0	Ruby ACL will be programmed in Ruby	Must have
2.0	Ruby ACL will be library for Ruby	Must have
2.1	Ruby ACL will be released as RubyGem	Must have
	Ruby ACL will have visual representation	Won’t have

Note: SRS priority= {must have, should have, nice to have, Won’t have}

Vstupy a Výstupy

Vstupy budou parametry: 1)Principals - which user(s)/group(s) can
2) Privileges - which operation(s) is about to perform
3) Objects - which data

Výstup: Allow/Deny specified in boolean

Ukazka použití

Příklad použití kontroly přístupu.

```
require 'Ruby-ACL'
require 'dbi'

acl = Racl.new()
acl.load("test1")

username = "pepanovak"
password = "tajneheslo"
desired_operation = "select"
desired_object = "dbi:OCIU:mydb/people"

if(acl.acl_check(username,desired_operation,desired_object)) then
  db = DBI.connect(separate_db(desired_object), username, password)
  query = "select * from people"
  stmt = db.prepare(query)
  stmt.execute
  while row = stmt.fetch do
    puts row.join(",")
  end
  stmt.finish
  db.disconnect
else
  puts "Access denied to #{desired_object}."
End
```

Příklad použití nastavení práva. `init_from_db` načte všechny objekty z databáze. Tj. *Principals* a *ResourceObject.*

```
require 'Ruby-ACL'

username = "pepanovak"
access_type = "deny"
desired_privilege = "create"
desired_object = "dbi:OCIU:mydb"

acl = Racl.new("test2")
acl.init_from_db("dbi:OCIU:mydb",“tabulka_Useru_a_Skupin“)
acl.set_new_ace(username,access_type,desired_privilege,desired_objec
t)
```

Use Case diagram

Znázorňuje roli uživatele vůči knihovně. Ruby ACL definuje 1 aktéra, kterým je uživatel/administrátor ACL.

Scénáře

Ověřování oprávnění k objektu

Uživatel má vytvořenou instanci RubyACL, která obsahuje pravidla.

Hlavní úspěšný scénář:

1) Uživatel zavolá metodu acl_check. Přes tuto metodu se dotází systému, jestli Uživatel/Skupina (ne)mají oprávnění ke zdrojovému objektu.
2a) Systém vrátí true v případě, že uživatel/skupina má specifikované nebo vyšší oprávnění.
2b) Systém vrátí false v případě, že uživatel/skupina nemají specifikované nebo vyšší oprávnění.

Rozšíření:
0a) Pokud neexistují pravidla v instanci, systém vrátí false, protože nenašel, žádné vyhovující pravidlo.

