Ohjelmointiprosessissa kohdatut ongelmat ja niiden ratkaiseminen työkalujen ja ohjelmistojen avulla

Jarmo Isotalo

Aine
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 16. joulukuuta 2015

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Jarmo Isotalo

Työn nimi — Arbetets titel — Title

Ohjelmointiprosessissa kohdatut ongelmat ja niiden ratkaiseminen työkalujen ja ohjelmistojen avulla

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Aine	16. joulukuuta 2015	15

Tiivistelmä — Referat — Abstract

Ohjelmistojen kehityksen ja jatkokehityksen aikana tulee ohjelmistokehittäjä kohtaa useita erilaisia kysymyksiä, joiden avulla ohjelmistokehittäjä pyrkii selvittämään ohjelmiston toimintaa. Erityisesti tällaisia kysymyksiä tulee silloin, kun tutustutaan itselle ennalta tuntemattomaan ohjelmakoodiin. Tarve tuntemattomaan ohjelmakoodiin tutustumiseen tulee sekä ohjelmointivirhettä avoimen lähdekoodin sovelluksesta korjattaessa että uutta ominaisuutta avoimen lähdekoodin sovellukseen lisättäessä. Näihin kysymyksiin vastaamista varten on kehitetty useita erilaisia työkaluja, jotka ovat avoimesti kaikkien saatavilla ja kaikkien käytettävissä. Näitä työkaluja tehokkaasti käyttämällä esille tulleisiin kysymyksiin vastaaminen helpottuu. Haasteena on se, etteivät ohjelmistokehittäjät useinkaan tunne kaikkia ongelmanratkaisuun soveltuvia työkaluja. Vaikka ohjelmistokehittäjät tuntisivat työkalun, osaavat he harvoin käyttää näitä työkaluja tehokkaasti osana omaa ohjelmointiprosessia. Erityisesti nuoret ohjelmistokehittäjät päätyvät useasti hyvin tehottomiin ratkaisuihin vastauksia etsiessään.

Esittelen tässä tutkielmassa yleisiä tilanteita, joissa usein kysyttyjä kysymyksiä tulee vastaan ja esittelen näihin tyypillisiä ja tehokkaita sekä tehottomia ratkaisukeinoja. Lisäksi esittelen muutamia vastaamiseen soveltuvia työkaluja sekä toimivia tapoja tilanteista, joissa työkalua ei oltukaan juuri näihin tilanteisiin tarkoitettu.

ACM Computing Classification System (CCS):
D.2.1 [Software Engineering]: Tools
D.2.2 [Software Engineering]: Programmer workbench

Avainsanat — Nyckelord — Keywords
ohjelmistotuotanto, ohjelmointiprosessi, tiedonhaku, ohjelmointiprosessin kysymykset

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Koodatessa ohjelmistoprojekteissa sekä ohjelmistojen ylläpidossa tulee vastaan erilaisia ongelmatilanteita, joiden selvittämiseen auttaa erilaisten selventävien kysymysten kysyminen ohjelmakoodin toiminnasta ja toimimattomuudesta. Näillä kysymyksillä pyritään luomaan kuvaa siitä, miten ohjelma toimii ja mistä osista se rakentuu [SSE15, SMDV06, DER12].

Erityisesti aloitettaessa tuntemattoman projektin parissa on niin tehtävän laajuutta kuin sopivan aloituskohdan löytymistä vaikea selvittää. Erityisesti projektin koon kasvaessa sopivan kohdan löytäminen vaikeutuu. Myös uusien ohjelmointirajapintojen (API) kohdalla on useasti vaikea hahmottaa ohjelmointirajapinnan rakenne [MXBK05] ja sen tarjoamien luokkien suhteet toisiin luokkiin. Kuten normaaleissa projekteissa niin myös rajapintojen käytössä suuri koko vaikeuttaa oikean kohdan ja tarvittavien luokkien löytämistä. Näiden ongelmien ratkaisemiseen on luotu useita erilaisia työkaluja, joita ohjelmoija voi käyttää. Ohjelmoijien käytössä on myös erilaisia sosiaalisia medioita, joista löytyy ratkaisuja ongelmiin.

Tutkielman rakenne muodostuu seuraavasti: alussa tarkastellaan tyypillisiä ongelmatilanteita ja niissä syntyneitä kysymyksiä, joita ohjelmoijat ongelmia ratkoessaan esittävät. Sen jälkeen tarkastellaan muutamia juuri näihin kysymyksiin vastaamiseen suunnattuja ohjelmistoja sekä lopuksi, miten sosiaalinen media voi auttaa vastaamaan ohjelmoijien kysymyksiin.

