

Millaisia kysymyksiä ohjelmoijat esittävät ja miten he hakevat niihin vastauksia

Jarmo Isotalo

Referaatti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 19. lokakuuta 2015

Sisältö

1	Johdanto	1
2	Tilanteita missä tulee ongelmia - miksi kyselee ja mitä	1
2.1	Tehtävään oleellisten aloituskohtien löytäminen	2
2.2	Riippuvuuksien hahmottaminen	3
2.3	Koodin konseptien selvittäminen	3
2.4	ohjelmointirajapinnan käyttäminen	4
3	Lainauksia ja tarinaa	5
4	Työkaluja avun hakuun	5
4.1	Ohjelmointiympäristö	6
4.2	Kysymys ja vastaus palstat	6
4.3	S ⁶ projekti	7
4.4	Jungloid louhinta	7
4.5	Whyline	8
4.6	GitHub	8
4.7	Wiki	9
4.8	Blogit	9
	Lähteet	10

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 ohjelmointirajapintojen (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 hahmottaa ohjelmointirajapinnan rakenne, ja sen tarjoamien luokkien suhteet toisiinsa luokkiin. Isommissa ohjelmointirajapinnoissa käytön haasteeksi muodostuu usein koon tuomat haasteet löytää ohjelmointirajapinnasta oleelliset kohdat sekä saada tyytit 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 tyyppillisiä kysymyksiä sekä tilanteita, missä ohjelmoijat näitä kysymyksiä esittävät. Sen jälkeen tarkastelemme muutamia juuri näihin 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 ohjelmistokehittäjä 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 työstettävä 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 ilmenee. Selvitämme myös millaisia kysymyksiä kussakin tilanteessa esitetään sekä tarkastelemme lyhyesti miten tällaisiin kysymyksiin haetaan vastauksia mahdollisesti erilaisia apuohjelmia käyttäen. Sen jälkeen tarkastelemme muutamia näihin kysymyksiin vastaamiseen käytettyjä työkaluja tarkemmin sekä tutustumme muutamiin erityisesti näihin kysymyksiin vastaamiseen tarkoitettuihin sovelluksiin.

2.1 Tehtävään oleellisten 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 tuntemattomasta koodista paljon aikaa, eikä se ole helppoa. Eräs tapa aloituskohdan löytämiseen on arvata sopivia aiheeseen liittyviä avainsanoja, 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, mutta 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 tekeminen pysähdyspisteiden sijainnille. Debuggeria voi myös käyttää edellä mainituilla tavoilla saadun, mahdollisesti oleellisen kohdan, oleellisuuden varmistamiseen [3].

Kuitenkaan mikään edellä mainituista tavoista ei ole erityisen tehokas löytämään oleellista aloituskohtaa. Haasteeksi kaikissa tavoissa muodostuu suhteellisen suuri määrä täysin epäolennaisia tuloksia, jotka vievät ohjelmistokehittäjän 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 hahmottaminen [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 ohjelmointiympäristöjen tarjoamat luokka- ja pakettitason tarkasteluun tarkoitetut työkalut sopivat hyvin.

2.3 Koodin konseptien selvittäminen

Kun oleellisen aloituskohdan läheiset riippuvuudet ja rakenne on selvillä, on seuraava askel tarkastella koodin korkean tason konsepteja, muodostaen samalla kokonaiskuvan ohjelmiston toiminnasta. Usein tätä selvittäessä tarkastellaan sitä, missä oleellisen aloituskohdan sisältävä esiintymä luokasta luodaan, mitä parametreja 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 [11, 3].

Ohjelmointiympäristöistä on myös 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 ohjelmakoodia voi nähdä kaikki kyseistä metodia käyttävät, sillä kääntäjä joka kodasta kertoo, että kyseistä metodia ei ole. Vastavasti luokan nimeä muuttamalla ja sen jälkeen ohjelmakoodia kääntämällä 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 useasti 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 ohjelmointirajapintojen käytössä [6]. Usein ohjelmistoympäristöt tarjoavat automaattista täydennystä vasta silloin kun ohjelmoija tietää mitä luokkia ja rajapintoja tarvitaan. Siten useimmiten ohjelmointirajapinnan käytön sisäistämiseen käytetään vain ohjelmointirajapinnan tarjoamaa sisäistä dokumentaatiota, mikä harvoin auttaa yleiskuvan luomisessa. Myös tästä dokumentaatiosta oleellisten kohtien löytäminen on vaikeaa, sillä sopivan avainsanan keksiminen on useasti 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 [14]. 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.

