“Why do we plan before implementing? The big danger in plunging right into a project is the possibility that we will paint ourselves into a corner. If we had a more flexible language, could this worry be lessened?”
- Paul Graham: On Lisp

Testausdokumentti

Työn testauksessa käytin alusta asti LISPistä tuttua testausfilosofiaa. Tulkkikielissä interaktiivinen testaus muuttuu olennaiseksi osaksi testausprosessia, sillä tulkkiohjelmat tarjoavat oivallisen alustan niin yksikkötestausten, kuin integraatiotestaustenkin tekemiseen lennosta samalla kuin itse ohjelman toiminnallisuutta kirjoitetaan; uuden funktion testaus on mahdollista parhaimmillaan jopa samaan aikaan kuin funktiota kirjoitetaan, ja virheiden korjaus voidaan tehdä hetkessä.

Siispä lähtökohtana LISP-tulkin testausta oli käyttää sille luontevaa testaustyökalua - tulkkia itseään. Oli helpompaa kirjoittaa suoraan tulkkiin niin tyypillisiä, kuin epätyypillisiäkin LISP-lauseita, tarkastella tulkin käyttäytymistä niiden suhteen, ja reagoida sitten sopivalla tavalla.

Tietysti suuri ongelma tuli vastaan siinä, että koska tulkkia itseään ei ole kirjoitettu ttulkilla, ei siinä esiintyviä ongelmia pysty samalla tavalla korjaamaan lennosta, vaan ne on erikseen korjattava Java-lähdekoodista. Mutta silti tulkin testaus tehostui huomattavasti.

Toisaalta oli asioita, joita ei voinut testata tällä tavalla tulkin itsensä avulla; parseri oli suunniteltu täysin erilliseksi kokonaisuudekseen, jonka virheitä ei tulkin avulla tehtyjen testien kautta voinut kovin helposti havaita. Tähän ongelmaan sopivat Javan perinteiset yksikkötestit paremmin.

Javan yksikkötestauksessa lähdin aluksi varmistamaan, että tarkoituksenmukainen käyttäytyminen oli taattu toimivan, ja vasta tämän jälkeen aloin testaamaan poikkeustapauksia. Projektin luonteen vuoksi raja-tapauksia ei juuri ole; S-lausejoko on hyväksyttävä Skeleton-lisp -lause, tai sitten se ei ole, ja kielioppi, joka määrää milloin lause on hyväksyttävä, on hyvin tarkkaan määritelty (vaikka aluksi yritinkin tehdä aivan liian mahtipontista kielioppia). Siispä, kieliopin rikkovat lauseet ovat kategorisesti niin erilaisia kuin kieliopin läpäisevät, että niiden välillä ei ole oikeastaan minkäänlaista rajamaastoa.