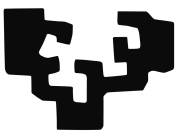
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

Mikroblogintza tekniken erabilera programazioan sortzen diren zalantzak ebazteko

---

Egilea

Jeremías Pérez Contell

2013

Laburpena

Dokumentu honek mikroblogintza plataformen gaia jorratzen duen gradu amaierako proiektu bat deskribatzen du, programazioan sor daitezkeen arazoak ebazteko testuinguru batetik ikusita.

Gaur egun Twitter moduko mikroblogintza aplikazioak zalantzak argitzeko gero eta gehiago erabiltzen dira. Erabiltzaileak laguntza eskaera adierazi eta sare sozialaren bidez bere kontaktuei helarazi egiten zaie. Erabiltzaileak galdera eta erantzunen kudeaketa sare sozial nahaste baten barruan egin beharko du.

Proiektuaren helburu nagusia mikroblogintza plataformetako kanala programazio lengoaien IDEen barruan bateratzea da, modu honetan laguntza eskatzeko prozesua IDEetan bertan kudeatzeko. Proiektuan zehar helburu honetarako garatutako tresnari CrowdCall izena eman zaio, jendetzari egiten zaion eskaera irudikatu nahian.

Honakoak dira CrowdCall-en ezaugarri nagusiak:

- Editore desberdinetara egokitzeko azpiegitura.
- Erabiltzaileari laguntza prozesuarekin elkarreragiteko aukera ematen dion interfaze grafiko intuitiboa.
- Mikroblogintza plataformetako eskaera eta erantzunen kudeaketa integratua.

Beraz, dokumentu honek CrowdCall garatzeko helburua izan duen proiektuaren nondik norakoak azaltzen dira, kudeaketa arloa nahiz arlo teknikoa landuta.

Hitz gakoak: CrowdCall, mikroblogintza, crowdsourcing eta DSL editoreak.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

Eskertzea

Onekin Taldeari, batez ere proiektuaren zuzendari *Óscar Diaz*i, haiekin lan egiteko aukera emateagatik eta proiektuan zehar eskaini didan laguntza itzelarengatik; eta *Cristóbal Arellano*ri, inplementazioaren inguruan emandako aholku guztiengatik.

Fakultateko nire klasekideei, karrerako momenturen batean nire klasekide izan direnei eta nire klasekideak ez izan arren nirekin harremana izan duten fakultateko kideei, elkarrekin bizitako momentu on guztiengatik.

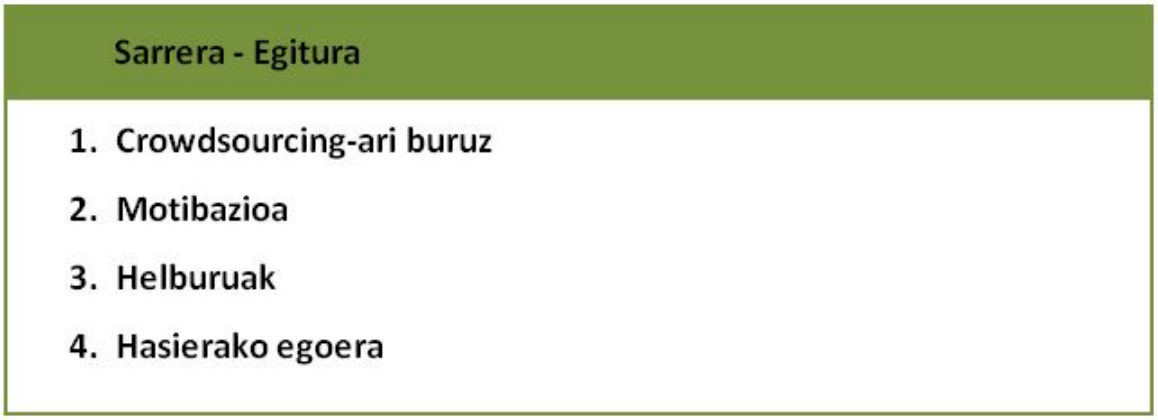
Karrera guztian zehar izandako irakasleei, Goi Mailako Ingeniaritzako lehenengo urtean izandakoei zein Graduko azkeneko hiru urteetan izandakoei, une honetaraino iristeko prestatzeagatik.

Azkenik, baina ez horregatik garrantzi gutxiagokoa, nire familiari, beti nire ondoan egoteagatik.

1. KAPITULUA

Sarrera

Atal honetan proiekturako sarrera egingo da. Lehenengo atalean Crowdsourcing-a zertan datzan azalduko da. Ondoren, proiektuaren motibazioa, helburuak eta hasierako egoera azalduko dira.



1.1 Crowdsourcing-ari buruz

Proiektu honen aurrekari bezala crowdsourcing praktikak har genitzake. Crowdsourcing terminoa ingelesezko *crowd* (jendetza) eta *outsourcing* (kanpo-errekrutamendua) hitzetatik dator, eta ekimen bat aurrera eramateko pertsona kopuru handi baten lankidetza baliatzearen praktika adierazten du. Bereziki Interneten garapenarekin batera ezagutzera eman den praktika da, honek pertsona askorekin batera harremanetan jarri eta koordinatzeko aukera eskaintzen duelako. Praktika honen adibideen artean Wikipedia webgunea aurkitzen da.

Crowdsourcing-a arazoen sorrera eta ebazpenerako eredu bat da. Orokorrean, arazoak elkar ezagutzen ez duten pertsona talde bati deialdi ireki baten bidez helarazten zaizkio, hauek ebatzi ditzaten. Normalean, erabiltzaile hauek komunitateetan elkartzen dira, ebazpenak proposatzeko eta proposatutakoak hobetzeko. Orduan, ebazpen hoberenak eskatzailearenak izatera pasatzen dira. Kasu batzuetan, ebazpenik hoberenak proposatzen dituzten erabiltzaileak sarituak izaten dira.

Crowdsourcing-aren abantailei dagokienez, guztietatik garrantzitsuena inolako gasturik izan gabe arazo bat ebazteko modu asko lortzeko ahalmena dela kontsidera daiteke. Bestalde, kolaboratzaileek parte-hartzeko motibazio ugari daude, harreman soziala eta estimulu intelektuala, esate baterako.

Hala ere, eredu honen aurkako kritiko asko daude. Hauek diotenez, kasu askotan praktika honen bidez garatutako proiektu baten kostua modu tradizionalean garatzeak suposatuko lukeen kostua baino handiagoa izan daiteke. Horrez gain, modu honetan garatutako proiektuek izan ditzaketen konfidentzialtasun arazoak nabarmentzen dituzte.

1.2 Motibazioa

DSL bat espezializatutako lengoaia bat da, domeinu jakin baterako ebazpenak inplementatzeko diseinatuta dagoena. Lengoaia hauek ebazpena domeinuko kontzeptuen terminoetan deskribatzen dute, helburu orokorreko programazio lengoaia (GPL) baten bidezko egikaritzapenaren ordez. DSL lengoaien adibideen artean MatLab (zenbaki-prozesatzeak egiteko), XUL (interfaze grafikoak sortzeko) edo SQL (datu-baseekin elkarreragiteko) aurki ditzakegu.

Helburua interesatuak soluzio programatikoak garatzeko prozesuan sartzea da, la-

guntzarik behar ez izatera iritsi arren. Hala ere, hau ez da beti gertatzen. Beharbada domeinuko adituek ez dituzte DSLaren gramatikari buruzko alderdi guztiak menperatzen, edo soluzioa lortzeko beharrezko daturen baten falta izan dezakete.

Arazo honi orain arte emandako irtenbidea sare profesionalei lotuta egon da. Bai aurrez aurreko laguntzaren bidez bai Interneteko foroen bidez, sare profesionalak langileek egindako galderak erantzuteaz arduratu dira. Konpainietako langileak geroz eta sakabanatuagoak daudenez, hauek SNS aplikazioetara jotzen hasi dira galderen argitze honekin jarraitzeko.

DSL lengoaiak domeinu berezikoak direnez, SNS aplikazioak lengoaia hauetako galderak egiteko erabiltzeak bi eragin nagusi ditu. Batetik, erabiltzaileek ez dute zertan programatzaileak izan, domeinuko adituak izan daitezke. Bestetik, galderak egiteko eta erantzuteko prozesua DSLen eraikitzaileen testuinguruan jartzen da. Honen ondorioz, egin daitezkeen galderei eta erantzunak tratatzeko moduari buruzko domeinuko ezagutza biltzen duten tresna laguntzaileak sor daitezke. Tresna hauek erabiltzaileari crowdsourcing plataforma definitu eta kudeatzeko beharra gainetik kentzen diote, eta ondorioz, denbora eta dirua aurrezteko baliagarriak dira.

Proiektu honetan mota honetako tresna bat garatu da. Garatutako tresna DSL editoreetan integratzeko diseinatuta dago, eta erabiltzaileek editore hauen bidez sortutako arazoen ebazpen prozesua kudeatuko du. Horretarako, arazoa ebazteko galderaren idazketa SNS aplikazioetan eta hauetan jasotako erantzunen kudeaketa tresnaren barruan egingo da.

1.3 Helburuak

Proiektuaren helburuak honakoak dira:

- Merkatuan dauden mikroblogintza plataformetara ohitzera, batez ere Twitter-en APIari erreparatuz.
- Programazioan sortzen diren zalantzak argitzeko prozesua errazten duen tresna baten garapena. Tresna honek ezaugarri hauek izan beharko ditu:
  - Mikroblogintza sareetan idatzitako laguntza eskaera eta erantzunen kudeaketa bateratua.

- Erabiltzaileari bere arazoaren ebazpen prozesuarekin elkarreragiteko aukera ematen dion interfaze grafikoa.
- Programazio lengoaietako IDE desberdinetan integratzeko mekanismoa.

1.4 Hasierako egoera

Proiektu hau Onekin ikerketa taldeak egindako lanetan eta idatzitako artikuluetan oinarritu da. Horrela, garatutako tresna talde honek aurretik egindako analisi lanaren ondorio da eta definitutako eredu kontzeptualean oinarritzen da. Eredu kontzeptual hau 3.1.1 azpiatalean azalduko da.

Bestalde, sortutako tresnaren interfaze grafikoa garatzeko ikerketa taldeak egindako zirriborro bat hartu da abiapuntu bezala.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Kapitulu honetan Proiektuaren Helburuen Dokumentua azaltzen da. Honek proiektuaren irismena, denboraren planifikazioa, baliabideen esleipena eta proiektua arrakastaz burutu ahal izateko garatutako kalitate, arrisku eta komunikazioak kudeatzeko planak biltzen ditu.

Proiektuaren Helburuen Dokumentua - Egitura
1. Proiektuaren irismena
2. Denboraren planifikazioa
3. Baliabideen esleipena
4. Komunikazio plana
5. Kalitatearen kudeaketa plana
6. Arriskuen kudeaketa plana

2.1 Proiektuaren irismena

Atal honetan proiektuaren irismena deskribatuko da. Horretarako, lehendabizi proiektuaren betekizunak azalduko dira. Ondoren, LDE diagrama baten bidez egin beharreko lana deskonposatuko da. Honen ostean, proiektuko emangarri nagusiak zerrendatuko dira, eta azkenik, proiektuaren mugarriak identifikatuko dira.

2.1.1 Betekizunak

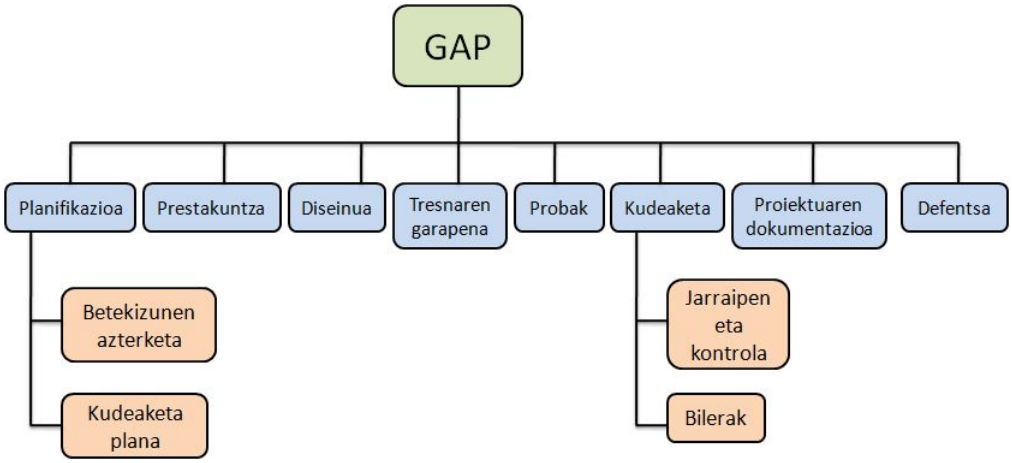
Proiektuaren helburu nagusia mikroblogintza plataformetako mezuak kudeatzeko tresna bat garatzea da. Garatutako tresna programazio lengoaietako IDEtan txertatu beharko da, eta programazio prozesuan zehar sortzen diren arazoak ebazteko balio beharko du.

Tresna batez ere *Sticklet* editorearekin batera lan egiteko pentsatuta dago. Ondorioz, lehendabizi tresna honetan bateratu beharko da, DSL honetako adierazpenak sortzeko prozesuan laguntza gisa erabiltzeko. Hau lortu ondoren, beste editore batzuetan erabili ahal izateko egokitu beharko da. Horretarako, tresnak konfiguratzeko aukera eman beharko du, editore desberdinetara moldatzeko.

Mikroblogintza plataformetan programazio zalantzak argitzeko laguntza eskatzeko prozesuak honako urratsak jarraitu beharko ditu:

1. erabiltzaileak laguntza eskaera adieraziko du
2. laguntza eskaera erabiltzaileak SNS plataformetan dituen kontaktuei helaraziko zaie
3. kontaktuek egindako erantzunak tresnaren barruan integratuko dira
4. erantzun hauetatik arazoa ebazteko baliagarriak diren datuak erauziko dira
5. erauzitako datuetatik arazoaren ebazpenerako egokiena aukeratuko da
6. emaitza IDEan integratuko da

Tresna garatzeko HTML5, Greasemonkey eta JavaScript teknologiak erabili beharko dira. Interfaze grafikoari dagokionez, erabiltzaileari ebazpen egoeraren berri emateko eta aldi berean berekin elkarreragiteko aukera eman beharko du. Horretarako, arazoaren ebazpen egoera islatzen duen grafo bat erabiliko da, JointJS eta Raphaël liburutegiak erabilita garatu beharko dena.



2.1 Irudia: Proiektuko lanaren deskonposaketa irudikatzen duen LDE diagrama.

2.1.2 LDE diagrama

2.1 irudian proiektuko lanaren deskonposaketa irudikatzen da. Jarraian, diagramako elementu bakoitzak irudikatzen duen lana deskribatuko da.

- **GAP** - Gradu Amaierako Proiektua irudikatzen du.
- **Planifikazioa** - proiektuko planaren garapena, proiektua arrakastaz burutzeko helburuarekin. Honen barruan, bi azpielementu bereiz daitezke:
  - **Betekizunen azterketa** - proiektua arrakastaz burutzeko bete beharreko puntuen azterketa.
  - **Kudeaketa plana** - proiektua modu egokian kudeatzeko plana garatzea.
- **Prestakuntza** - tresna garatu ahal izateko beharrezko gaitasun teknikoak lortzea.
- **Diseinua** - tresnak izango duen arkitektura, osagaiak, interfazeak eta bestelako ezaugarrien definizioa.
- **Tresnaren garapena** - tresna inplementatzea.
- **Probak** - tresnak izan ditzakeen akatsak bilatzea probak eginez.
- **Kudeaketa** - proiektuan egindako lana kudeatzea. Elementu hau aldi berean bi azpielementuetan deskonposatzen da:
  - **Jarraipen eta kontrola** - proiektuan egindakoari buruzko datuak lortu eta planifikazioan egindakoekin alderatu.

  - **Bilerak** - proiektuko pare-hartzaileak biltzea, proiektuaren eta produktuaren inguruko erabakiak hartzeko.

- **Proiektuaren dokumentazioa** - proiektuaren memoriaren eta sor daitezkeen bestelako dokumentuen idazketa.
- **Defentsa** - proiektua epaimahaiaren aurrean defendatzeko aurkezpena prestatzea.

2.1.3 Emangarriak

Proiektuan bi emangarri mota bereiz daitezke: garatu beharreko produktuarekin lotura estua dutenak eta proiektuarekin zerikusia dutenak.

Produktuarekin lotutakoak

Garatu beharreko produktuarekin lotutako emangarrien artean, honakoak aurkituko dira:

- *Sticklet* editorearekin bateragarria den tresna.
- IDE desberdinetarako bateragarria den tresna.
- Tresna instalatzeko eskuliburua.
- Erabiltzailearen eskuliburua

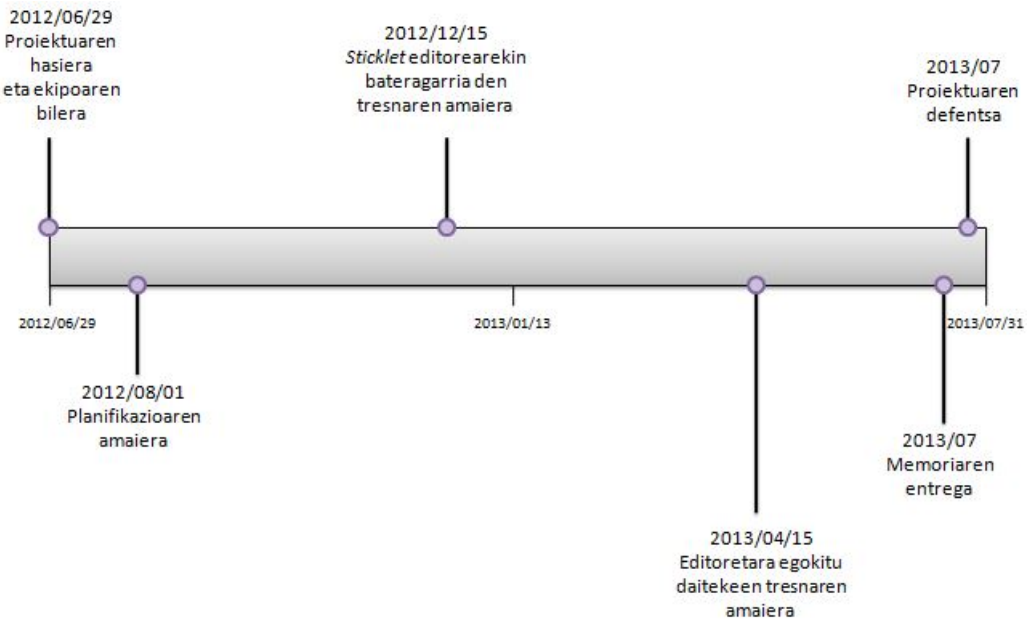
Proiektuarekin lotutakoak

Proiektuaren kudeaketarekin lotutako emangarriei dagokienez, honakoak bereiz daitezke:

- Proiektuko plana.
- Proiektuaren memoria dokumentua.
- Proiektuaren defentsarako gardenkiak.

2.1.4 Mugarriak

2.2 irudiko diagraman proiektuak izango dituen mugarri nagusiak ikus daitezke. Bertan, tresnaren garapenaren bi faseen amaiera azaltzen da. Hala ere, planifikazioa egiterakoan aurreikusitako gutxi gorabeherako datak baino ez dira, eta dira zertan zehatz-mehatz bete.



2.2 Irudia: Proiektuko mugarri nagusiak biltzen dituen diagrama.

2.2 Denboraren planifikazioa

Atal honetan proiektuan burutuko diren atazen kudeaketa egingo da. Aurrena, atazak identifikatuko dira eta horietako bakoitzak izango duen iraupenaren estimazio bat egingo da. Ondoren, ataza horien antolaketa egingo da, kronograma batean kokatuz.

2.2.1 Atazak

Hauek dira identifikatu diren atazak, arloka sailkatuta:

Planifikazioa:

- *Betekizunak aztertu*: proiektuko betekizunak bildu eta ulertzea.
  - Estimatutako iraupena: 3 ordu.
- *Kudeaketa plana garatu*: proiektua modu egoki batean kudeatzeko planaren sorrera. Plan honek proiektuaren irismena, denboraren planifikazioa, baliabideen esleipena, komunikazio plana eta kalitatea eta arriskuak kudeatzeko planak bilduko ditu.
  - Estimatutako iraupena: 27 ordu.

- Arlo honetan estimatutakoa guztira: 30 ordu.

Prestakuntza:

- *HTML5 eta JavaScript prestakuntza*: teknologia hauen inguruko ezagutzak barneratzea.
  - Estimatutako iraupena: 8 ordu.
- *Greasemonkey prestakuntza*: teknologia honen inguruko ezagutzak barneratzea, garapen prozesuan zehar aplikatu ahal izateko.
  - Estimatutako iraupena: 3 ordu.
- *JointJS eta RaphaëlJS prestakuntza*: liburutegi hauek erabili ahal izateko beharrezko kontzeptuak barneratzea.
  - Estimatutako iraupena: 5 ordu.
- *Twitter prestakuntza*: Twitter-en APIa erabili ahal izateko beharrezko kontzeptuak barneratzea.
  - Estimatutako iraupena: 7 ordu.
- Arlo honetarako estimatutakoa guztira: 23 ordu.

Diseinua:

- *Erabilpen kasuak*: tresnak izango dituen erabilpen kasuak identifikatu eta garatzea.
  - Estimatutako iraupena: 7 ordu.
- *Tresnaren egitura*: tresnak izango duen egituraren diseinua.
  - Estimatutako iraupena: 15 ordu.
- Arlo honetarako estimatutakoa guztira: 22 ordu.

Tresnaren garapena:

- *Interfaze grafikoa sortu*: tresnaren interfaze grafikoak izango dituen pantailen definizio eta sorrera.
  - Estimatutako iraupena: 15 ordu.
- *Mikroblogintza plataformak integratu*: SNS plataformen kudeaketa garatutako tresnaren bidez egiteko mekanismoen inplementazioa.
  - Estimatutako iraupena: 10 ordu.
- *Erauzketa inplementatu*: erantzunetatik baliagarriak diren datuak erauzteko mekanismoen inplementazioa.
  - Estimatutako iraupena: 10 ordu.
- *Aukeraketa inplementatu*: erauzitako datuetatik egokiena aukeratzeko mekanismoen inplementazioa.
  - Estimatutako iraupena: 10 ordu.
- *Editoreetako egokitzapena inplementatu*: tresna editoreetara moldatzeko moduaren inplementazioa.
  - Estimatutako iraupena: 20 ordu.
- Arlo honetarako estimatutakoa guztira: 65 ordu.

Probak:

- *Probetarako ingurunea prestatu*: probak burutu ahal izateko beharrezko ingurunearen prestakuntza.
  - Estimatutako iraupena: 5 ordu.
- *Probatu*: tresna nahiz tresnaren osagaiak probatu.
  - Estimatutako iraupena: 10 ordu.
- Arlo honetarako estimatutakoa guztira: 15 ordu.

Kudeaketa:

- *Proiektuaren jarraipena*: proiektuko jarraipen eta kontrola burutzea.
  - Estimatutako iraupena: 5 ordu.
- *Bilerak*: proiektuko interesatuen arteko bilerak.
  - Estimatutako iraupena: 20 ordu.
- Arlo honetarako estimatutakoa guztira: 25 ordu.

Proiektuaren dokumentazioa:

- *Memoriaren egitura prestatu*: proiektuaren memoriak izango duen egitura prestatzea.
  - Estimatutako iraupena: 3 ordu.
- *Memoria idatzi*: proiektuaren memoriaren idazketa.
  - Estimatutako iraupena: 100 ordu.
- *Memoria berrikusi*: proiektuaren memoriaren berrikusteak, akatsak eta hobetu daitekeen puntuak bilatuz.
  - Estimatutako iraupena: 7 ordu.
- Arlo honetarako estimatutakoa guztira: 110 ordu.

Defentsa:

- *Aurkezpena prestatu*: proiektua epaimahaiaren aurrean defendatzeko gardenkiak prestatzea.
  - Estimatutako iraupena: 7 ordu.
- *Entseguak egin*: defentsaren entseguak egitea, gardenkien edukia barneratzeko.
  - Estimatutako iraupena: 3 ordu.
- Arlo honetarako estimatutakoa guztira: 10 ordu.

Estimazio hauek gehituta, proiektu guztian 300 ordu sartzea estimatuta dagoela ikus daiteke.

2.2.2 Atazen antolaketa

2.3 irudiko Gantt diagraman aurreko atalean identifikatutako atazak noiz burutuko diren adierazten da.

2.3 Baliabideen esleipena

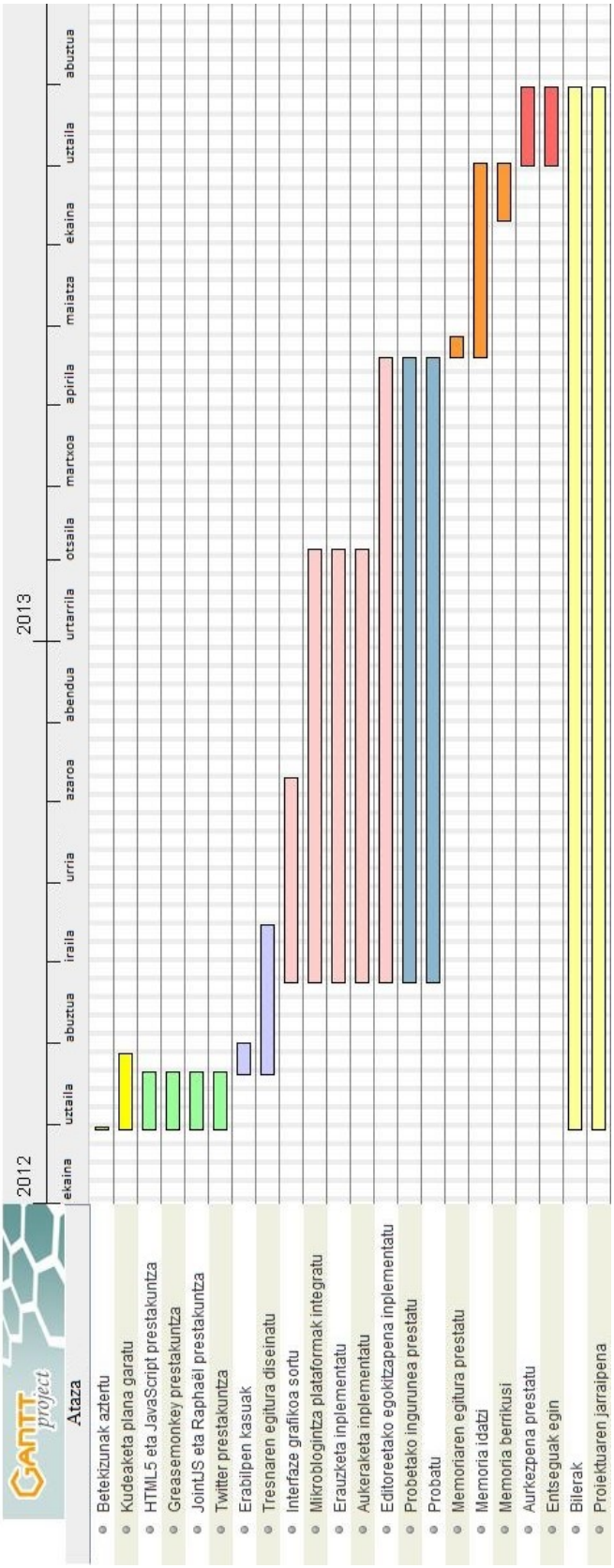
Proiektu honetarako beharrezko baliabideak minimoak izango dira, proiektuaren izaera informatikaren arlo fisikotik nahiko urrun dagoelako. Errendimendu altua eskatzen duten softwareen beharrik aurreikusten ez denez, ia edozein ordenagailurekin lan egin daitekeela esan daiteke.

Honakoa da erabiliko den hardwarea:

- Ordenagailu pertsonal eramangarria, edonon lan egiteko aukera ematen duena. *Windows Vista* eta *Ubuntu 12.04* sistema eragileak instalatuta dauzka. Ordenagailuaren erabilera eskatzen duten proiektuko ataza guztiak burutzeko erabiliko da.
- Datuak leku batetik bestera garraiatzeko eta egindakoaren segurtasun kopiak gordetzeko erabiliko den 4GB-eko USB memoria.

Software aldetik, hauek dira erabiliko diren baliabideak:

- *Windows Vista* sistema eragilea.
- *Mozilla Firefox* web nabigatzailea, tresnaren azpiegitura gisa. Nabigatzailearen plugin hauek erabiliko dira:
  - *Greasemonkey*
  - *Firebug*
  - *HttpFox*
- *Notepad++* testu editorea, tresna garatzeko.
- *Microsoft Word* eta *TexMaker* LaTeX editorea, dokumentuak sortzeko.
- *Microsoft Excel* jarraipeneko datuak biltzeko.



2.3 Irudia: Proiektuko Gantt diagrama

- *Microsoft Visio*, UML diagramak eta bestelako grafikoak sortzeko.
- *StarUML*, tresnaren diseinurako UML diagramak egiteko.
- *GanttProject*, kronogramak egiteko.
- *Windows*eko *Recortes* tresna, dokumentazioari erantsiko zaizkion *screenshoot*ak egiteko.

2.4 Komunikazio plana

Atal honetan komunikazioarekin erlazionatuta dauden gaiak aztertzen dira, proiektuko komunikazio prozesuak era eraginkorrean eman daitezen. Batetik, proiektuko interesatuak identifikatuko dira, eta bestetik, informazio banaketa nola egingo den zehaztuko da.

2.4.1 Interesatuen identifikazioa

Honakoak dira proiektu honetan aurkitu daitezkeen interesatuak:

Ikaslea	Jeremías Pérez	jperez085@ikasle.ehu.es
Tutorea	Óscar Díaz	oscar.diaz@ehu.es
Laguntzailea	Cristóbal Arellano	cristobal.arellano@ehu.es

2.1 Taula: Proiektuko interesatuak.

2.4.2 Informazioaren banaketa

Proiektuko interesatuen arteko komunikazioa 2.4.1 taularen eskuineko zutabeko posta elektronikoak erabilita egingo da.

Bestalde, interesatuak noizean behin bilduko dira proiektu zein produktuari buruzko erabakiak hartzeko. Proiektuaren iraupena luzea denez, ez dira bilera periodikoak egingo;

beharrezkoak direnean egingo dira. Horrela, bilerak ahalik eta emankorrenak izatea bilatzen da. Bileretarako deialdia posta elektronikoz egingo da, eta bileran parte hartuko duten interesatuen artean data eta ordua adostu beharko da. Gainera, bilerara bertaratuko diren interesatuek bileran jorratuko diren gaien berri izan beharko dute hau burutu aurretik.

Azkenik, Wiki bat kudeatuko da, proiektuaren egoeraren berri emateko, eta proiektuaren memoria idazteko erabilgarria izan daitekeen informazioa antolatzeko.

2.5 Kalitatearen kudeaketa plana

Proiektu guztian zehar egiten diren ataza guztietan, ahalik eta kalitate mailarik handiena bilatuko da, bai proiektuaren kudeaketan bai produktuan. Atal honetan, kalitate elementuak zerrendatuko dira, eta bakoitzarentzako lortu nahi den neurria eta minimo onargarria finkatzeaz gain, neurtzeko modua ere azalduko da. Jarraian datozen azpiataletan identifikatutako kalitate elementu hauek azalduko dira, arloka sailkatuta.

