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 ohjelmistorajapintojenen (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 ohjelmistorajapinnan 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

Kun sopivan aloituskohta on tiedossa, voi viimein tarkastella kyseistä kohtaa ohjelmakoodisa, sen suhdetta muuhun projektiin. Seuraava asken on siis aloituskohdan alueelta riippuvuuksien hahmoittaminen [11]. Tässä ohjelmoija tarkastelee itse oleellisia luokkia selvittäen tarkemmin niiden suhdetta muuhun ohjelmakokonaisuuteen, keskittyen kuitenkin läheisiin luokkiin ja niissä oleviin rakenteisiin ja metodeihin. Tyypilisesti tässä selvitetään aluksi mitä metodeja kyseinen luokka toteuttaa, mitä tietorakenteita se käyttää sekä mihin luokkiin se viittaa. Myös mitä luokkia oleellinen luokka perii, mitä viitatut ja lähistön muut luokat perivät ja mitä rajapintoja ne toteutavat. Tarkastelu etenee hitaasti laajemalle alueelle, alkaen itse oleellisesta luokasta hitaasti laajentuen ohjelmoijan syventäessään käsitystään ohjelman rakenteesta. Näihin kysymyksiin vastaamiseen riitää ohjelmointiympäristöjen valmiit työkalut, jotka on tarkoitettu juuri koodin navigointiin.

2.3 Koodin konseptien selvittämnen

Alkupaikan sekä läheisten riippuvuuksien selvittäminen on hyvä alku; kuitenkin usein on tarpeen saada selville tarkempi kokonaiskuva ohjelmiton oleellisesta osasta ja siten parantaa yleiskuvaa ohjelmiston toiminnasta. Tämän selvittämiseksi haetaan usein vastauksia kysymyksiin, missä tämä olio luodaan, mitä parametrejä sille annetaan, miten oikeaoppisesti käytetään tiettyä tietorakennetta sekä miten ohjelma käsittelee tietyt tapaukset. Ohjelmoijat tarkastelevat löydetyn oleellisen kohdan suhdetta ohjelmistoon, mistä ja mitkä luokat käyttävät tätä, ja mitä tehdessä, mitkä luokat kapseloivat tämän ja miksi ne tekevät niin [11, 3]

2.4 Miten tätä käytetään - apit

Niin ohjelman toimintaa tarkastellessa, kuin uutta ominaisuutta luodessa tulee useasti vastaan tuntemattomia ohjelmointirajapintoja (API). Joskus ohjelmointirajapinnat ovat helppokäyttöisiä, mutta useasti, varsinkin isompien kirjastojen (Library) ohjelmistorajapinnat ovat monipuolisia ja siten vaikeammin sisäistettävissä. Aluksi, uuden ohjelmointirajapinnan kohdatessaan ohjelmoija tyypillisesti aloittaa selvittämällä, kuinka eri tyypit liittyvät toisiinsa. Tämä tapahtuu osin samoin, kuten yllä mainitussa riippuvuuksien hahmoittaminen osiossa, mutta painottuu pitkälti dokumentaatioon ja luokka kaavioihin. Dokumentaatiosta sopivien luokkien löytämisen yksi suurimmista haasteista on oikeiden avainsanojen arvaaminen. Myös haasteita ohjelmistorajapintoihin tutustuttaessa tuottaa eri luokista esiintymien luominen. Erityisesti silloin, kun luokka ei tarjoa julkista konstruktoria, vaan käytössä on rakentaja (builder) tai tehdas (factory) ohjelmointityylit. Myös luokkien dokumentoimattomat oletukset sekä ohjelmoijan mielestä epätyypillinen toteutus tuottaa haasteita [2].

2.5 hakemisen haasteet

He havaitsivat, että ohjelmistokehittäjillä oli toive, miten ohjelmistorajapinta toimisi, ja he ärsyyntyivät, kun ohjelmistorajapinta ei toiminutkaan odotetusti [2] Eri hakutapojen vaikutus, he havaitsivat noin puolen tutkittavista hakeneen esimerkkejä ja dokumentaatiota webistä, kun taas loput käyttivät vain paikallista dokumentaatiota. Kävi ilmi, että tällä hakutapaerolla ei ollut kummepaa vaikutusta tehtävästä selviämisessä. He havaitsivat, että tutkittavat aliarvioivat koodiesimerkkien haun vievän ajan. Tosin useat tutkittavat eivät onnistuneet löytämään sopivia esimerkkejä ja lopulta tyytyivät paikalliseen dokumentaatioon. [2]

