Millaisia kysymyksiä ohjelmoijat esittävät ja miten he hakevat niihin vastauksia

Jarmo Isotalo

Referaatti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 19. lokakuuta 2015

Sisältö

1 Johdanto

Ihmiset ovat aina hakeneet apua oppiakseen ja suoriutuakseen paremmin erilaisista tehtävistä. Ohjelmistoprojekteissa ja niiden ylläpidossa tulee esille useita kysymyksiä ohjelmakoodin toiminnasta ja toimimattomuudesta. Näillä kysymyksillä pyritään usein luomaan kuvaa siitä, miten ohjelma toimii ja mistä osista se rakentuu [9, 11, 2]. Kysymykset liittyvät niin vieraiden ohjelmointirajapintojenen (api) käyttöön [6] kuin yleisemmin.

Erityisesti aloittaessa tehtävää itselle tuntemattoman projektin parissa, on tehtävän laajuutta vaikea tiedostaa, saati sitä, mistä päin koodia sopiva aloituskohta löytyy. Erityisesti projektien koon kasvaessa sopivan kohdan löytäminen vaikeutuu. Myös uusien ohjelmointirajapintojen (API) kohdalla on useasti vaikea hahmoittaa ohjelmointirajapinnan rakenne, ja sen tarjoamien luokkien suhteet toisiin luokkiin. Isommissa ohjelmointirajapinnoissa käytön haasteeksi muodostuu usein koon tuomat haasteet löytää ohjelmointirajapinnasta oleelliset kohdat sekä saada tyypit muunnettua sopivasti ohjelmointirajapinnan tukemiin tyyppeihin. Näiden haasteiden ratkaisemiseen on useita erilaisia työkaluja sekä sosiaalisia medioita, mistä apua saattaa hakea.

Tutkielman rakenne muodostuu seuraavasti. Alussa tarkastelemme tyypillisiä kysymyksiä sekä tilanteita, missä ohjelmoijat näitä kysymyksiä esittävät. Sen jälkeen tarkastelemme muutamia juuri näinin kysymyksiin vastaamiseen suunnattuja ohjelmistoja ja miten sosiaalisen media voi tarjota vastaukset näihin kysymyksiin.

2 Tilanteita missä tulee ongelmia - miksi kyselee ja mitä

Ohjelmoitaessa tulee usein esiin erilaisia kysymyksiä, joilla ohjelmoja pyrkii paremmin sisäistämään koodin toimintaa. Kuitenkin näihin kysymyksiin vastaaminen ei ole aina helppoa saati nopeaa; tällaisiin kysymyksiin vastaamiseen saattaa kulua jopa puolet ohjelmointiin käytetystä ajasta [3]. Erityisesti tällaisia kysymyksiä tulee mieleen silloin, kun työskentelee uuden ominaisuuden tai muutostyön parissa. Sekä erityisesti silloin, kun työsetettävä ohjelmakoodi on ennalta tuntematonta. Näihin kysymyksiin vastatessaan, pyrkii ohjelmoija sisäistämään koodin toimintaa tarvittaessa määrin saadakseen tehtävän tehtyä kunnollisesti.

Usein kysymykseen vastausta haettaessa tulee esiin useita kysymykseen liittyviä, alkuperäistä kysymystä tarkentavia kysymyksiä, joihin vastaaminen lopulta edesauttaa alkuperäiseen kysymykseen vastaamista [11].

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

2.1 Tehtävään oleelisten aloituskohtien löytäminen

Uuden ohjelmointi tehtävän aloittaminen on usein haastavaa, vaikka tiedossa olisi suunnilleen, mitä on tarkoitus tehdä. Oli kyseessä sitten olemassa olevan ohjelmiston toimintavirheen korjaaminen tai uuden ominaisuuden tuominen 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.

Oleellisen kohdan löytäminen ohjelmakoodista vie erityisesti tuntematttomasta koodista paljon aikaa, eikä se ole helppoa. Eräs tapa aloituskohtan löytämiseen on arvata sopivia aiheeseen liittyviä avainsajona, ja niiden perusteella hakee kohtia ohjelmakoodista. Tässä haasteena on kuitenkin niin oikeiden avainsanojen keksiminen sekä epäoleellisten tulosten suuri määrä, mikä hidastaa prosessia entisestään [8].