2.5.1 Proiektuaren kudeaketa

- **Kalitate elementua**: bilera kopurua.
  Azalpena: proiektuko interesatuen arteko komunikazioan gabeziak ez egoteko, bilera kopuru minimo bat ezarriko da.
  Neurtzeko modua: egindako bilera kopurua neurtuta.
  Minimo onargarria: gutxienez 10 bilera.
  Lortu nahi den neurria: 15-25 bilera.

2.5.2 Garatutako produktua

- **Kalitate elementua**: tresna editoreetan integratzea.
  Azalpena: garatutako tresna editore desberdinetan integratzeko gai izan behar da, beraz, zenbat eta editore gehiagotan integratu, tresnaren akoplamendu gaitasuna orduan eta handiagoa izango da.
  Neurtzeko modua: garatutako tresna erabili ahal izateko egokitutako editore kopurua.
  Minimo onargarria: 2.
  Lortu nahi den neurria: 2 baino gehiago.

- **Kalitate elementua**: mikroblogintza plataformak tresnan integratzea.
  Azalpena: garatutako tresnak mikroblogintza plataformetako mezuak kudeatuko ditu. Beraz, zenbat eta gehiago kudeatzeko aukera eman, produktuaren erabilgarritasuna orduan eta handiagoa izango da.
  Neurtzeko modua: tresnan arrakastaz integratutako mikroblogintza plataforma kopurua.
  Minimo onargarria: 1.
  Lortu nahi den neurria: 1 baino gehiago.

2.5.3 Proiektuaren memoria

- **Kalitate elementua**: dokumentuaren luzera minimoa.
  Azalpena: dokumentu baten luzera kalitatezkoa izatearen sinonimo ez izan arren, ez da komenigarria memoria oso laburrak idaztea.
  Neurtzeko modua: dokumentuaren orrialde kopurua neurtuta.
  Minimo onargarria: gutxienez 70 orrialde.
  Lortu nahi den neurria: gutxienez 90-120 orrialde.

- **Kalitate elementua**: kapituluen atal kopuru minimoa.
  Azalpena: dokumentuak egitura ahalik eta sendoena izateko, kapituluek izan beharreko atal kopuru minimoa ezarriko da. Horrela, azpiatal gutxiko kapitulu asko sortzea saihesten da, eta dokumentuaren antolaketa hobetzea eragiten du.
  Neurtzeko modua: kapitulu guztien atal kopurua neurtuta.
  Minimo onargarria: gutxienez 3 atal kapituluko.
  Lortu nahi den neurria: 4-6 atal kapituluko.

2.5.4 Proiektuaren defentsarako gardenkiak

- **Kalitate elementua**: gardenki bakoitzeko ideia nagusi kopurua.
  Azalpena: gardenkiek testu asko edukitzeak entzuleen arreta galtzea eragin dezake. Beraz, komenigarria da gardenkiak ez gainkargatzea.
  Neurtzeko modua: gardenki bakoitzak duen ideia nagusien kopurua neurtuta.
  Minimo onargarria: gehienez 3 ideia nagusi gardenkiko.
  Lortu nahi den neurria: ideia nagusi bakarra gardenkiko.

2.6 Arriskuen kudeaketa plana

Atal honetan, proiektua arrakastaz gauzatzea arriskuan jar dezaketen elementuak identifikatu eta arrisku-mailaren arabera ordenatzen dira. Arrisku bakoitza zertan datzan azaltzeaz gain, proiektuan izango lukeen eragina, gertatzeko probabilitatea eta ekiditeko estrategia edo behin gertatuta honen eragina gutxitzeko estrategia zehaztuko dira.

- **Arriskua**: informazioaren galera.
  Azalpena: informatikaren sorreratik presentzia handiko arriskua izan da hau. Informazioa galtzeko modu asko dago: makinen arazo teknikoak, hondamendi naturalak, malware-a, erabiltzaileen hanka-sartzeak, etab.
  Eragina: oso handia. Informazio-sistema kaskar baten ondorioz proiektuko informazioa galtzeak konponbiderik gabeko kalteak eragingo lituzke, eta galdutakoa berriro errepikatu beharko litzateke.
  Gertatzeko probabilitatea: txikia. Ikaragarriko eragina izan arren, gaur egun ez da oso ohikoa.
  Ekiditeko estrategia: arrisku hau ekiditeko proiektuan informazio-sistema seguru bat erabiliko da. Horrela, informazioaren segurtasun kopia periodikoak egingo dira, eta leku desberdinetan biltegiratuko dira: ordenagailu eramangarrian, USB memoria batean eta Interneteko hodeian, *Dropbox* aplikazioa erabilita.

- **Arriskua**: ordenagailua hondatzea.
  Azalpena: ordenagailuek ez dute betirako irauten, beti funtzionatzeari uzteko aukera dago. Asko dira ordenagailu baten osagaiak (prozesadorea, txartel nagusia...) eta hauetako bat funtzionatzeari uztea nahikoa da makina osoa erabilgaitza geratzeko.
  Eragina: handia. Proiektuan ordenagailu bakarra erabiltzea pentsatuta dagoenez, hau hondatuz gero lan egiteko erreminta galduko da. Gainera, informazio-sistemaren kudeaketa txarra egiten bada (ikus *informazioaren galera* arriskua), proiektuko informazioa galtzeko arriskua dago.
  Gertatzeko probabilitatea: altua. Ordenagailu bat noiz hondatuko den aurreikustea oso zaila da, baina ordenagailu zaharrek hondatzeko aukera gehiago dute. Proiektuan erabiliko den ordenagailua 2007an erosi zen, eta gainera, bateriak iraganean arazo asko eman ditu. Beraz, ordenagailua hondatzeko probabilitatea altua dela esan daiteke.
  Eragina gutxitzeko estrategia: proiektua garatzeko Onekin ikerketa taldeak eskainitako ordenagailu bat erabiltzeko aukera dago. Beraz, proiektuaren garapenerako

erabiliko den ordenagailuak huts eginez gero, fakultateko edo liburutegiko ordenagailuak eskura izateaz gain, Onekin taldeak utzitako bat ere erabili ahal izango da.

- **Arriskua**: garatutako tresnak erabiltzen duen zerbitzuren batek erabiltzeari uztea.
  Azalpena: kanpo-zerbitzuak erabiltzeak beti arriskua izan dezake, hauen funtzionamendu egokia gure esku ez dagoelako. Edozein arrazoia dela medio zerbitzua erori edo aldatu egiten bada, zerbitzu hori erabiltzen duten aplikazioen erabilgarritasuna murriztuta geratuko da.
  Eragina: ertaina. Aplikazio bat funtzionatu ahal izateko zerbitzu baten menpe badago, honek huts eginez gero aplikazioa guztiz aldatu beharko da.
  Gertatzeko probabilitatea: baxua. Orokorrean zerbitzuak eskaintzen dituzten konpainiak nahiko fidagarriak izaten dira, eta ez dute hutsegiterik izaten. Bestalde, nahiko egonkorrak izaten dira, eta zerbitzuen bertsioek denbora asko irauten dute.
  Ekiditeko estrategia: garatutako tresna ez da zerbitzu baten menpe egongo, eta zerbitzu bat erabiltzea beharrezkoa den kasuetan, ahal den heinean hornitzaile desberdinek eskainitakoak erabiltzeko prest egon beharko da. Bestalde, bakarrik konpainia fidagarrietako zerbitzuak erabiliko dira, ezustekoak ekiditeko.

- **Arriskua**: denbora-estimazioak okerrak izatea.
  Azalpena: baliteke planifikazioa egiterakoan egindako denbora estimazioak okerrak izatea, eta estimatutakoaren eta benetan burututakoaren arteko aldea zabala izatea.
  Eragina: aldakorra. Benetan dedikatutako ordu kopurua estimatutakoa baino askoz altuagoa bada, proiektuan atzerapenak sor daitezke, eta emangarrien kalitatea murriztuta ikus daiteke. Bestalde, benetan dedikatutako ordu kopurua estimatutakoa baino txikiagoa bada, kalitatea hobetzeko erabil daiteke.
  Gertatzeko probabilitatea: ertaina. Proiektu honen iraupena luzea denez, zaila da denbora estimazio egokiak egitea, eta ondorioz, huts egiteko aukera gehiago dago.
  Eragina gutxitzeko estrategia: jarraipen eta kontrol datuetan alde handia dagoela detektatzen bada, birplanifikazio bat egin daiteke lan-karga nolabait arintzeko edo bestelako lan-lerroak planteatzeko.

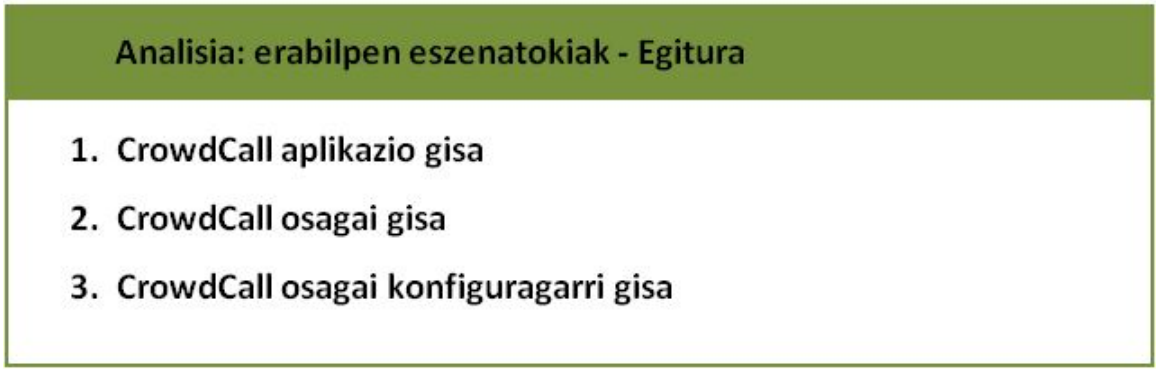
- **Arriskua**: proiektuaren defentsaren atzerapena.
  Azalpena: arrazoi akademikoak direla eta proiektuaren defentsa aurreikusitako datan egitea ezinezkoa izatea.

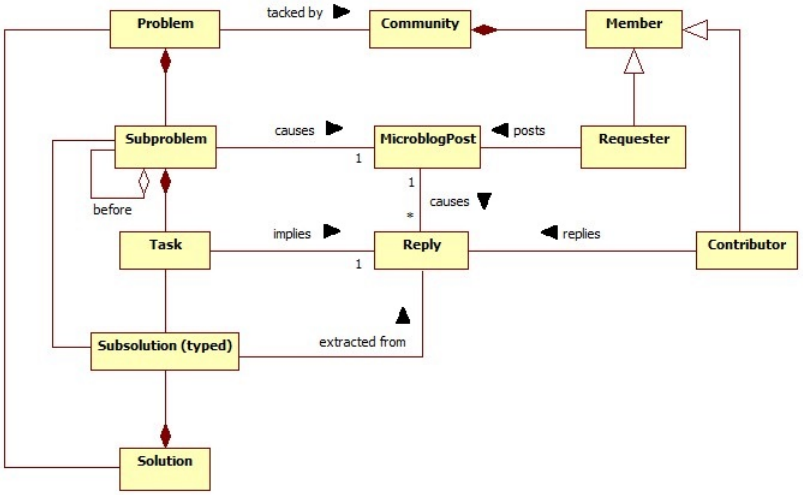
Eragina: txikia. Gertaera honek ez luke proiektuan eragin asko izan beharko, hau gertatzerakoan proiektua amaitzear egongo litzatekeelako.
Gertatzeko probabilitatea: aldakorra. Ezin da inoiz aurreikusi zer gertatuko den irakasgai bakoitzean.
Eragina gutxitzeko estrategia: birplanifikazio bat egin beharko litzateke, ez baita komenigarria defentsa prestatzen denetik egiten den arte hainbeste denbora igarotzea.

3. KAPITULUA

Analisia: erabilpen eszenatokiak

Proiektuan CrowdCall izeneko tresna garatu da. Kapitulu honetan garatutako tresna horren analisi kontzeptuala azaltzen da: aplikazio independente bat izatetik, DSL editoreek erabil dezaketen osagai konfiguragarri bat izatera iritsi arte. Horretarako egoera bakoitzean adibideak txertatuko dira, irakurleak azalpenaren haria ez galtzeko.





3.1 Irudia: CrowdCall-en eredu kontzeptuala

3.1 Crowdcall aplikazio gisa

Atal honetan garatutako tresna aplikazio independentea (bestelako editoreen beharrik ez duena) balitz bezala tratatzen da, erabiltzaileak honen funtzionamendua errazago ulertu dezan. Horretarako, adibide sinple bat planteatzen da.

3.1.1 Eredu kontzeptuala

CrowdCall mezu jarioak kudeatzeko tresna da. 3.1 irudian hurrengo paragrafoetan azalduko den eredu kontzeptualaren irudikapen grafiko bat ikus daiteke. Berrikuntza *Problem* kontzeptuaren sarreran datza. *Problem* bat helburu komun bati erantzuten dioten mezu multzo bat da. Multzokatze kontzeptu hau ez dago SNS tresnetan, hauetan mezuak elementu independente gisa ikusten baitira, haien arteko erlazio kausalik gabe.

Arazo edo *Problem* bat azpiarazo txikiagotan deskonposa daiteke, eta *Subproblem* multzo gisa ikus dezakegu. Azpiarazo hauek menpekotasun erlazioak izan ditzakete haien artean: azpiarazo jakin bat ebazten hasi aurretik beste baten ebazpena ezagutzeko beharra.

Problem bati SNS aplikazio batean ekin egiten zaio, eredu kontzeptualean *Community* izena hartzen duena. *Community*ak erabiltzaile edo *Member*ez osatuta daude. Hauek bi paper joka ditzakete arazoaren ebazpenean: *Requester* edo laguntza eskatzailea, eta *Contributor* edo kolaboratzailea.

Arazoa osatzen duen azpiarazo bakoitzari *Requester* batek SNS aplikazioan idatzitako mezu edo *MicroblogPost* bat dagokio, azpiarazo bakoitzeko laguntza eskaera gisa. Bestalde, kolaboratzaileek eskaera honi erantzunez *Reply*ak idatziko dituzte. Hauetako bakoitzak ataza edo *Task* bat suposatuko du.

Subproblem bakoitzaren *MicroblogPost*ak eragindako *Reply*etatik *Subproblem*arentzako azpisoluzio edo *Subsolution*a erauziko dugu, azpisoluzioak aldez aurretik finkatutako mota batekoak izango direlako. *Problem*a osatzen duten *Subproblem*en *Subsolution*ak elkartuta, erroan dagoen arazoarentzako emaitza edo *Solution*a lortzen da.

3.1.2 NASDAQ adibidea

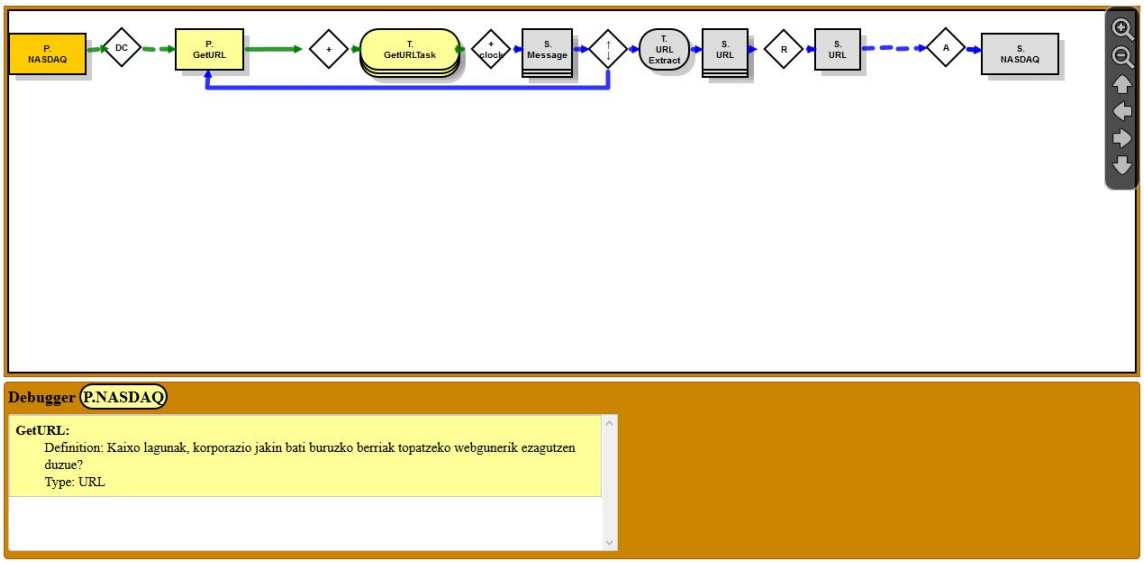
Garatutako tresnak GUI bat eskaintzen du Problem baten sorrera, eguneraketa eta ebazpen prozesuetan laguntzeko. Jarraian datorren adibidea irakurleak 3.1.1 atalean azaltzen den eredu kontzeptuala barneratzeko helburua du, eta bidenabar, CrowdCall-en funtzionamendu eta ohitzeko erabiliko da.

Demagun erabiltzaile batek NASDAQ konpainiei buruzko berrien inguruko webgune bat bilatzen ari dela, eta CrowdCall tresna erabili duela Twitter-en dituen jarraitzaileei laguntza eskatzeko. Horretarako, eskaera hau helarazi egingo die bere jarraitzaileei: “Kaixo lagunak, korporazio jakin bati buruzko berriak topatzeko webgunerik ezagutzen duzue?”. Kasu honetan sortuko litzatekeen arazo edo *Problem*a azpiarazo edo *Subproblem* bakarraz osatuta egongo litzateke: webgunearen bilaketa. Bestalde, Twitter izango litzateke adibide honetako SNS aplikazio edo *Community*a, eta laguntza eskatzen duen erabiltzailea edo *Requester*a, aldiz, adibide honen protagonista.

CrowdCall tresna martxan jartzerakoan, erabiltzaileak 3.2 irudian ikus daitekeen interfaze grafikoa ikus ahal izango du. Honen egitura oso sinplea da. Goiko zatian, arazoaren ebazpen prozesuan erabiltzailea gidatzeko grafo bat dago. Grafo honek, aldi berean, arazoa irudikatzen du: honen erroan dagoen nodoak *Problem*a izango litzateke, eta honen semeak (kasu honetan bakarra) *Subproblem*ak izango lirateke.

Erabiltzailea Twitter-en sartzen bada, tresnak laguntza eskaera idatzi duela ikusi ahal izango du. Idatzitako eskaera hau eredu kontzeptualean *Subproblem*ari dagokion *MicroblogPost*a izango litzateke.

Goiko zatiko grafoak zehazten duen prozesua jarraituta, erabiltzailea eskaera eta honi egindako erantzunak kudeatzeko pantaila batera iritsiko da, 3.3 irudian azaltzen den moduan. Pantaila honetan laguntza eskaera egin duen erabiltzailearen jarraitzaileek la-



3.2 Irudia: NASDAQ adibideko erabiltzaileak CrowdCall martxan jartzerakoan ikusiko lukeen pantaila.

3.3 Irudia: Laguntza eskaera eta mezuak kudeatzeko pantaila.

3.4 Irudia: Kolaboratzaileek proposatutako azpisoluzioak erakusten dituen pantaila.

3.5 Irudia: Azpiarazoarentzako lortutako azpisoluzioa.

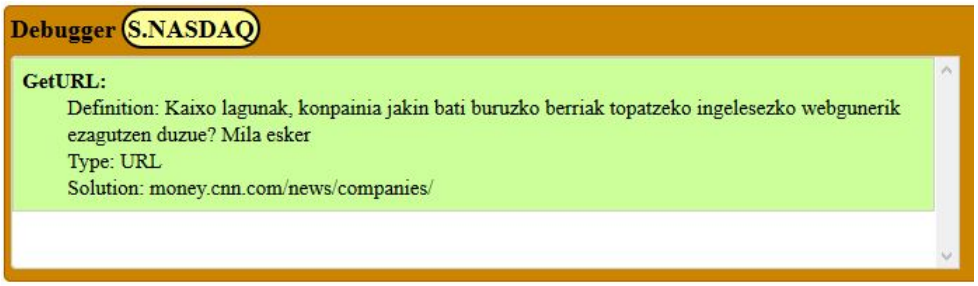
guntza eskaerari egindako erantzunak azaltzen dira. Eredu kontzeptualari berriz helduta, Jon, Mikel eta Igor, erantzunak idatzi dituzten erabiltzaileak, adibide honetako *Contributor*ak izango lirateke. Erabiltzaile hauek, laguntza eskaera egiten duen erabiltzailea bezala *Community*ko *Member*ak dira. Bestalde, azken hauek idatzitako mezuak *Reply*ak izango litzateke.

Laguntza eskaera egiterakoan, erabiltzaileak argi utzi du URLak bilatzen ari zela, eta horri esker CrowdCall-ek jasotako erantzunetatik baliagarriak diren datuak erauzi ahal izango ditu. Horrela, mota bereko azpisoluzio edo *Subsolution* sorta lortuko da. Eredu kontzeptualean *Subsolution*ak *Typed* edo mota finkokoak direla esaterakoan hori adierazi nahi da. 3.4 irudiko irudiak adibideko erantzunetatik erauzitako azpisoluzioak erakusten ditu.

Behin erantzunetatik baliagarriak diren datuak erauzi eta gero, aukera guztiak bakarrera laburtu beharko dira. Erabiltzaileak eskatutako datuak URL motakoak direnez, azpisoluziorik hoberena zein den jakiteko Alexak (http://www.alexa.com) web orriak jasotzen duten trafikoaren arabera egiten duen sailkapenean oinarritu gaitezke. Horrela, Alexa-ren web orritik datuak eskuratuta, CrowdCall-ek erauzitako azpisoluzioen artean denetatik hoberena lortuko du.

Modu honetan, tresnak *Subproblem*arentzako *Subsolution*a lortuko du. 3.5 irudian azpisoluzio hoberena aukeratu eta gero tresnaren interfaze grafikoak erabiltzaileari erakutsiko liokeena.

Adibide honetako *Problem*a *Subproblem* bakarraz osatuta dagoenez, arazoaren *So-*



3.6 Irudia: Arazoarentzako soluzioa erakusten duen pantaila.

*lution*a azpiarazo bakar honen *Subsolution*ak bakarrik osatzen du. 3.6 irudiak arazoaren soluzioa osatzen duten azpisoluzioak erakusten ditu, baina, *Subproblem* bakarra dagoenez, *Subsolution*a bakarra da.

3.2 CrowdCall osagai gisa

Arazoak ez dira ezerezetik sortzen, baizik eta testuinguru zehatz batean sortzen dira. Bereziki, software aplikazioaren garapenean sortzen diren arazoei ekingo diogu. Gehiago zehaztuta, DSLen erabileran sortzen direnak (adibidez, SQL, kalkulu orrietako formulak, eta abar).

Hemen *Problem* bat sortzen da kodearen espezifikazioa osatzeko beharrezkoa den daturen bat ezagutzen ez denean. Adibidez, erabiltzaile batek ez ditu baimenak ezagutzen edo ez daki zein tala erabili behar duen SQL kontsulta baten zati bat osatzeko. Kasu honetan, arazoa SQL editorearen testuinguruan sortzen da (adibidez, Oracle-ren SQL Worksheet).

Beraz, oraingo helburua CrowdCall-en funtzionalitatearen zati bat API bezala eskaintzea da. Hau da, beste aplikazio batzuk (editoreak, esate baterako) funtzionaltasun hau erabili ahal izango dute CrowdCall-en APIari egindako deien bidez.

APIan ondorengo metodoak definitu dira:

- getProblems
- deleteProblem
- getProblemsSolution
- createProblem

- configureCrowdCall

D eranskinean API honetan definitutako metodoak sakonean azalduko dira.

Definitutako API honen bidez, CrowdCall aplikazio bat izatetik beste aplikazioek erabil dezaketen osagaia izatera igaroko da. Horrela, kanpo editore batek CrowdCall tresna erabili ahal izango du, esate baterako, Sticklet editoreak. Editore hau 3.2.1 azpiatalean deskribatuko da, 3.2.2 azpiataleko adibidea ulertu ahal izateko.

3.2.1 Sticklet

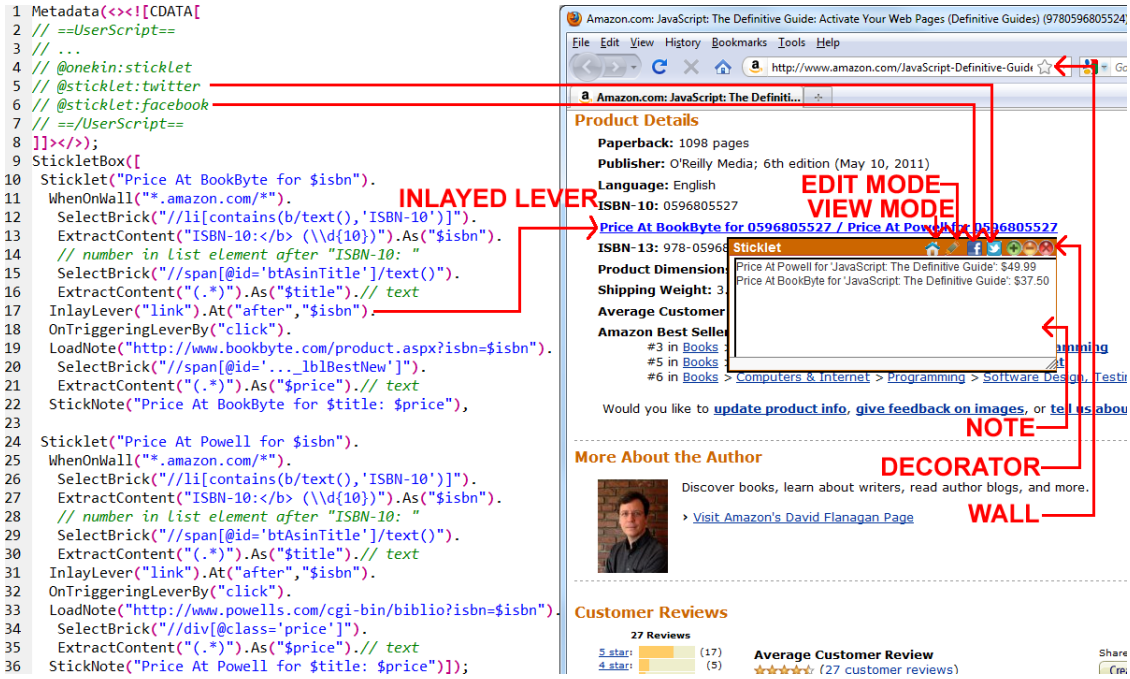
Sticklet weba areagotzeko DSL tresna da (www.sticklet.org). Horretarako, erabiltzaileak webean nabigatzeko esperientzia modu erraz eta seguru batean pertsonalizatzeko aukera ematen du. Beste hitz batzuetan, web orri jakin baten barruan nabigatzen den bitartean, hau areagotu daiteke beste web orri edo web zerbitzu batetik lortutako informazioa erabilita.

Greasemonkey-k web orriak pertsonalizatzeko aukera ematen du, JavaScript kode zati txikiak (scriptak) erabilita. Script hauetako ehunka adibide userscripts.org webgunean aurki daitezke, webgune askotarako egindakoak. Hala ere, script hauen sorrera oso konplexua izan daiteke erabiltzaile arruntarentzat, eta Greasemonkey erabiltzeak sor ditzakeen segurtasun arazo batzuen ondorioz, hauen zabalketa oztopatzen da.

Sticklet-ek arazo honi aurre egiten dio domeinu bereziko lengoaia (DSL) bat definitu eta erabilita. Erabiltzaile arruntak ez du gehiago JavaScript lengoaiaz script bat idazten, baizik eta *StickletBox*ak. Modu honetan eginez, erabiltzaile arruntak adierazkortasuna galtzen du, baina erraztasuna eta segurtasuna hobetzen dira.

Sticklet-en oinarria weba erabiltzaileak definitutako eranskailuekin apaindu daitekeen hormatzat hartzea da. Horma-eranskailu bikoteak areagotze-unitate bat osatzen du. Aurrerantzean sticklet hitza erabiliko da unitate bakoitza izendatzeko.

Demagun Amazon-en liburu bat kontsultatzen ari garen bitartean, liburu horrek beste liburutegi batean duen salneurria ikusi ahal izateko eranskailu bat txertatu nahi dugula. Helburu honetarako Greasemonkey erabiliko bagenu, 300 kode lerro inguruko JavaScript lengoaiaz idatzitako script bat beharko genuke. 3.7 irudian Sticklet tresnaren bidez gauza bera egiteko beharko genukeen kode zatia ikus daiteke. DSLaren egitura horma (*walls*), adreilu (*bricks*), nota (*notes*) eta palankaz (*levers*) osatuta dago.



3.7 Irudia: Sticklet tresnarekin egindako web areagotzeko adibidea, horri dagokion kodearekin batera.

Hormak (11. lerroa) Web orrien gaineko bistak balira bezala ikus daitezke. Horma batek adierazpen erregular batekin bat egiten duten URLak dituzten web orriak biltzen ditu (*WhenOnWall* klausula). Sticklet baten hormak, adreiluen existentziarekin batera, stickletaren irismena definitzen du. Esate baterako, aipatutako adibidean, horma hedatzen da ISBN adreilu bat duten Amazon-eko orri guztietara.

Adreiluak (12-16 eta 20-21 lerroak) Izen bat jasotzen duten HTML dokumentuetako nodoak dira, eta datuen erauzketarako, bereizketarako, irismena finkatzeko eta geruzak sortzeko balio dute. Adreilu batek honako osagaiak ditu:

- XPath adierazpen bat, nodoa zehazteko (*SelectBrick* klausula)
- Adierazpen erregular bat, nodotik edukia erauzteko (*ExtractContent* klausula)
- Adreiluaren izena (*As* klausula).

Notak (22. lerroa) Testua eta adreiluak batzen dituzten adierazpenak dira (*StickNote* klausula). Adreiluak hormetatik nahiz URL bidez atzigarriak diren zerbitzuetatik lor daitezke (*LoadNote* klausula, 19. lerroa). Aipatu adibidean, BookByte-ri eskaera bat

egiten zaio, non URLaren parametroak aurretik erauzitako adreiluetatik lortu baitira (adibidez, $isbn). Eskaeraren erantzuna adreilu berri bat lortzeko erabiltzen da: $price. Azkenik, jatorri desberdinetako adreiluak nota osatzeko erabiltzen dira (*StickNote* klausula). Apaingarri batek notak inguratzen ditu, eta lekuz aldatu, handitu, txikitu eta ixteko aukera ematen dute. Notek atzigarri egon beharko lukete erabiltzailea horman sartu bezain laster. Hala ere, hau ez da beti nahi den portaera. Adibidez, Amazon-en sar zaitezke liburu bat erosteko asmorik izan gabe. Horren ondorioz, notak zuzenean pantailaratzeak eraginkortasuna kaltetu dezake, benetan behar ez den informazioa ikustera behartuz. Horregatik, nota bat itsasteko erabiltzailearen parte-hartzea beharko da: palanka bati eragitea.

Palankak (17. lerroa) Notak erabiltzaileak nahi dituenean itsasteko aukera ematen dute. Palanken posizioa adreiluen arabera egiten da. Gure adibidean, palanka bat (esteka bezala egikaritua) $isbn adreiluaren ostean txertatzen da. Palanka bat adreilu baten aurretik itsatsi, edo zuzenean adreilu bat ordezkatzeko aukera ematen da. Palanka bati eragiterakoan, URL bidez atzigarria den zerbitzu bat aktibatzen da. Aztertutako adibidean, palankaren gertaera klik bat da, edozein DOM gertaera onartzen den arren. Klik egiterakoan, BookByte webguneari eskaera bat egiten zaio; ondoren, salneurria erauzten da, eta azkenik, nota kargatzen da.