Useat ohjelmointiympäristöjen työkalut koodin täydennyksessä yms, eivät tarjoa kunnollista tukea työhön liittyvien tyyppien tunnistamiseen, vaan olettavat, että ohjelmistokehittäjä tuntee jo tarvittavat tyypit. [2]

3 Tyypillisiä ongelmia

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. Esimerkkien haku on oleellinen osa nykyaikaista ohjelmistokehitystä [13], siksi on erityisen tärkeää, että hakutyökalut tarjoavat mahdollisuuden esimerkkien paikantamiseen. 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 helpommin haluamansa vastauksen. He havaitsivat ExampleOverflown löytävän sopivia esimerkkejä useissa tapauksissa yhtä hyvin tai paremmin kuin StackOverflow [13].

[1]

Esimerkkien haku on oleellinen osa nykyaikaista ohjelmistokehitystä [13]

4 Työkaluja avun hakuun

4.1 Grep

4.2 IDE

Eclipsen tarjoama tuki koodin täydennykseen olettaa, että ohjelmoija tietää tarpeelliset luokat, eikä ohjelmointiympäristö tarjoa mahdollisesti sopivia luokkia ohjelmoijalle [6].

4.3 S^6 projekti

S^6 projekti pyrkii helpottamaan sopivan koodin hakemista ja siten koodin uudelleenkäyttöä. Projekti mahdollistaa sopivien luokkien ja metodien haun siten, että ohjelmoija tarjoaa testitapauksia, joiden rakennetta ja semantiikkaa automatisoidusti tarkastelemalla projekti pystyy tarkentamaan hakutuloksia. Projekti pyrkii myös automatisoimaan tarvittavat muunnokset, jotta valmiit, eri tyyppejä käyttävät metodit, toimisivat oikein ahjelmoijan tarpeisiin. He korostavat, että hausta ei tule tehdä liian tarkaa, sillä harva toteutusratkaisu on lopullinen, vaan toteutus ja sen rakenne voi tarvittaessa mukautua sopimaan tarjolla olevaan koodiin. Siis vain korkean tason tieto siitä, mikä haluttu lopputulos on tiedossa. [8]

4.4 Jungloid louhinta

Suuri määrä mahdollisia ohjelmointirajapintoja tekee kaikkien tarpeellisten ohjelmointirajapintojen ulkoa opettelun mahdottomaksi sekä vaikeuttaa tarpeeseen sopivan ohjelmointirajapinnan löytämistä. Mahdollisesti sopivan ohjelmointirajapinnan löydettyään, haasteeksi saattaa muodostua se, että se tarvitsee eri tyypit, kuin mitä on tarjolla. Lähtötyypin muuntaminen sovivaan tyyppiin ei ole aina helppoa eikä suoraviivaista. Joskus tyyppi tulee kierrättää usean muun tyypin kautta, jotta kohdetyyppiin päästään. Se tuottaa ohjelmoijalle joskus paljon haasteita, eityisesti silloin jos tarjolla olevat tyypit eivät ole tuttuja. Tässä avuksi tulee Jungloid louhinta. Jungloidit määritellään seuraavasti: $\lambda x.\, e \;:\; \tau_{in} \;\rightarrow\; \tau_{out}$. 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. Kun haussa tiedetään lähtö tyyppi τ_{in} sekä kohde tyyppi τ_{out} voidaan tyyppien suhde verkosta hakea mahdollisia reittejä näiden kahden tyypin väliltä, perus verkko algoritmeillä [6]

4.5 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].

4.6 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 vastauksista tietopankkia, josta on helpooa hakea itselleen sopivaa kysymystä ja siihen valmista vastausta. Yksi sivustojen tärkeimmistä perjaitteista on keskit-

työ vain aiheeseen, ja tarjota selkeästi laadukkaimmat vastaukset, merkaten niidne uskottavuuden selkeästi. StackOverflow on osa Stack Exchangea. [1].

4.7 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]

4.8 Javadoc

4.9 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]

4.10 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 ohjelmistorajapintojen dokumentaatiota. Wikit koetaan myös sähköpostilistoja ja muita perinteisempiä keskustelu sekä dokumentaatiomuotoja helpommiksi ja selkeämmiksi käyttää [5]. GitHub pages mainuttu

4.11 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].

4.12 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.