Toinen tapa sopivan aloituskohdan löytämiseen on ohjelmistoympäristön (IDE) erilaiset luokka ja pakkaus kaavioiden tarkasteluun tarkoitetut työkalut. Nämä saattavat nopeuttaa hakua, muta edelleen haasteena on sopivien avainsanojen löytäminen [8].

Myös debuggeria voi käyttää sopivan aloituskohdan kohdan löytämiseen, liittämällä ohjelmakoodiin pysähdyspisteitä (break point) ja tarkastelemalla ohjelmaa suorittaessa, pysähtyykö ohjelman suorittaminen näihin pysähdyspisteisiin. Tässä haasteena, kuten aiemmissakin tavoissa, on sopivien arvausten tekemisnen pysähdyspisteiden sijainnille. Debuggeria voi myös käyttää edellä mainituilla tavoilla saadun, mahdollisesti oleellisen kohdan, oleellisuuden varmistamiseen [3].

Kuitenkaan mikään edellämainituista tavoista ei ole erityisen tehokas löytämään oleellista aloituskohtaa. Haasteeksi kaikissa tavoissa muodostuu suhteellisen suuri määrä täysin epäolennnaisia tuloksia, jotka vievät ohjelmoian aikaa ja saattavat pahimmillaan johdatella ohjelmoija pitkäksi aikaa tarkastelemaan väärää aluetta ohjelmakoodista, ennen kuin kohdan epäolennaisuus selviää [3].

2.2 Riippuvuuksien hahmottaminen

Sopivan aloituskohdan ollessa tiedossa, voi viimein keskittyä kunnolla tarkastelemaan tarkastella kyseistä kohtaa ohjelmakoodista ja sen suhdetta muuhun projektin ohjelmakoodiin, sekä aloituskohdan ohjelmakoodissa olevien käytössä luokkien riippuvuuksien hahmoittaminen [11].

Tarkastelemalla itse aloituskohdan ohjelmakoodissa olevia luokkia ja niiden suhdetta loppuun ohjelmakoodiin, kuitenkin aluksi keskittyen läheisiin luokkiin, niiden rakenteisiin ja metodeihin, on tarkoitus pyrkiä parantamaan ja sisäistämään omaa käsitystä juuri oleellisesta ohjelmakoodista. Tässä tyypillisesti tarkastellaan, mitä tietorakenteita ja luokkia oleellinen ohjelmakoodin kohta käyttää sekä mitä metodeja luokassa, jossa oleellisen kohdan sijaitsee on. Tyypillisesti myös tarkastellaan perintää, sekä mitä rajapintoja nämä toteuttavat, keskittyen tarkastelussa vain lähimpien luokkien tarkasteluun, muodostaen pieneltä alueelta tarkan yleiskuvan [11]. Tätä ohjelmointiympäristöt tukevat usein hyvin. Monet ohjelmointiympäristöt tarjoavat helpon navigoinnin luokkien ja niiden riippuvuuksien välillä. Myös ohjelmointiympärisöjen tarjoamat luokka- ja pakettitason tarkasteluun tarkoitetut työkalut sopivat hyvin.

2.3 Koodin konseptien selvittämnen

Kun oleellisen aloituskohdan läheiset riippuvuudet ja rakenne on selvillä, on seuraava askel tarkastella koodin korkean tason konsepteja, muodostaen samalla kokonaiskuvan ohjelmiston toiminnasta. Usein tätä selvittässä tarkastellaan sitä, missä oleellisen aloituskohan sisältävä esiintymä luokasta luodaan, mitä parametrejä sille annetaan. Tietorakenteiden kohdalla myös tiedorakenteen oikeaoppisen käytön selvittäminen auttaa. Lisäksi tarkastellaan sitä, missä oleellisen kohdan luokkaa ja sen esiintymiä käytetään, sekä mitkä kapseloivat kyseisiä esiintymiä ja mihin niitä käytetään [11, 3].

Ohjelmointiympäristöistä on myös tässä apua. Useat ohjelmointiympäristöt tarjoavat mahdollisuuden hakea paikkoja, mistä metodia kutsutaa 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 ohjemakoodia voi nähdä kaikki kyseistä metodia käyttävät, sillä kääntäjä joka kodasta kertoo, että kyseistä metodia ei ole. Vastaavasti luokan nimeä muuttamalla ja sen jälkeen ohjelmankoodia kääntämäällä ja virheitä tarkastelemalla saa selville, missä luokkaa on käytetty. Tämä toki on poikkeuksellinen tapa, mutta aloittelijoiden keskuudessa tyypillinen.