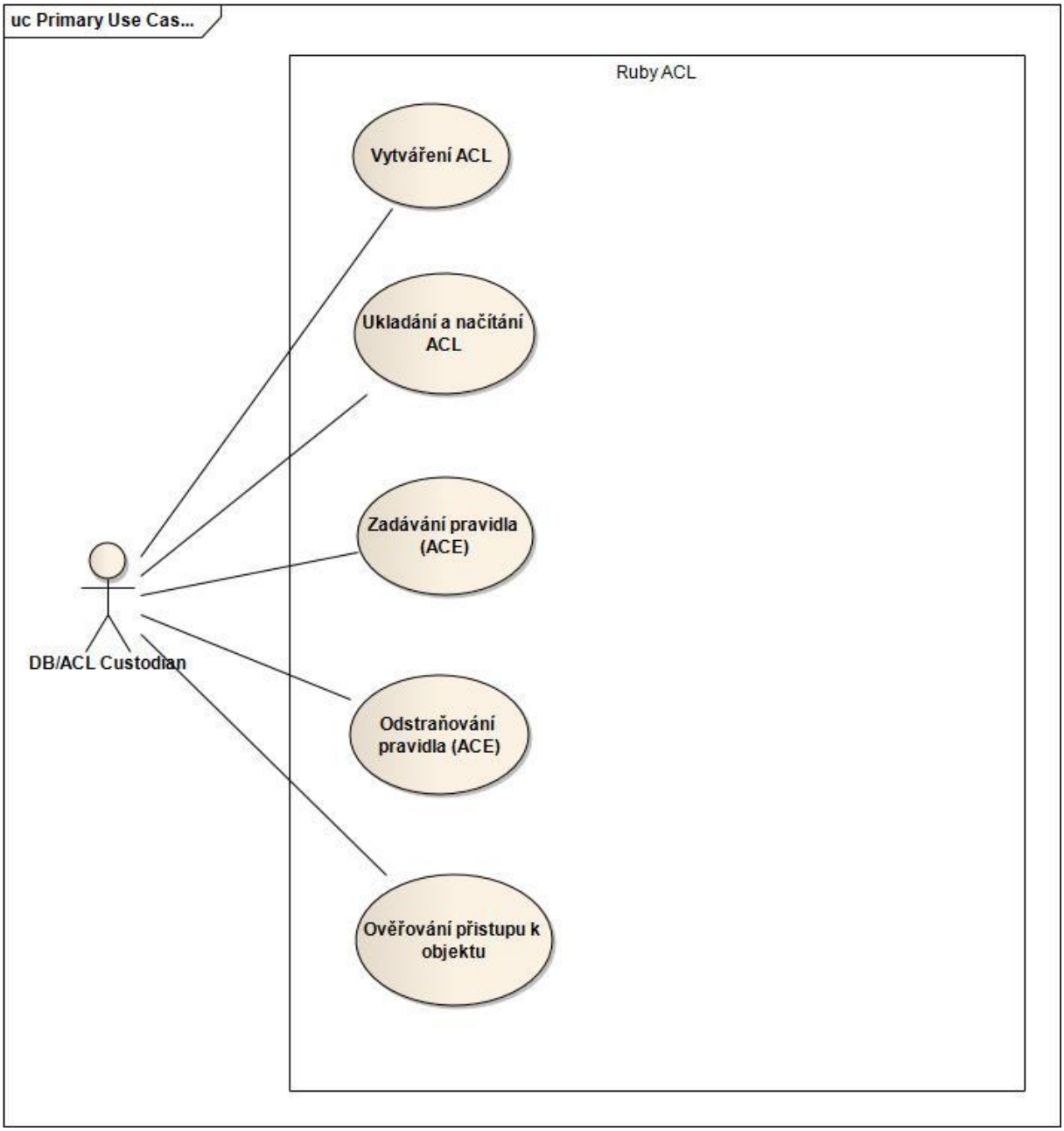
Zadávání pravidla (ACE)

Uživatel má vytvořenou instanci RubyACL, která obsahuje pravidla.

Hlavní úspěšný scénář:

1) Uživatel zavolá metodu set_new_ace a specifikuje údaje (Uživatel/skupina, typ přístupy (allow/deny), oprávnění, zdrojový objekt
2) Systém nastavý pravidlo a vrátí 0, když vše proběhlo v pořádku, a 1 když nastala chyba.