Esimerkkien haku on oleellinen osa nykyaikaista ohjelmistokehitystä [14]

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

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

A,b,c ehdoittavat ratkaisuksi kysymyksiin vastaamiseen työkalua, joka tarkkailee ohjelmoijan toimintaa automaattisesti samalla tunnistuen 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 Työkaluja avun hakuun

Ohjelmistokehittäjä ei ole aina yksin tyhmän tekstieditorin kanssa, ilman apuvälineitä hakemassa vastauksia kysymyksiin, joita ohjelmointiprosessin aikana tulee vastaan. Vaan ohjelmistokehittäjiä varten on luotu erilaisia ohjelmistoja 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 esittelen 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 tekstieditorista, 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 Kysymys ja vastaus palstat

Kysymys ja vastaus (Q&A) palstat ovat yleinen tapa hakea vastauksia ongelmiin kaikilla elämän osa-alueilla. Sama pätee ohjelmointikehityksessä, kysymys ja vastaus palstoja käytetään paljon etsittäessä sopivia ratkaisuja ongelmiin sekä haettaessa esimerkkejä tietyn ohjelmointirajapinnan käyttöön. StackExchangen StackOverflow on erityisen suosittu ohjelmistokehittäjien keskuudessa. Sivusto keskittyy tarkasti pysymään vain siinä olennaisessa, oli ohjelmistokehitys aiheisissa kysymyksissä. 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 paremmuus järjestyksessä.

Näin sivulta omaa kysymystä vastaavan kysymyksen löytäjä löytää nopeasti aiheeseen sopivimman kunnollisen vastauksen [1].

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, ja tarjoaa optimoidun haun niistä. ExampleOverflow tarjoaa koodihaussa aina viisi sopivinta vastausta, ja vastauksen sopimattomaksi merkitessään saa tilalle aina seuraavaksi sopivimman vaihtoehdon. Sivusto tarjoaa suoraan koodi esimerkkejä ja niiden alta linkit alkuperäisiin kysymyksiin ja vastauksiin. Sivuston tavoite on tehdä esimerkkien löytäminen vaivattomammaksi. Kirjoittajien havaintojen mukaan ExampleOverflow löytävän sopivia esimerkkejä useissa tapauksissa yhtä hyvin tai paremmin kuin StackOverflow [14].

StackOverflown käyttäjiä tarkasteltaessa havaittiin, että GitHubissa aktiiviset ovat keskimääräistä aktiivisempia myös vastaamaan StackOverflowsa esitettyihin kysymyksiin. Vastaavasti GitHubissa vähemmän aktiiviset olivat aktiivisempia kysymään apua StackOverflowssa [13]

4.3 S⁶ projekti

S⁶ projekti pyrkii helpottamaan sopivan koodin hakemista ja siten koodin uudelleenkäyttöä, sekä erilaisten ohjelmistokirjastojen löytämistä. S⁶-projekti mahdollistaa sopivien luokkien ja metodien haun siten, että ohjelmoija tarjoaa testitapauksia, joiden rakennetta ja semantiikkaa automatisoidusti tarkastelemalla S⁶ -projekti pystyy tarkentamaan hakutuloksia. Projekti pyrkii myös automatisoimaan tarvittavat muunnokset eri tyyppien (Type) välillä, jotta valmiit, eri tyyppisiä käyttävät metodit, toimisivat oikein ohjelmistokehittäjän tarpeisiin. Haku ei kuitenkaan tarjoa vain juuri ohjelmoistokehittäjän tarjoamaan esimerkkiin sopivaa vastausta, vaan myös suunnilleen siihen sopiva ratkaisu. Näin siksi, että ohjelmistokehittäjä useasti tietää vain suuntaa-antavasti, mitä oikeasti tarvitsee [8].