2.4 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 usesti ensin varmistamalla, että kyseinen ohjelmointirajapinta varmasti tarjoaa tarkoitukseen sopivan toiminnallisuuden. Tämän jälkeen voi aloittaa tarkemman tutustumisen ohjelmointirajapintaan. Vastaavasti, kuten uuteen ohjelmistoon tutustuttaessa, pyritään aluksi paikantaa ne luokat ja metodit, jotka ovat tarpeen. Ja sitten selvittämään niiden ja muiden ohjelmointirajapinnan tyyppien välisiä suhteita. Myös dokumentaation lukeminen auttaa, mutta se harvoin tarjoaa yleiskuvaa siitä, mihin luokka yleiskuvassa asettuu, sekä minkä muiden luokkien kanssa tätä tyypillisesti käytetään [2].

Ohjelmointiympäristöt tarjoavat usein heikosti apua uusien ohjelmistoraapintojen käytössä [6]. Usein ohjelmistoympäristöt tarjoavat automaattista täydennystä vasta silloin kun ohjelmoija tietää mitä luokkia ja rajapintoja tarvitaan. Siten useimmiten ohjelmointirajapinnan käyttön 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 usesti haastavaa [8]. Toinen tapa tutustua ohjelmointirajapinnan käyttöön on etsiä omaan tarkoitukseen sopivia esimerkkejä kyseisen ohjel-

mointirajapinnan käytöstä. Tällöin haasteeksi muodostuu esimerkkien suuri määrä sekä esimerkkien vaihteleva laatu [13]. Kuitenkin riippuen käytössä olevasta ohjelmointirajapinnasta, sopivien esimerkkien löytäminen on joskus hyvinkin haastavaa, ja ohjelmoijat, jotka hakevat esimerkkejä, useasti myös päätyvät palaamaan ohjelmointirajapinnan dokumentaatioon [2]. Tämä kuitenkin riippuu pitkälti käytössä olevasta ohjelmointirajapinnasta sekä siitä, miten tyypillistä asiaa sillä on tekemässä.

3 Lainauksia ja tarinaa

Sosiaalisesta media, kuten Facebook ja Imdb, tarjoavat paljon hyödyllisiä suosituksia, kuten mitä elokuvia katsoa ja mitä tuttavat ja lähipiiri tekee. Sivustot, kuten StackOverflow, ovat vastaavasti ohjelmistokehittäjille erityisesti suunnattuja sosiaalisia medioita. Haasteeksi kuitenkin tällaisten sivujen kohdalla tulee esimerkkien liiallinen määrä. Hyvänä esimerkkinä toimii myös Zagalskyn, Barzilayn ja Yehudain luoma sivusto ExampleOverflow, joka kerää StackOverflowsta koodiesimerkkejä ja pyrkii karsimaan sopimattomat ja turhat esimerkit pois, jolloin esimerkin hakija löytäisi nopeammin ja helpommim haluamansa vastauksen. He havaitsivat ExampleOverflown löytävän sopivia esimerkkejä useissa tapauksissa yhtä hyvin tai paremmin kuin StackOverflow [13].

Esimerkkien haku on oleellinen osa nykyaikaista ohjelmistokehitystä [13]

Ohjelmointiympäristö auttaa useissa tyypillisissä tilanteissa, mutta sen tekstieditoirin automaattinen tekstin täydennys ei tyypillisesti osaa kuitenkaan ehdottaa, aiemmin ohjelmoijalle sopivia luokkia [6].

4 Työkaluja avun hakuun

Ohjelmistokehittäjä ei ole aina yksin tyhmän tekstieditoirin kanssa, ilman apuvälineitä hakemassa vastauksia kysymyksiin, joita ohjelmointiprosessin aikana tulee vastaan. Vaan ohjelmistokehittäjiä varten on luotu erilaisia ohjemistoja ja apuvälineitä ratkaisemaan juuri näitä ongelmia. Kuitenkin yksi haasteista on sopivan työkalun valinta oikeaan tehtävään. Kaikki ohjelmoijan auttamista varten tehdyt työkalut kun eivät kaikkiin kysymyksiin osaa vastata eikä lainkaan auttaa niissä. Toinen haaste on myös se, että vaikka

ohjelmistokehittäjä valitsee sopivan työkalun, käyttää hän siitä vain osaa tarkoitukseen luoduista ominaisuuksista [4].