2 Tutkimusmenetelmä

Tutkielmassa käytetyt artikkelit valittiin hakemalla aiheeseen sopivilla avainsanoilla artikkeleita Google Scholarista sekä ACM:stä. Artikkelit valikoituivat haun tuloksista otsikon, tiivistelmän sekä artikkelin ensimmäisten kappaleiden perusteella. Avainsanaperusteisen haun jälkeen valittuna oli kaksi lähtöartikkelia, joista lähdettiin liikkeelle ns. lumipallotekniikalla käyden läpi lähdeartikkelieen viitteet ja viittaukset sekä näiden artikkelien viitteet ja viittaukset arvioiden artikkelin oleellisuutta otsikon ja tiivistelmän perusteella. Hakuavaruuden kasvettua noin kahteensataan artikkeliin lopetettiin määrätietoinen uusien artikkelien etsiminen ja keskityttiin pienentämään mahdollisesti oleellisten artikkelien joukkoa artikkelien otsikon, tiivistelmän ja ensimmäiseten kappaleiden perusteella. Näin hakuavaruudesta saatiin valit-

tua noin 15 varmasti tutkielmaan sopivaa artikkelia. Kirjallisuuskatsauksen edetessä mukaan tuli myös muutama aiemmassa karsinnassa hylätty, mutta valituissa artikkeleissa kiinnostavasti viitattu artikkeli.

2.1 Tutkimuskysymykset

1. Millaisia ongelmia ohjelmistokehittäjät kohtaavat ohjelmoidessaan.
2. Miten kokemus vaikuttaa kohdattuihin ongelmiin.
3. Miten ohjelmistokehittäjät ratkaisevat kohtaamansa ongelmat.
4. Mitä työkaluja näiden ongelmien ratkaisemiseen on.
5. Mitä työkaluja ohjelmointiprosessin tueksi on.

3 Tilanteita, joissa ongelmia esiintyy

Tutkiessaan jo olemassaolevaa koodia ohjelmoija pyrkii erilaisilla loogisilla kysymyksillä sisäistämään koodin toimintaa. Näihin kysymyksiin vastaaminen ei ole aina helppoa saati nopeaa: kysymysten vastausten löytymiseen saattaa kulua jopa puolet ohjelmointiin käytetystä ajasta [KAM05]. Erilaisia kysymyksiä tulee mieleen erityisesti silloin, kun työskentelee uuden ominaisuuden tai muutostyön parissa tai kun työstettävä ohjelmakoodi on ennalta tuntematonta. Ohjelmoija ei voi löytää kunnollista vastausta ongelmiinsa perehtymättä riittävässä määrin ohjelmakoodin toimintaan.

Usein kysymykseen vastausta haettaessa tulee esiin useita uusia kysymykseen liittyviä, alkuperäistä kysymystä tarkentavia kysymyksiä, joihin vastaaminen lopulta edesauttaa alkuperäiseen kysymykseen vastaamista [SMDV06].

Alla tarkastelemme erilaisia tilanteita ohjelmointiprosessissa, jossa kysymyksiä tyypillisesti ilmenee. Selvitämme myös, millaisia kysymyksiä kussakin tilanteessa esitetään sekä tarkastelemme lyhyesti, miten tällaisiin kysymyksiin haetaan vastauksia mahdollisesti erilaisia apuohjelmia käyttäen. Sen jälkeen tarkastelemme muutamia vastaamiseen käytettyjä työkaluja tarkemmin sekä tutustumme muutamiin erityisesti näihin kysymyksiin vastaamiseen tarkoitettuihin sovelluksiin.

3.1 Tehtävän oleellisten aloituskohtien löytäminen

Uuden ohjelmointitehtävän aloittaminen on usein haastavaa, vaikka tiedossa olisi suurinpiirtein se, mitä on tarkoitus tehdä. Tilanne on sama, oli sitten kyseessä olemassaolevan ohjelmiston toimintavirheen korjaaminen tai uuden ominaisuuden lisääminen ohjelmaan. Erityisen haastavaksi aloittamisen tekee se, jos ohjelman koodi ei ole ennalta tuttua. Tällöin ennen tehtävän aloittamista tulee selvittää itselleen tarkemmin ohjelman rakennetta ja toimintaa sekä selvittää, mitkä ovat oleellisia kohtia tehtävän kannalta.

Eräs tapa aloituskohdan löytämiseen on arvata sopivia aiheeseen liittyviä avainsanoja, joiden perusteella voi hakea kohtia ohjelmakoodista. Tässä metodissa on haasteena niin oikeiden avainsanojen keksiminen [Rei09] kuin epäoleellisten tulosten suuri määrä, mikä hidastaa prosessia entisestään.

Toinen tapa sopivan aloituskohdan löytämiseen on ohjelmistoympäristön luokka- ja pakkauskaaviotyökalut. Nämä saattavat nopeuttaa hakua, mutta myös näiden käytön haasteena on sopivien avainsanojen löytäminen [Rei09].

Myös virheenjäljitintä (debugger) voi käyttää sopivan aloituskohdan paikantamiseen merkitsemällä ohjelmakoodiin pysähdyskohtia (break point) mahdollisesti sopiviin kohtiin. Seuraamalla ohjelman suoritusta virheenjäljittimellä näkee helposti, pysähtyykö ohjelman suoritus merkittyihin pysähdyskohtiin - eli suoritetaanko koodia näiden pysähdyskohtien kohdalta. Tässäkin, kuten aiemmassakin aloituskohtien paikantamiseen käytetyissä tavoissa, on haasteena sopivien pysähdyskohtien arvaaminen.