3.2.2 CrowdCall-en erabilera-adibidea Sticklet testuinguruan

3.2.1 azpiatalean Sticklet zer den eta Sticklet bat eratzeko kodearen egitura laburki azaldu ondoren, adibide bat jarriko dugu. Adibidean, Sticklet baten kodea osatzeko CrowdCall tresna erabiliko da. Demagun burtsa-agente bati Sticklet tresnaren erabilera gomendatu diotela, kontsulta-lana asko errazten baitu. Baina Sticklet bat osatzea erronka handia izan daiteke, XPath adierazpenak, adierazpen erregularrak eta informazio-iturri egokiak ezagutzea eskatzen baitu. Hori dela eta, CrowdCall erabili ahal izateko Sticklet-ek duen hedapena erabiltzea erabakitzen du, Twitter-en dituen jarraitzaileei laguntza eskatzeko asmoz.

3.8 irudian erabiltzaileak sortutako stickleta ikus daiteke. Irudi honetan, “crowd:” adierazpenaren agerpenak islatzen duenez, *SelectBrick* (14-16 eta 26. lerroak) *ExtractContent* (17-19 eta 27. lerroak) eta *LoadNote* (23. lerroa) klausulak osatzeko laguntza eskatzen da. Sticklet-ek CrowdCall-ekin batera lan egiteko daukan hedapenak “crowd:” adierazpen hauek detektatzen dituenean CrowdCall-en eskuetan utziko du kodea osatzeko ardura.

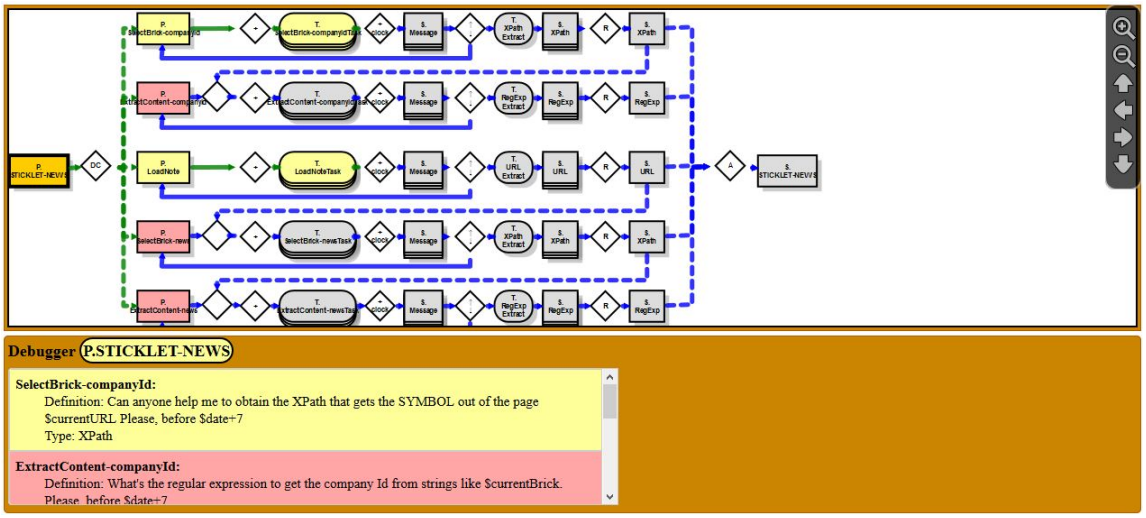
```
1  Metadata(<><![CDATA[
2  // ==UserScript==
3  // @name           Stock mashup
4  // ...
5  // @sticklet.crowd
6  // @sticklet.crowd:social    Twitter
7  // @sticklet.crowd:mediation content
8  // @sticklet.crowd:retention ownership
9  // ==/UserScript==
10 ]]></>)
11 StickletBox([
12  Sticklet("news").
13  WhenOnWall("*.nasdaq.com/quotes/*").
14   SelectBrick("crowd:Can anyone help me to
15  obtain the XPath that gets the SYMBOL out of
16  the page $currentURL. Please, before $date+7").
17   ExtractContent("crowd:What's the regular
18  expression to get the company Id from strings
19  like $currentBrick. Please, before $date+7")
20   .As("$companyId").
21  InlayLever("button").At("after","$companyId").
22  OnTriggeringLeverBy("click").
23  LoadNote("crowd:Dear fellows, any website for
24  news of a given corporation.
25  Please, before $date+7").
26   SelectBrick("crowd:?").
27   ExtractContent("crowd:?").As("$news").
28  StickNote("$news")
29 ]);
```

3.8 Irudia: Sticklet-en barruan CrowdCall tresna erabiltzeko kode adibidea.

Kodean zehar aldagai laguntzaile batzuk ikus daitezke, laguntza eskatzeko orduan erabilgarriak izango direnak. Aldagai hauek $ karaktereaz hasten dira, eta honako hauek dira: $currentURL (16. lerroan; aldagai honek sticklet-a deitzen duen web orriaren URLa adierazten du), $currentBrick (19. lerroan; *SelectBrick* klausularen emaitza adierazten du) eta $date+7 (16., 19. eta 25. lerroetan; stickleta sortu den egunetik hasita 7 egun barru dagoen data adierazten du). Laguntza eskaerak Twitter-en idazterakoan, aldagai hauek ordezkatu egingo dira. Bestalde, 26. eta 27. lerroetan “crowd:?” adierazpena ikus daiteke. Adierazpen hau duten klausulak (*SelectBrick* eta *ExtractContent*) ebatzi ahal izateko, aurrena 23. lerroko *LoadNote* klausularen ebazpena behar da, ezinezkoa baita web orri batetik informazioa erauzteko XPath adierazpena eta adierazpen erregularrak eskatzea, web orriaren URLa aurretik jakin gabe. Orduan, galdera ikurraren bidez *LoadNote* klausularen ebazpena lortzen denean bi laguntza eskaera hauek definituko direla adierazten da.

CrowdCall tresna martxan jartzerakoan, 3.9 pantaila azalduko zaio erabiltzaileari. Pantailako grafoan antzeman daitekeenez, arazo bat sortu da, bost azpiarazoz osatua, hauek izanik azpiarazo hauen identifikatzaileak:

- *SelectBrick-companyId*



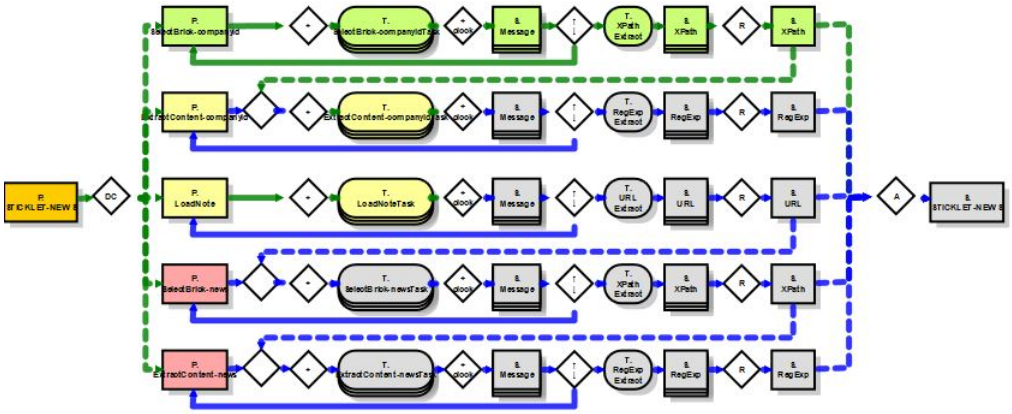
3.9 Irudia: CrowdCall erabiltzen sticklet bat sortzeko. Hasierako egoera.

- *ExtractContent-companyId*
- *LoadNote*
- *SelectBrick-news*
- *ExtractContent-news*

3.1.2 azpiataleko adibidean azpiarazo bakarra zegoen eta, ondorioz, ez zegoen azpiarazoen arteko menpekotasunik. Oraingo honetan, aldiz, 5 azpiarazo desberdin dauzkagu, eta bakarrik bi (grafoan horiz margotutako nodoak dauzkatenak) besteengandik independenteak dira. Beste hiruak (grafoan gorriz azaltzen direnak) bi azpiarazo independente hauetako baten menpe daude, eta ondorioz, hauek ebazten hasi aurretik, hauek ebatzi beharko dira. Menpekotasun hau grafoan islatzen da, azpiarazo bati dagokion azpizuhaitzaren amaierako nodoa eta beste azpiarazo bati dagokion azpizuhaitzaren hasieran dagoen operadorea lotzen dituen gezi etenaren bidez. Geziaren noranzkoak menpekotasun erlazioa adierazten du, geziaren helburuan dagoen azpiarazoa geziaren jatorrian dagoenaren menpekoa izanik.

Horrela, erabiltzaileak *SelectBrick-companyId* azpiarazoa ebazten duenean, tresnak *ExtractContent-companyId* azpiarazoari dagokion laguntza eskaera Twitter-en idatziko du, eta erabiltzailea ebazten hasi ahal izango da. 3.10 irudiak grafoak egoera honetan izango lukeen itxura adierazten du.

Antzeko zerbait gertatuko da *LoadNote* azpiarazoa ebazterakoan, baina kasu honetan, *SelectBrick-news* azpiarazoari dagokion laguntza eskaera galdera ikur batez osatuta

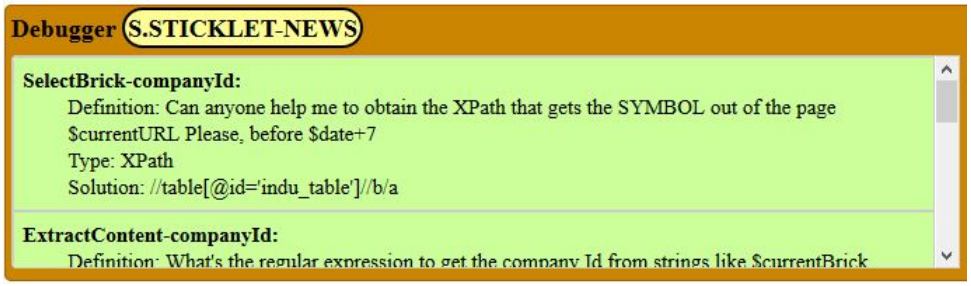


3.10 Irudia: CrowdCall erabiltzen sticklet bat sortzeko. *ExtractContent-companyId* azpiarazoaren menpekotasuna islatzen duen grafoa.

3.11 Irudia: CrowdCall erabiltzen sticklet bat sortzeko. *SelectBrick-news* azpiarazoa berriz definitzen.

zegoenez (*LoadNote* azpiarazoaren ebazpenaren menpe zegoelako), 3.11 irudian ikus daitekeen moduan erabiltzaileari azpiarazoa berriz definitzea eskatuko zaio. Hau egiterakoan, erabiltzaileak arazoaren ebazpen prozesuan aurrera jarraitu ahal izango du.

Azkenik, azpiarazo guztiak ebatzi ondoren, grafoa erabat osatuta azalduko da, eta azkeneko emaitza-nodoan klik eginez, 3.12 irudian ikus daitekeen moduan, definitutako azpiarazo bakoitzarentzako azpisoluzioa ikusi ahal izango da. Horrela, erabiltzaileak sortu nahi zuen stickleta osatuko du.



3.12 Irudia: CrowdCall erabiltzen sticklet bat sortzeko. Amaierako egoera.

3.3 CrowdCall osagai konfiguragarri gisa

Arazo baten ebazpena domeinuaren menpekoa da kasu askotan. Azpiarazoen mota, ebazpenerako estrategiak edo soluzioen identifikazioa domeinuaren izaeraren araberakoak izan daitezke. Horrenbestez, CrowdCall-ek konfigurazio mekanismo bat eskaintzen du, erabili nahi den domeinura egokitu ahal izateko. Konfigurazio hau editoreak egiten duenez, funtzionaltasun hau API gisa eskaintzen da.

Honakoak dira konfigura daitezkeen elementuak:

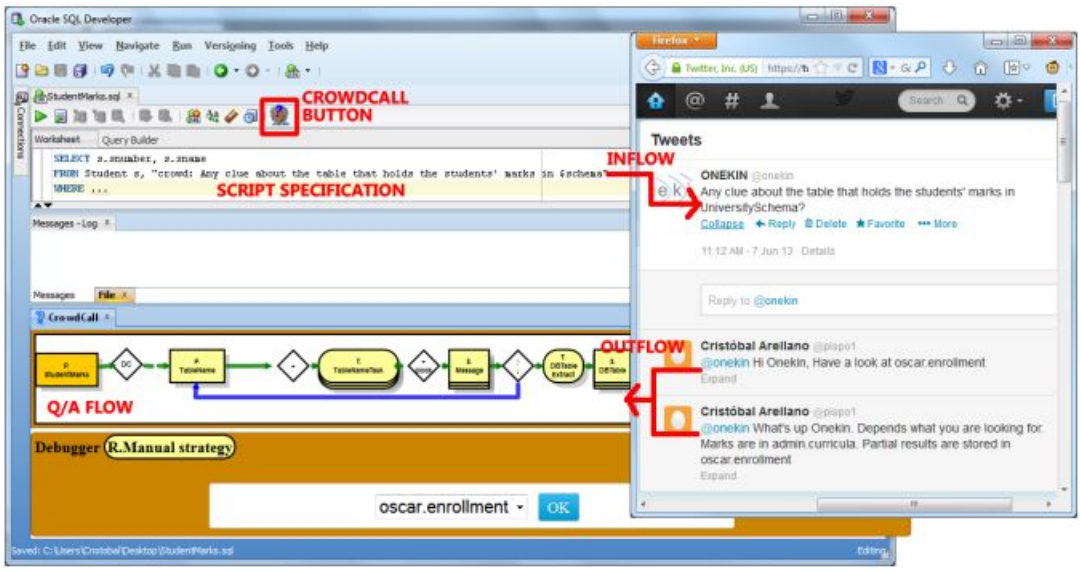
- *DSLexpression2Problem*: domeinuaren adierazpenetik abiatuta arazoaren sorkuntzarako modulua.
- *subproblem2Context*: azpiarazoak testuinguruan jartzeko modulu.
- *subproblem2MicroblogPost*: SNSak kudeatzeko modulua.
- *reply2Subsolution*: erantzunetan azpisoluzioak identifikatzeko modulua, datu motetan oinarrituta.
- *subsolutions2Subsolution*: azpisoluzioak bakarrera laburtzeko modulua.

Elementu guzti hauen konfigurazioa CrowdCall tresna deitu baino lehen egin behar da, ConfigureCrowdCall() funtzioari deituta. Funtzio hau tresna martxan jartzen den bakoitzean exekutatu beharko da. Funtzio hau parametrizatzeko modua D eranskinean sakonean aztertuko da.

Jarraian, hiru kasu praktikoen bidez, bakoitza kanpo-editore batean oinarrituta, konfigurazioa egiteko mekanismo baten beharra azalduko da. Lehenik, Oracle-ren SQL Developer editorean oinarritutako adibide bat azalduko da, SQL kontsulta bat osatu ahal izateko. Ondoren, Google-ren kalkulu orrien editorearen gainean egindako adibide bat, kalkulu batzuk egin ahal izateko formula eskuratzeko. Azkeneko adibidea 3.2.1 atalean aurkeztutako tresnan oinarrituko da, weba areagotzeko *sticklet* bat osatu ahal izateko.

3.3.1 SQL adibidea

Hona hemen Oracle-ren SQL Developer editorean oinarritutako adibide bat, CrowdCall tresnaren konfigurazioaren motibazioa azalduko da. Kasu honetan, beraz, SQL izango da adibide honetan landuko den DSLa. 3.13 irudian adibide honen irudikapen bat ikusten da.



3.13 Irudia: CrowdCall tresna *Oracle-ren SQL Developer* editorearekin bateratuta lan egiten.

Demagun erakunde batean denbora gutxi daraman langile batek datu jakin batzuk lortzeko beharrezko taulak aurkitzeko arazoak dituela. Horren ondorioz, SELECT FROM klausularen osatu ahal izateko laguntza eskatzea erabakitzen du. Horretarako, Oracle-ren SQL Developer editorean, kontsultaren FROM klausulan honako adierazpena txertatzen du: “crowd: Any clue about the table that holds the students’ marks in $schema?”. CrowdCall-ekin bateragarria izateko editoreari egindako hedapenak “crowd:” klausula detektatzerakoan CrowdCall-en eskuetan utziko du adierazpena tratatzeko ardura.

Adierazpen hori ezagututa, beharrezkoa da *Problem*a erauztea. Erauzketa hau *DSLexpression2Problem* moduluaren bidez egiten da. Honek SQL adierazpena sarrera bezala dauka, eta emaitza gisa *Subproblem*ez osatutako *Problem* bat sortzen du. Kasu honetan azpiarazo bakarra lortzen da: “Any clue about the table that holds the students’ marks in $schema?”. Kontuan hartu behar da azpiarazo hori testuinguru zuzen batean ematen dela (adibidez, SQLn testuingurua datu-basearen eskema da).

Azpiarazo hori zehazteari begira, bere testuingurua esplizitu egitea beharrezkoa da. Lan hori *azpiarazoak testuinguruan jartzeko modulua*ri dagokio. Azpiarazoa testuinguruan ipini eta gero, non argitaratu nahi dugun erabaki beharko dugu. *subproblem2Message* moduluari dagokio azpiarazoa microblogintza plataforma batean argitaratzeko lana (adibidez, Twitter bidez). Argitaratu eta gero, *Contributors*ek azpiarazo horri erantzun ahal izango diote (adibidez, Twitter-en erantzunak sortuta). Behin erantzun guztiak bildu eta gero, hauetatik azpisoluzioak erauzteko unea iritsi da.

Lan honen arduraduna *message2Subsolution* modulua da. SQL adierazpenaren kasu honetan, eta azpiarazo honetarako, modulu honek datu-baseko taulen izenak identifikatu behar ditu. Azkenik, azpisoluzio guzti horiek erauzi eta gero, bakar batera laburtzeko unea iritsi da.

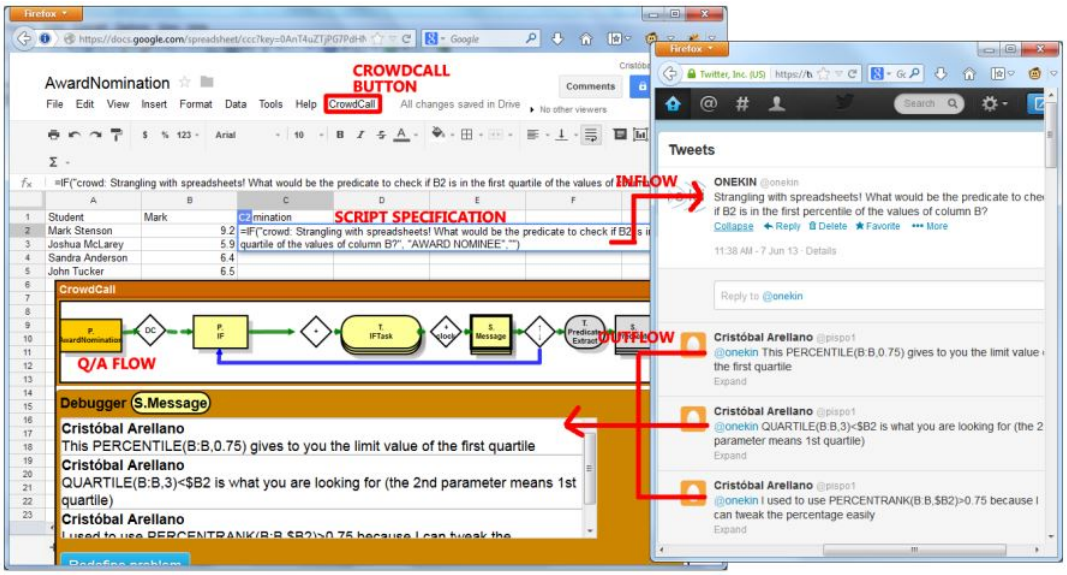
subsolutions2Subsolution modulua lan hau egiteaz arduratzen da. SQL taulen izenen kasuan, modulu honek erabiltzailearen eskuetan uzten du azpisoluzioaren aukeraketa lana. Estrategia honi "manuala" deitzen zaio.

3.3.2 Kalkulu orrien adibidea

Jarraian azalduko den adibidea kalkulu orriak editatzeko Google-ren editorean oinarritzen da. Bertan, DSL berezi bat erabiltzen da kalkulu orrietako gelaxkei formula konplexuak lotzeko. Formula hauek arlo desberdinetakoak izan daitezke: arlo matematikokoak, finantza-arlokoak, eta abar (Google-ren kalkulu orrien editoreak ehunka funtzio onartzen ditu). Erabiltzaileak normalean erabiltzen diren formulak erabiltzera ohituta daude, baina baliteke beste arlo batzuetako formulak lortzeko laguntza behar izatea. Mikroblogintza teknikak erabiltzea dokumentazioa begiratzearen alternatiba bihurtzen ari da.

Ikasleak saritzeko hautagaiak lortzeko kasua har dezagun. Batzordeko kide batek lehen kuartilan dauden notak dituzten ikasleak lortu behar ditu. Kalkulu orriak hiru zutabe ditu: ikaslearen izena, nota, eta hautagaitzaren zutabea. Zutabe honek funtzio kondizional bat definituta izan beharko luke: ikaslearen nota lehenengo kuartilan badago, gelaxkan "AWARD NOMINEE" pantailaratu beharko litzateke. Zoritxarrez, erabiltzaileak ez ditu lehenagotik kuartilak erabili, eta ondorioz, bere Twitter-eko jarraitzaileei laguntza eskatzera behartuta ikusten da. Horregatik, gelaxkako IF klausula ondorengo adierazpenarekin osatzen du: "crowd: Strangling with spreadsheets! What would be the predicate to check if B2 is in the first quartile of the values of column B?". Adibide hau 3.14 irudian irudikatuta azaltzen da.

Oracle-ren SQL Developer editorearen kasuan bezala, Google-ren kalkulu orrien editoreak hedatuta egon beharko da "crowd:" klausula detektatzen denean CrowdCall tresnari deitzeko. Gainera, editorearen testuingurura egokitzeko konfiguratu beharko da. Honek esan nahi du balizko ebazpenek izan beharreko datu motak, jasotako ebazpenak bakarrera laburtzeko estrategiak, etab. aldez aurretik zehaztu behar direla. Behin konfiguratuta, CrowdCall tresnak dakin nola maneiatu IF azpiarazoak. Kasu honetan, balizko ebazpenen tarteak Googleren kalkulu orrietan erabil daitezkeen funtzioak biltzen ditu.



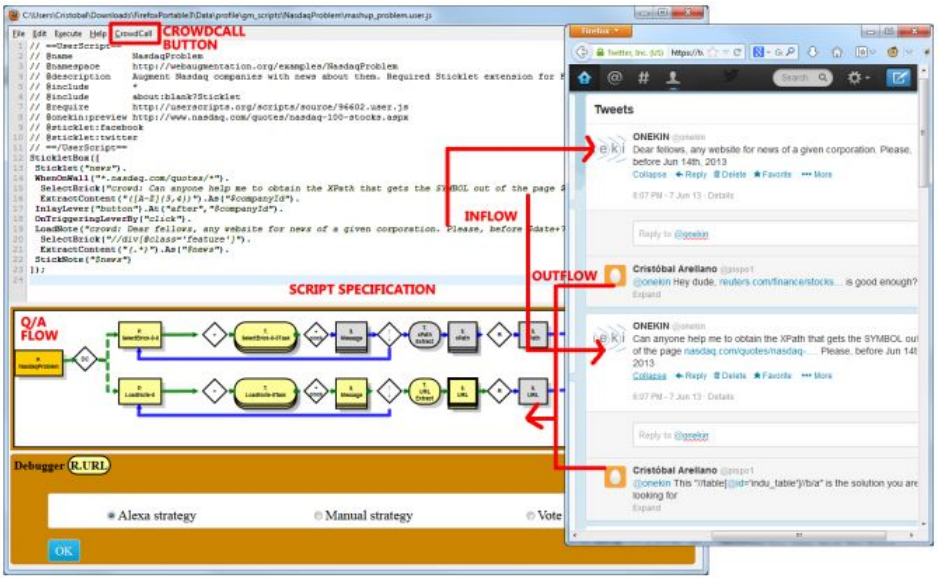
3.14 Irudia: CrowdCall tresna Google-ren kalkulu orrien editorearekin bateratuta lan egiten.

Bestalde, kasu honetan ezin da aplikatu 3.3.1 atalean aipatutako estrategia "manuala" laburtze-estrategia bezala, erabiltzaileak editoreko formulen arteko semantika desberdintasunak ezagutzen ez dituelako. 3.14 irudian ikus daitekeenez, erantzunetan PERCENTILE, QUARTILE eta PERCENTRANK aukerak aurki ditzake, Nahiz eta erabiltzaileak aukera hauen artean hoberena zein den ebaluatzeko gaitasunik ez izan, formula hauek aplikatzerakoan sortzen den emaitzan oinarrituta aukeratu dezake. Erabiltzaileak sintaxia ez du ezagutzen, baina bai badaki formulak egin beharrekoa. Beraz, balizko ebazpen bakoitzak sortzen duen emaitzen artean, lortu nahi duen emaitzarekin hobekien bat egiten duena aukera dezake. Honi "aurrebista" estrategia deritzo. CrowdCall tresnak aukera guztiak korritzen ditu, bakoitzak sortzen duen irteera erakutsita. Azkenik, erabiltzaileari nahi duenarekin hobekien bat etortzen den ebazpena zein den galdetzen zaio.

3.3.3 Sticklet adibidea

3.2.1 azpiatalean azaltzen den moduan, Sticklet weba areagotzeko tresna da. Har dezagun berriz 3.2.2 azpiatalean azaldutako adibidearen enuntziatua: burtsa-agente batek bere lana arintzeko NASDAQ webgunea areagotu nahi du beste webguneren bateko informazioarekin. Horretarako Sticklet tresna erabili nahi du, baina kodea idazterakoan zenbait arazo ditu.

3.15 irudiak adibide honetarako erdi osatutako Sticklet script bat erakusten du. Bertan, klausula batzuk osatuta daude, baina beste batzuk osatzeko arazoak izan dituenez,



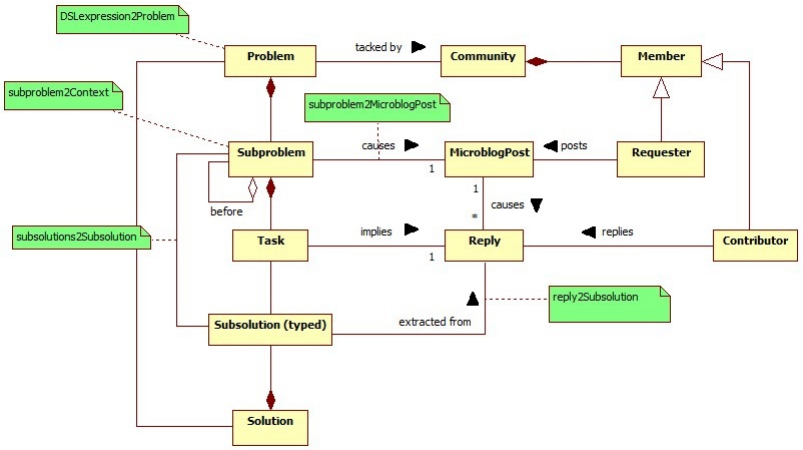
3.15 Irudia: CrowdCall tresna Sticklet editorearekin bateratuta lan egiten.

klausula horiek osatzeko “crowd:” adierazpena txertatu du. Adierazpen hauek interpretatu ahal izateko, Sticklet editorea CrowdCall tresnarekin lan egiteko hedatuta egon beharko da. Bestalde, Sticklet-en testuingurura egokitzeko konfiguratu beharko da. Horrela, Sticklet-en motorrak “crowd:” adierazpen bat detektatzerakoan, CrowdCall tresna martxan jarriko du, eta honek jakingo du nola tratatu stickleta osatzerakoan sortzen diren arazoak.

Adibide honetan, erabiltzaileak XPath adierazpen bat eta URL bat eskatzen ditu Twitter bidez. Orduan, CrowdCall gai izan beharko da jasotako erantzunetatik mota honetako adierazpenak erauzteko. Hau ere konfigurazio orduan egin beharreko lana da.

Bestalde, erabiltzailearen nahietara hobekien egokitzen den URLa eta XPath adierazpena zein den aukeratu ahal izateko modua ere adierazi beharko da konfigurazioa egiteko orduan. XPath adierazpenen kasuan, 3.3.2 azpiataleko adibidean bezala, erabiltzaileak ez du gaitasunik jasotako balizko ebazpen guztiak ikusita hoberena zein den bere kabuz aukeratzeko. Baina adibide horretan gertatzen den bezala, ebazpen horiek sortzen duten emaitzak ikusita, hau da, XPatha URL baten gainean aplikatuta lortzen den emaitza aztertuta, bere nahietara gehien egokitzen den XPath adierazpena aukeratzeko gai izango da. Beraz, 3.3.2 azpiataleko kasuan bezala, “aurrebista” estrategia erabili daiteke horretarako.

URLei dagokionez, aldiz, webgune baten kalitatea www.alexa.com webgunean duen ospearen arabera neurtu daiteke. Hori dela eta, kolaboratzaileek proposatutako URLak modu honetan (“Alexa” estrategia) sailka daitezke ebazpen zerrenda bakarrera laburtzeko.



3.16 Irudia: CrowdCall-en modulu konfiguragarrien kokapena eredu kontzeptualean.

3.3.4 Laburpena

Atal honetan azaldutako adibideen bidez, konfigurazioa egitearen motibazioa azaldu da, kasu errealetan oinarrituta. CrowdCall-ek hainbat editorerekin batera lan egitea nahi bada, editoreen testuingurura egokitzeko aukera eskaini behar da. Horrela, CrowdCall-ek DSL editore bakoitzean sortzen diren arazoak ulertu ditzake.

Kapituluaren hasieran aipatzen den bezala, bost dira konfigura daitezkeen elementu edo moduluak. 3.16 irudian modulu hauetako bakoitzak eredu kontzeptualean jokatzen duen papera islatuta azaltzen da.

- *DSLexpression2Problem* moduluari DSL editoreetako adierazpenetatik abiatuta tresnak ulertu ditzakeen *Problem*ak sortzeko prozesua dagokio.

- *subproblem2Context* moduluak *Subproblem*ak hartu eta editorearen testuinguruan jartzen ditu.

- *subproblem2MicroblogPost* moduluak *Subproblem*ak hartu eta *Community*etan laguntza eskaerak hartzen ditu.

- *reply2Subsolution* moduluak laguntza eskaerari egindako *Reply*etatik baliagarria den informazioa, hots *Subsolutionak*, erauzteaz arduratzen da.

- *subsolutions2Subsolution* moduluaren betebeharra *Reply*etatik erauzitako *Subsolution*en artean hoberena aukeratzeaz arduratzen da.

