Aalto-yliopisto
Perustieteiden korkeakoulu
Informaatioverkostojen koulutusohjelma

Laiska evaluointi funktionaalisessa ohjelmoinnissa

Kandidaatintyö, välipalautus V1

7. helmikuuta 2016

Atte Keinänen

Sisältö

1 Johdanto

Laiska evaluointi on 1970-luvulta juontuva ohjelmoinnin konsepti, jonka ytimessä on turhan ja liian aikaisen laskennan välttäminen ohjelmakoodia suorittaessa. Eli jos esimerkiksi listan alkiota ei koskaan tarvita, niin sen arvoakaan ei ole syytä laskea. Vastaavasti jos arvoa tarvitaankin, sitä ei lasketa listaa muodostettaessa, vaan vasta sitten kuin sitä on tarve käyttää.

Laiska evaluointi on nimityksenä vakiintunut, mutta "laiskuuden" osalta hieman harhaanjohtava. Paremmin konseptia kuvaa sen englanninkielinen synonyymi *call-by-need evaluation*, joka korostaa sitä, että laskentaa tehdään vasta kun tietoa tarvitaan. Toinen tulkinta on, että laskentaa viivytetään niin kauan, kunnes arvo on pakko laskea.

Laiskan evaluoinnin tekee aiheena kiinnostavaksi se, että sitä hyödynnetään kasvavissa määrin niin ohjelmointikielien suunnittelussa kuin yksittäisissä sovelluskirjastoissa. Esimerkiksi uudehkot ohjelmointikielet, kuten Scala ja Clojure, tarjoavat erityisiä syntaktisia rakenteita sitä varten. Toisaalta myös monet suositut apukirjastot, esimerkiksi JavaScript-kielen Immutable.js, Lazy.js ja Bacon.js, hyödyntävät sen periaatteita.

Laiskaa evaluointia ei ole yhtä tapaa tehdä "oikein", vaan käyttökohteiden ja toteutustapojen diversiteetti on suuri. Myös tavoiteltavat hyödyt ovat erilaisia. Verrataan esimerkiksi ohjelmointikielen ja apukirjaston laatijoiden tavoitteita: ohjelmointikielen laatija voi käyttää laiskaa evaluointia kielen ilmaisuvoiman kasvattamiseen, kun taas apukirjaston laatija voi päästä sen avulla kilpailevia kirjastoja parempaan suorituskykyyn.

Ymmärrän tässä työssä laiskan evaluoinnin laajasti, ja pyrin luomaan kokonaiskuvaa konseptin käyttökohteista ja toteutustavoista. Seuraavaksi määrittelen tarkemmin, mitä tarkoitan laiskalla evaluoinnilla, ja mitä muita käsitteitä konseptiin tiiviisti liittyy. Sen jälkeen esittelen tutkimuskysymykset, joihin työni pyrkii vastaamaan.

1.1 Keskeisten käsitteiden määrittely

Lauseke tarkoittaa ohjelmointikielen ilmaisua, jolle voidaan määrittää arvo. Esimerkiksi aritmeettiset operaatiot, muuttujien nimet ja funktiokutsut ovat lausekkeita.

Evaluointi tarkoittaa tarkoittaa lausekkeen arvon laskemista.

Evaluointistrategia tarkoittaa ohjelmointikielen sääntöjä, jotka määrittävät, missä tilanteissa lausekkeita evaluoidaan.

Laiska evaluointi tarkoittaa evaluointistrategiaa, joka viivyttää lausekkeen evaluointia saakka, kunnes sitä tarvitaan (Watt, 2004). Käsitettä voi käyttää kuvaamaan myös yksittäistä lauseketta abstraktimman operaation, kuten esimerkiksi kokoelmien käsittelyn, suorittamisen viivästämistä vastaavaan tapaan.

1.2 Tutkimuskysymykset ja -menetelmät

Haluan saada kartoitettua työlläni laiskan evaluoinnin käyttökelpoisuutta tutkimuksessa ja työelämässä, ja haluan myös tuottaa sellaista tietoa, joka olisi käyttökelpoista laiskan evaluoinnin konseptien opetuksessa. Päädyin näiden tavoitteiden kautta seuraaviin tutkimuskysymyksiin:

1. Mikä on laiskan evaluoinnin merkitys nykypäivänä?
2. Mitkä ovat laiskan evaluoinnin hyödyt ja haitat?

Kysymyksessä 1 laiskan evaluoinnin merkityksen tutkiminen tarkoittaa, että tarkastelen (1) laiskan evaluoinnin historiaa, (2) laiskan evaluoinnin leviämistä moderneihin ohjelmointikieliin sekä kielien apukirjastoihin, (3) laiskan evaluoinnin kautta kehittyneitä uusia ohjelmoinnin konsepteja ja (4) laiskaa evaluointia käyttävien teknologioiden hyödyntämistä tutkimuksessa ja työelämässä. Toteutan kysymyksen 1 tarkastelun kirjallisuuskatsauksen avulla, keskittyen akateemisiin papereihin ja tukien akateemista tutkimusta etenkin viimeisimpien, 2010-luvulla tapahtuneiden kehityskulkujen osalta myös blogitekstien avulla.

Kysymyksessä 2 selvitän sitä, millaisia subjektiivisia mielipiteitä laiskaa evaluointia hyödyntäviä teknologioita käyttävillä ihmisillä on sekä laiskasta evaluoinnista yleisesti, että kysymyksen 1 tarkastelussa esiin tulleista ohjelmointikielistä ja apukirjastoista. Tässä yhdistän sekä kirjallisuuskatsauksen että mielipidetutkimuksen tekniikoita.

Lähteet

David A Watt. *Programming language design concepts*. John Wiley & Sons, 2004.