Virheenjäljitintä voidaan käyttää myös toisella tapaa: varmistetaan oleellinen kohta [KAM05] - käyttämällä muilla tavoilla saatuja arvauksia aloituskohdasta pysähdyspisteiden kohtana, jolloin virheenjäljittimellä tarkistetaan, suoritetaanko ohjelmakoodia oikeasti näistä kohdista.

Kuitenkaan mikään edellä mainituista tavoista ei ole erityisen tehokas löytämään oleellista aloituskohtaa. Haasteeksi kaikissa tavoissa muodostuu suhteellisen suuri määrä täysin epäolennaisia tuloksia [KAM05], jotka vievät ohjelmistokehittäjän aikaa ja saattavat pahimmillaan johdatella ohjelmoijan pitkäksi aikaa tarkastelemaan väärää aluetta ohjelmakoodista, ennen kuin kohdan epäolennaisuus selviää hänelle.

Tärkeää on kuitenkin löytää oikea aloituskohta, olipa keino mikä tahansa.

3.2 Riippuvuuksien hahmottaminen

Kun sopiva aloituskohta on tiedossa, voi ohjelmoija viimein keskittyä tarkastelemaan kyseistä ohjelmakoodin kohtaa, sen suhdetta ja riippuvuuksia muuhun ohjelmakoodiin [SMDV06].

Tarkastelemalla aloituskohdan ohjelmakoodissa olevia luokkia ja niiden suhdetta muuhun ohjelmakoodiin, kuitenkin aluksi keskittyen läheisiin luokkiin, niiden rakenteisiin ja metodeihin, on tarkoitus pyrkiä parantamaan ja sisäistämään omaa käsitystä oleellisesta ohjelmakoodista. Tässä tyypillisesti tarkastellaan, mitä tietorakenteita ja luokkia oleellisessa ohjelmakoodin kohdassa käytetään sekä mitä metodeja on luokassa, jossa oleellinen kohta sijaitsee. Tyypillisesti tarkastellaan myös luokan perintää sekä sen toteuttamia rajanpintoja keskittyen tarkastelussa vain lähimpien luokkien tarkasteluun, jolloin muodostetaan pieneltä alueelta tarkka yleiskuva [SMDV06]. Useat ohjelmointiympäristöt tukevat yllämainittua hyvin. Monet ohjelmointiympäristöt tarjoavat helpon navigoinnin luokkien ja niiden riippuvuuksien välillä. Myös ohjelmointiympäristöjen tarjoamat luokka- ja pakettitason tarkasteluun tarkoitetut työkalut auttavat ohjelmoijaa visaisissa kysymyksissä.

3.3 Koodin konseptien selvittäminen

Kun oleellisen aloituskohdan läheiset riippuvuudet ja rakenne ovat selvillä, on seuraava askel tarkastella koodin korkean tason konsepteja. Samalla muodostetaan kokonaiskuva ohjelmiston toiminnasta. Usein tätä selvitettäessä tarkastellaan sitä, missä oleellisen aloituskohdan sisältävä esiintymä luokasta luodaan ja mitä parametrejä sille annetaan. Tietorakenteiden kohdalla myös tietorakenteen oikeaoppisen käytön selvittäminen auttaa. Lisäksi tarkastellaan sitä, missä oleellisen kohdan luokkaa ja sen esiintymiä käytetään sekä mitkä kapseloivat kyseisiä esiintymiä ja mihin niitä käytetään [SMDV06, KAM05].

Ohjelmointiympäristöistä on tässä apua. Useat ohjelmointiympäristöt tarjoavat mahdollisuuden hakea paikkoja, mistä metodia kutsutaan sekä paikkoja, missä tiettyyn tyyppiin (Type) viitataan. Nämä auttavat selvittämään kyseisen kohdan käytön laajuutta ja paikkoja ohjelmakoodissa. Aloittelijoille tavanomainen tapa vastaavaan selvitykseen on esimerkiksi muuttaa metodin määrittelyä (signature) siten, että metodia ei enää löydy. Tämän jälkeen kääntämällä ohjelmakoodin uudestaan voi nähdä kaikki

kyseistä metodia käyttävät kohdat, sillä kääntäjä raportoi käännösvirheestä niistä kohdista, jotka käyttivät (yrittävät käyttää) kyseistä metodia. Vastaavasti luokan nimeä muuttamalla ja sen jälkeen ohjelmankoodia kääntämällä ja virheitä tarkastelemalla saa selville, missä luokkaa on käytetty. Tämä toki on poikkeuksellinen tapa, mutta aloittelijoiden keskuudessa tyypillinen.