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

4.1 Ohjelmointiympäristö

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

4.2 S^6 projekti

S^6 projekti pyrkii helpottamaan sopivan koodin hakemista ja 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 tarkentamaan hakutuloksia. Projekti pyrkii myös automatisoimaan tarvittavat muunnokset eri tyyppien (Type) välillä, jotta valmiit, eri tyyppejä käyttävät metodit, toimisivat oikein ahjelmoijan tarpeisiin. Haku ei kuitenkaan tarjoa vain juuri ohjelmoistokehittäjän tarjoamaan esimerkkiin sopivaa vastausta, vaan myös suunnilleen siihen sopiva ratkaisuja. Näin siki, että ohjelmoja useasti tietää vain suuntaa-antavasti, mitä oikeasti tarvitsee [8].

4.3 Jungloid louhinta

Suuri määrä mahdollisia ohjelmointirajapintoja tekee kaikkien tarpeellisten ohjelmointirajapintojen ulkoa opettelun mahdottomaksi sekä vaikeuttaa tar-

peeseen soveltuvan ohjelmointirajapinnan löytämistä. Mahdollisesti sopivan ohjelmointirajapinnan löydettyään, haasteeksi saattaa muodostua se, että se tarvitsee eri tyypit, kuin mitä on tarjolla. Lähtötyypin muuntaminen ohjelmistorajapinnan tarpeeseen sovivaan tyyppiin ei ole aina helppoa eikä suoraviivaista. Joskus tyyppi tulee kierrättää usean muun tyypin kautta, jotta kohdetyyppiin päästään. Se tuottaa ohjelmoijalle paljon haasteita, eityisesti silloin jos tarjolla olevat tyypit eivät ole tuttuja.

Jungloid louhinta pyrkii auttamaan ohjelmoijaa tällaisessa tilanteessa. Jungloidit määritellään seuraavasti: $\lambda x.\, e \;:\; \tau_{in} \rightarrow \tau_{out}$ ja jungloidhaku määritellään parina: (τ_{in}, τ_{out}) missä τ_{in} ja τ_{out} ovat tyyppejä, siis mistä tyypistä, mihin tyyppiin. Jungloid louhinnan taustalla on tietovarasto erilaisista tyypeistä ja niiden suhteista. Haku toimii siten, että ku tiedetään lähtö tyyppi τ_{in} sekä kohde tyyppi τ_{out} voidaan tyyppien välille hakea erilaisia reittejä tyyppien suhde verkosta. Tämä haku onnistuu perus verkkoalgoritmeillä. Haun laadun parantamiseksi, tyyppejä voidaan muunnella ja yleistää ennen haun suorittamista. Esimerkiksi perintää tai rajapintoja käyttävä luokka voidaan tulkita jonnain yläluokan tyyppinä tai rajapinnan tyyppinä, kun jungloid haku suoritetaan [6].

4.4 Whyline

Ohjelmoijat kysyvät usein miksi jotain tapahtuu ja miksi jotain ei tapahtunut osana ongeman ratkaisu prosessia. Kuitenkin suuri osa tarjolla olevista ohjelmakoodin debuggaus työkaluista ei tarjoa helppoa mahdollisuutta näihin kysymyksiin vastaamiseen, vaan tarjoavat vain tapoja tarkastella ohjelmaa ja sen tilaa tietyssä kohtaa ohjelman suoritusta, jättäen suuren osan ohjelman suorituksen tarkastelusta ja ongelmien tunnistamisesta ohjelmoijan vastuulle. Whyline luotiin ohjaamaan tarkemmin ongelmanratkaisu prosessia, ohjaamaan oikeisiin kysymyksiin vastaamista ja siten vähentää debuggaamiseen käytettyä aikaa [4].

4.5 Stack Exchange

Stack Exchange on joukko kysymys ja vastaus sivustoja, jotka kattavat useita eri aihealueita, niin ruoanlaitosta, tee-se-itse projekteista kuin ohjelmoinnista. Sivustolla voi esittää kysymyksiä ja niihin saa useasti nopeasti käyttäjiltä vastaukset. Sivusto kokoaa laadukkaista kysymyksistä ja vastauksis-

ta tietopankkia, josta on helpooa hakea itselleen sopivaa kysymystä ja siihen valmista vastausta. Yksi sivustojen tärkeimmistä perjaitteista on keskittyä vain aiheeseen, ja tarjota selkeästi laadukkaimmat vastaukset, merkaten niidne uskottavuuden selkeästi. StackOverflow on osa Stack Exchangea. [1].