DSLexpression2Problem modulua editore bakoitzean sortutako adierazpenetatik abiatuta tresnak ulertu ditzakeen arazoak sortzeaz arduratzen da. Editore bakoitzak adierazpenak formatu desberdin batean pasatzen dizkio CrowdCall-i. Beraz, editoreko adierazpenak CrowdCall adierazpenetara eraldatzeko beharra sortzen da. *subproblem2Context* moduluak azpiarazo bakoitzari dagokion laguntza eskaera osatzen du. Honen adibide argi bat Sticklet editorea da. Bertan, laguntza eskaeretan $currentURL edo $currentBrick adierazpenak erabil daitezke. Lehenengoak stickleta deitzen duen web orriaren URLarekin ordezkatzen da. Bigarrena, aldiz, *SelectBrick* azpiarazo baten emaitzarekin ordezkatzeko erabiltzen da. Ikus daitekeen moduan, adierazpen hauek bakarrik zentzua dute Sticklet editorearen barruan. Beraz, nabarmena da editore bakoitzak hau konfiguratu beharko duela.

Azaldutako adibide guztietan Twitter erabili da laguntza eskaerak aditzera emateko mikroblogintza plataforma bezala. Baina Twitter erabiltzen den moduan, edozein SNS plataforma erabil daiteke, esate baterako, Facebook. Beraz, editore bakoitzak ikusi beharko du azpiarazoaren izaerak eskatzen duen profila zein plataformako erabiltzaileekin hobekien bat etortzen den, eta *subproblem2MicroblogPost* moduluaren konfigurazioa horren arabera egin beharko du.

Erantzunen barruan dauden azpisoluzioen identifikazioari dagokionez, Twitter-eko mezuetatik (testu lauz osatutakoak) URLak, XPath adierazpenak, adierazpen erregularrak, Googleren kalkulu orrietako formulak edo SQL taulen izenak modu desberdinetan erauzi egiten dira. Beraz, editorearen ardura da editorean bertan sor daitezkeen azpiarazoetako balizko ebazpenak kolaboratzaileek idatzitako mezuetatik erauzteko metodologia definitzea, eta *reply2Subsolution* modulua konfiguratzerako orduan adieraztea.

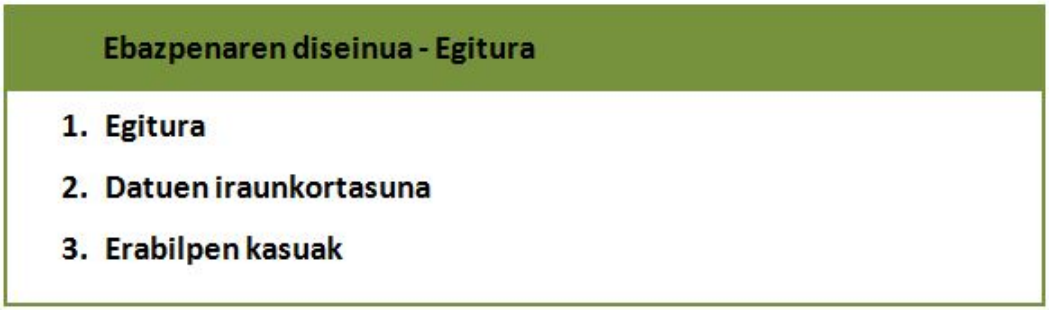
Azkenik, azpiarazo bakoitzaren izaerak balizko ebazpenak azpisoluzio bakarrera laburtzeko modua baldintzatzen du. Horrela, azpisoluzioa URL motakoa denean, adibidez, “Alexa” estrategia erabili daiteke, www.alexa.com webguneko sailkapenean oinarrituta. Bestalde, azpisoluzioa XPath edo Googleren kalkulu orrietako formula bat denean, “aurrebista” estrategia erabil daiteke, erabiltzaileak azpisoluzioek sortuko luketen emaitzen artean aukeratzeko. Beste kasu batzuetan estrategia “manuala” erabiltzeko aukera eman daiteke, erabiltzaileak balizko ebazpenen artean egokiena aukeratzeko. Ondorioz, editore bakoitzak *subsolutions2Subsolution* modulua konfiguratu beharko du sor ditzakeen azpiarazoetako azpisoluzioen artean aukeraketa egiteko mekanismo egokiak eskaintzeko.

Beraz, CrowdCall editore bakoitzak sor ditzakeen azpiarazoetara egokitzeko konfigurazio mekanismo bat eskaini behar da.

4. KAPITULUA

Ebazpenaren diseinua

Kapitulu honetan ebazpenaren diseinuaren nondik norakoak azaltzen dira. Aurrena, ebazpenaren egitura azalduko da, tresnak dituen modulu desberdinen klase diagramak deskribatuta. Ondoren, datuen iraunkortasunerako aukeratutako mekanismoa azalduko da. Azkenik, tresnaren erabilpen kasuak deskribatuko dira, bakoitzaren sekuentzia diagrama azalduz.



4.1 Egitura

3.3 atalean azaldutakoaren arabera, aplikazioak honako bost elementu edo modulu konfiguragarri ditu:

- *DSLexpression2Problem*: Domeinuaren adierazpenetik abiatuta arazoaren sorkuntzarako modulua.
- *subproblem2Context*: azpiarazoak testuinguruan jartzeko modulua.
- *subproblem2MicroblogPost*: SNSak kudeatzeko modulua.
- *reply2Subsolution*: erantzunetan azpisoluzioak identifikatzeko modulua, datu motetan oinarrituta.
- *subsolutions2Subsolution*: azpisoluzioak bakarrera laburtzeko modulua.

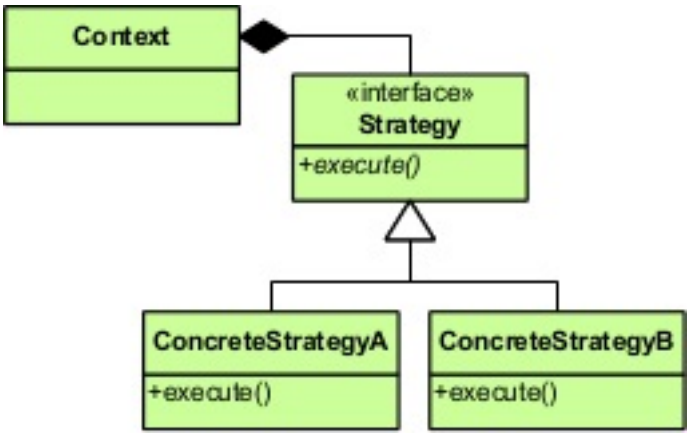
DSLexpression2Problem moduluari dagokionez, konfigurazioa egiten den bakoitzean, CrowdCall-ek domeinuaren adierazpenetatik abiatuta arazoak sortzeko algoritmo bakarra jasoko du DSL editoretik. Hori dela eta, modulu hau metodo bat balitz bezala har daiteke.

Beste lau moduluei dagokienez, aldiz, DSL editoreak algoritmo bat baino gehiago defini dezake modulu bakoitzarentzat, eta exekuzio garaian erabiliko dena erabakiko da. Adibidez, *reply2Subsolution* moduluaren kasuan, Sticklet editorek algoritmo desberdinak definitu beharko ditu, mezuetatik XPath adierazpenak, adierazpen erregularrak edo URLak erauzi ahal izateko. Hori dela eta, *Strategy* diseinu patroia erabiltzea erabaki da. Jarraian, diseinu patroi hau zertan datzan eta lau modulu hauetan patroi honen aplikazioaren nondik norakoak azalduko dira.

D eranskinean modulu hauek inplementatzeko argibideak emango dira, adibideak txertatuz ulergarritasuna handitzeko.

4.1.1 *Strategy* diseinu patroia

Strategy softwarearen garapenerako diseinu patroia da. Patroi hau portaera patroien sailkapenean sartzen da, ataza bat burutzeko egin beharreko objektuen arteko mezu-trukea zehazten baitu. *Strategy* patroian, algoritmo multzo bat gordetzen da, eta bezero-objektuak



4.1 Irudia: *Strategy* diseinu patroiaren klase diagrama.

hauetako zein erabiltzea komeni zaion erabaki dezake, eta bere beharren arabera dinamikoki aldatu.

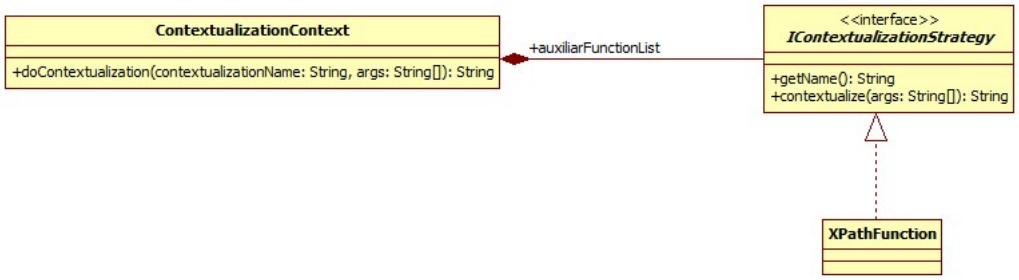
4.1 irudian ikus daitekeen moduan, patroiaren aplikazioan, hiru parte-hartzaile hauek bereizten dira:

- Estrategia (*Strategy*): algoritmo guztiek bete beharreko interfaze bat definitzen du. Testuinguruak interfaze hau erabiliko du estrategia konkretu bat deitzeko.
- Estrategia konkretua (*ConcretStrategy*): estrategiak definitutako interfazea erabilita algoritmo bat inplementatzen du.
- Testuingurua (*Context*): algoritmoak erabiltzen dituen elementua da. Estrategia konkretu bat erabiltzeko estrategia horretarako erreferentzia bat gordetzen du.

CrowdCall-en patroi hau aplikatzeko, JavaScripten objektuetara orientatutako programazio lengoaia izaera erabiliko da. Hala ere, CrowdCall-en modulu desberdinetako estrategia konkretuen erabilerak ez du jarraitasunik izango, hau da, algoritmo konkretu bat exekutatzean ez da kontuan hartuko zein den aurretik exekutatu dena. Hori dela eta, estrategia konkretu bat exekutatzen den aldiro estrategiaren izena beharrezkoa izango da, eta ondorioz, testuinguruan ez da aukeratutako estrategiarako erreferentzia gordeko. Beraz, ez da *Strategy* diseinu patroiaren aplikazio puru bat egingo.

4.1.2 subproblem2Context modulua

4.2 irudiko klase diagraman ikus daitekeen bezala, modulu honen egituraren diseinuan *Strategy* diseinu patroiaren aplikazioan bereizten diren elementuak azaltzen dira.



4.2 Irudia: *subproblem2Context* moduluaren klase diagrama

Batetik, DSL editoreek inplementatu beharreko interfazea dago (*ICotextualizationStrategy*). Honek bi metodo dauzka: *getName()* eta *contextualize()*, interfazea inplementatzen duten estrategia konkretu guztiek inplementatu beharko dituztenak. Lehenengoa estrategiaren izena atzitu ahal izateko metodoa da; bigarrenak aldiz, azpiarazoak DSL editorearen testuinguruan ipintzeko balio du.

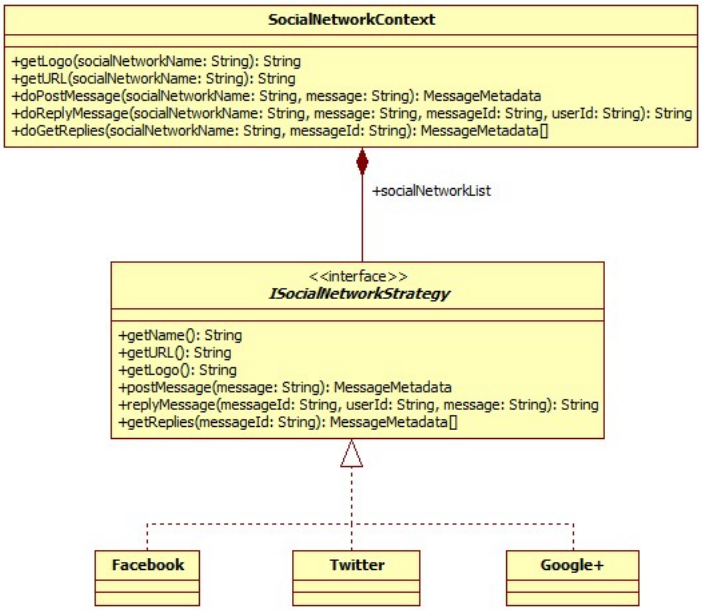
Bestetik, interfaze honen inplementazio baten adibidea dugu (*XPathFunction*), Sticklet editoreak erabiltzen duena. Honek interfazeak definitutako metodoak inplementatzen ditu. Klase honek egiten duen *contextualize()* metodoaren inplementazioan, XPath adierazpen bat eta URL bat emanda, URL horri dagokion web orriaren gainean XPath adierazpenarekin bat datorren testu katea itzultzen du.

Azkenik, estrategia konkretu desberdinak erabiliko dituen testuingurua dago (*ContextualizationContext*). Horretarako, *doContextualization()* metodoa deitu beharko du, estrategiaren izena eta honek jasoko dituen argumentuak pasata.

4.1.3 subproblem2MicroblogPost modulua

4.3 irudiko klase diagraman ikus daitekeen bezala, modulu honen kasuan estrategia konkretu bakoitza ez dago algoritmo bakar batez osatuta; SNS plataforma jakin bat kudeatzeko metodoak (erabiltzailearen logina egiaztatu, mezu bat idatzi, mezu bati erantzun eta erantzunak lortu) biltzen ditu.

Horrela, estrategia konkretuek inplementatu beharreko interfazean (*ISocialNetworkStrategy*), atributuak atzitzeko metodoez (*getName()*, *getURL()* eta *getLogo()*) gain, estrategia konkretu bakoitzak inplementatu beharreko metodoak aurki daitezke: *postMessage()*, *replyMessage()* eta *getReplies()*.



4.3 Irudia: *subproblem2MicroblogPost* moduluaren klase diagrama

Estrategia konkretuei dagokienez, klase diagraman interfaze honen hiru inplementazio azaltzen dira: *Twitter*, *Facebook* eta *Google+*.

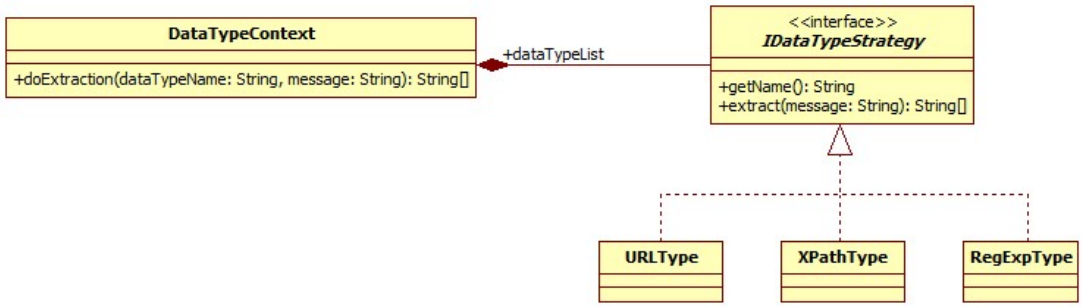
Azkenik, testuinguru klasea ere aurkitu dezakegu: *SocialNetworkContext*. Honek interfazean definitutako metodoak deitu ahal izateko metodoak definituta dauzka (*getLogo()*, *getURL()*, *doPostMessage()*, *doReplyMessage()* eta *doGetReplies()*). Estrategia konkretu bat erabiltzeko, nahikoa du erabili nahi duen estrategia konkretuaren izena metodo hauen *socialNetworkName* parametro gisa pasatzearekin.

4.1.4 reply2Subsolution modulua

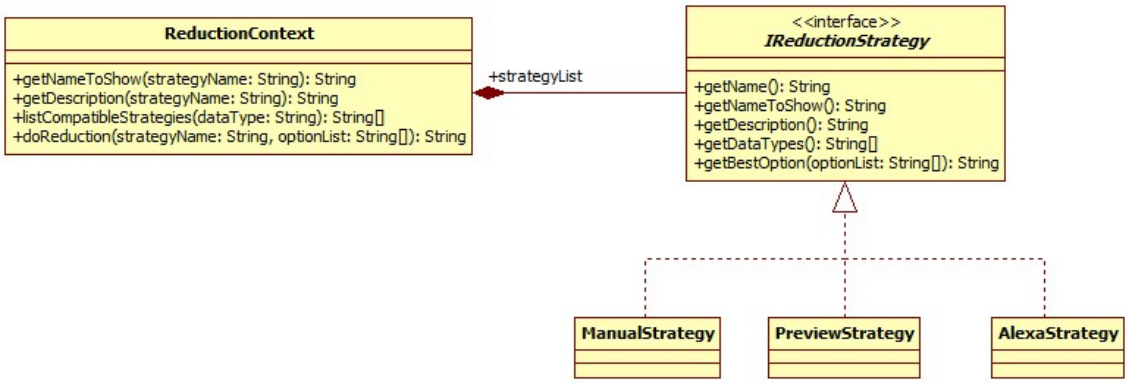
Modulu honen egituran *Strategy* patroiak definitzen dituen hiru elementu motak (testuingurua, estrategia konkretuek inplementatu beharreko interfazea eta estrategia konkretuak) azaltzen dira. 4.4 irudiko klase diagraman islatuta ikusten da.

Interfazeak (*IDataTypeStrategy*) estrategia konkretuek inplementatu beharreko metodoak zehazten ditu: *getName()* eta *extract()*. Lehenengoa estrategia bakoitzaren izena atzitu ahal izateko metodoa da, eta bigarrena, aldiz, mezu batetik abiatuta datu mota jakin batekin bat egiten duten adierazpenak erauzteaz arduratzen da.

Klase diagraman, interfazea inplementatzen duten estrategia konkretuen adibide mo-



4.4 Irudia: *reply2Subsolution* moduluaren klase diagrama



4.5 Irudia: *subsolutions2subsolution* moduluaren klase diagrama

duan Sticklet editoreak erabiltzen dituenak azaltzen dira (*URLType*, *XPathType* eta *RegExpType*).

Azkenik, *DataTypeContext* klaseak testuinguru papera jokatzen du: interfazearen inplementazioak *dataTypeList* zerrendan gordetzen ditu, eta horietako bat exekutatu nahi duenean, nahikoa du *doExtraction* metodoari aukeratutako estrategiaren izenarekin deitzearekin.

4.1.5 subsolutions2subsolution modulua

Modulu honen egiturari dagokionez, *Strategy* diseinu patroia aplikatuta, 4.5 irudian ikus daitekeen klase diagrama lortu da. Bertan, patroiak definitutako hiru klase motak bereizten dira: testuinguru klasea, (*ReductionContext* klasea), interfaze abstraktua irudikatzen duen klasea (*IReductionStrategy*) eta interfaze abstraktuaren gauzatzeak (*AlexaStrategy*, *PreviewStrategy* eta *ManualStrategy*).

Interfaze abstraktuak aukeraketa estrategia batek inplementatu beharko dituen metodoak zehazten ditu: *getName()*, *getNameToShow()*, *getDescription()* eta *getDataTypes()* interfazearen gauzatzeen eremuak atzitu ahal izateko, eta *getBestOption()* pasatako aukerak bakarrera laburtzeko.

Interfaze abstraktuaren gauzatzeei dagokienez, klase diagraman 3 kapituluan aipatzen diren batzuk adierazten dira: *AlexaStrategy*, *PreviewStrategy* eta *ManualStrategy*.

Azkenik, testuinguru klaseari dagokionez, estrategia konkretu zerrenda bat gordetzen du *strategyList* eremuan. Horrez gain, estrategia konkretuetako eremuak atzitzeko metodoak (*getNameToShow()* eta *getDescription()*) eta estrategia konkretu bakoitzak definitutako aukeraketa algoritmoa exekutatzeko metodo bat (*doReduction()*) eskaintzen ditu. Hauek erabiltzeko nahikoa izango da parametro bezala estrategia konkretuaren izena pasatzearekin. Bestalde, lana errazteko, datu mota bat adierazten duen karaktere kate bat pasata horrekin bateragarriak diren estrategien izenak itzultzen dituen metodoa definitu da (*listCompatibleStrategies()*).

4.2 Datuen iraunkortasuna

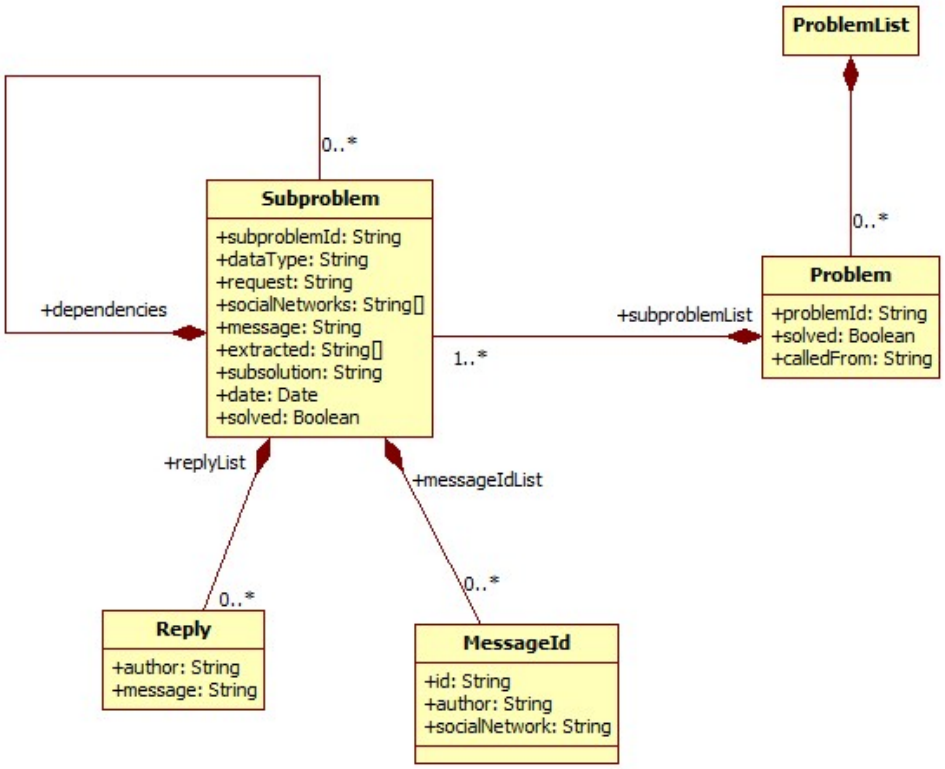
Tresnak betetzen duen funtzioa dela eta, saio bateko datuak gorde behar dira, ondorengo saioetan erabili ahal izateko. Adibidez, erabiltzaile batek saio batean azpiarazo baten azpisoluzioa lortzen badu, azpisoluzio hori nolabait biltegiratu beharko da, tresna hasieratzen den hurrengoetan berriro ez egiteko.

Tresna inplementatzeko erabiliko diren teknologiak direla eta, helburu honetarako Greasemonkeyk *GM_setValue* eta *GM_getValue* funtzioen bidez eskaintzen duen biltegiratze sistemaz baliatuko da. Hau erabilita, JSON notazioko objektuak karaktere kateak bilakatuta gorde daitezke.

4.6 irudian ikus datekeen moduan, biltegiratutako datu-egitura *ProblemList* batez osatuta dago. Hau zerrenda bat da, eta hainbat *Problem* biltzen ditu.

*Problem*etako bakoitza arazo independente bati dagokio, eta honako eremuak gordetzen ditu:

- Arazoaren identifikatzaile bat (*problemId*).
- Arazoa ebatzita dagoen edo ez adierazten duen aldagai boolearra (*solved*).



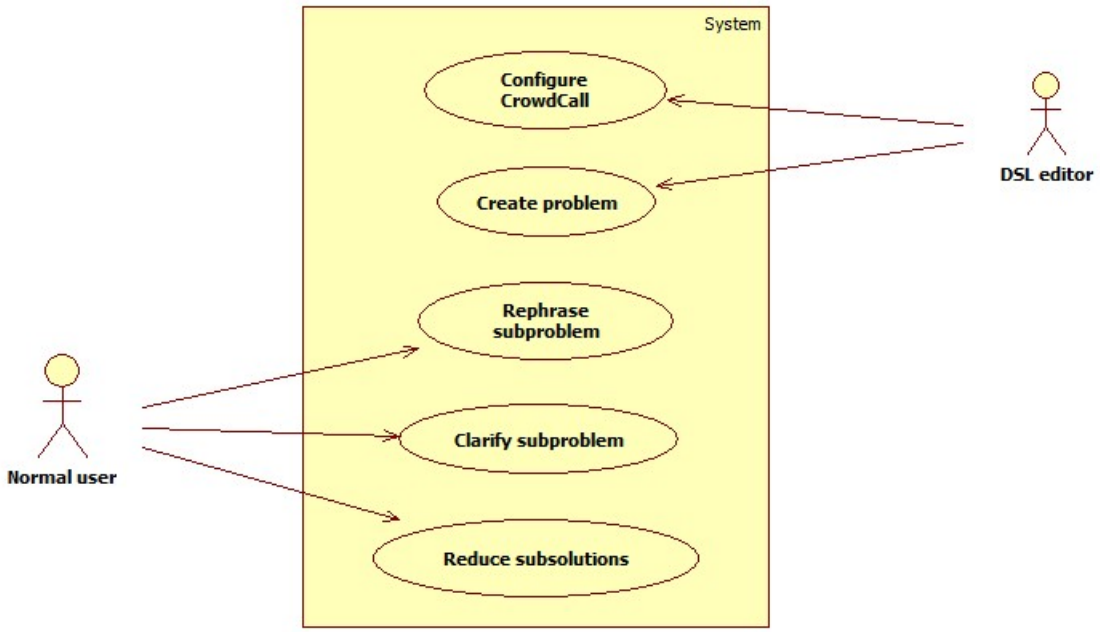
4.6 Irudia: Biltegiratzen den datu-egitura adierazten duen diagrama

- CrowdCall tresnaren deitzailearen erreferentzia (*calledFrom*) gordetzen ditu.
- *subproblemList* eremuan arazoa osatzen duen azpiarazo edo *Subproblem* zerrenda bat gordetzen da.

Problem bateko *subproblemList*ean gordetzen den *Subproblem* bakoitzarentzako honako eremuak gordetzen dira:

- Identifikatzaile bat (*subproblemId*).
- Espero den datu mota (*dataType*).
- Azpiarazoari dagokion laguntza eskaera (*request*).
- Laguntza eskaera argitaratuko den mikroblogintza sare zerrenda (*socialNetworks*).
- Mikroblogintza sareetan argitaratutako mezua (*message*).
- Erantzunetatik erauzitako balizko ebazpenak (*extracted*).
- Azpiarazoarentzako azpisoluzioa (*subsolution*).
- Laguntza eskaera egin deneko data (*date*).
- Azpiarazoa ebatzita dagoen edo ez adierazten duen boolearra (*solved*).
- Mikroblogintza sareetan mezuak hartu duen identifikatzaile zerrenda (*messageIdList*). Hauetako bakoitzak, aldi berean, *MessageId* egitura bat osatzen du, eta mikroblogintza sarearen izena (*socialNetwork*), sare horretan erabiltzaileak duen identifikatzailea (*author*) eta argitaratutako mezuak duen identifikatzailea (*id*) gordetzen ditu.
- Mikroblogintza sareetan eskaerak jasotako erantzunak (*replyList*). Hauetako bakoitzak, aldi berean, *Reply* egitura bat osatzen du, erantzuna (*message*) eta erantzun horren egilearen identifikatzailea (*author*) gordetzen dituena.
- Azpiarazoak beste azpiarazoekiko dituen menpekotasunak (*dependencies*). Eremu hau zerrenda bat da, eta azpiarazoen identifikatzaileak bildu ditzake. Azpiarazo batek bere *dependencies* eremuan beste azpiarazo batzuen identifikatzaileak edukitzeak ebazten hasi baino lehen adierazitako azpiarazoak ebatzi behar direla esan nahi du.

Aipatzekoa da hasiera batean eremu gehienak hutsak egoten direla, eta ebazpen prozesuak aurrera egin ahala betetzen joaten direla.



4.7 Irudia: CrowdCall-en erabilpen kasuen diagrama

4.3 Erabilpen kasuak

Atal honetan CrowdCall-en bereiz daitezkeen aktoreak, eta hauei lotuta dauden erabilpen kasuak azalduko dira, aktore hauen eta sistemaren artean sortzen den elkarrekintza argitzeko. 4.7 irudian erabilpen kasuen diagrama ikus daiteke, jarraian egingo diren azalpenen eskematzat har daitekeena.

4.3.1 Aktoreak

4.7 diagraman ikus daitekeenez, CrowdCall-en bi aktore mota bereizten dira: erabiltzaile arruntak (*normal user*) eta DSL editoreak (*DSL editor*). Ikus daitekeenez, azpiarazoak ebatzi ahal izateko laguntza ematen duten kolaboratzaileak diagramatik kanpo gelditu dira. Izan ere, hauek ez dute CrowdCall-ekin zuzenean elkarreragiten, SNS plataforma desberdinekin baizik. Gainera, kolaboratzaileek ez dute zertan CrowdCall-en erabilera-ren berri izan, hauen ekarpenak Twitter bezalako mikroblogintza sare baten bidez egiten baitira.

Aurretik aipatu den moduan, DSL editorea tresna honetan bereiz daitezkeen bi aktore motetako bat da. Erabiltzaile mota hau ez da pertsona fisiko bat, baizik eta aplikazio

informatiko bat. CrowdCall DSL editoreetan txertatua izateko eginda dago; ondorioz, editore hauek CrowdCall-ekin elkarreragin beharko dute. Erabiltzaile mota honen adibideak 3.3 ataleko adibideetan aurki ditzakegu: Oracle SQL Developer, Google Spreadsheet eta Sticklet.

CrowdCall tresnak izan dezakeen beste erabiltzaile mota erabiltzaile arrunta da. Erabiltzaile mota honek arazoa ebazteko CrowdCall-en interfaze grafikoarekin elkarreragiten du, eta kasu honetan bai, pertsona fisiko bat izango da. Erabiltzaile arruntak CrowdCall DSL editore baten barruan erabiliko du, DSL horretako adierazpen bat osatzeko laguntza eskatzeko. Erabiltzaile mota honen adibideak 3.3 ataleko adibideetan aurki ditzakegu: DSL editoreen atzean dauden pertsonak izango litzateke.

Hurrengo azpiataletan, aktore hauei dagozkien erabilpen kasuak azalduko dira, hauetako bakoitza zertan datzan aipatuta. Aurrena, DSL editoreei lotutako erabilpen kasuak azalduko dira: “CrowdCall konfiguratu” (*configure CrowdCall*) eta “arazo bat sortu” (*create problem*), eta ondoren gauza bera egingo da erabiltzaile arruntei lotutakoekin: “azpiarazo bat berriz definitu” (*rephrase subproblem*), “azpiarazo bat argitu” (*clarify subproblem*) eta “azpiarazo baten azpisoluzioak laburtu” (*reduce subsolutions*).

4.3.2 CrowdCall konfiguratu

Izenburua: CrowdCall konfiguratu.