3.4 Ohjelmointirajapinnan käyttäminen

Niin ohjelman toimintaa tarkastellessa kuin uutta ominaisuutta luodessa tulee useasti vastaan ennalta tuntemattomia ohjelmointirajapintoja (API). Joskus ohjelmointirajapinnat ovat helppokäyttöisiä, mutta useasti, varsinkin isompien kirjastojen (Library) kohdalla, ohjelmointirajapinnat ovat monipuolisia ja siten vaikeammin sisäistettävissä. Uuteen ohjelmointirajapintaan tutustuminen aloitetaan useain ensin varmistamalla, että kyseinen ohjelmointirajapinta varmasti tarjoaa tarkoitukseen sopivan toiminnallisuuden. Tämän jälkeen voi aloittaa tarkemman tutustumisen ohjelmointirajapintaan. Vastaavasti, kuten uuteen ohjelmistoon tutustuttaessa, pyritään aluksi paikantamaan ne luokat ja metodit, jotka ovat tarpeen. Niiden löydyttyä ryhdytään selvittämään löydettyjen luokkien ja metodien suhteita muihin ohjelmointirajapinnan käyttämiin tyyppeihin (type). Myös dokumentaation lukeminen auttaa, mutta se harvoin tarjoaa yleiskuvaa siitä, mihin luokka yleiskuvassa asettuu tai minkä muiden luokkien kanssa sitä tyypillisesti käytetään [DER12].

Ohjelmointiympäristöt tarjoavat vain heikosti apua uusien ohjelmointirajapintojen käytössä [MXBK05]. Usein ohjelmistoympäristöt tarjoavat automaattista täydennystä vasta silloin, kun ohjelmoija tietää, mitä luokkia ja rajapintoja tarvitaan. Niinpä useimmiten ohjelmointirajapinnan käytön sisäistämiseen käytetään vain ohjelmointirajapinnan tarjoamaa sisäistä dokumentaatiota, mikä harvoin auttaa yleiskuvan luomisessa. Myös tästä dokumentaatiosta oleellisien kohtien löytäminen on vaikeaa, sillä sopivan avainsanan keksiminen on useasti haastavaa [Rei09]. Toinen tapa tutustua ohjelmointirajapinnan käyttöön on etsiä omaan tarkoitukseen sopivia esimerkkejä kyseisen ohjelmointirajapinnan käytöstä. Tällöin haasteeksi muodostuu esimerkkien suuri määrä sekä esimerkkien vaihteleva laatu [ZBY12]. Lisäksi riippuen käytössä olevasta ohjelmointirajapinnasta sopivien esimerkkien löytäminen

on joskus hyvinkin haastavaa, ja ohjelmoijat, jotka hakevat esimerkkejä, päätyvät usein palaamaan ohjelmointirajapinnan dokumentaatioon [DER12]. Tämä kuitenkin riippuu pitkälti käytössä olevasta ohjelmointirajapinnasta sekä siitä, miten tyypillistä asiaa sillä on tekemässä.

4 TODO: Ohjelmoijan kokemuksen vaikutus kohdattuihin ongelmiin

5 TODO: Miten ohjelmistokehittäjät ratkaisevat kohtaamansa ongelmat

6 Työkaluja ohjelmointiprosessin tueksi

Ohjelmistokehittäjä ei ole aivan yksin "tyhmän"tekstieditorin kanssa etsiessään vastauksia ohjelmointiprosessin aikana esiintulleisiin kysymyksiinsä, sillä ohjelmistokehittäjiä varten on luotu erilaisia ohjelmistoja ja apuvälineitä työn tueksi. Yksi haasteista on sopivan työkalun valinta kulloinkin kyseessä olevaan tehtävään/kysymykseen: kaikki ohjelmoijan avuksi tehdyt työkalut kun eivät osaa vastata kaikkiin kysymyksiin tai kenties lainkaan auttaa juuri akuuttiin kysymykseen. Toinen haaste on se, että vaikka ohjelmistokehittäjä valitsee sopivan työkalun, käyttää hän todennäköisesti vain osaa tarkoitukseen luoduista ominaisuuksista [KM04].

Alla esittelen tyypillisiä ohjelmistokehittäjän elämän helpottamiseksi luotuja työkaluja. Osa työkaluista on tarkoitettu auttamaan yleisellä tasolla, kun taas toiset auttavat vain pienessä, mutta sitäkin haastavammassa osassa.

Ohjelmoitaessa on aina tärkeää uusiokäyttää mahdollisimman paljon olemassaolevaa koodia ja kirjoittaa uutta koodia vain mikäli valmista toteutusta ei ole olemassa. Koodia uusiokäyttämällä säästyy saman, jo olemassaolevan koodin uudelleen kirjoittamiselta ja samalla riski uusien virheiden (bugien) tuottamisesta pienenee. Ohjelmakoodin uusiokäyttämisessä on kuitenkin myös riskinsä; ohjelmakoodissa voi olla tuntemattomia ohjelmointivirheitä, tietoturvariskejä tai käytössä oleva algoritmi saattaa olla aivan liian tehoton/hidas.