4.6 Stackoverflow

16 minjoonaa uniikkia vierailiaa [1] [1]

Havaittiin, että aktiivisimmat GitHubin käyttäjät ovat kokeneempia ja kysyvät hyvin vähän kysymyksiä StackOverflowssa mutta ovat aktiivisia vastaamaan kysymyksiin, kun taas vähemmän aktiiviset GitHubissa ovat aktiivisemmin kysymässä apua StackOverflowssa. [12]

TODO tapa löytää sopiva työkalu sopivaan kysymykseen vastaamiseen on vaikeeta [10]

A,b,c ehdoittavat 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. [10]

4.7 Javadoc

4.8 OSS - Github

GitHub tarjoaa palveluita niin yksittäisille kuin organisaatioille hallita julkisia ja yksityisiä ohjelmistovarantoja (repo) Git versionhallintajärjestelmällä. Github hostaa yli 5 miljoonaa avoimen lähdekoodin koodivarastoa. Ison repositoriomäärän hallintaa helpoittamaan GitHubissa voi merkata itseään mahdollisesti kiinnostavia repoja tähdellä. Tähdet on tarkoitettu nimenomaakuvaamaan kiinnostusta ja merkkaamaan repoja itselleen myöhempää käyttöä varten. Ei niinkään osoittamaan että todella tykkää ja käyttää kyseisen repon ohjelmaa [1]. GitHubissa voi myös seurata projekteja sekä GitHubiin rekisteröityneitä käyttäjiä. GitHub tarjoaa myös kielipohjaisen katselumahdollisuuden niin projekteihin, jotka ovat suosittuja, kuin projekteihin joiden suosio on äskettäin noussut huomattavasti. [1]

4.9 Wikit

Wikit ovat yhteisöllinen tapa luoda kattavia tietopankkeja, missä usean ihmisen tietotaito ja osaaminen yhdistyvät luoden asiasta kiinnostuneille laadukasta materiaalia. Wikejä käytetään myös avoimen lähdekoodin ohjelmien dokumentaatioon sekä parantamaan ohjelmointirajapintojen dokumentaatiota. Wikit koetaan myös sähköpostilistoja ja muita perinteisempiä keskustelu sekä dokumentaatiomuotoja helpommiksi ja selkeämmiksi käyttää [5]. GitHub pages mainuttu

4.10 Blogit

Sosiaalisen median kasvun myötä on blogien käyttö kasvanut myös ohjelmistokehittäjien keskuudessa. Isoissa avoimen lähdekoodin projekteissa, kuten Postgresql, Gnome ja Python julkaistaan keskimäärin noin kahdeksan tunnin välein uusi blogi julkaisu. Julkaisut ovat keskimäärin 150-273 sanaa ja ne käsittelevät pääosin joko ohjelmiston vaatimuksia, ohjelmiston ympärille muodostunutta yhteisöä, ohjelmistosta lisätiedon kertomista, ohjelmiston käyttöönottoa ja jakelua, prosessin koordinointia ja hallintaa, lisätietoja siitä, miten asia on toteutettu ohjelmistossa, suunnitteluratkaisuista sekä ylläpidosta. Useimmiten aihe liittyy asiaan, jonka parissa kirjoittaja on äskettäin työskennellyt. [7].

4.11 specifying what to search for

Avainsanahaut tuottavat usein paljon turhia tuloksia avainsanat, rakenne, metodisignaturet ja loopit yms (ast) tosin toi on turhaa infoa, tee testcaseja ja arvo keywordejä But even if the programmer could be precise here, it would not be enough. As programmers become more precise as to what they want, the odds of identifying code that exactly matches their specifications approaches zero.

Lähteet

[1] 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`.

[2] 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`.

[3] 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`.

[4] 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`.

[5] 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`.

[6] 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`.

[7] 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`.

[8] 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, Washing-

ton, DC, USA, 2009. IEEE Computer Society, ISBN 978-1-4244-3740-5. http://dx.doi.org/10.1109/SUITE.2009.5070020.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.