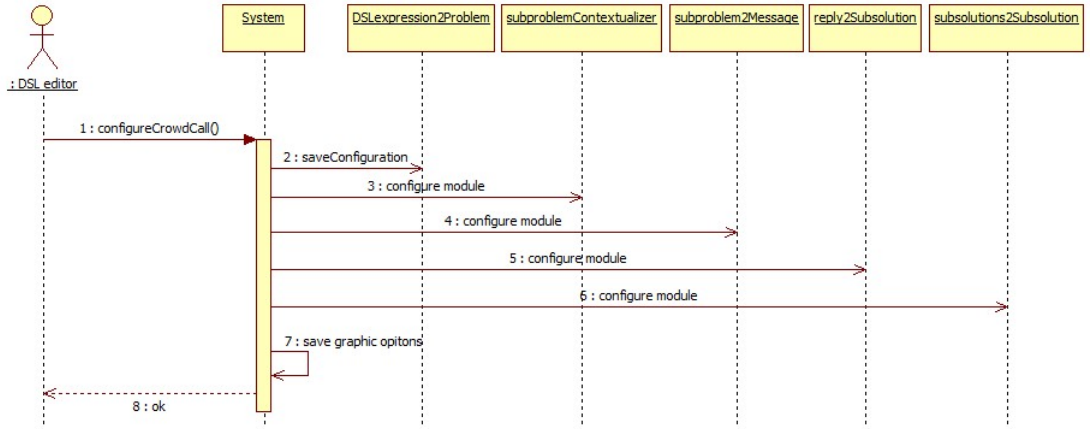
Aktore nagusia: DSL editorea.

Helburuak:

- CrowdCall tresna konfiguratzea DSL editorearen testuinguruan jartzeko.

Aurrebaldintzak:

- Konfigurazioak D eranskinean deskribatzen diren baldintzak betetzea.
- DSL editorea CrowdCall tresna erabiltzeko hedatuta egotea.



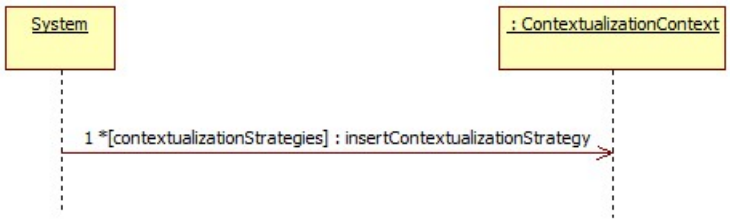
4.8 Irudia: *Configure CrowdCall* erabilpen kasuaren sekuentzia diagrama

Azalpena: Garatutako tresna edozein DSL editorerekin lan egiteko pentsatuta dago; beraz, CrowdCall editore jakin baten testuinguruan jartzeko modua eskaini behar da. 3.3 atalean deskribatzen den moduan, editorearen interfaze grafikora egokitzeko beharrezko informazioaz gain, bost dira konfigura daitezkeen elementu edo moduluak, DSL editore batetik bestera aldatu egiten direlako:

- Domeinuaren adierazpenetik abiatuta arazoaren sorkuntzarako modulua (*DSLexpression2Problem*).
- Azpiarazoak testuinguruan jartzeko modulua (*subproblem2Context*).
- SNSak kudeatzeko modulua (*subproblem2MicroblogPost*).
- Erantzunetan azpisoluzioak identifikatzeko modulua, datu motetan oinarrituta (*reply2Subsolution*).
- Azpisoluzioak bakarrera laburtzeko modulua (*subsolutions2Subsolution*)

Konfigurazio hau egiteko API dei bat definituta dago, eta DSL editoreak CrowdCall tresna hasieratu aurretik egin beharko du. D eranskinean sakonean azalduko dira konfigurazio prozesuaren nondik norakoak.

Sekuentzia diagrama: CrowdCall-en konfigurazio prozesua modu honetan egiten da: moduluak banaka konfiguratzen dira; ondoren editorearen interfaze grafikora egokitzeko informazioa gordetzen da, eta azkenik editoreari konfirmazio mezu bat bidaltzen zaio.



4.9 Irudia: *Strategy* diseinu patroiaren egitura jarraitzen duten moduluen konfigurazioaren sekuentzia diagrama.

Ikus daitekeenez, *DSLexpression2Problem* moduluren konfigurazioa DSL editoreak definitutako funtzioa gordetzean datza.

Bestalde, 4.1 atalean *Strategy* diseinu patroiaren egitura jarraitzen duten moduluak(*subproblemCont*
subproblem2MicroblogPost, *reply2Subsolution* eta *subsolutions2Subsolution*) konfiguratzeko orduan, 4.9 irudian ikus daitekeen sekuentzia diagrama jarraitzen da: editoreak sistemari pasatako estrategia konkretu guztiak korritu eta dagokion testuinguru klasearen barruan txertatzen dira.

4.3.3 Arazo bat sortu

Izenburua: Arazo bat sortu.

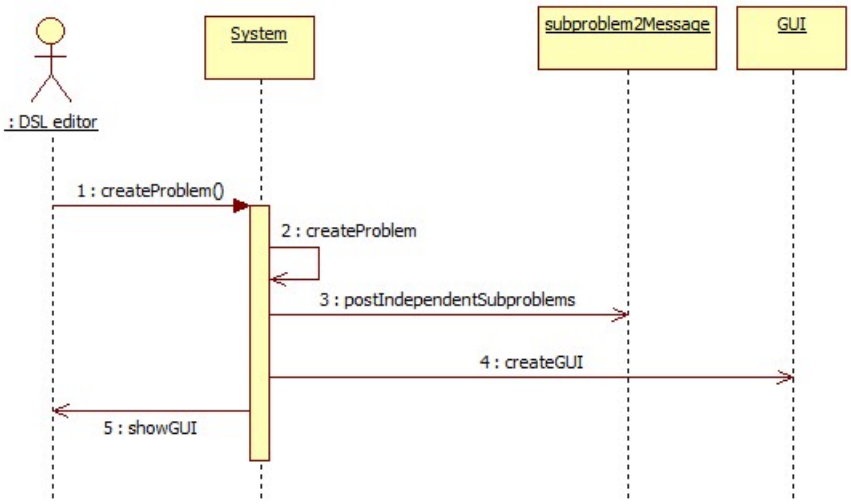
Aktore nagusia: DSL editorea.

Helburuak:

- Erabiltzaile arruntak ebatzi nahi duen arazoaren sorrera.
- CrowdCall martxan jartzea.

Aurrebaldintzak:

- CrowdCall-en konfigurazioa eginda egotea.
- DSL editorea CrowdCall tresna erabiltzeko hedatuta egotea.



4.10 Irudia: *Create problem* erabilpen kasuaren sekuentzia diagrama.

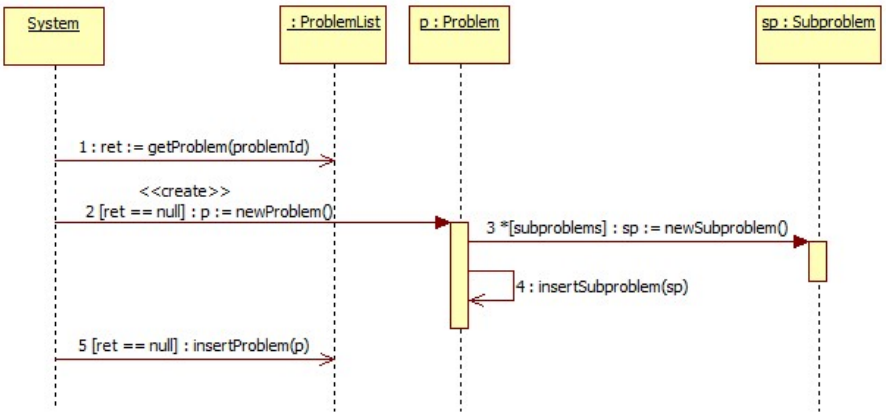
Azalpena: CrowdCall-en konfigurazioa egin ostean, arazo bat sortzen da, ondorenean kolaborazioz ebatziko dena. Horrela, arazoa sortu eta biltegiratu egingo litzateke, lehenagotik sortuta egongo ez balitz, eta tresnaren interfaze grafikoa abiatuko litzateke.

Arazo baten sorrera erabilpen kasurik konplexuena da. Erabiltzaile arrunt bat, CrowdCall bateratuta daukan DSL editore bat erabiltzen ari den bitartean, DSL horretako adierazpen bat osatzeko laguntza eskatzera behartuta ikusten da. Horretarako, editoreak CrowdCall erabiltzeko eskaintzen duen aukera erabiliko du, honen bidez arazoa sortu ahal izateko.

Honetan oinarrituta, erabilpen kasu honetako aktorea DSL editorea izango litzateke, eta ez erabiltzaile arrunta, hala iruditu arren. Erabiltzaile arruntak DSL editoreak eskainitako funtzionalitate bat erabiliko du, eta hau da CrowdCall-ekin elkarreragiteaz arduratuko dena. Beraz, kasu honetako benetako aktorea DSL editorea dela ondoriozta daiteke.

Sekuentzia diagrama: Arazo bat sortzeko prozesua 4.10 irudiko sekuentzia diagraman islatuta agertzen da. Prozesua honakoa da: aurrena arazoa sortzen da; ondoren, inolako menpekotasunik ez duten azpiarazoei dagozkien laguntza eskaerak azpiarazo bakoitzak adierazten dituen SNS plataformetan idazten dira. Azkenik, interfaze grafikoa martxan jartzen da.

4.10 irudiko sekuentzia diagraman, arazo bat sortzeko prozesua eta inolako menpekotasunik ez duten azpiarazoei dagozkien laguntza eskaerak SNS plataformatan idazteko



4.11 Irudia: Arazo bat sortzeko prozesuaren sekuentzia diagrama.

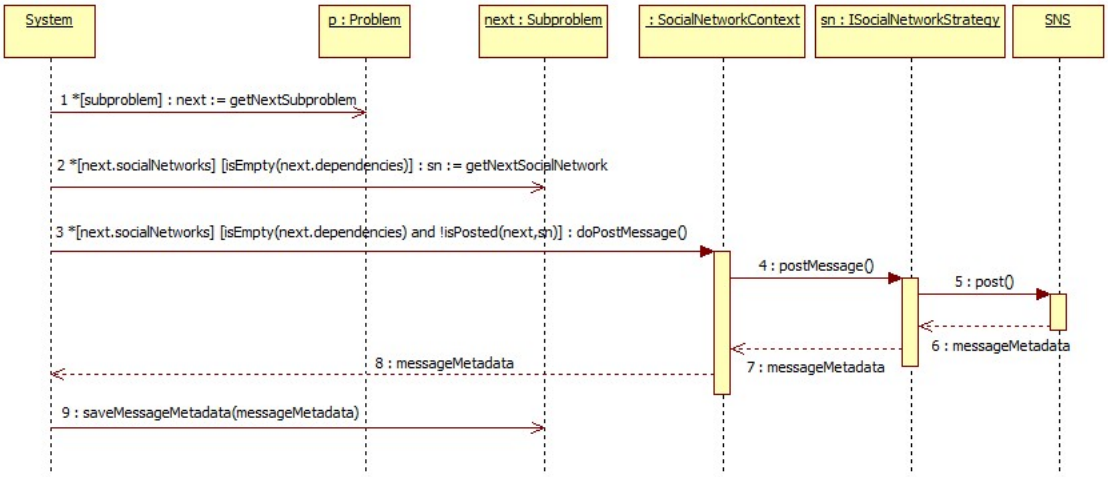
prozesua sinplifikatuta azaltzen dira, irakurgarritasun kontuak direla medio. 4.11 eta 4.12 irudietako sekuentzia diagramek bi prozesu hauek azaltzen dituzte.

Arazo bat sortzeko prozesua honakoa da: lehendabizi arazoa aurretik sortuta zegoen ikusten da. Hala ez bada, arazoa sortu, azpiarazoak sortu eta arazoaren barruan txertatu egiten dira. Azkenik, sortu berri dugun arazoa arazoak biltegiratzen dituen zerrendan txertatzen da.

Laguntza eskaerak SNS plataformetan idazteko prozesuari dagokionez, berriz, prozesua honakoa da: arazoaren azpiarazoak korritzen dira, eta inolako menpekotasunik ez badute (*dependencies* hutsa bada), azpiarazoaren *socialNetworks* zerrenda korritzen da. Zerrenda honetan azpiarazoa ebazteko erabiliko diren SNS plataformak biltzen dira. Zerrendako elementu bakoitzeko *subproblem2MicroblogPost* moduluko testuinguru klaseko *doPostMessage()* metodoari deituko zaio. Metodo honen *socialNetworkName* parametro gisa zerrendako elementua erabiliko da, eta *message* parametro gisa, azpiarazoaren *request* eremua. Testuinguru klase honek dagokion estrategia konkretuko *postMessage()* deituko du, eta honek, azkenik, mezua SNS plataforman idatziko du. Ondoren, SNS plataformak itzulitako mezuaren metadatuak objektuz objektu sistemarenganaino iritsiko dira, eta honek azpiarazoaren *messageId* eremuan txertatuko ditu (ikus 4.2).

4.3.4 Azpiarazo bat berriz definitu

Izenburua: Azpiarazo bat berriz definitu.



4.12 Irudia: *Create problem* erabilpen kasuko mezua idazteko fasearen sekuentzia diagrama.

Aktore nagusia: Erabiltzaile arrunta.

Helburuak:

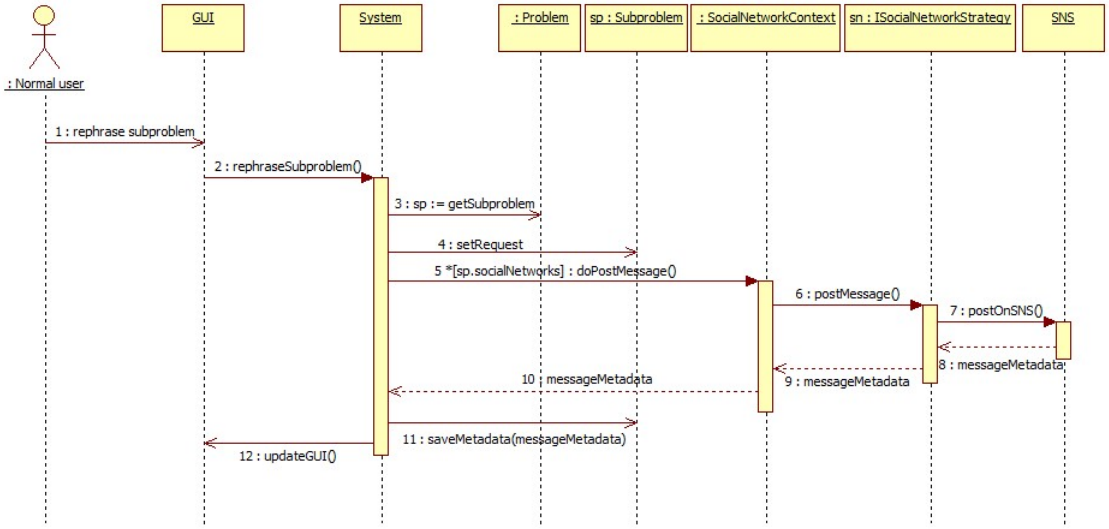
- Azpiarazoa definitzerakoan egindako adierazkortasun akatsak konpontzea.
- Kolaboratzaileek azpiarazoarentzat proposatutako ebazpenak egokiagoak izatea.

Aurrebaldintzak:

- Arazoa sortuta egotea.
- CrowdCall tresnaren interfaze grafikoa hasieratuta egotea.

Azalpena: Esperientziak dionenez, azpiarazo bat definitzerako orduan ez da beti eskatu nahi dena ongi adierazten. Adierazkortasun gabezi hauek direla eta, gaizki-ulertuak sor daitezke kolaboratzaileen artean, eta honen ondorioz, erabiltzaileak ez ditu hasiera batean espero zituen ebazpenak jasotzen.

Honi aurre egiteko, CrowdCall tresnak bi aukera ematen dizkio erabiltzaileari: azpiarazoa berriz definitzea edo eskaerari erantzunez azpiarazoa ebazteko argibideak ematea. Erabilpen kasu honetan lehenengo aukera lantzen da. Prozesua honakoa da: erabiltzaileak azpiarazoaren definizio berria proposatu eta gero, CrowdCall tresnak definizio zaharra berriarengatik ordezkatzen du, eta ondoren konfiguratutako SNS plataforman laguntza



4.13 Irudia: *Rephrase subproblem* erabilpen kasuaren sekuentzia diagrama.

eskaera berria idazten du. Erabiltzailearen eskuetan geratzen da laguntza eskaera zaharra ezabatzea, edo honi erantzutea ordura arte erantzun dutenei eskerrak emanez eta azpiarazo berrira bideratuz.

Sekuentzia diagrama: 4.13 irudian sekuentzia diagrama ikus daiteke. Diagramak deskribatutako prozesua honakoa da: hasteko, erabiltzaile arrunt batek interfaze grafikoaren bitartekaritzari esker, sistemari azpiarazo bat berriz definitzeko eskatzen dio. Sistemak azpiarazoa bilatzen du arazoaren *subproblemList*ean, eta ondoren, honen definizioa aldatzen du *setRequest* deiak adierazten duen moduan. Hau egin ostean, azpiarazoaren *socialNetworks* zerrenda korritzen da, honen elementuak *socialNetworkName* parametro bezala erabilita, *subproblem2MicroblogPost* moduluko testuinguru klaseko *doPostMessage()* mezuari deitzeko. Honek, aldi berean, SNS plataformen kudeaketa inplementatzen duten estrategia konkretuak deitzen ditu, eta hauek dira SNS plataformetan mezua idazten dutenak. SNS plataformak mezua idazterakoan itzulitako metadatuak objektuz objektu sistemaraino iristen dira, eta honek azpiarazoan biltegiratzen ditu geroago erabili ahal izateko. Azkenik, sistemak interfaze grafikoa eguneratzen du.

4.3.5 Azpiarazo bat argitu

Izenburua: Azpiarazo bat argitu.

Aktore nagusia: Erabiltzaile arrunta.

Helburuak:

- Azpiarazoa definitzerakoan egindako adierazkortasun akatsak konpontzea.
- Kolaboratzaileek azpiarazoarentzat proposatutako ebazpenak egokiagoak izatea.

Aurrebaldintzak:

- Arazoa sortuta egotea.
- CrowdCall tresnaren interfaze grafikoa hasieratuta egotea.

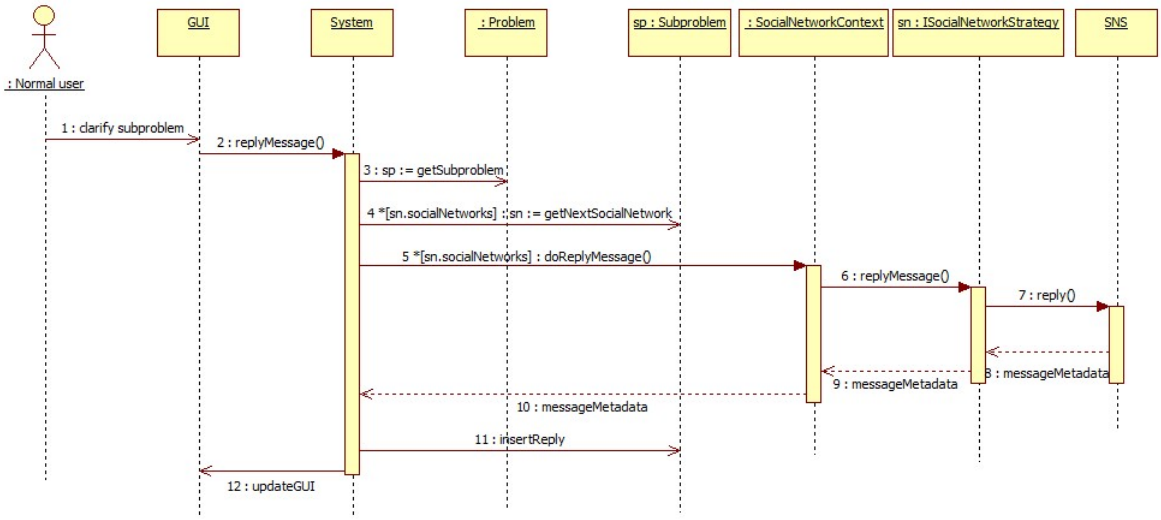
Azalpena: "Azpiarazo bat berriz definitu" erabilpen kasuari dagokion azpiatalean, hots, 4.3.4 azpiatalean, azpiarazo bat definitzerakoan egindako. adierazkortasun akatsei aurre egiteko CrowdCall tresnak erabiltzaileari bi aukera ematen dizkiola aipatzen da: azpiarazoa berriz definitzea edo eskaerari erantzunez azpiarazoa ebazteko argibideak ematea. Aipatutako azpiatal horretan lehenengo aukera lantzen da; oraingo honetan bigarrenari helduko zaio.

Erabilpen kasu hau laguntza eskaera erantzutean oinarritzen da. Horregatik, laguntza eskaerari egindako erantzunak erakusten dituen interfazeko pantailan mezuak idazteko formulario bat erakutsiko da. Bertan idatzitako mezuek laguntza eskaerari dagokion SNS plataformako mezua erantzungo dute, eta azpiarazoaren ebazpenean laguntzen duten kolaboratzaileek ikusi ahal izango dituzte, erabiltzaileak emandako argibideak proposatutako ebazpenak emateko orduan kontuan hartuz.

Sekuentzia diagrama: Erabilpen kasu honetako sekuentzia diagrama (ikus 4.14 irudia) *rephrase subproblem* erabilpen kasua aztertzerakoan azaldutakoaren oso antzekoa da: erabiltzaile arrunt batek interfaze grafikoaren bidez sistemari azpiarazo bat argitzeko agindua ematen dio. Honek, arazoaren azpiarazo zerrendatik azpiarazoa lortu, eta azpiarazoko *socialNetworks* zerrenda korritzen da. Ondoren, *subproblem2MicroblogPost* moduluko testuinguru klaseko *doReplyMessage* metodoa deitzen da, parametro hauekin:

- *socialNetworkName*: zerrendako elementua

Honek dagokion SNS plataformaren kudeaketa inplementatzen duen estrategia konkretua deitzen du, parametro hauek erabilita (*socialNetworkName* salbu), eta hau da SNS plataforman mezua erantzuteaz arduratzen dena. Ondoren, SNS plataformak itzulitako mezuaren metadatuak objektuz objektu pasatzen dira, azkenean azpiarazoaren *replyList* eremuan txertatu arte. Azkenik, interfaze grafikoa eguneratzen da idatzitako mezu berria pantailaratuz.



4.14 Irudia: *Clarify subproblem* erabilpen kasuaren sekuentzia diagrama.

- *message*: erabiltzaileak interfaze grafikoko formularioan idatzitako mezua
- *messageId*: azpiarazoak *MessageIdList* zerrendako SNS plataformaren izenarekin bat egiten duen *MessageId*aren *id* eremua (ikus 4.2)
- *userId*: azpiarazoak *MessageIdList* zerrendako SNS plataformaren izenarekin bat egiten duen *MessageId*aren *author* eremua (ikus 4.2)

4.3.6 Azpiarazo baten emaitzak laburtu

Izenburua: Azpiarazo baten emaitzak laburtu.

Aktore nagusia: Erabiltzaile arrunta.

Helburuak:

- Azpiarazoaren ebazpena lortzea.

- Arazoaren ebazpen prozesuan aurrera egitea.

Aurrebaldintzak:

- Arazoa sortuta egotea.
- CrowdCall tresnaren interfaze grafikoa hasieratuta egotea.
- Azpiarazoari dagokion eskaerari erantzunez gutxienez mezu bat egotea.
- Azpiarazoari dagokion eskaerari erantzuten dioten mezuen artean, gutxienez batek erabiltzaileak bilatzen duen datu motarekin bat egiten duen adierazpen bat barne hartzea.

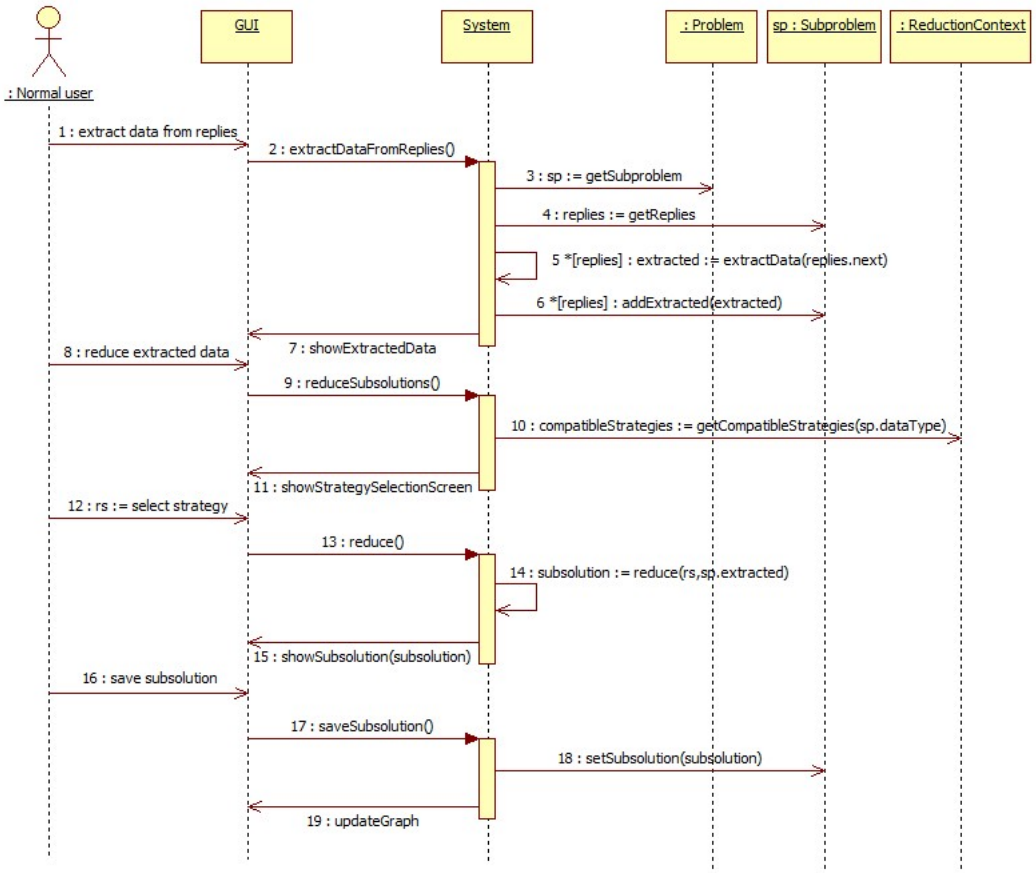
Azalpena: CrowdCall erabiltzaile batek egindako laguntza eskaerak kolaboratiboki ebaztean oinarritzen da. Beraz, kolaboratzaileek proposatutako ebazpen hauek tratatu eta emaitza bakar batera laburtzeko mekanismoa eskaini behar du. Lan honen ardura eskaera egiten duen erabiltzaileari dagokio, interfaze grafikoko grafoan zehaztuta dauden urratsak jarraituta.

Jasotako balizko ebazpenak bakarrera laburtzeko prozesuan DSL editoreak konfigurazio urratsean zehaztutako bi modulu kontuan hartuko dira, erantzunen barruan azpisoluzioak identifikatzeko orduan (reply2Subsolution modulua) eta azpisoluzioen aukeraketa egiteko orduan (subsolutions2Subsolution).

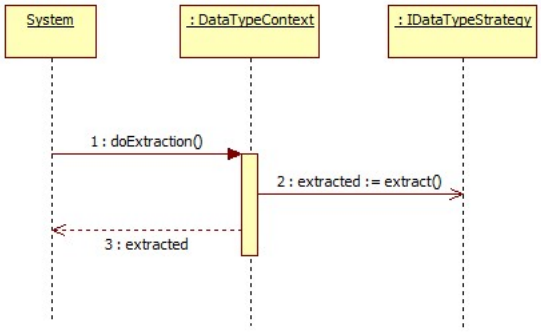
Azpisoluzioak laburtzeko prozesua arrakastaz burutu ahal izateko beharrezkoa da erabiltzaileak egindako eskaerari erantzunez gutxienez mezu bat egotea eta mezu hauen artean gutxienez azpiarazoan zehaztuta dagoen datu motarekin bat egiten duen adierazpen bat erauzi ahal izatea. Bestela, interfaze grafikoko grafoaren bidez prozesuarekin aurrera egitea galaraziko da.

Sekuentzia diagrama: 4.15 irudian *reduce subsolutions* erabilpen kasuaren sekuentzia diagrama azaltzen da. Bertan, azpisoluzioen erauzketa eta aukeraketa faseak sinplifikatuta azaltzen dira, diagrama hobeto ulertu ahal izateko.

Bertan ikus daitekeenez, erabiltzaile arruntak erantzunetatik azpiarazoan zehaztutako datu motarekin bat egiten duten adierazpenak erauzteko agindua ematen dio sistemari, interfaze grafikoko grafoko nodo baten bidez. Sistemak arazoan azpiarazoa bilatu



4.15 Irudia: *Reduce subsolutions* erabilpen kasuaren sekuentzia diagrama.



4.16 Irudia: *Reduce subsolutions* erabilpen kasuko erauzketa fasearen sekuentzia diagrama.

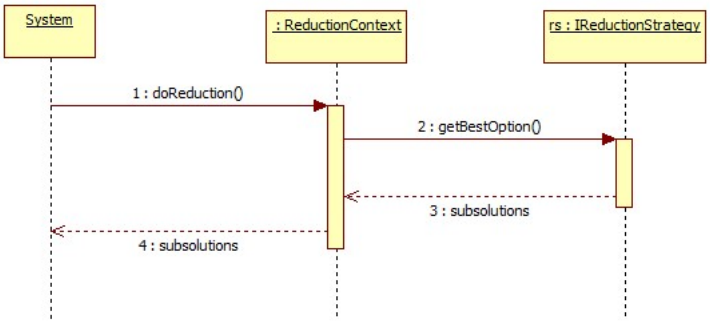
(*getSubproblem*), eta azpiarazotik erantzunak lortzen ditu (*getReplies*). Ondoren, erantzunak korritzen joaten da, eta bakoitzarentzat azpiarazoan zehaztutako datu motarekin bat egiten duten adierazpenak erauzi egiten dira (*extractData*). Hau egin eta gero, *extracted* aldagaian erauzitako adierazpenak daude, eta honen edukia azpiarazoan biltegiratzen da (*addExtracted*).

Hau egin eta gero, erabiltzaileari erauzitako adierazpenak erakusten zaizkio, eta bakar batera laburtzeko aukera ematen zaio. Interfaze grafikoaren bidez sistemari hau egiteko agintzerakoan, sistemak azpiarazoan adierazitako datu motarekin bateragarriak diren *subsolutions2subsolution* moduluko estrategia konkretuen zerrenda bat eskuratzen du (*getCompatibleStrategies*), eta erabiltzaileari erakusten dio interfaze grafikoaren bidez.

Une honetan erabiltzaileak erabili nahi duen estrategia aukeratuko du, eta sistemak erauzketa egin ondoren (*reduce*), erabiltzaileari estrategia hori aplikatzearen emaitza erakutsiko dio. Orduan, erabiltzaileak lortutako azpisoluzioa azkeneko azpisoluzio bezala baieztatu beharko du interfaze grafikoaren bidez. Hala eginez gero, sistemak azpisoluzioan biltegiratuko du (*setSubsolution*), eta interfaze grafikoko grafoa eguneratuko du (*updateGraph*).

Erauzketa faseari dagokionez, 4.16 irudiko sekuentzia diagraman azaltzen den moduan burutzen da. Sistemak *reply2Subsolution* moduluko *doExtraction* metodoa deitzen du, parametro bezala azpiarazoan adierazten den datu motaren izena, eta erantzun bat pasata. Honek datu motaren izen bera daukan estrategia konkretuaren *extract* metodoa exekutatuko du, emaitza gisa erauzitako adierazpenak itzuliko dituena.