6.1 Ohjelmointiympäristö

Ohjelmointiympäristö on ohjelmisto, joka pyrkii tukemaan ohjelmistokehittäjän arkea tarjoamalla kaikki olennaisimmat työkalut ohjelmiston kehittämiseen. Ohjelmointiympäristö koostuu tyypillisesti tekstieditorista, joka tukee ohjelmakoodin kirjoittamista tarjoamalla automaattista tekstin täydennystä ja ohjelmakoodin väritystä. Ohjelmointiympäristö tarjoaa myös tyypillisesti sisäänrakennetun kääntäjän, joka kertoo ohjelmakoodin syntaksivirheistä samalla kun ohjelmakoodia kirjoitetaan. Useissa ohjelmointiympäristöissä on myös sisäänrakennettu virheen etsintä, joka avustaa ohjelmakoodin suorituksen tarkastelun ohjelmakoodille rivi riviltä. Ohjelmointiympäristöt tarjoavat myös tiedostoselaimen projektin tiedostoille sekä erilaisia luokka- ja pakettitason visualisointi- sekä selaustyökaluja.

6.2 Kysymys- ja vastauspalstat

Kysymys- ja vastauspalstat (Q&A) ovat yleinen tapa hakea vastauksia ongelmiin kaikilla elämän osa-alueilla. Sama pätee ohjelmointikehityksessä: kysymys- ja vastauspalstoja käytetään paljon etsittäessä sopivia ratkaisuja ongelmiin sekä haettaessa esimerkkejä tietyn ohjelmointirajapinnan käyttöön. StackExchangen StackOverflow on erityisen suosittu ohjelmistokehittäjien keskuudessa. Sivusto keskittyy tarkasti pysymään olennaisessa eli ohjelmistokehitysaiheisten kysymyksissä ja niiden vastauksissa. Sivuston toimintaperiaate on yksinkertainen: se kerää ison osan aiheen asiantuntijoita käyttäjikseen, jotka vastaavat toistensa kysymyksiin. StackOverflowlla on yli 16 miljoonaa eri käyttäjää kuukaudessa. Käyttäjät saavat pisteitä hyvistä vastauksista, käyttäjät äänestävät jokaiseen kysymykseen tulleet vastaukset ja ne näytetään paremmuusjärjestyksessä. Näin sivulta omaan kysymykseensä vastausta etsivä ohjelmoija löytää nopeasti aiheeseen sopivimman kunnollisen vastauksen [BBS13].

StackOverflown ja muiden vastaavien sivustojen käyttäjien haasteeksi muodostuu valtava esimerkkien ja vastausten määrä. Zagalskyn, Barzilayn ja Yehudain sivusto ExampleOverflow kokoaa StackOverflowsta parhaiksi vastauksiksi merkityt koodiesimerkit. Lisäksi se tarjoaa optimoidun haun niistä. ExampleOverflow tarjoaa koodihaussa aina viisi sopivinta vastausta ja vastauksen sopimattomaksi merkitessään käyttäjä saa tilalle aina seuraavaksi

sopivimman vaihtoehdon. Sivusto tarjoaa suoraan koodiesimerkkejä, joiden alla on linkit alkuperäisiin kysymyksiin ja vastauksiin. Sivuston tavoitteena on tehdä esimerkkien löytäminen mahdollisimman vaivattomaksi. Kirjoittajien havaintojen mukaan ExampleOverflow löytää sopivia esimerkkejä useissa tapauksissa yhtä hyvin tai paremmin kuin StackOverflow [ZBY12].

StackOverflown käyttäjiä tarkasteltaessa havaittiin, että GitHubissa aktiiviset ovat keskimääräistä aktiivisempia myös vastaamaan StackOverflowssa esitettyihin kysymyksiin. Vastaavasti GitHubissa vähemmän aktiiviset olivat aktiivisempia kysymään apua StackOverflowssa [VFS13]

6.3 TODO: API doc ja projektien doc

6.4 S^6 projekti

Vaikka avointa lähdekoodia on paljon tarjolla, ei sen uudelleen käyttäminen ole kuitenkaan helppoa. Erityisen vaikeaa on sopivan ohjelmakoodin löytäminen.

S^6-projekti pyrkii helpottamaan sopivan koodin hakemista ja tukee siten koodin uudelleenkäyttöä sekä erilaisten ohjelmistokirjastojen löytämistä. S^6-projekti mahdollistaa sopivien luokkien ja metodien haun siten, että ohjelmoija tarjoaa testitapauksia, joiden rakennetta ja semantiikkaa automatisoidusti tarkastelemalla S^6-projekti pystyy rajaamaan hakutuloksia. Projekti pyrkii myös automatisoimaan tarvittavat muunnokset eri tyyppien (Type) välillä, jotta eriävät tyypit eivät rajoittaisi järjestelmän tarjoamia hakutuloksia liikaa [Rei09]. Haku ei kuitenkaan tarjoa ainoastaan juuri ohjelmistokehittäjän tarjoamaan esimerkkiin sopivaa vastausta, vaan näyttää myös suunnilleen siihen sopiva ratkaisuja. Tämä ominaisuus on kehitetty S^6-projektiin siksi, että ohjelmistokehittäjä tietää usein vain suuntaa-antavasti, mitä hän oikeasti tarvitsee.