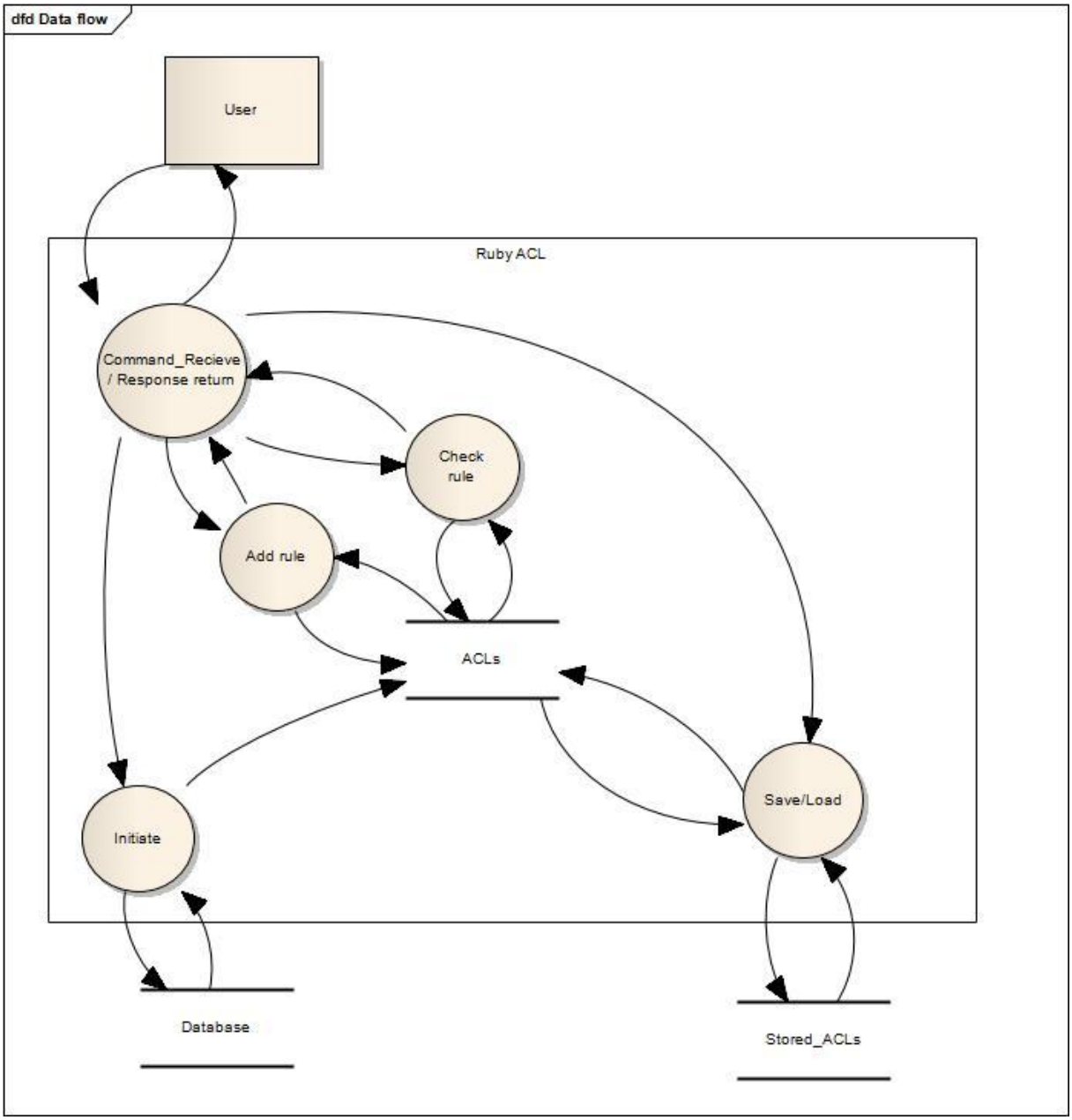
Poznámka: Jde vlastně o přiřazování oprávnění uživatelům na objekt



Všechny případy užití předpokládají vytvořenou instanci RubyACL, která má vytvořená nějaká pravidla.

Data flow diagram

Data flow diagram (diagram datových kroků) znázorňuje tok dat mezi jednotlivými funkcemi aplikace. Popisuje funkce a jejich vazby.



ACLs je instance RubyACL, kterou se aktuálně pracuje. Do ní systém ukládá změny provedené uživatelem.

Stored_ACLs – je uložiště s jedním nebo více již vytvořených Access Control Listů

Class diagram

Class diagram znázorňuje základní stavební prvky aplikace a jejich vazby. ACL (Access Control List) se skladá