Bestalde, aukeraketa faseari dagokionez, 4.17 irudiko sekuentzia diagraman azaltzen den moduan burutzen da. Sistemak *subsolutions2Subsolution* moduluko *doReduction* metodoa deitzen du, parametro bezala erabiltzaileak aukeratutako estrategiaren izena



4.17 Irudia: *Reduce subsolutions* erabilpen kasuko aukeraketa fasearen sekuentzia diagrama.

eta erantzunetatik erauzitako adierazpenak pasata. Honek erabiltzaileak aukeratutako estrategia konkretuaren *getBestOption* metodoa exekutatuko du, emaitza gisa azpisoluzio bakarra itzulita.

5. KAPITULUA

Ebazpenaren inplementazioa

Atal honetan ebazpenaren inplementazioaren nondik norakoak azalduko dira. Lehenengo atalean, inplementazioan erabilitako tresnen analisia egingo da. Ondoren, inplementazioaren ezaugarri nagusiak aipatuko dira. Honen ostean, tresnaren interfaze grafikoaren, modulu konfiguragarrien eta DSL editoreetan integratzeko mekanismoaren inplementazioari buruzko xehetasun batzuk azalduko dira.

Ebazpenaren inplementazioa - Egitura

1. Erabilitako tresnen analisia
2. Inplementazioaren ezaugarriak
3. Interfaze grafikoa
4. Modulu konfiguragarriak
5. Editoreetan integratzeko mekanismoa

5.1 Erabilitako tresnen analisia

CrowdCall tresnaren garapenean hiru tresnen erabilera nabarmendu daiteke:

- Greasemonkey
- JointJS
- RaphaëlJS

Greasemonkey tresnaren azpiegitura gisa erabili da, eta JointJS eta RaphaëlJS liburutegiak, aldiz, interfaze grafikoa garatu ahal izateko. Jarraian, tresna hauen analisi bat egingo da.

5.1.1 Greasemonkey

Greasemonkey Mozilla Firefox-erako gehigarri bat da, kode zati txikien instalazioaren bidez web orri jakinen portaera aldatzeko aukera (*augmented browsing* izenaz ere ezaguna) ematen duena.

Greasemonkey hainbat helburutarako erabil daiteke: orrien itxura pertsonalizatzeko, web orriei funtzionalitate berriak eransteko (adibidez, erosketa webguneetako salneurri konparaketak txertatzeko), errenderizazio erroreak konpontzeko, web orri desberdinetako datuak konbinatzeko, eta abar.

Gehigarri honek ez du aldaketarik egiten bere kabuz. Horretarako, beharrezkoa da *user scripts* izeneko kode zatiak instalatzea, JavaScript bidez ekintza oso zehatzak egikaritzeko. Bestalde, kode zatiaren exekuzioa nabigatzaileak karga egin aurretik zein ondotik burutu daiteke.

Greasemonkeyko *user scriptak* JavasCript-ez idatzita daude, eta DOM interfazearen bidez maneiatzen dute web orriko edukia. Orokorrean, scriptak web orri zehatz baterako edo domeinu zehatz baterako (domeinu zehatz bateko web orri guztietarako) idazten dira, domeinu guztietako web orriei aplikatzeko kode daitekeen arren.

Greasemonkey erabiltzaileei scriptak idatzi, deskargatu eta haien liburutegi pertsonalean gordetzeko aukera ematen zaie. Ondoren, erabiltzaileak bisitatutako web orri batek liburutegi pertsonaleko script batean zehaztutako helbidearekin bat egiten badu, Greasmonkeyk scripta deituko du.

Greasemonkeyko scriptek JavaScriptek eskaintzen duen edozein modutan eralda dezakete web orri bat, zenbait segurtasun murrizketarekin. Bestalde, scriptek beste web orri eta web-zerbitzuak atzi ditzakete domeinu-murrizketarik gabeko XMLHTTP eskaeraren bidez, jatorrizko web orriko edukian kanpo-edukia txertatzeko aukera emanez.

User scriptei user.js luzapena ematen zaie, eta Greasemonkeyk instalatzeko aukera ematen du nabigatzailean luzapen hori daukan URL bati eskaera bat egiten zaionean. Kodearekin batera, Greasemonkeyko scriptek metadatuak gordetzen dituzte: scriptaren izena, deskribapen bat, scriptak behar dituen baliabideak, URL izen-espazio bat izen bereko scriptak bereizteko eta scripta deitu behar den edo ez erabakitzeko URL patroiak.

Greasemonkey script bat idaztea web orrietan txertatzen daitekeen JavaScript kodea idaztearen antzekoa da, aurretik aipatutako domeinu-murrizketarik gabeko XMLHttpRequest eskaerak bezalako ezaugarri gehigarriak erabiltzeko aukerarekin. Firefox nabigatzailerako hedapen bat idaztearekin alderatuta, *user scriptak* idaztea web programaziorako konplexutasun maila baxuagoko modu bat da. Hala ere, modu honetan idatziriko scriptak mugatuta daude Mozillak inposatutako segurtasun murriztapenak direla eta. Adibidez, Greasemonkeyko scriptek ez dute Firefoxeko osagai asko atzitzeko baimenik: deskarga kudeatzailea, sarrera/irteera prozesuak, eta abar. Gainera, Greasemonkeyko scriptak bat datozen web orrien instantzia bakoitzeko martxan jartzen dira. Hori dela eta, zaila egiten da elementu listak globalki kudeatzea. Hala ere, garatzaileek cookieak eta Greasemonkeyk eskaintzen dituen APIak (`GM_setValue` eta `GM_getValue`, esaterako) erabiltzen dituzte arazo hau gainditzeko.

5.1.1 taulan Greasemonkey-ren abantaila eta desabantailak zerrendatzen dira.

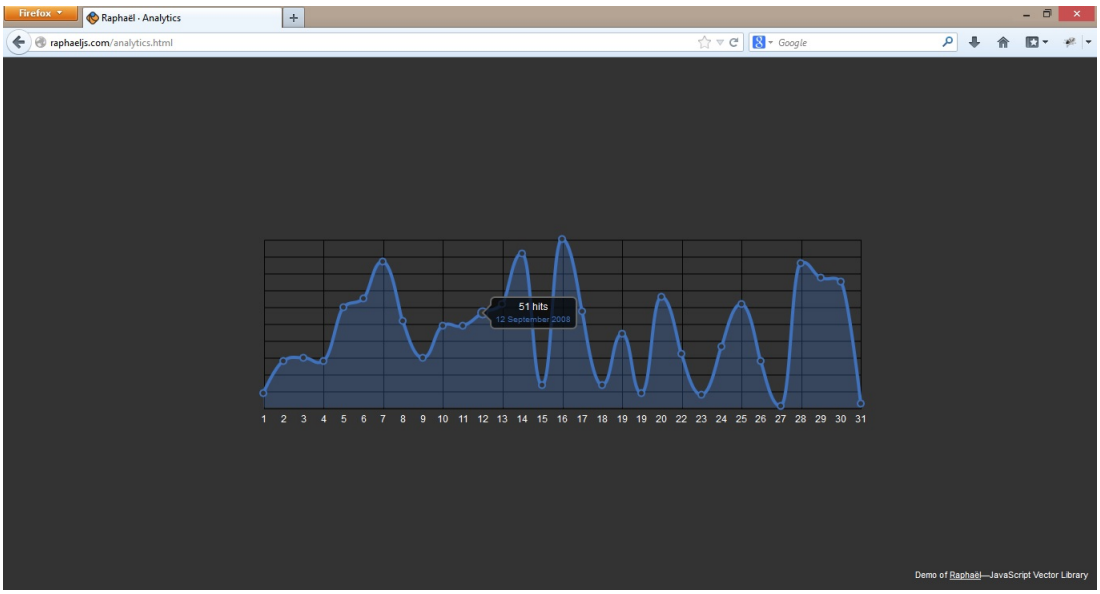
5.1.2 RaphaelJS

Raphaël webean irudiekin lan egiten laguntzen duen JavaScript liburutegi txiki bat da. Horretarako, liburutegiak alde batetik W3Cren SVGren inguruko gomendioak erabiltzen ditu, eta bestetik, VML oinarritzat hartzen du irudiak sortzeko. Honek esan nahi du sortutako irudi bakoitzak aldi berean DOM objektu izaera ere daukala, eta honen ondorioz, JavaScript gertaera maneiatzaileak esleitzen ahal zaizkio edo ondorenean eraldatu. Liburutegiaren helburua irudi bektorialen marrazketarako egokigailu erabilerraza eta nabigatzaile desberdinekin bateragarria egitea da.

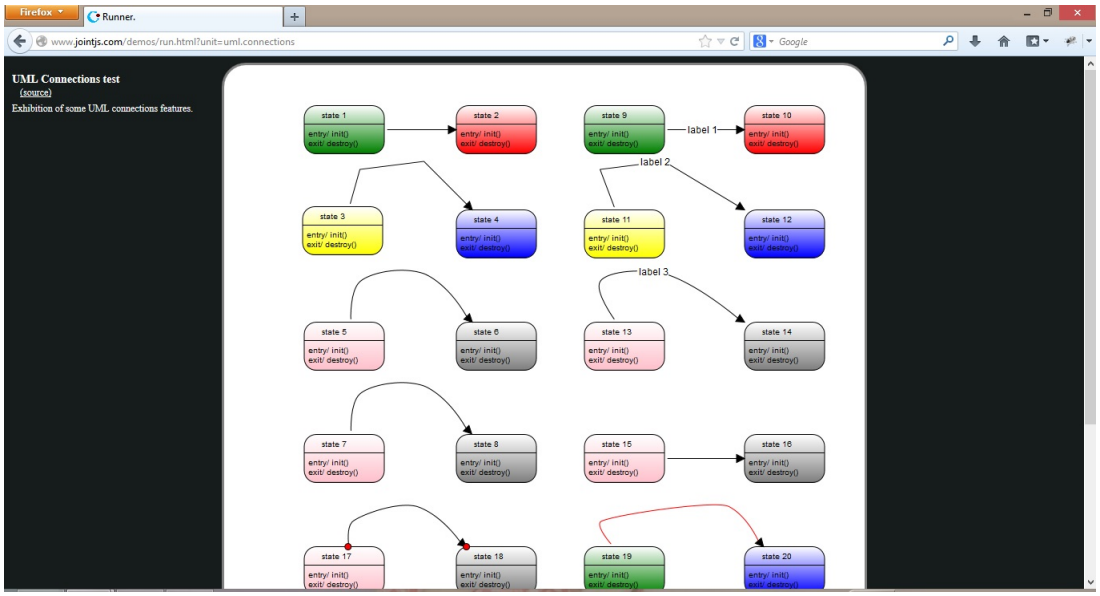
5.1.2 irudian liburutegi honekin garatutako aplikazio baten adibidea ikus daiteke.

Abantailak	Desabantailak
+ Edozein web orri eraldatzeko aukera. + Milaka script eskuragarri. + Scriptak instalatzeko erraztasuna. + Script bat erraz desgaitu daiteke Greasemonkey-ren interfazea erabilita + Scriptak idazten ikasteko laguntza + Komunitate handi bat atzean	- Scripten garapena ez dago guztiz araututa; ondorioz, segurtasun akatsak arazo bat bihur daitezke. - Web orri bererako script bat baino gehiago izateak batzuetan gatazkak sortzen ditu

5.1 Taula: Greasemonkey-ren abantaila eta desabantailak



5.1 Irudia: RaphaëlJS liburutegiarekin garatutako aplikazio baten adibidea



5.2 Irudia: JointJS liburutegiarekin garatutako aplikazio baten adibidea

5.1.3 JointJS

JointJS diagramak sortzeko JavaScript liburutegi bat da. Liburutegi honekin sortutako diagramak erabat interaktiboak izan daitezke. Joint liburutegia egokia da diagramak sortzeko tresna bat garatzeko zein diagramak argitaratzeko, 5.1.3 irudiko aplikazioak erakusten duen moduan.

Honakoak dira liburutegiak eskaintzen dituen ezaugarri nagusiak:

- Objektuak mota desberdinetako geziak erabilita haien artean lotzeko aukera
- Objektu eta loturekin elkar eragiteko aukera
- Zenbait gertaeretarako maneiatzaile pertsonalizatuak definitzeko aukera
- Lerro kurbatu leunak
- Diagrama ezagunetako (ERD, Organizational chart, FSA, UML, PN, DEVS, LDM) elementuak erabiltzeko prest
- Diagrama hierarkikoak
- Serializazioa (JSON formatutik/JSON formatura, SVG esportazioa bakarrik nabigatzaile bateragarrietan)

- Hedagarritasuna
- Pertsonalizagarritasuna

5.2 Inplementazioaren ezaugarriak

CrowdCall-en inplementazioa JavaScript programazio lengoaia erabiliz egin da. Lengoaia hau interpretatutakoa da, eta batez ere bezeroaren aldean erabiltzen da, nabigatzailearekin batera.

Programazio lengoaia hau Mozilla Firefox-en Greasemonkey *plugin*arekin konbinatu da. *Plugin* honek ez du ezer bere aldetik egiten, baina JavaScript-ez idatzitako *script*ak instalatzerakoan edozein web orritako itxura eta funtzionamendua alda daiteke. Hau da CrowdCall-ekin bilatzen dena: nabigatzailean exekutatzen diren DSL editoreen itxura zein funtzionamendua aldatzea garatutako tresna txertatu ahal izateko. Beraz, tresna *userscript* bat izango da.

Bestalde, JavaScript programazio lengoaiak "jatorri bereko politika" jarraitzen du. Politika hau gaur egungo nabigatzaileen segurtasun neurririk garrantzitsuenetariko bat da, eta *script*ek bere jatorria ez duten baliabideak atzitzeko duten gaitasuna mugatzen du. Modu honetan, tresnak ezin izango luke SNS aplikazioekin komunikatu, eta tresnak ez luke zentzurik izango. Baina Greasemonkey-ren APIak eskainitako *GM_xmlhttpRequest* metodoari esker muga hau gainditzea ahalbidetzen du, beraz, tresnaren funtzionamendurako funtsezkoa da.

JavaScript erabiliz aplikazio konplexuak garatzen direnean, araztaile bat erabiltzea komenigarria da. Kasu honetan, Mozilla Firefox-erako Firebug *plugin*a erabili da helburu honetarako. Araztatze lanetarako erabiltzeaz gain, beste helburuetarako ere baliagarria suertatu da: DOMa ikuskatzeko eta sareko trafikoa ikusteko, batez ere.

5.3 Interfaze grafikoa

Tresnaren interfaze grafikoaren garapenean bi zati desberdin bereiz daitezke. Alde batetik, erabiltzailea arazoaren ebazpen prozesuan zehar gidatzeko grafoa dago. Atal hau egonkor mantenduko da arazoaren ebazpen prozesu guztian; aldatuko dena grafoaren egoera izango da. Atal hau garatzeko JointJS eta RaphaëlJS liburutegiak erabili dira. Bestetik,

interfaze grafikoaren gainerako osagaiak inplementatzeko tresna txertatzen den web orriko DOMa eta CSSa zuzenean eraldatu dira. Beste atal hau ebazpen egoeraren arabera aldatuko da.

Grafoari dagokionez, aipatutako bi liburutegietan definitutako APIak erabilita ezaugarri hauek inplementatu dira:

- Nodoen koloreen kudeaketa. Azpiarazoei dagozkion grafoko nodoek lau koloretakoak izan daitezke: gorriak, berdeak, horiak edo grisak, azpiarazoaren ebazpen prozesuaren egoera islatuta.
- Nodoen arteko konektoreen kolorearen kudeaketa. Hasieran konektoreak urdinez margotuta daude eta azpiarazoaren ebazpen prozesua aurrera doan heinean, konektoreak berdez margotzen joaten dira.

Inplementatutako ezaugarri hauen helburua erabiltzaileek grafoak adierazten duen prozesua ahalik eta hobekien jarraitzea da.

Pantailaren gainerako osagaiei dagokienez, zenbait "pantailaren" beharra identifikatu da, erabiltzaileari datuak erakusteko eta honek sartutako datuak jasotzeko. Hauek dira inplementatutako "pantaila" horiek:

- Arazoa osatzen duten azpiarazoen laburpena erakusteko pantaila.
- Azpiarazo jakin bati dagokion laguntza eskaera erakusteko pantaila.
- Azpiarazo jakin bati dagokion laguntza eskaera eta erantzunak kudeatzeko pantaila.
- Azpiarazo jakin bati dagokion laguntza eskaera bati egindako erantzunak erakusteko pantaila.
- Azpiarazo jakin bat berriz planteatzeko pantaila.
- Laguntza eskaeretatik erauzitako datuak bistaratzeko pantaila.
- Azpisoluzioen laburketarako erabiliko den estrategiaren aukeraketarako pantaila.
- Laburketa estrategiaren emaitza erakusteko pantaila.
- Azpiarazo bati dagokion azpisoluzioa bistaratzeko pantaila.
- Arazoari dagokion soluzioa bistaratzeko pantaila.

5.4 Modulu konfiguragarriak

Bost dira tresnak dituen modulu konfiguragarriak. Hauen inplementazioa bi modu desberdinetan burutu da: alde batetik *DSLexpression2Problem* moduluaren inplementazioa dago, eta bestetik, gainerako lauena (*subproblem2Context*, *subproblem2MicroblogPost*, *reply2subsolution* eta *subsolutions2subsolution*). Bereizketa honen arrazoia moduluen izaerari esker azaldu daiteke.

DSLexpression2Problem moduluaren kasuan CrowdCall tresna konfiguratzen duen DSL editoreak bakarrik berak bidalitako adierazpenetatik abiatuta *Problem* bat sortzeko metodologia adierazten du, eta hau bakarra izango da arazoaren ebazpen prozesu osoan. JavaScript programazio lengoaiak funtzio deietan funtzioak argumentu moduan bidaltzea eta pasatako funtzioak exekutatzea ahalbidetzen du, beraz, egokiena metodologia hori funtzio baten bidez pasatzea da.

Beste lau moduluei dagokienez, aldiz, konfigurazio urratsean bakoitzerako gutxienez estrategia bat adieraziko du DSL editoreak. Orduan, kasu honetan egitura konplexuago bat behar da, estrategia desberdin hauek kudeatu ahal izateko. Horregatik, testuinguru klase bat sortu da hauetako modulu bakoitzerako, estrategia guztiak bilduko dituena. Bestalde, estrategiak identifikatu ahal izateko izen bat edukiko dute, *getName* funtzioaren bidez itzuliko dutena. Horrela, modulu bateko estrategia jakin bat exekutatu nahi denean, moduluaren testuinguru klaseko estrategia zerrendan izen hori daukana bilatuko da eta honek definitutako algoritmoa aplikatuko da.

subsolutions2subsolution moduluaren kasuan, testuinguru klasean funtzio lagungarri bat inplementatu da, datu mota baten izena pasata datu mota horrekin bateragarriak diren laburtze estrategien izenak itzultzen dituena. Funtzio hau beharrezkoa da erabiltzaileari kasu bakoitzean erabili ditzakeen estrategien zerrenda erakutsi ahal izateko.

5.5 Editoreetan integratzeko mekanismoa

CrowdCall editoreetan integratzeko mekanismoari buruz hitz egiterakoan, tresnaren interfaze grafikoa DSL editoreetan txertatzea eta editoreek tresnaren funtzionalitateak erabili ahal izateko inplementatutako mekanismoa bereizi behar dira.

Interfaze grafikoari dagokionez, modurik sinpleena aurrera eraman da. DSL editoreak DOM elementu bat erreserbatu beharko du bertan CrowdCallen interfazea txertatze-

ko, eta *id* bat esleitu beharko dio. Ondoren, CrowdCall-i *id* hori pasako dio, CrowdCall-ek izango dituen altuera eta zabaleraz gain. CrowdCall gainerakoaz arduratuko da: elementua DOM zuhaitzean bilatu eta horren barruan hasieratuko du interfaze grafikoa.

Editoreak tresnaren funtzionalitateak erabili ahal izateko, API bat definitu da, metodo hauekin:

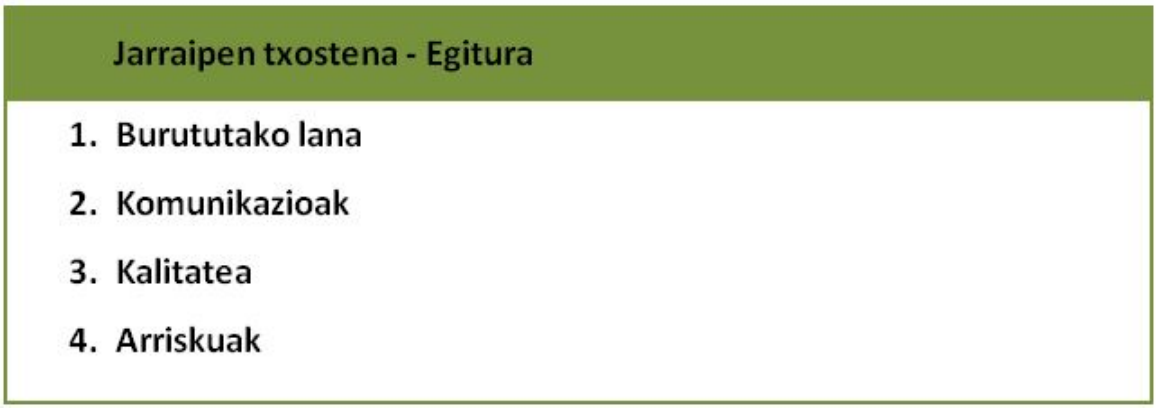
- *getProblems*
- *deleteProblem*
- *getProblemsSolution*
- *createProblem*
- *configureCrowdCall*

Metodo hauekin nahikoa da DSL editoreek arazoen kudeaketa egin ahal izateko, arazoak sortu, eskuratu eta ezabatzeko aukera ematen baitute. Gainera, *configureCrowdCall* metodoaren bidez tresna editorearen testuingurura egokitzeko aukera ematen da.

6. KAPITULUA

Jarraipen txostena

Kapitulu honetan proiektuaren jarraipen txostena deskribatzen da. Honetan, lehendabizi proiektuan zehar burututako lana planifikatutakoarekin alderatzen da. Ondoren, komunikazioak, kalitatea eta arriskuen kudeaketaren inguruan egindakoa deskribatzen da, hauetan ere planifikatutakoarekin alderatuta.



6.1 Burututako lana

Produktuaren iraupen totala 304 ordutakoa izan da. Hasieran planifikatutako 300 orduekin alderatuta desbideratze txikia egon da, % 1.4koa, hain zuzen.

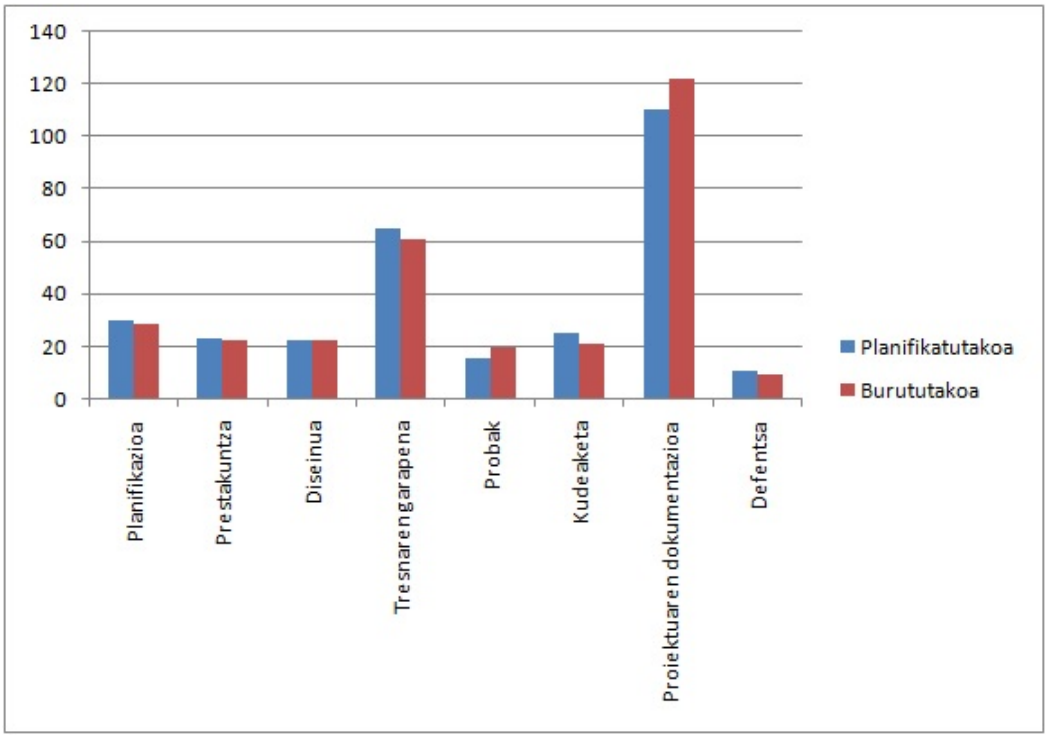
Jarraian datorren taulan arlo bakoitzari dedikatutako ordu kopurua planifikatutakoarekin alderatzen da, egondako desbideratzea adieraziz:

Arloa	Planifikatutakoa orduak	Dedikatutakoa	Desbideratzea
Planifikazioa	30	28	% -6,7
Prestakuntza	23	22	% -4,3
Diseinua	22	22	% 0
Tresnaren garapena	65	61	% -6,2
Probak	15	19	% 26,7
Kudeaketa	25	21	% -16
Proiektuaren dokumentazioa	110	122	% 10,9
Defentsa	10	9	% -10

6.1 Taula: Proiektuaren arlo bakoitzari dedikatutako ordu kopurua.

6.1 irudian taulako datuak diagrama batean biltzen dira. Ikus daitekeenez, proiektu osoaren desbideratzea % 1.4koa izan bada ere, arloetako desbideratzeak nahiko handiak izan dira. Horrela, lau arloetako desbideratzea % 10aren berdina edo altuagoa izan da: *probak*, *kudeaketa*, *proiektuaren dokumentazioa* eta *defentsa*. Defentsaren prestakuntzaren kasua alde batera utziko da, % 10eko desbideratzea izan arren, aldea bakarrik ordu batekoa izan delako. Hiruetatik esanguratsuena proben inguruko arloa da, % -26.7ko desbideratzearekin. Horren arrazoia honakoa da: garatutako modulu bakoitzerako probak egin behar izan dira, hauen funtzionamendu egokia bermatzeko. Kudeaketaren kasuan, aldiz, % 16ko desbideratzea lortu da, bakarrik 10 bilera egin direlako, hasieran aurreikusitakoa baino gutxiago. Bestalde, proiektuaren dokumentazioaren kasuan, ordu gehiago dedikatu dira memoriaren elaborazioan, sartutako ordu horiek dokumentuaren kalitatean islatzeko asmoz.

Azkenik, 6.2 irudian ataza bakoitzari dedikatutako ordu kopurua adierazten da, berriz ere planifikatutakoarekin alderatuta. Diagramak islatzen duenez, oso ataza gutxi planifikatutako denbora berean burutu dira; gehienak desbideratze txikiak jasan dituzte.



6.1 Irudia: Arlo bakoitzeko atazei dedikatutako ordu kopurua planifikatutakoarekin alderatuta.

6.2 Komunikazioak

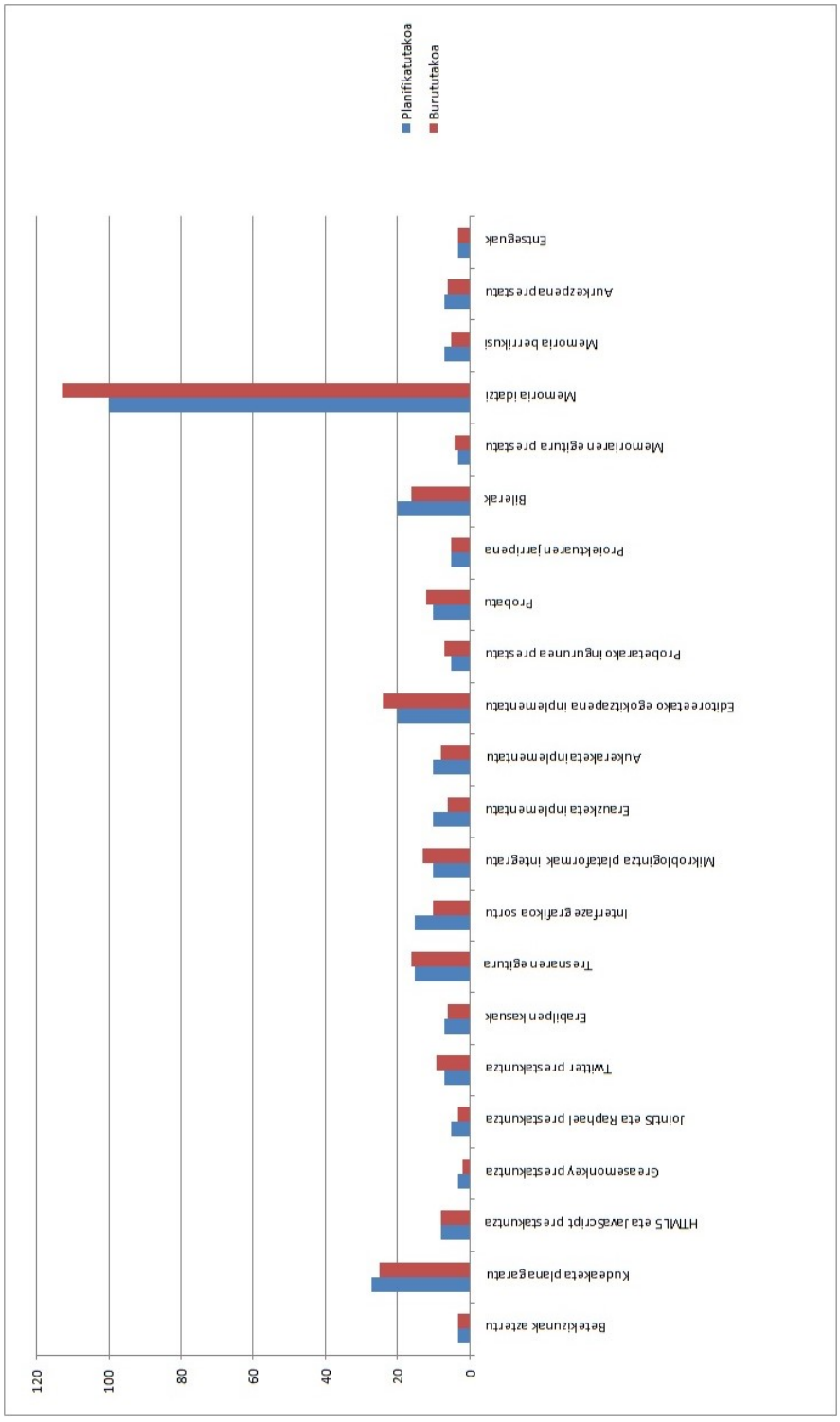
Komunikazioen kudeaketa proiektuaren puntu hobegarrietako bat izan da, kudeaketa planean adierazitako ideia batzuk bete ez direlako.

Proiektuko interesatuen arteko mezu trukeari dagokionez, planean zehaztutako posta elektronikoak erabiliz gauzatu da, eta zentzu honetan ez da inolako arazorik egon.

Bileren kasuan, egindako planifikazioan bilerak noizean behin egingo zirela adierazten zen, baina ez zen periodikotasunik zehazten bilerak ez emankorrak saihesteko. Hemen aipatutakoa bete arren, bilera gehiago egitea espero zen.