6.5 Jungloid-louhinta

Suuri määrä mahdollisia ohjelmointirajapintoja tekee kaikkien tarpeellisten ohjelmointirajapintojen ulkoa opettelun mahdottomaksi sekä vaikeuttaa tarpeeseen soveltuvan ohjelmointirajapinnan löytämistä. Mahdollisesti sopivan ohjelmointirajapinnan löydettyään ohjelmoijan haasteeksi saattaa muodostua se, että ohjelmointirajapinta tarvitsee eri tyypit kuin mitä on tarjolla. Lähtötyypin muuntaminen ohjelmistorajapinnan tarpeeseen sopivaan tyyp-

piin ei ole aina helppoa eikä suoraviivaista. Joskus tyyppi tulee kierrättää usean muun tyypin kautta, jotta kohdetyyppiin päästään. Se tuottaa ohjelmoijalle paljon haasteita erityisesti silloin, jos tarjolla olevat tyypit eivät ole ohjelmoijalle entuudestaan tuttuja.

Jungloid-louhinta pyrkii auttamaan ohjelmoijaa tällaisessa tilanteessa. Jungloidit määritellään seuraavasti: $\lambda x.\, e \;:\; \tau_{in} \rightarrow \tau_{out}$ ja jungloid-haku määritellään parina: (τ_{in}, τ_{out}), missä τ_{in} ja τ_{out} ovat tyyppejä, joilla kuvataans mistä tyypistä mihin tyyppiin muunnos halutaan tehdä. Jungloid-louhinnan taustalla on tietovarasto erilaisista tyypeistä ja niiden suhteista. Haku toimii siten, että kun tiedetään lähtötyyppi τ_{in} sekä kohdetyyppi τ_{out}, voidaan tyyppien välille hakea erilaisia reittejä tyyppien suhdeverkosta. Tämä haku onnistuu perusverkkoalgoritmeilla. Haun laadun parantamiseksi tyyppejä voidaan muunnella ja yleistää ennen haun suorittamista. Esimerkiksi perintää tai rajapintoja käyttävä luokka voidaan tulkita jonain yläluokan tyyppinä tai rajapinnan tyyppinä, kun jungloid-haku suoritetaan [MXBK05].

6.6 Whyline

Ohjelmointiprosessin aikana ohjelmistokehittäjien kysymyksiin vastaamiseksi on luotu myös ohjelma Whyline, joka pyrkii auttamaan ohjelmistokehittäjiä vastaamaan seuraavanlaisiin kysymyksiin:

- Miksi jokin toimii
- Miksi jokin ei toimi odotetusti
- Miksi jotain tapahtuu
- Miksi jotain ei tapahdu.

Tämän tyyppisiin kysymyksiin harva työkalu tarjoaa vastauksia - suurin osa työkaluista tarjoaa korkeintaan mahdollisuuden tarkastella ohjelman toimintaa suorituksen aikana, mutta jättää kokonaan kysymykseen vastauksen selvittämisen ohjelmoijan tehtäväksi. Whyline pyrkii helpottamaan tämän tyyppisiin kysymyksiin vastaamista; se ei kerro suoraan oikeaa vastausta, vaan ohjaa ja opastaa ongelmanratkaisuprosessissa ja siinä käytettyjen työkalujen tehokkaassa käytössä - ja siten vähentää ohjelmointiongelman ratkaisuun käytettyä aikaa [KM04].

6.7 GitHub

GitHub tarjoaa niin yksittäisille ohjelmistokehittäjille kuin organisaatioille keinon hallita julkisia ja yksityisiä ohjelmistovarantoja (repository) Git-versionhallintajärjestelmällä. GitHubissa on yli 5 miljoonaa avointa lähdekoodivarastoa. Ison lähdekoodivarastomäärän hallintaa helpottamaan GitHubissa voi merkitä itseään mahdollisesti kiinnostavia lähdekoodivarastoja tähdellä. Tähdet on tarkoitettu nimenomaan kuvaamaan kiinnostusta ja auttamaan lähdekoodivarastojen löytymistä helposti myöhempää käyttöä varten - ei niinkään osoittamaan pitävänsä tai käyttävänsä kyseisen lähdekoodivaraston ohjelmaa[BBS13], kuten merkistä voisi olettaa. GitHubissa voi myös seurata muiden projekteja sekä GitHubiin rekisteröityneitä käyttäjiä.

GitHub tarjoaa kielipohjaisen lajittelun suosittujen lähdekoodivarastojen tarkasteluun. Lisäksi GitHubista voi hakea ohjelmakoodia kielen ja avainsanojen perusteella [BBS13]. Kuten muissakin avainsanapohjaisissa hauissa avainsanojen arvaaminen/löytäminen on haastavaa [Rei09], mutta sopivan ohjelmointirajapinnan käyttöesimerkin haku on taas muita työkaluja helpompaa, sillä sopivat avainsanat, luokat ja metodit ovat tyypillisesti silloin tiedossa. GitHubista esimerkin hakemisessa on myös se hyvä puoli, että koodia ei ole kirjoitettu vain esimerkiksi, vaan se on oikeasti jossain tuotantokäytössä.