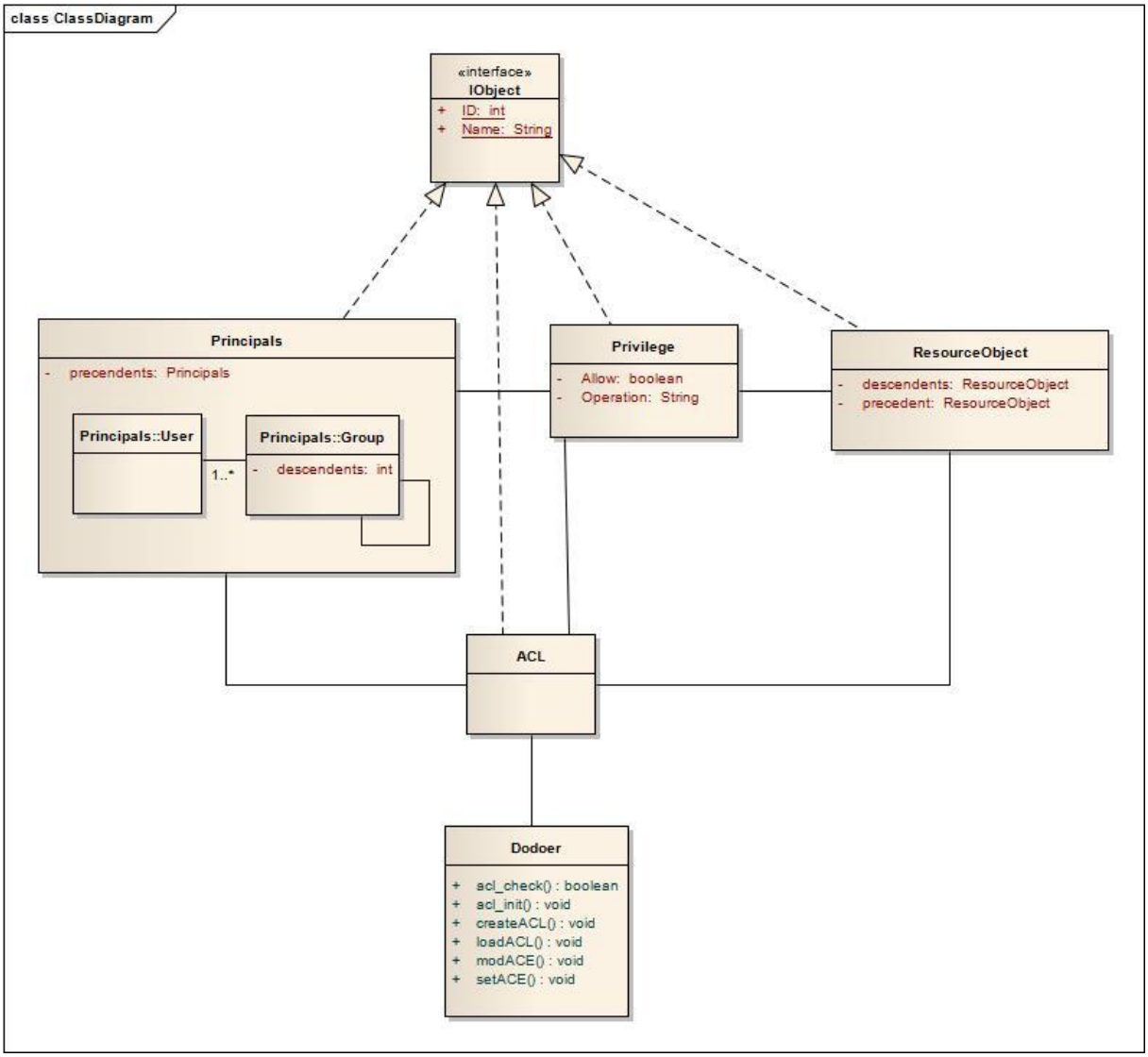
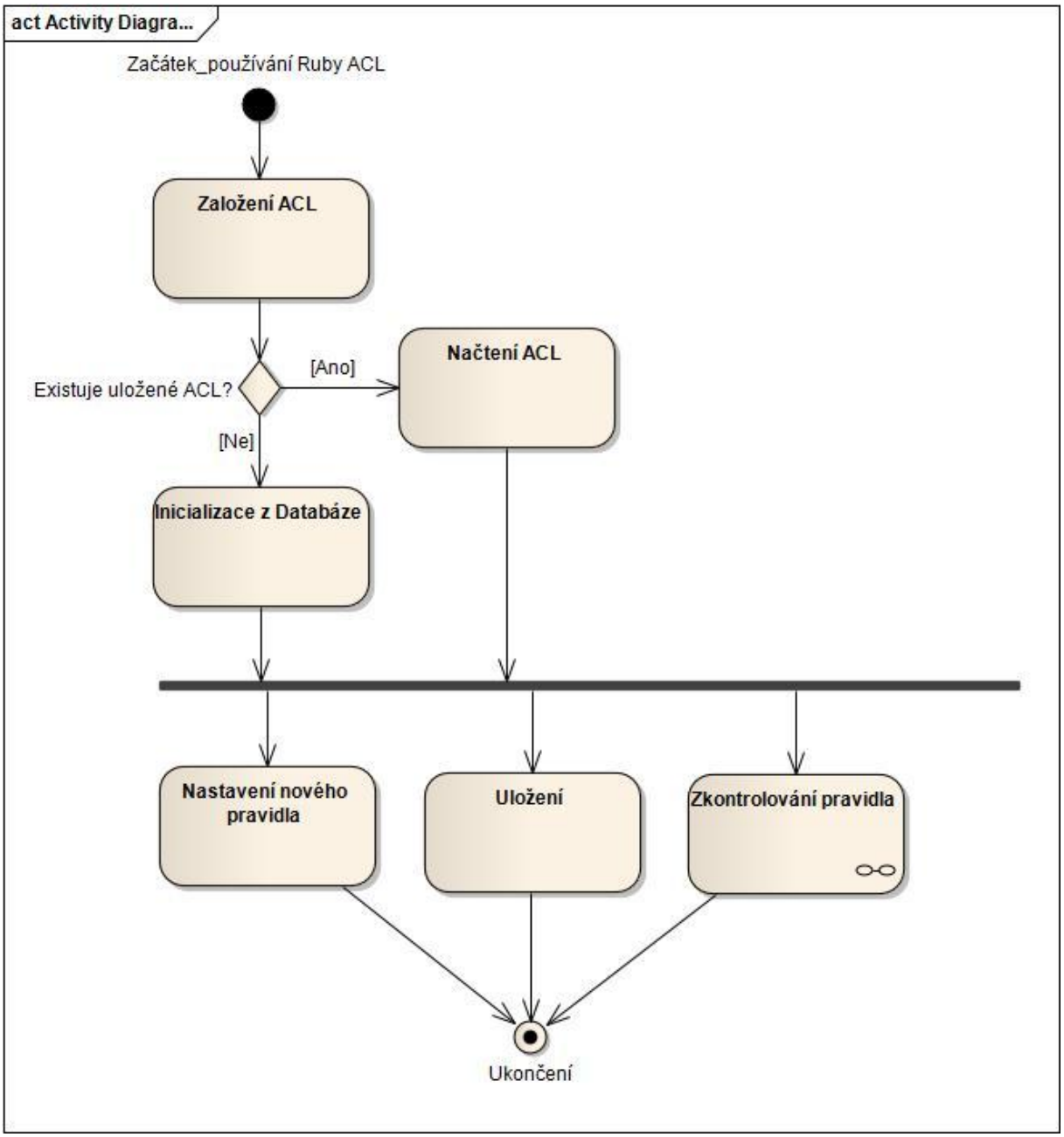
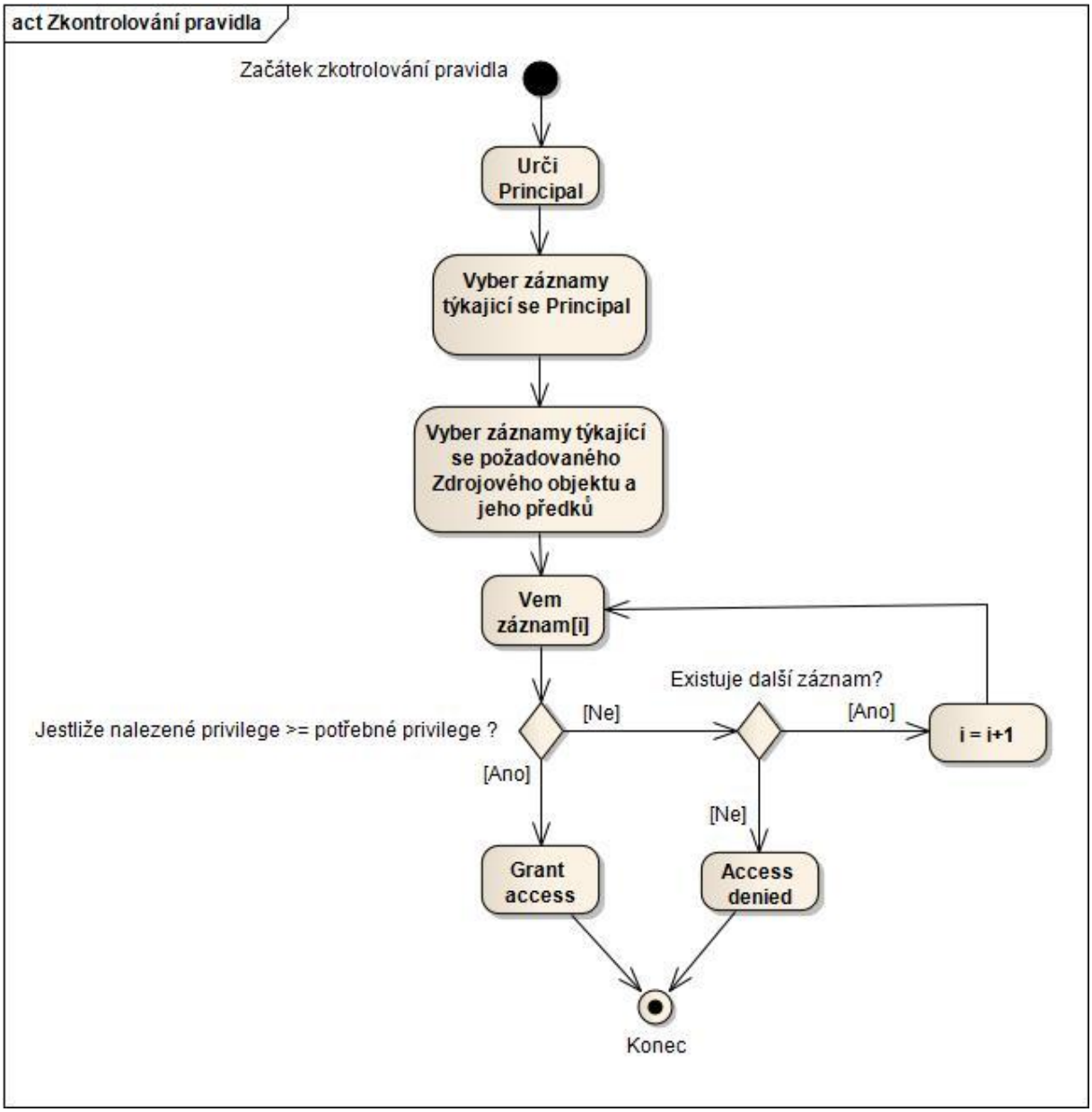


Diagram aktivit





Slovník

Fine-Grained *access control* = jemné řízení přístupu. Komukoliv lze nastavit jakékoliv práva na jakýkoliv objekt

ACL = Zkratka pro Access Control List, neboli seznam řízení přístupu. ACL je složeno z ACE

ACE = Access Control Entry, Záznam řízení přístupu. Př: **Jarda** má *zakázan* zápis do DB/TClanky

Principals = Objekt vyžadující přístup. V příkladu nahoře se jedná o **Jardu**

ResourceObject = Objekt ke kterému Principal žádá přístup. V př. Nahoře se jedná o DB/TClanky

Access_Type = Allow nebo Deny

RubyGem = je balíčkovací systém navrhnutý pro tvorbu, sdílení a instalaci knihoven. RubyGem je podobný *apt-get*, ale zaměřený na Ruby.