Bilera deialdiak egiteko protokoloari dagokionez, planean adierazitakoa jarraitu da: bilerako parte-hartzaileen artean bileren data eta ordua posta elektronikoz adostu da. Mezu-truke honetan bileran landuko ziren gaiak ere aipatu dira.

Azkenik, Wiki-aren kudeaketari dagokionez, proiektuaren hasieran gaurkotuta mantendu da. Baina proiektua aurrera joan ahala honen kudeaketa alde batera uzten joan da.



6.2 Irudia: Ataza bakoitzari dedikatutako ordu kopurua planifikatutakoarekin alderatuta.

6.3 Kalitatea

Kalitateari dagokionez, kudeaketa planean identifikatutako kalitate elementuek proiektuko atal desberdinak hobetzen lagundu dute. Hona hemen proiektuko atal desberdinen kalitate mailaren neurketa, plan horretan identifikatutako elementuak erabilita.

6.3.1 Proiektuaren kudeaketa

Proiektuaren kudeaketaren kalitate-maila neurtzeko identifikatutako elementu bakarra identifikatu zen: egindako bilera kopurua. Planaren ezarritakoaren arabera, kudeaketaren kalitate-maila onargarria izateko minimoa 10 bileratan ezarri zen, proiektuan egindako bilera kopurua, hain zuzen. Ondorioz, elementu honi dagokionez, proiektuaren kudeaketan onargarria den kalitate-maila minimoa lortu da, lortu nahi zen baliotik (15-25 bilera) urrun.

6.3.2 Garatutako produktua

Garatutako produktuari dagokionez, bi kalitate-elementu identifikatu ziren: tresna editoreetan bateratzea eta mikroblogintza plataformak tresnan bateratzea.

Tresna editoreetan bateratzeko gaitasunari dagokionez, minimo onargarria bi editoretan ezarri zen eta lortu nahi zen neurria, bi baino gehiagotan. Proiektuan zehar tresnak bi editoreekin lan egitea lortu da (Sticklet eta Google-ren kalkulu orrien editorea). Beraz, honi dagokionez kalitate-maila minimoa lortu da.

Modu berean, tresnan integratutako mikroblogintza plataforma kopurua beste adierazle bezala hartu zen. Kasu honetan ere minimo onargarria lortu da: mikroblogintza sare bakarra (Twitter).

6.3.3 Proiektuaren memoria

Proiektuaren memoriaren kasuan, bi ziren identifikatutako kalitate-elementuak: dokumentuaren luzera minimoa eta kapituluen atal kopuru minimoa.

Dokumentuaren luzera minimoari dagokionez, minimo onargarria (70 orrialde) erraz gainditzea lortu da, eta ez hori bakarrik: lortu nahi zen balio-tartearen (90-120 orrialde) barruan egotea lortu da. Beraz, dokumentuaren luzera kalitate onekoa dela esan daiteke.

Bestalde, memoriaren kalitate-maila neurtzeko kapituluen atal kopuru minimoa neurtzea erabaki zen. Kapitulu batek izan beharreko atal kopuru minimoa 3tan finkatu zen, eta dokumentuko kapitulu guztiek baldintza hau betetzen dute. Hala ere, kapitulu gehienak 3 ataletan geratzen dira, eta ez dira lortu nahi ziren 4-6 ataletaraino iristen. Dena den, kalitate-elementu honek memoriaren elaborazioan izandako garrantzia oso handia izan da, dokumentuaren egituraren sorreran eragin handia izan duelako.

6.3.4 Proiektuaren defentsarako gardenkiak

Proiektuaren defentsarako gardenkien kalitate-maila neurtzeko elementu bakarra identifikatu zen: gardenki bakoitzaren ideia nagusi kopurua. Honekin gardenki bakoitzak gehienez ere 3 ideia nagusi (eta ahal zen heinean, bakarra) izatea bilatzen zen. Prestatutako gardenkiek minimo onargarria betetzen dute, baina oso gutxitan lortu da ideia bakarra egotea. Ondorioz, kalitate-maila egokia lortu dela esan daiteke.

6.4 Arriskuak

Proiektuan egindako arriskuen kudeaketari esker, proiektua arazorik gabe aurrera eramatea lortu da. Hala ere, ezin izan dira arrisku guztiak saihestu.

Proiektuan eraginik handiena izan zezakeen arriskua informazioaren galera zen. Hau ekiditeko, arriskuen kudeaketa planean definitutako estrategia aurrera eraman da. Honi esker, arrisku hau guztiz saihestu da, eta zentzu honetan ez da inolako arazorik izan.

Ordenagailua hondatzeko arriskuaren inguruan, ezin izan da ezer egin arriskua ekiditeko: ordenagailua matxuratu zen bateriaren arazo bat dela eta. Konpontze lanen iraupena dela eta (hilabete eta erdi), planean arriskuaren eragina gutxitzeko definitu zen estrategia aplikatu behar izan zen. Horrela, fakultateko eta etxeko ordenagailuak erabili ziren proiektuarekin aurrera jarraitzeko.

Proiektuaren defentsaren atzerapenaren kasuan eta atazen estimazio okerraren kasuan, ez da arazorik eduki, nahiz eta ataza batzuetarako estimatutako iraupena eta dedikatutako denboraren artean alde txiki batzuk egon.

Azkenik, beste konpainiek eskainitako zerbitzuek funtzionatzeari uzteko arriskuari dagokionez, ez da arriskua saihesteko definitu zen estrategia guztiz aplikatu. Arriskuen kudeaketa planean adierazten zenez, ahal zen heinean zerbitzu bakar baten menpe gera-

tzea saihestu behar zen, baina denbora kontuak direla eta, mikroblogintza plataforma bakarra kudeatzeko modua inplementatu da: Twitter. Honek ondorio negatiboak izan ditu: plataformaren APIak jasandako birmoldaketaren ondorioz, funtzio batzuk berriz inplementatu behar izan dira. Bestalde, kudeaketa planean definitutako estrategia berean bakarrik konpainia fidagarrietako zerbitzuak erabiliko zirela erabaki zen. Twitter fidagarritzat hartzea erabaki zen, eta honek eskainitako zerbitzuak etenik jasan ez dituztela ikusita, erabakia ongi hartu zela ondoriozta daiteke.

7. KAPITULUA

Ondorioak

Kapitulu honetan hainbat puntu landuko dira. Lehendabizi, tresna probatzeko Onekin taldeak egindako saiakera bat eta saiakeratik ondorioztatutakoa azalduko da. Ondoren, proiektuaren balorazio orokorra egingo da, lortutako emaitzei buruzko hausnarketak eginez. Honen ostean, proiektu honetan zehar ikasitako lezioak azalduko dira, etorkizunerako baliagarriak izan daitezkeenak. Bukatzeko, gai honen inguruan etorkizunean egin beharreko lana deskribatzen da.

Ondorioak - Egitura

1. Saiakera
2. Ondorioak
3. Ikasitako lezioak
4. Etorkizunerako lan-lerroak

7.1 Saiakera

UPV/EHUko Onekin ikerketa-taldeak saiakuntza bat aurrera eraman zuen CrowdCall-en erabileraren inguruko ondorioak ateratzeko helburuarekin. Atal hau saiakuntza horietaz arituko da.

Aipatutako esperimentua CrowdCall Stickleten barruan erabiltzean zetzan: subjektuei, irakaskuntza asmoak direla eta, Wikipediako XMLri buruzko artikulua beste webgunetako edukiarekin areagotzeko scriptak idaztea eskatu zitzaien. Esperimentuaren subjektu gisa DSLa erabiltzearekin ohituta zeuden lau irakasle aukeratu ziren. Lana errazteko, CrowdCall-en interfazeari buruzko 30 minututako azalpen bat eman zitzaien. Subjektuek CrowdCall erabili behar zuten Twitteren zituzten jarraitzaileei laguntza eskatzeko tresna gisa. Horrela, tresna hau erabili zuten XPath adierazpenen inguruan zituzten arazoak konpontzeko, eta, batez ere, XMLri buruzko webgune interesgarriak eskatzeko. Hurrengo paragrafoetan saiakuntzaren emaitzak azalduko dira.

Alde batetik, subjektuak CrowdCall-en interfazearekin elkarreragiteko eta fluxua jarraitzeko gai izan ziren. Sagua erabiltzea tweetak atzitzeko eta ebazpenaren fluxua jarraitzeko modu naturaltzat hartu zuten. Interfazea bera aipatzeaz gain, bi subjektuek mikroblogintza DSL editorearen barruan txertatzeak zentratzea eta trazabilitatea hobetzen zituela jakinarazi zuten. Aldi berean, subjektu guztientzat tresnaren abantailarik garrantzitsuena aurrezten zen denbora zen. Barneratutako mikroblogintzak mezu bat igortzea klik batera jartzen du.

Bestalde, kolaboratzaileak oso urriak izan ziren: ataza bakoitzeko batez beste 3.5 egon ziren. Balio baxu hauek subjektuek aukeratutako SNS plataforman zuten ikusgarritasun baxuaren (5 jarraitzaile batez beste) ondorioa izan zitekeen. Gainera, atazek domeinuari buruzko jakintza eskatzen dutenez (adibidez, XMLri buruzko webguneak ezagutzea), praktikarako komunitateak baliatzea gomendatzen da eskaera irekiak egitea baino (crowdsourcingean egiten den moduan). Honek, ordea, ekarpenak egiteko sustagarrien ideia planteatzen du. Domeinuko adituek oso lanpetuta ibiltzen dira (denbora gabezia), non diruak ez baitu rol garrantzitsu bat jokatzen, crowdsourcingean bezala. Autoretza eta elkarrekikotasuna sustagarri nagusi bezala azaleratzen dira. Hau oraindik ikusteke dago.

7.2 Ondorioak

Proiektu honetan lan asko egin da, eta lanaren fruituak ikusita ongi egin dela esan daiteke. DSL editore askotan bateratu daiteken tresna sortzea lortu da, proiektuaren hasieran ezarritako helburu garrantzitsuetako bat. Honek garrantzi handia dauka gaur egun, era askotariko editoreak aurki daitezkeelako.

Hala ere, helburu nagusia tresna erabiltzaileentzat baliagarria izatea da, eta egindako saiakera erreferentzia bezala hartuta, helburu hau lortu dela esan daiteke. Orokorrean, tresnarekin egindako saiakera horretatik ondorio positiboak atera ziren: tresnak arazoen ebazpen prozesuan denbora aurrezteko balio zuen. Hala ere, ez ziren kolaborazio-maila oso itxaropentsuak lortu, baina esan beharra dago saiakerako subjektuek saiakeran erabili zen sare sozialean zeukaten ikusgarritasuna nahiko kaskarra zela. Bestalde, beste ikuspuntu batetik begiratuta, tresnaz harro egoteko arrazoirik dago: tresnak erabiltzaileei eskaintzen dizkien aukerak oso zabalak eta era askotarikoak dira.

Gainera, DSL editoreen garatzaileen ikuspuntutik begiratuta, mikroblogintza sareak arazoen ebazpenerako erabiltzea oso erakargarria izan daiteke. Alde batetik, sare sozialek duten ikaragarrizko oihartzuna dela eta, editoreak ezagutzera emateko modu eraginkorra izan daiteke. Bestetik, editoreetako erabiltzaileek positiboki baloratuko lukete laguntza eskatzeko modu berri hau, gaur egun laguntza foroetan burutu beharreko bilaketa oso astuna izan daitekeelako.

Bestalde, proiektuaren ildo nagusia gaur egun pil-pilean dagoen gai bat da. 2.0 webaren ekarpen handienetako bat erabiltzaileen parte-hartzea izan da. Gaur egun sare sozialen erabilera-maila orain dela urte batzuk pentsaezina zen, eta hauek etengabe hazten ari direla kontuan hartuta, etorkizun oparo bat izango dutela aurreikus daiteke. Arrakasta honen arrazoietako bat sare sozialei ematen zaien erabilera-sorta zabala da. Horretan oinarrituta, ez da erokeri bat sare sozialek daukaten boterea programazioan sortzen diren arazoak ebazteko erabili daitekeela pentsatzea.

Formazioari dagokionez, ezaugarri hauetako proiektu bat garatzea nire etorkizun profesionalerako baliagarria izango delakoan nago, barneratutako ezagutza tekniko guztiak zein proiektuen kudeaketaren arloan hartutako eskarmentua dela eta. Eskala altuko proiektu bat egiten dudan lehenengo aldia denez, normala den bezala akatsak eta hobetzeko puntuak egon dira, baina ziur nago akats hauek ere etorkizunerako probetxugarriak izango direla.

Oro har, beraz, proiektuaren emaitzak positiboak izan dira, arlo desberdinetan: nire

formazioan aurrerapauso izugarria emateaz gain, komunitatearentzat erabilgarria izan daitekeen tresna bat garatu da, eta gainera, erabiltzaileek elkarri laguntzea sustatzen duena, gaur egungo gizartean geroz eta gutxiago ikusten den ohitura.

7.3 Ikasitako lezioak

Proiektu honetatik etorkizunerako baliagarriak izan daitezkeen lezio batzuk ikasi dira, jarraian zerrendatuko direnak.

- Proiektuen kudeaketan, sarritan saihestu ezin diren arriskuak identifikatzen dira. Saihestu ezin daitezkeenez, ezin da estrategiarik definitu hauek ekiditeko. Kasu hauetan, arriskua gauzatzen bada honen eragina txikiagotzeko estrategia bat definitu beharko da.
- Dokumentu bat idazten hasi baino lehen honek izango duen egitura ongi finkatzeak asko laguntzen du idazketa prozesuan, atal bakoitzean azaldu beharrekoa argi edukitzen delako. Horrela, egitura finkatzen ematen den ordu kopuruak eragin nabarmena izango du dokumentuaren idazketari dedikatutako zaizkion ordu kopuruan.
- API bat definitzearen zati oso garrantzitsu bat metodo bakoitzaren deskribapen sakona egitea da, sarrera/irteera parametroak ongi azalduta. Hau ezinbestekoa da erabilia izatea bilatzen bada. Bestalde, azalpenen aurkezpena ere zaindu beharreko puntua da, aurkezpen on batek erabiltzeko erraztasun sentsazioa transmititzen duelako.
- Zerbitzu-hornitzaile batek eskainitako zerbitzu bat erabiltzeak (API bat, esate baterako) zerbitzu-hornitzaile horren menpe egotea esan nahi du. Ezin da suposatutzat eman zerbitzua bere horretan betirako mantenduko dela, zerbitzuaren edozein aldaketak gure aplikazioan eragin zuzena suposatzen duelako. Beraz, eragin hau minimizatzeko periodikoki zerbitzuaren egoeraz arduratzea komeni da, eta ahal den heinean, beste aukerak maneiatzea.

7.4 Etorkizuneko lan-lerroak

Etorkizunari begira, CrowdCall hobetzeko lerroa jarraitu beharko litzateke. Horretarako, lehentasunik handieneko betebeharra tresna beste SNS plataformetara zabaltzea izango

litzateke, momentuz bakarrik Twitter-ekin probatu delako.

CrowdCall hobetzeko ildo honetatik jarraituta, interfaze grafikoa hobetzea komenigarria izan beharko litzateke, atal honi proiektuan zehar garrantzi handirik eman ez zaiolako.

Bestalde, tresna editoreetan integratzeko mekanismoa hobetu beharko litzateke. Horretarako, beharrezkoa da editore gehiagotan integratzen saiatzea, mekanismoaren mugak identifikatu ahal izateko.

Azkenik, saiakera gehiago egitea interesgarria izango litzateke, balizko erabiltzaileen feedback-ak tresnaren garapenean hartu beharreko norabidea erabakitzen lagunduko lukeelako.

8. KAPITULUA

Bibliografia

Artikuluak

Built-in Crowdsourcing into Domain-Specific Languages
Díaz, O. eta Arellano, C., 2012.

Integrating Microblogging into Domain Specific Language Editors
Díaz, O. eta Arellano, C., 2013.

Beste argitalpenak

Domain-Specific Languages
Fowler, M., 2010.

Webguneak

Onekin Taldea (UPV/EHU Euskal Herriko Unibertsitatea)
http://onekin.org

Crowdsourcing (Wikipedia: The Free Encyclopedia)
http://en.wikipedia.org/wiki /Crowdsourcing

Social networking services (Wikipedia: The Free Encyclopedia)
http://en.wikipedia.org/wiki/Social_networking_service

Sticklet editorea
http://sticklet.org

JointJS
http://www.jointjs.com

RaphaëlJS
http://raphaeljs.com

Twitter Rest API v1.1
https://dev.twitter.com/docs/api/1.1

Eranskinak

A. ERANSKINA

Akronimoak

CSS: Cascading Style Sheets

DOM: Document Object Model

DSL: Domain Specific Language

GPL: General Programming Language

GUI: Graphical User Interface

IDE: Integrated Development Environment

LDE: Lanaren Deskonposaketa Eskema

SNS: Social Networking Service

SVG: Scalable Vector Graphics

UML: Unified Modeling Language

URL: Uniform Resource Locator

VML: Vector Markup Language

W3C: World Wide Web Consortium

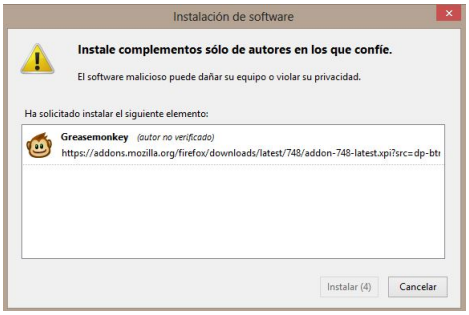
B. ERANSKINA

CrowdCall instalatzeko eskuliburua

CrowdCall Mozilla Firefox nabigatzailerako bateragarria da. Tresna nabigatzailean instalatzeko prozesua oso sinplea da, eta ez luke 2 minutu baino gehiago iraun beharko. Hauek dira jarraitu beharreko urratsak:

1. CrowdCall-ek funtzionatu ahal izateko beharrezkoa da Greasemonkey *plugin*a instalatuta edukitzea. *Plugin* hau instalatzeko pluginaren web orrira jo.

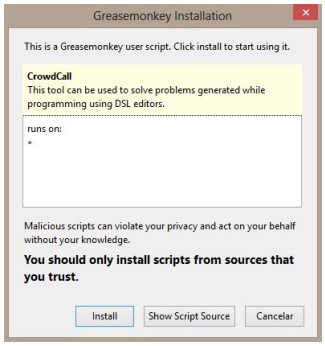
2. “Gehitu Firefox(e)ra” botoia sakatu eta agertuko den konfirmazio leihoko mezua onartu.



3. Nabigatzailea berrabiaraziko da, eta nabigazio-barraren eskuinaldean tximino baten aurpegia irudikatzen duen ikonoa ikusiko da. Bertan klikatu beharko dugu Greasemonkey gaitzeko. Ikonoaren gainean klik egiterakoan kolorez aldatuko da.

4. CrowdCall-en iturburu kodea duen *script*a deskargatu eta idazmahaian kokatu.

5. *Script*a idazmahaitik nabigatzailera arrastaka eraman. Azalduko den konfirmazio pantailako mezua onartuta, CrowdCall zure nabigatzailean instalatuta egongo da.



C. ERANSKINA

Erabiltzailearen eskuliburua

Zer da CrowdCall?

CrowdCall programazioan sortzen diren zalantzak argitzeko DSL editoreetan txerta daitekeen erreminta da. Tresna honek erabiltzailea arazo hauen ebazpen prozesuan zehar gidatzen du.

Zalantza hauek argitzeko, tresna honek laguntza eskaerak erabiltzailearen SNS aplikazioetan (Twitter, Facebook...) idazten ditu, eta honen kontaktuak dira arazoetako ebazpenak proposatuko dituztenak. CrowdCall aplikazio hauetan egiten den mezu-trukea kudeatzeaz arduratzen da, eta egindako laguntza eskaerak nahiz honen erantzunak tresnaren barruan integratzen dira. Gainera, jasotako erantzunetik arazoa ebazteko informazio baliagarria erauzteaz arduratuko da, erabiltzaileari lana erraztuz. Azkenik, erabiltzailearen beharretara gehien egokitzen den ebazpena aukeratzeko prozesua ere sinplifikatzen du. Eta guztia DSL editoretik atera gabe!

Arazoak

CrowdCall arazoen ebazpenerako tresna da. Hala ere, hauek independenteki tratatzeko konplexuegiak izan daitezke. Horregatik, arazoak azpi-arazotan deskonposatzen dira, horietako bakoitzaren ebazpena zehatzagoa izateko. Modu honetan, gainera, batzuetan azpi-arazoen arteko menpekotasun erlazioa sor daiteke: azpi-arazo bat ebazten hasi ahal iza-

teko beste baten ebazpena behar da. CrowdCall arduratuko da arazoa azpi-arazoetan deskonposatzeaz eta, modu berean, azpi-soluzioak soluzio bakar batean bateratzeaz.

Azpiarazoen definizioa

CrowdCall DSL editore batekin erabili ahal izateko, DSL editorea tresnarekin lan egin ahal izateko hedatuta egon beharko da. Editore bakoitzak arazoak definitzeko bere sintaxi propioa defini dezake, baina laguntza eskaera baten formulazioa DSL editore batetik bestera aldatzen den arren, editore guzietan erabili daitezkeen komodin batzuk daude:

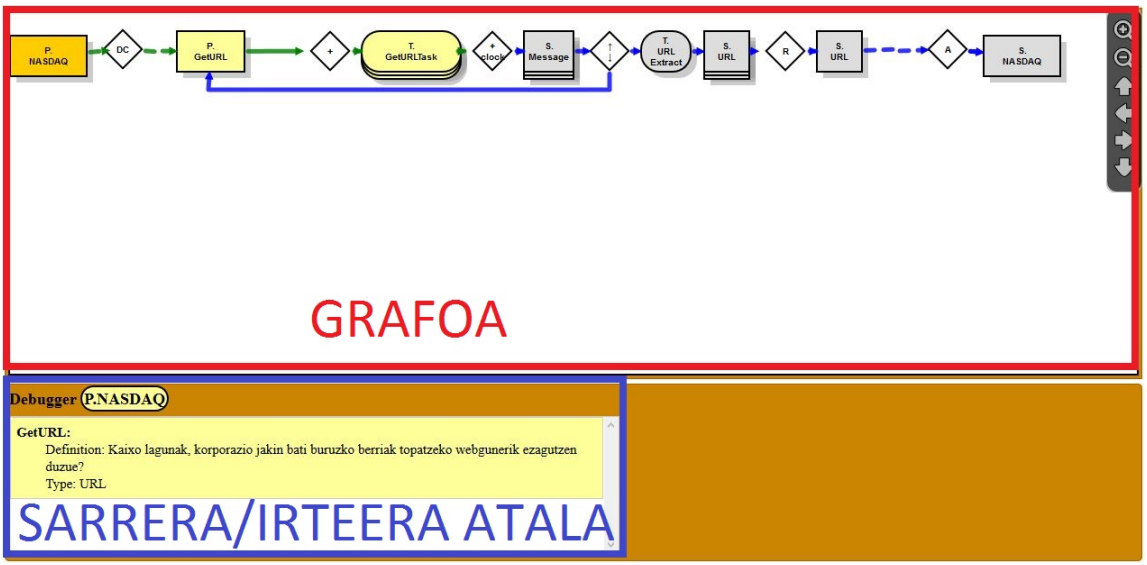
- Demagun azpi-arazo baten ebazpenak aste honetan jaso nahi direla. Data eskuz idazteaz arduratu beharrean, *$date+7* komodina erabili daiteke. Horrela, azpi-arazoaren formulazioan *$date* komodinari *x* zenbakia gehituz, gaurko egunetik hasita *x* egunetara dagoen data idatziko da komodinaren ordez.
- Modu berean, $currentURL komodina erabil daiteke CrowdCall martxan jartzen deneko web orriaren URLa ordezkatzeko.

Interfaze grafikoa

Interfaze grafikoaren betebeharra arazoaren ebazpen prozesuan zehar erabiltzailearekin elkarreragitea da. Arazoa sortu bezain laster interfaze grafikoa bistaratuko da, tresna erabiltzen duen DSL editorean bateratua.

Interfaze grafikoaren atalak

Interfaze grafikoa bi ataletan bereizten da: arazoaren ebazpen-egoera islatzen duen grafo nabigarria eta egoera bakoitzeko erabiltzaileari xehetasun gehiago erakusteko eta erabiltzailearen sarrerarako erabiltzen den atala.



Grafoko elementuak

Grafoaren atalean, zenbait elementuk osatutako grafoa ikus dezakegu. Hauek mota desberdinetakoak izan daitezke. Alde batetik, arazo-nodoak ditugu, “P.” karakterez hasten direnak. Nodo mota hauek hasierako arazoa zein honen azpiarazoak irudikatzeko balio dute. Bestalde, ataza-nodoak ditugu, “T.” karakterez hasten direnak. Nodo hauek ataza bati dagozkie, eta hauetan klik eginez gero, ataza honen konfigurazio parametroak pantailaratuko dira, hala izanez gero. Beste alde batetik, soluzio-nodoak aurki ditzakegu, “S.” karakterez hasten direnak. Hauek azpi-arazo bakoitzaren ebazpen prozesuan sortzen diren tarteko emaitzak irudikatzeko erabiltzen dira, baita azpi-arazo nahiz azpiarazoek osatzen duten arazoaren azkeneko ebazpenak ere. Azkenik, operadoreak ikus ditzakegu, erronbo itxurako nodo zuriak. Azken hauetan klikatuz gero ez da ezer azalduko ebazpen-egoerari buruzko xehetasun gehiago emateko atalean. Operadore bakoitzak esanahi desberdina du:

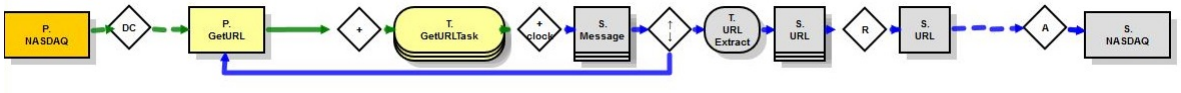
- *Divide-and-Conquer* operadoreak hasierako arazoa azpiarazo desberdinetan banatzeko prozesua irudikatzen du.
- *Aggregate* operadoreak azpiarazo guztien emaitzen bateratzea irudikatzen du, hasierako arazoaren emaitza lortzeko.

- *Ambiguity* operadoreak azpiarazo baten anbiguotasuna ekiditeko egin daitekeen birplanteaketa irudikatzen du.
- *Reduce* operadoreak emaitza multzo batetik bakarrera laburtzeko prozesua irudikatzen du.
- *Multiply* eta *Merge* operadoreek azpiarazo bereko ebazpen desberdinak ebazpen posibleen multzo batean batzeko prozesua irudikatzen dute.

Bestalde, nodoak bakunak edo anizkoitzak izan daitezke. Nodo anizkoitzek mezu eta azpisoluzioen anizkoiztasuna adierazten dute.

Grafoaren kolorea

Koloreari dagokionez, grafoaren nodo nahiz konektoreak kolorez aldatzen joaten dira arazo eta azpiarazoen ebazpen-mailaren arabera. Horrela, azpi-arazo bat irudikatzen duten nodoak lau koloretakoak izan daitezke: gorriak (oraindik ezin da hasi azpiarazoa ebazten, beste azpiarazo baten ebazpenaren menpe dagoelako), horiak (azpiarazoa ebazpen hasi da, baina oraindik ez da emaitzarik lortu), berdeak (azpiarazoa ebatzita dago) edo grisak (azpiarazoa ebazten hasi da, baina nodoak adierazten duen egoerara iristeko aurretik burutu beharreko eragiketaren bat egiteke dago).



Bestalde, nodoen arteko konektoreak urdinak (grafo edo azpigrafoaren uneko egoera oraindik ez da konektorera iritsi) edo berdeak (grafo edo azpigrafoaren uneko egoerak konektorea gainditu du) izan daitezke.

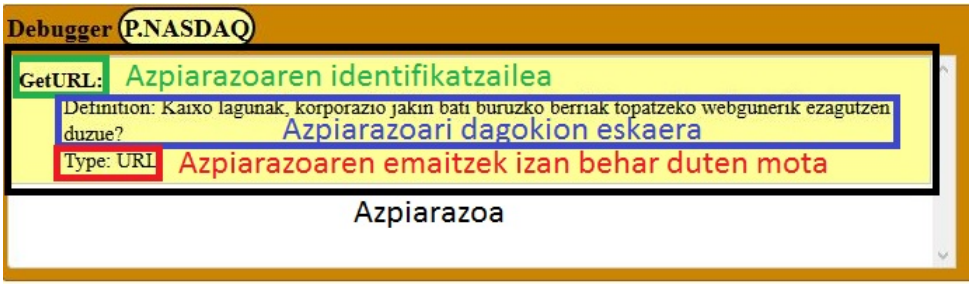
Grafoaren barruko nabigazioa

Grafoaren barruan nabigatzeko sagua nahiz teklatuko geziak erabil daitezke. Gezien bidez grafoaren barrutik desplazatzeko aukera ematen da. Saguari dagokionez, klik egin eta kurtsorea arrastatuta, grafoaren bidez desplazatzeko aukera ematen da. Bestalde, saguko ezkerreko botoian klik bikoitza eginez gero, grafoan zoom egingo da. Efektu bera lortuko dugu saguaren gurpiltxoa erabilita, alde batera biratuz gero grafoa handituz, eta beste

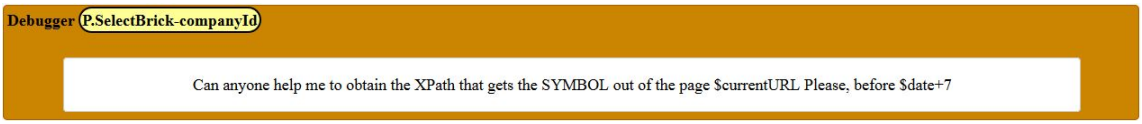
aldera, berriz, txikituz. Honez gain, grafoko nodoak (operadoreak salbu) sakatuta erabiltzaileak tresnarekin elkarreragin ahal izango du, eta komunikazio hau interfaze grafikoaren bigarren atalean egingo da, hots, ebazpen egoerari buruzko xehetasunak adierazteko atalean. Azkenik, grafoaren goi-eskuin izkinan sei botoi ikus daitezke. Goiko biak, luparen itxura dutenak, zooma kontrolatzeko erabiltzen dira, grafoa handitu nahiz txikitzeko. Beste lauak, berriz, grafoaren barruan desplazatzeko erabil daitezke.

Arazo baten ebazpena

Interfazea osatzen duen beste atalari dagokionez, hots, arazoaren ebazpen-egoera bakoitzean sarrera/irteera gisa erabiltzen den atalari dagokionez, egitura desberdina izango du ebazpen-egoeraren arabera. Grafoaren erro nodoan klik eginez, arazoa osatzen duten azpiarazoen laburpen bat ikusiko da. Azpiarazo bakoitzeko, bere identifikatzailea, azpiarazoari dagokion laguntza eskaera eta azpiarazoaren subsoluzioek izan beharreko datu mota adierazten dira. Gainera, koloreen bidez bakoitzaren ebazpen-egoera adierazten da. Gorriak ebazten hasi gabe dagoela adierazten du; horiak, ebazteko prozesuan dagoela, eta berdeak jadanik ebatzita dagoela.

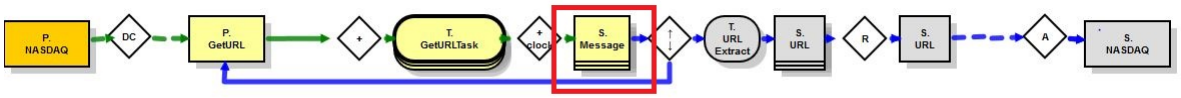


Azpiarazo bati dagokion azpigrafoko erro nodoaren gainean klik eginez gero, interfaze grafikoaren sarrera/irteera atalean azpiarazoari dagokion laguntza eskaera bistaratzen da.