6.8 Wiki

Wikit ovat yhteisöllinen tapa luoda kattavia tietopankkeja, joissa usean ihmisen tietotaito ja osaaminen yhdistyvät luoden asiasta kiinnostuneille laadukasta materiaalia. Wikejä käytetään myös avoimen lähdekoodin ohjelmien dokumentointiin sekä parantamaan ohjelmointirajapintojen dokumentaatiota. Wikit koetaan myös sähköpostilistoja ja muita perinteisempiä keskusteluseka dokumentointimuotoja helpommiksi ja selkeämmiksi käyttää juuri yhteisöllisen muokkauksen ja tiedon ajantasaisuuden vuoksi [Lou06]. Myös GitHub tarjoaa helppokäyttöisiä wikejä ohjelmistojen dokumentointiin. [sta15]

6.9 Blogit

Sosiaalisen median kasvun myötä on blogien käyttö on kasvanut myös ohjelmistokehittäjien keskuudessa. Isoissa avoimen lähdekoodin projekteissa kuten Postgresql:ssä, Gnomessa ja Python:ssa julkaistaan keskimäärin noin

kahdeksan tunnin välein uusi blogijulkaisu. Julkaisut ovat keskimäärin 150-273 sanan pituisia ja ne käsittelevät pääosin

- ohjelmiston vaatimuksia
- ohjelmiston ympärille muodostunutta yhteisöä
- ohjelmistosta lisätiedon kertomista
- ohjelmiston käyttöönottoa ja jakelua
- suunnitteluratkaisuja
- ylläpitoa
- prosessin koordinointia ja hallintaa
- lisätietoja siitä, miten asia on toteutettu ohjelmistossa.

Useimmiten aihe liittyy sellaiseen kokonaisuuteen, jonka parissa kirjoittaja on äskettäin työskennellyt. [PM11]. Myös henkilökohtaiset ohjelmistokehittäjien blogit toimivat paikkana löytää tietoa uusista ohjelmointirajapintojen ominaisuuksista sekä esimerkkejä niistä.

7 Sosiaalisen median hyödyntäminen

Sosiaalinen media, kuten Facebook ja Imdb, tarjoavat paljon hyödyllisiä suosituksia kaikille internetin käyttäjille, kuten mitä elokuvia kannattaisi katsoa tai mitä tuttavat ja lähipiiri tekevät parhaillaan. Ammattilaisten sivustot, kuten StackOverflow, ovat vastaavasti ohjelmistokehittäjille erityisesti suunnattuja sosiaalisia medioita.

Esimerkkien haku on oleellinen osa nykyaikaista ohjelmistokehitystä [ZBY12].

Ohjelmointiympäristö auttaa useissa tyypillisissä tilanteissa, mutta sen tekstieditorin automaattinen tekstin täydennys ei tyypillisesti osaa kuitenkaan ehdottaa, aiemmin ohjelmoijalle sopivia luokkia [MXBK05].

TODO tapa löytää sopiva työkalu sopivaan kysymykseen vastaamiseen on vaikeaa [SM08]

A,b,c ehdottavat ratkaisuksi kysymysiin vastaamiseen työkalua, joka tarkkailee ohjelmoijan toimintaa automaattisesti samalla tunnistaen kohdat,

missä ohjelmoijalla on tyypillisesti vaikeuksia. Sillä on myös valmiiksi määritelty informaatio auttamaan alkuun pääsyssä, joka auttaa löytämään sopivan työkalun vastaamaan yleisiin kysymyksiin. Työkalu myös ajanmittaa auttaisi ohjelmoijaa oppimaan yhä tehokkaammin käyttämään tarjolla olevia työkaluja ja ohjaa niiden käyttöön aktiivisesti. [SM08]

Ohjelmoitaessa on aina tärkeää uusiokäyttää mahdollisimman paljon olemassaolevaa koodia ja kirjoittaa uutta koodia vain mikäli valmista toteutusta ei ole olemassa. Koodia uusiokäyttämällä säästyy niin saman koodin uudelleen kirjoittamiselta ja samalla riski uusien bugien tuottamisesta pienenee.

8 Yhteenveto

Ohjelmistotuotantoprosessin aikana ohjelmistokehittäjän kohtaamiin kysymyksiin vastaaminen ei ole aina yksinkertaista, mutta ohjelmoijan kannattaa tutustua erilaisiin apuvälineisiin sen sijaan, että yrittää ratkoa ongelmiaan yksin. Ohjelmoijan työn tueksi on luotu useita erilaisia työkaluja, jotka kehittyvät jatkuvasti. Ohjelmoijan on tarpeen tuntea eri työvälineet ja pyrkiä löytämään niistä itselleen sopivin/sopivimmat ja opetella niiden käyttöä, jotta ohjelmointityö olisi mahdollisimman tehokasta.