4.4 Jungloid louhinta

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

peeseen soveltuvan ohjelmointirajapinnan löytämistä. Mahdollisesti sopivan ohjelmointirajapinnan löydettyään, haasteeksi saattaa muodostua se, että se tarvitsee eri tyyppit, kuin mitä on tarjolla. Lähtötyypin muuntaminen ohjelmistorajapinnan tarpeeseen sopivaan tyyppiin ei ole aina helppoa eikä suoraviivaista. Joskus tyyppi tulee kierrättää usean muun tyyppin kautta, jotta kohdetyyppiin päästään. Se tuottaa ohjelmoijalle paljon haasteita, erityisesti silloin jos tarjolla olevat tyyppit 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 tyyppistä, mihin tyyppiin. Jungloid louhinnan taustalla on tietovarasto erilaisista tyypeistä ja niiden suhteista. Haku toimii siten, että kun tiedetään lähtö tyyppi τ_{in} sekä kohde tyyppi τ_{out} voidaan tyyppien välille hakea erilaisia reittejä tyyppien suhde verkosta. Tämä haku onnistuu perus verkkoalgoritmeilla. 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].

4.5 Whyline

Ohjelmointiprosessin aikana ohjelmistokehittäjien kysymyksiin vastaamiseksi on luotu ohjelma Whyline, joka pyrkii auttamaan ohjelmistokehittäjiä vastaamaan kysymyksiin, kuten miksi jokin toimii tai ei toimi odotetusti, miksi jotain tapahtuu tai ei tapahdu. Juuri tämän tyyppisiin kysymyksiin harva työkalu tarjoaa vastausmahdollisuutta. 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; ei suoraan oikeaa vastausta kertomalla, vaan ohjaamalla ja opastamalla ongelmanratkaisu prosessissa ja siinä käytettyjen työkalujen tehokkaassa käytössä - ja siten vähentämällä siinä käytettyä aikaa [4].

4.6 GitHub

GitHub tarjoaa niin yksittäisille ohjelmistokehittäjille kuin organisaatioille hallita julkisia ja yksityisiä ohjelmistovarantoja (Repository) Git version-

hallintajärjestelmällä. GitHubissa on yli 5 miljoonaa avoimen lähdekoodivarastoa. Ison lähdekoodivarastomäärän hallintaa helpottamaan GitHubissa voi merkata itseään mahdollisesti kiinnostavia lähdekoodivarastoja tähdellä. Tähdet on tarkoitettu nimenomaan kuvaamaan kiinnostusta ja merkkamaan lähdekoodivarastoja itselleen myöhempää käyttöä varten - ei niinkään osoittamaan että todella tykkää ja käyttää kyseisen lähdekoodivaraston ohjelmaa[1]. GitHubissa voi myös seurata projekteja sekä GitHubiin rekisteröityneitä käyttäjiä.

GitHub tarjoaa kielipohjaisen lajittelun suosittujen lähdekoodivarastojen tarkasteluun. Lisäksi GitHubista voi hakea ohjelmakoodia kielen ja avainsanojen perusteella [1]. Kuten muissakin avainsanapohjaisissa hauissa avainsanojen arvaaminen on haastavaa [8], mutta sopivan ohjelmointirajapinnan käyttöesimerkin haku on taas 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 käytössä.

4.7 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 dokumentaatioon sekä parantamaan ohjelmointirajapintojen dokumentaatiota. Wikit koetaan myös sähköpostilistoja ja muita perinteisempiä keskustelu sekä dokumentaatiomuotoja helpommiksi ja selkeämmiksi käyttää juuri yhteisöllisen muokkauksen ja tiedon ajantasaisuuden vuoksi [5]. Myös GitHub tarjoaa helppokäyttöisiä wikejä ohjelmistojen dokumentointiin. [12]

4.8 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 koordinoitua ja hallintaa, lisätietoja sii-

tä, miten asia on toteutettu ohjelmistossa, suunnitteluratkaisuksista sekä ylläpidosta. Useimmiten aihe liittyy asiaan, jonka parissa kirjoittaja on äskettäin työskennellyt. [7]. Myös blogit toimivat paikkana löytää tietoa uusista ohjelmointirajapintojen ominaisuuksista sekä esimerkkejä niistä.

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

- [13] 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>.
- [14] 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>.