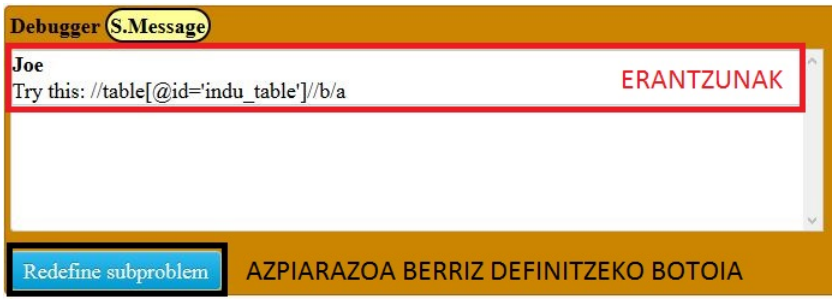


Azpiarazo baten ebazpen prozesua irudikatzen duen azpigrafoko lehenengo atazanodoan (ezkerretik hasita) klik eginez gero, azpiarazo horri dagokion laguntza eskaera eta eskaera horri erantzunez bidalitako mezuak ikusi ahal izango dira. Pantaila honek hiru osagai nagusi dauzka: laguntza eskaera eta erantzunak zerrendatzen dituen atala, azpiarazoaren ebazpenari buruzko argibideak emateko erabil daitekeen formularioa eta hurrengo urratsera igarotzeko botoia. Formularioaren bidez, kolaboratzaileek egin ditzaketen galderak erantzuteko aukera ematen da. Funtzionamendua oso sinplea da: ezkerreko testu koadroan idatzi, eta eskuineko botoia ("Send Message" testua daukana) sakatu. Bestalde, azpiarazoa ebatzi ahal izateko nahiko erantzun jaso dituzula ikusterakoan, "I've got enough answers" botoia sakatu eta hurrengo urratsera pasatzeko aukera izango duzu. Botoi hau gaituta agertu ahal izateko, laguntza eskaerak gutxienez erantzun bat izan behar du.

"I've got enough answers" botoia sakatuz gero, azpiarazoaren ebazpen egoerak aurrera egingo du, eta grafoko hurrengo nodoa gaituko da, kolore grisetik horira pasatuz.



Berriki gaitu den nodoan klik eginez gero, grafoko hurrengo nodoa gaituko da. Aldi berean, aurreko pantailaren oso antzekoa den pantaila azalduko da. Bertan, jasotako erantzunen zerrenda bat azalduko da, eta horrekin batera, "Redefine problem" botoia ikus daiteke. Botoi hau baliagarria da laguntza eskaeran nahi duguna ez dugunean ongi azaldu, eta argibideak ematen ibili beharrean, eskaera berri bat sortu nahi dugunean.

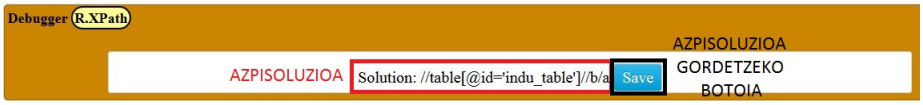


Laguntza eskaera berriz definitzeko pantailako sarrera/irteera atala formulario batez osatuta dago, eta bere funtzionamendua oso sinplea da: laguntza eskaeraren definizio berria testu eremuan idatzi eta "Redefine" botoia sakatu.

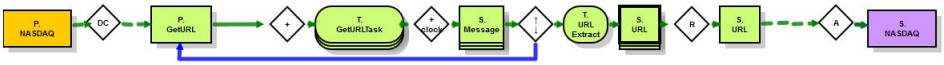
Laguntza eskaera berriz definitu beharrean, aktibatutako azkeneko nodoan klik eginez gero, ez da eguneraketarik egingo sarrera/irteera atalean; horren ordez, hurrengo nodoa gaituko da, prozesuarekin jarraitu ahal izateko. Gaitutako nodoan klik eginez gero, kolaboratzaileek idatzitako erantzunetatik erauzitako adierazpenak zerrendatuko dira. Horrez gain, "Reduce" testua daukan botoia ere azalduko da; honek balizko ebazpen guztietatik egokiena aukeratzeko mekanismoa eskaintzen du.

Botoi hau sakatuta, azpiarazoak definitutako datu motarentzat bateragarriak diren aukeraketa estrategiak erakusten zaizkio erabiltzaileari. Pantaila honetan, estrategiez gain, "OK" testua duen botoi bat dago.

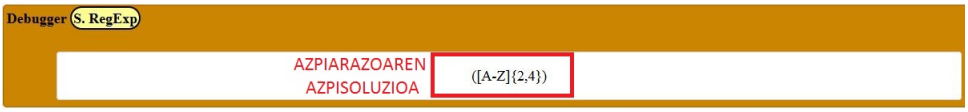
"OK" botoian klik eginez, aukeraketa estrategia aplikatu eta emaitza erabiltzaileari erakutsiko zaio. Honek, azpisoluzioa erabiltzaileari erakusteaz gain, "Save" testua duen botoiaren bidez, azpisoluzioa azkeneko emaitza bezala baieztatzeko aukera ematen da.



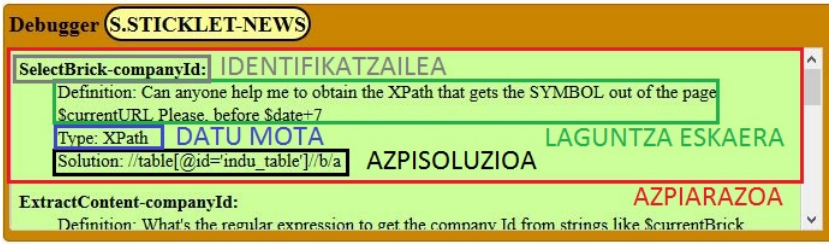
Botoi hau sakatzean, azpiarazoari dagokion azpigrafoak kolore berdea hartzen du.



Azpiarazo bat ebatzita dagoenean, honi dagokion azpigrafoko emaitza-nodoan klik eginez gero, azpiarazoaren azpisoluzioa pantailaratuko da.



Azkenik, arazoaren soluzioa irudikatzen duen emaitza-nodoan klik eginez, arazoa osatzen duten azpiarazoak zerrendatuko dira, bakoitzaren identifikatzailea, definizioa, datu mota eta azpisoluzioa erakutsita.



D. ERANSKINA

CrowdCall-en APIa

CrowdCallen APIan ondorengo metodoak definituta daude:

- getProblems
- deleteProblem
- getProblemsSolution
- createProblem
- configureCrowdCall

Jarraian, metodo hauek sakonean azalduko dira.

getProblems()

Deskribapena: Metodo honek CrowdCall tresnak sortutako arazoen zerrenda itzultzen du, hauen kudeaketa egin ahal izateko.

Sarrera: Metodo honek ez du parametrorik definituta.

Irteera: Funtzio honek objektu bat itzultzen du, JSON notazioan, honako eremuak izanik:

Izena	Mota	Deskribapena
status	String	Kontsulta ongi exekutatu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak kontsulta arazorik gabe exekutatu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.
problemList	String[]	Arazoen identifikatzaile-zerrenda da.

Erabileraren adibidea:

```
var ret = getProblems();
if(ret.status == "ok"){
  var i;
  for(i=0;i<ret.problemList.length;i++){
    alert(ret.callList[i]);
  }
}
else{
  alert(ret.error);
}

```

deleteProblem(problemId)

Deskribapena: Metodo honek arazo jakin bat ezabatu egiten du, existitzen bada, edo errore mezu bat itzultzen du, hala ez bada.

Sarrera: Metodo honek parametro bakarra definituta dauka.

Izena	Mota	Deskribapena
problemId	String	Ezabatu nahi den arazoaren identifikatzailea.

Irteera: Funtzio honek objektu bat itzultzen du, JSON notazioan, honako eremuak izanik:

Izena	Mota	Deskribapena
status	String	Ezabaketa ongi exekutatu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak ezabaketa arazorik gabe exekutatu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.

Erabileraren adibidea:

```
var ret = getProblems();
if(ret.status == "ok"){
  var i;
  for(i=0;i<ret.problemList.length;i++){
    DeleteProblem(ret.problemList[i]);
  }
}
else{
  alert(ret.error);
}

```

getProblemsSolution(problemId)

Deskribapena: Metodo honek ebatzita dagoen arazo jakin baten ebazpena lortzen du.

Sarrera: Metodo honek ondorengo parametroak definituta dauzka:

Izena	Mota	Deskribapena
problemId	String	Arazoaren identifikatzailea.

Irteera: Funtzio honek objektu bat itzultzen du, JSON notazioan, honako eremuak izanik:

Izena	Mota	Deskribapena
status	String	Exekuzioa ongi burutu den adierazten du. Hiru balio posible dauzka: “ok”, “error” eta “not_found”. • “ok” balioak exekuzioa arazorik gabe burutu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du • “not_found” balioak adierazten du ez dela eskatutako arazoa aurkitu
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.
subsolutionList	Array	Laguntza arazoaren azpiarazo bakoitzarentzako identifikatzailea-ebazpena bikotea adierazten du. Beraz, zerrendako posizio bakoitzak bi eremutako JSON notazioko objektua izango ditu: • *subproblemId*: azpiarazoaren identifikatzailea adierazten duen karaktere-katea. • *subsolution*: tresna erabilita azpiarazoarentzat lortutako azpisoluzioa

Erabileraren adibidea:

```
1  var ret = getProblemsSolution("Sticklet1");
2  if(ret.status == "ok"){
3    var i;
4    for(i=0;i<ret.subsolutionList.length;i++){
5      alert(ret.subsolutionList[i].subproblmId+" azpiarazoaren azpisoluzioa
     honakoa da: "+ret.subsolutionList[i].subsolution);
6    }
7  }
8  else if(ret.status == "error"){
9    alert(ret.error);
10 }
11 else{
12   alert("Ez da arazorik aurkitu Sticklet1 identikatzailearekin.")
13 }
14
```

createProblem(DSLExpression,callback)

Deskribapena: Metodo honek arazo bat sortzen du eta tresnaren interfaze grafikoa martxan jarri egiten da. Identifikatzaile bereko arazo bat aurretik egongo balitz, interfaze grafikoa zuzenean martxan jarriko litzateke, arazoa sortu gabe.

Sarrera: Metodo honek ondorengo parametroak definituta dauzka:

Izena	Mota	Deskribapena
DSLExpression	String	Arazoa sortzeko erabiliko den adierazpena. *configureCrowdCall* metodoaren bidez konfiguratzen den *DSLexpression2Problem* modulua adierazpenetik abiatuta arazoa sortzeaz arduratuko da.
callback	Function	Arazoa ebazterakoan exekutatuko den funtzioa.

Irteera: Funtzio honek objektu bat itzultzen du, JSON notazioan, honako eremuak izanik:

Izena	Mota	Deskribapena
status	String	Exekuzioa ongi burutu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak exekuzioa arazorik gabe burutu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.

Erabileraren adibidea:

```
1 var cb = function(){
```

```
2     alert("Arazoa ebatzi da");
3   }
4   var ret = createProblem("cell5D-crowd:How can I check if B1 is in the first
      percentile of the values of column B? Thanks",cb);
5   if(ret.status == "error"){
6     alert(ret.error);
7   }
8
```

configureCrowdCall(GUIDetails, DSLExpression2Problem, subproblem2Context, subproblem2Message, message2Subsolution, subsolutions2Subsolution)

Deskribapena: Metodo honen bidez, interfaze grafikoari buruzko xehetasunak adierazteaz gain, tresna osatzen duten bost elementu edo moduluak konfiguratuko dira. 5 modulu horiek honakoak dira:

- Domeinuaren adierazpenetik abiatuta arazoaren sorkuntzarako modulua (*DSLExpression2Problem*).
- Azpi-arazoak testuinguruan jartzeko modulua (*subproblem2Context*).
- SNSak kudeatzeko modulua (*subproblem2MicroblogPost*).
- Erantzunetan azpi-soluzioak identifikatzeko modulua, datu motetan oinarrituta (*reply2Subsolution*).
- Azpi-soluzioak bakarrera laburtzeko modulua (*subsolutions2Subsolution*)

Tresnaren konfigurazioa arazoa sortzeko eta interfaze grafikoa abiarazteko deia egin aurretik (*createProblem*) egin beharko da.

Parametroak: Metodo honek honako parametroak definituta dauzka.

Izena	Mota	Deskribapena
GUIDetails	JSON	CrowdCall-en interfaze grafikoari buruzko xehetasunak.
DSLExpression2Problem	Function	*DSLExpression2Problem* moduluaren konfigurazioa.
subproblem2Context	IContextualizationStrategy[]	*subproblem2Context* moduluaren konfigurazioa.
subproblem2Message	ISocialNetworkStrategy[]	*subproblem2Message* moduluaren konfigurazioa.
message2Subsolution	IDataTypeStrategy[]	*message2Subsolution* moduluaren konfigurazioa.
subsolutions2Subsolution	IReductionStrategy[]	*subsolutions2Subsolution* moduluaren konfigurazioa.

Parametro hauen konplexutasuna dela eta, banaka deskribatuko dira.

GUIDetails

Parametro hau JSON notazioa jarraitzen duen objektua da, honako eremuak dituena

Izena	Mota	Deskribapena
elementId	String	Eremu honek CrowdCall-en interfaze grafikoa txertatzeko erabiliko den HTML elementuaren *id* atributua adierazten du.
height	Integer	Eremu honek CrowdCall-en interfaze grafikoaren altuera adierazten du. Eremu hau azaltzen ez bada *elementId* eremuak erreferentziatzen duen elementuaren altuera osoa hartuko da.
width	Integer	Eremu honek CrowdCall-en interfaze grafikoaren zabalera adierazten du. Eremu hau azaltzen ez bada *elementId* eremuak erreferentziatzen duen elementuaren altuera osoa hartuko da.

DSLExpression2Problem

Parametro hau DSL editoreko adierazpenetatik abiatuta CrowdCall-eko arazoak sortuko dituen funtzioa da. Funtzio honi *createCall()* metodoari pasatako *DSLExpression* argumentuarekin deituko zaio. Bestalde, funtzio honen irteera balioa honako eremuak dituen arazoa, JSON notazioaz adierazita:

Izena	Mota	Deskribapena
problemId	String	Arazoaren identifikatzailea.
subproblemList	Array	Arazoa osatzen duten azpi-arazoen zerrenda. Zerrenda honetako elementu bakoitza JSON notazioko objektu bat izango da, honako eremuak izango dituena: • *subproblemId*, String motakoa. Azpi-arazoaren identifikatzailea adierazten du. • *dataType*, String motakoa. Azpi-arazoaren azpi-soluzioek izan beharreko datu motaren izena. *dataType* balioa itzultzen duen *getName()* metodoa daukan objektua egon beharko *reply2Subsolution* modulurako zehaztutako estrategien artean. Bestela, errore mezu bat itzuliko da arazoa sortzerako orduan (*createProblem*). • *request*, String motakoa. Azpi-arazoari dagokion laguntza eskaera da. • *socialNetworks*, String zerreda bat. Azpi-arazoaren ebazketarako erabiliko den SNS plataforma. Zerrendako elementu bakoitzeko, String bera itzultzen duen *getName()* metodoa duen *subproblem2MicroblogPost* moduluan objektu bat egon beharko da. Bestela, errore mezu bat itzuliko da arazoa sortzerako orduan (*createProblem*).

Jarraian datorren kode-zatiko f aldagaia eremu honen adibide sinpleenetariko bat da. Adibidetik ondoriozta daitekeenez, editoreak azpi-arazo bakarreko arazoak sortuko ditu, eta hauek sortzeko adierazpenek formatu hau jarraituko dute “problemId:subproblemId:laguntza eskaera”.

```
var f = function(dslExp){
  var lag = dslExp.split(":");
  var i;
  var request;
  for(i=2;i<lag.length;i++){
    request = request + lag[i];
  }
  var problem = {
    "problemId":lag[0],
    "subproblemList":[
      {"subproblemId":lag[1],
      "dataType":"URL",
      "request":request,
      "socialNetworks":["Twitter"]
      }
    ]
  }
  return var;
}
```

subproblem2Context

Parametro hau IContextualizationStrategy interfaze abstraktua inplementatzen duen klaseren bateko objektuen zerrenda da. Interfaze abstraktua inplementatzen duten klaseek bi metodo inplementatu beharko dituzte:

- *getName()*. Metodo honek klase honetako objektuen identifikaziorako erabiliko den String bat itzultzen du.

- *contextualize(args)*. Metodo hau azpi-arazo bati dagokion laguntza eskaerak testuinguruan jartzeko erabiliko da. Horrela, CrowdCall-en motorrak laguntza eskaera batean $func("funtzioa",arg1,arg2,...) adierazpena detektatzerakoan, IContextualizationStrategy interfazea inplementatzen duten klaseko objektuen artean, *getName* metodoak *funtzioa* itzultzen duena bilatuko da. Orduan, objektu honen *contextualize* metodoari deituko zaio, argumentu bezala *arg1*-k eta *arg2*-k osatutako zerrenda pasatuta. Metodo honek karaktere kate bat itzuliko du, eta honek laguntza eskae-

ran adierazitako adierazpena ordezkatuko du. Metodoak *null* balioa itzultzen badu, karaktere kate huts batek adierazpena ordezkatuko du.

Hona hemen IContextualizationStrategy interfaze abstraktua inplementatzen duen klase baten adibidea:

```
1  function XPathFunction(){
2    this.name = "returnXpath";
3  }
4  XPathFunction.method("getName", function(){
5    return this.name;
6  }
7  XPathFunction.method("contextualize", function(args){
8    var url = args[0];
9
10   var l = GM_xmlhttpRequest({
11     method: "GET",
12     url: url,
13     synchronous: true});
14
15   var a=document.createElement("div");
16   a.innerHTML=l.responseText;
17   var lag = document.evaluate(args[1], a, null, XPathResult.
       FIRST_ORDERED_NODE_TYPE, null );
18   if(lag!=null){
19     var text = lag.singleNodeValue.textContent;
20     return text;
21   }
22   else{
23     return null;
24   }
25 })
```

subproblem2MicroblogPost

Parametro hau ISocialNetworkStrategy interfaze abstraktua inplementatzen duen klaseren bateko objektuen zerrenda da. Interfaze abstraktua inplementatzen duten klaseek metodo hauek izan beharko dituzte:

- *getName()*. Metodo honek klase honetako objektuen identifikaziorako erabiliko den karaktere kate bat itzultzen du.

- *getURL()*. Metodo honek klaseak kudeatzen duen SNS plataforman *login* egiteko web orriaren URLa itzuliko du.

- *getLogo()*. Metodo honek klaseka kudeatzen duen SNS plataformaren logoa itzultzen du.

- *postMessage(message)*. Metodo honen bidez, SNS aplikazioan *message* karaktere kateko mezua idatziko da. Metodo honek JSON notazioko objektu bat itzuliko du, honako eremuekin:

Izena	Mota	Deskribapena
status	String	Exekuzioa ongi burutu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak exekuzioa arazorik gabe burutu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.
author	String	Erabiltzaileak SNS aplikazioan duen identifikatzailea.
messageId	String	Idatzitako mezuak SNS aplikazioan duen identifikatzailea.

- *replyMessage(messagId, userId, message)*. Metodo honek SNS aplikazioan *userId* erabiltzaileak idatzitako eta *messageId* identifikatzailea duen mezuari erantzunez

message mezua idatziko da. Metodo honen irteera *postMessage* metodoaren irteera bera da.

- *getReplies(messageId)*. Metodo honek SNS aplikazioan *messageId* identifikatzailea duen mezuaren erantzunak zerrendatuko dira. Honen irteera balioa JSON notazioko objektu bat da, honako eremuekin:

Izena	Mota	Deskribapena
status	String	Exekuzioa ongi burutu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak exekuzioa arazorik gabe burutu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.
replyList	Array	Mezuari egindako erantzunen zerrenda bat. Zerrendako elementu bakoitzak eremu hauek dituen JSON objektua da: • *author*: mezua idatzi duen erabiltzailea SNS aplikazioan identifikatzeko erabiltzen den karaktere-katea. • *message*: SNS aplikazioko mezua.

Hona hemen ISocialNetworkStrategy interfaze abstraktua inplementatzen duen klase baten adibidea:

```
1  function Twitter(){
2    this.authenticityToken = null;
3    this.userId = null;
4  }
5  Twitter.method("getName", function(){
6    return "Twitter";
7  });
8  Twitter.method("getLogo", function(){
9    return "https://abs.twimg.com/a/1373572090/images/resources/twitter-bird-
       light-bgs.png";
10 });
11 Twitter.method('checkLogin', function(){
12   var l = GM_xmlhttpRequest({
13       method: "GET",
14       url: "https://twitter.com",
15       synchronous: true,
16       headers: {
17       "Cookie": document.cookie
18       }
19   });
20   if(l.status==200){ // OK
21     var lag = document.createElement("div");
22     lag.innerHTML= l.responseText;
23
24     var aux = document.evaluate( "//input[@id='init-data']/@value", lag, null,
        XPathResult.STRING_TYPE, null );
25
26     var aux2 = JSON.parse(aux.stringValue);
27     if(!aux2.loggedIn){
28       return {"status":"loggedOut"};
29     }
30     else{
31       // Look for the authenticity token, neccesary for posting messages on
       Twitter
32       if(aux2.formAuthenticityToken==null){ // Authenticity token not found
33         return {"status":"error", "error":"Authenticity token not found"};
34       }
35       else{
```

```
36          this.authenticityToken = aux2.formAuthenticityToken;
37          if(aux2.screenName==null){ // Twitter username not found
38            return {"status":"error", "error":"User name not found"};
39          }
40          else{
41            this.userId = aux2.screenName;
42            return {"status":"loggedIn"};
43          }
44        }
45      }
46    }
47    else{
48      return {"status":"error", "error":"Error while connecting to Twitter."};
49    }
50  });
51  Twitter.method('postMessage', function(message,problemId){
52    var jsonArgs = this.checkLogin();
53    if(jsonArgs.status == "loggedIn"){
54      var l = GM_xmlhttpRequest({
55        method: "POST",
56        url: "https://api.twitter.com/1/statuses/update.json",
57        synchronous: true,
58        data: "status="+escape(message)+"&post_authenticity_token="+this.
      authenticityToken,
59        headers: {
60        "Content-Type": "application/x-www-form-urlencoded",
61        "Accept": "application/json,text/javascript,*/*;q=0.01",
62        "X-PHX": true,
63        "X-Requested-With": 'XMLHttpRequest',
64        "Referer": "https://api.twitter.com/receiver.html"
65        }
66      });
67      if(l.status==200){ // OK
68        var jsonresponse = eval('('+l.responseText+')');
69        var messageId = jsonresponse.id_str;
70        return {"status":"ok","author":this.userId,"messageId":messageId};
71      }
72      else{
73        return {"status":"error", "error":"Error while posting tweet"};
74      }
75    }
```

```
76    else if(jsonArgs.status == "loggedOut"){
77      return jsonArgs;
78    }
79    else if(jsonArgs.status == "error"){
80      return jsonArgs;
81    }
82  });
83  Twitter.method('replyMessage', function(messageId,userId,message){
84    jsonArgs = this.checkLogin();
85    if(jsonArgs.status == "loggedIn"){
86      var mes = '@'+userId+' '+message;
87      var l = GM_xmlhttpRequest({
88          method: "POST",
89          url: "https://api.twitter.com/1/statuses/update.json",
90          synchronous: true,
91          data: "status="+escape(mes)+"&in_reply_to_status_id="+messageId+"&
      post_authenticity_token="+this.authenticityToken,
92          headers: {
93          "Content-Type": "application/x-www-form-urlencoded",
94          "Accept": "application/json,text/javascript,*/*;q=0.01",
95          "X-PHX": true,
96          "X-Requested-With": 'XMLHttpRequest',
97          "Referer": "https://api.twitter.com/receiver.html"
98          }
99      });
100     if(l.status==200){ // OK
101       var jsonresponse = eval('('+l.responseText+')');
102       var messageId = jsonresponse.id_str;
103       return {"status":"ok","author":this.userId,"messageId":messageId};
104     }
105     else{
106       return {"status":"error", "error":"Error while replying tweet"};
107     }
108   }
109   else if(jsonArgs.status == "loggedOut"){
110     return jsonArgs;
111   }
112   else if(jsonArgs.status == "error"){
113     return jsonArgs;
114   }
115 });
```

```
116 Twitter.method('getReplies', function(messageId){
117   var jsonArgs = this.checkLogin();
118   if(jsonArgs.status == "loggedIn"){
119     var l = GM_xmlhttpRequest({
120           method: "GET",
121           synchronous: true,
122           url: "https://api.twitter.com/1/statuses/mentions.json?
    include_entities=true&count=800",
123           headers: {
124           "Accept": "application/json,text/javascript,*/*;q=0.01",
125           "X-PHX": true,
126           "X-Requested-With": 'XMLHttpRequest',
127           "Referer": "https://api.twitter.com/receiver.html"
128           }
129         });
130     if(l.status==200){
131       var jsonresponse = eval('('+l.responseText+')');
132       var replies = [];
133       var i,j;
134       for (i=jsonresponse.length-1;i>=0;i--){
135         if(jsonresponse[i].in_reply_to_status_id_str==messageId){
136           // Delete mentions
137           var str_aux = new String(jsonresponse[i].text);
138           str_lag = str_aux.replace(/^@[^\s]+\s/gi,'');
139           // Unwrap URLs
140           for (j=0;j<jsonresponse[i].entities.urls.length;j++){
141             str_lag = str_lag.replace(jsonresponse[i].entities.urls[j].url,
    jsonresponse[i].entities.urls[j].expanded_url);
142           }
143           if(jsonresponse[i].user.screen_name!=this.userId){
144             replies.push({author:jsonresponse[i].user.name,message:str_lag});
145           }
146         }
147       }
148       return {"status":"ok", "replyList":replies};
149     }
150     else{
151       return {"status":"error", "error":"Error while connecting to Twitter."};
152     }
153   }
154   else if(jsonArgs.status == "loggedOut"){
```

```
155     return jsonArgs;
156   }
157   else if(jsonArgs.status == "error"){
158     return jsonArgs;
159   }
160 });
161 });
```

reply2Subsolution

Parametro hau IExtractionStrategy interfaze abstraktua inplementatzen duen klaseren bateko objektuen zerrenda da. Interfaze abstraktu hau inplementatzen duten klaseek bi metodo hauek izan beharko dituzte:

- *getName()*. Metodo honek klase honetako objektuen identifikaziorako erabiliko den karaktere kate bat itzultzen du.
- *extract(message)*. Metodo honek karaktere batetik baldintza batzuk betetzen dituzten adierazpenak itzultzen ditu, datu mota bati lotuta dagoenez. Metodo honek *message* karaktere katea jasotzen du sarrera gisa, eta karaktere kate zerrenda bat itzultzen du: erauzitako adierazpen zerrenda.

Hona hemen interfazea inplementatzen duen klase baten adibidea:

```
1  function URLType(){
2  }
3  URLType.method("extract", function(){
4    return "URL";
5  }
6  URLType.method("extract", function(message){
7    var extracted = [];
8    validator = /[-a-zA-Z0-9@:%_\+.~#?&//=]{2,256}\.[a-z]{2,4}\b(\/[-a-zA-Z0-9@
       :%_\+.~#?&//=]*)?/gi;
9    if(aux = lag.match(validator)){
10     var j;
11     for(j=0;j<aux.length;j++){
12       extracted.push(aux[j]);
13     }
```

```
14    }
15    return extracted;
16  });
```

subsolutions2Subsolution

Parametro hau IReductionStrategy interfaze abstraktua inplementatzen duen klaseren bateko objektuen zerrenda da. Interfaze abstraktu hau inplementatzen duten klaseek bi metodo hauek izan beharko dituzte:

- *getName()*. Metodo honek klase honetako objektuen identifikaziorako erabiliko den karaktere kate bat itzultzen du.
- *getNameToShow()*. Metodo honek erabiltzaileari laburtze estrategia aukeratzen zaion pantailan objektuak izango duen izena itzultzen du, karaktere kate motako balio baten bidez.
- *getDescription()*. Metodo honek estrategia deskribatzen duen karaktere katea itzultzen du.
- *getDataTypes()*. Metodo honek objektuarekin bateragarriak diren datu motak zerrendatzen ditu. *dataType* eremuan balio hori duten azpi-arazoak ebazteko erabili ahal izango da objektuko estrategia.
- *getBestOption(optionList)*. *optionList* parametroa karaktere kate zerrenda bat da. Metodo honek JSON notazioko objektu bat itzultzen du, eremu hauekin:

Izena	Mota	Deskribapena
status	String	Exekuzioa ongi burutu den adierazten du. Bi balio posible dauzka: “ok” eta “error”. • “ok” balioak exekuzioa arazorik gabe burutu dela adierazten du • “error” balioak exekuzioan erroreren bat gertatu dela adierazten du
error	String	Exekuzioan erroreren bat gertatuz gero (*status == “error”*), gertatutako errorearen deskribapen labur bat ematen du.
bestOption	String	Pasatako aukeren artean laburtze estrategia aplikatuta lortzen den azpisoluzioa

Hona hemen interfazea inplementatzen duen klase baten adibidea:

```
1  function AlexaStrategy(){
2  }
3  AlexaStrategy.method('getName', function(){
4    return "AlexaStrategy";
5  });
6  AlexaStrategy.method('getNameToShow', function(){
7    return "Alexa strategy";
8  });
9  AlexaStrategy.method('getDescription', function(){
10   return "This strategy returns best ranked option based on Alexa.";
11 });
12 AlexaStrategy.method('getDataTypes', function(){
13   return ["URL"];
14 });
15 AlexaStrategy.method('getOptionsRank', function(pos,callback){
16   var st = this;
17   //alert(pos+"elementua"+st.options[pos]);
```

```
18
19 });
20 AlexaStrategy.method('getBestOption', function(options){
21   var regExp = new RegExp(/Alexa\sTraffic\sRank:<\/span>\n<a href="[^"]+"
     >[^<]+(?=<\/a>)/i);
22   var ranks[];
23   var i;
24   for(i=0;i<options.length;i++){
25     var response = GM_xmlhttpRequest({
26       method: "GET",
27       url: "http://www.alexa.com/search?q="+st.options[pos],
28       synchronous: true
29     });
30     if(response.status==200){
31       var aux = regExp.exec(response.responseText);
32       var lag = new String(aux);
33       var rank = lag.replace(/Alexa\sTraffic\sRank:<\/span>\n<a href="[^"]+">\
     n/i, '');
34       rank = rank.replace(/\,/gi,'');
35       ranks[i]=rank;
36     }
37     else{
38       return {"status":"error", "error":"This strategy failed, try another one
     ."};
39     }
40   }
41   var bestRank;
42   var bestRankPos=-1;
43   for(i=0;i<st.options.length;i++){
44     var r = new Number(st.ranks[i]);
45     if(r!='No Data'&&r!="error"){
46       if((bestRankPos==-1)||(r<bestRank)){
47         bestRankPos=i;
48         bestRank=r;
49       }
50     }
51   }
52   if(bestRankPos!=-1){
53     return {"status":"ok", "bestOption":options[bestRankPos]);
54   }
55   else{
```

```
56      return {"status":"error", "error":"This strategy failed, try another one."
        };
57    }
58  });
```