Eri työvälineet on tehty erilaiseen käyttötarkoitukseen. Yksi parhaista työkaluista on Jungloid-louhinta [MXBK05] , jonka avulla on mahdollista ratkaista suuri osa ohjelmointirajapintojen käyttöön ja tyyppeihin liittyvistä kysymyksistä. Ohjelmistokehitysprosessin aikana tulevien kysymysten vastaamiseen liittyvät työkalut, kuten Whyline, on tarkoitettu ohjaamaan ohjelmistokehittäjä käyttämään tehtävään tarkoitettuja työkaluja ja opettaa niiden tehokkaampaa käyttöä [KM04].

Lähteet

[BBS13] Begel, Andrew, Bosch, Jan ja Storey, Margaret Anne: *Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder*. IEEE Softw., 30(1):52–66, tammikuu 2013, ISSN 0740-7459. `http://dx.doi.org/10.1109/MS.2013.13`.

[DER12] Duala-Ekoko, Ekwa ja Robillard, Martin P.: *Asking and Answering Questions About Unfamiliar APIs: An Exploratory Study.* Teoksessa *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, sivut 266–276, Piscataway, NJ, USA, 2012. IEEE Press, ISBN 978-1-4673-1067-3. http://dl.acm.org/citation.cfm?id=2337223.2337255.

[KAM05] Ko, Andrew J., Aung, Htet ja Myers, Brad A.: *Eliciting Design Requirements for Maintenance-oriented IDEs: A Detailed Study of Corrective and Perfective Maintenance Tasks.* Teoksessa *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, sivut 126–135, New York, NY, USA, 2005. ACM, ISBN 1-58113-963-2. http://doi.acm.org/10.1145/1062455.1062492.

[KM04] Ko, Andrew J. ja Myers, Brad A.: *Designing the Whyline: A Debugging Interface for Asking Questions About Program Behavior.* Teoksessa *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, sivut 151–158, New York, NY, USA, 2004. ACM, ISBN 1-58113-702-8. http://doi.acm.org/10.1145/985692.985712.

[Lou06] Louridas, Panagiotis: *Using Wikis in Software Development.* IEEE Softw., 23(2):88–91, maaliskuu 2006, ISSN 0740-7459. http://dx.doi.org/10.1109/MS.2006.62.

[MXBK05] Mandelin, David, Xu, Lin, Bodík, Rastislav ja Kimelman, Doug: *Jungloid Mining: Helping to Navigate the API Jungle.* SIGPLAN Not., 40(6):48–61, kesäkuu 2005, ISSN 0362-1340. http://doi.acm.org/10.1145/1064978.1065018.

[PM11] Pagano, Dennis ja Maalej, Walid: *How Do Developers Blog?: An Exploratory Study.* Teoksessa *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, sivut 123–132, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0574-7. http://doi.acm.org/10.1145/1985441.1985461.

[Rei09] Reiss, Steven P.: *Specifying What to Search for.* Teoksessa *Proceedings of the 2009 ICSE Workshop on Search-Driven Development-*

Users, Infrastructure, Tools and Evaluation, SUITE '09, sivut 41–44, Washington, DC, USA, 2009. IEEE Computer Society, ISBN 978-1-4244-3740-5. http://dx.doi.org/10.1109/SUITE.2009.5070020.

[SM08] Shepherd, David C. ja Murphy, Gail C.: *A Sketch of the Programmer's Coach: Making Programmers More Effective*. Teoksessa *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '08, sivut 97–100, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-039-5. http://doi.acm.org/10.1145/1370114.1370139.

[SMDV06] Sillito, Jonathan, Murphy, Gail C. ja De Volder, Kris: *Questions Programmers Ask During Software Evolution Tasks*. Teoksessa *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '06/FSE-14, sivut 23–34, New York, NY, USA, 2006. ACM, ISBN 1-59593-468-5. http://doi.acm.org/10.1145/1181775.1181779.

[SSE15] Sadowski, Caitlin, Stolee, Kathryn T. ja Elbaum, Sebastian: *How Developers Search for Code: A Case Study*. Teoksessa *Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 1600 Amphitheatre Parkway, 2015.

[sta15] staff, GitHub: *About GitHub Wikis*, 2015. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm, vierailtu 2015-10-19 .

[VFS13] Vasilescu, Bogdan, Filkov, Vladimir ja Serebrenik, Alexander: *StackOverflow and GitHub: Associations Between Software Development and Crowdsourced Knowledge*. Teoksessa *Proceedings of the 2013 International Conference on Social Computing*, SOCIALCOM '13, sivut 188–195, Washington, DC, USA, 2013. IEEE Computer Society, ISBN 978-0-7695-5137-1. http://dx.doi.org/10.1109/SocialCom.2013.35.

[ZBY12] Zagalsky, Alexey, Barzilay, Ohad ja Yehudai, Amiram: *Example Overflow: Using Social Media for Code Recommendation.* Teoksessa *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, sivut 38–42, Piscataway, NJ, USA, 2012. IEEE Press, ISBN 978-1-4673-1759-7. http://dl.acm.org/citation.cfm?id=2666719.2666728.