eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Lana

Software Produktu-Lerroen (SPL) arkitektura bistaratzeko aplikazioaren garapena

Egilea

Iosu Salaberri Intxaurrondo

2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Lana

Software Produktu-Lerroen (SPL) arkitektura bistaratzeko aplikazioaren garapena

Egilea

Iosu Salaberri Intxaurrondo

Zuzendariak

Arantza Irastorza, Maider Azanza

Laburpena

Proiektu honen helburu nagusia Software Produktu-Lerroekin (SPL) lanean hasten diren garatzaile berriei laguntzeko aplikazio baten sorrera da, SPL-aren domeinua eta funtzionamendua modu errazagoan azaltzen duena. Horretarako anotazioen bidezko SPL-ak erabili dira oinarri gisa, Onekin ikerkuntza taldeak garatutako *WacLine* SPL-a esaterako.

Aplikazioaren prototipoak bi modulu desberdin ditu: pure::variants SPL-ak minatzen dituen meatzaritza modulua, *SPLMiner*, eta aurreko moduluak erauzitako informazioa erabiliz bisualizazioak sortzen dituen modulua, *InsideSPL*.

Memoria honetan prototipoa sortzeko prozesu osoa azaltzen da, hasierako proposamen eta plangintzatik amaierako kalitate probetara.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Egun, softwarea gartzeko paradigma anitz daude, bakoitzak bere abantaila eta desabantailekin. Ohikoena, edo behintzat unibertsitatean gehien lantzen dena, software produktu bakarra emaitzatzat duen prozesua da. Nahiz eta produktu hori burutzeko metodologia desberdinak egon (*agile*, *scrum*, ...) prozesu honen emaitza produktu bakarra izango da beti.

Hala ere, badago bestelako programazio paradigma bat produktu desberdin asko izan ditzakeela emaitza moduan: Software Produktu-Lerroa. Paradigma honek domeinu zehatz batean lan egiten du beti, eta software desberdinen antzekotasunak eta berezitasunak egitura bakar batean jasotzen ditu, nahiko konplexua izan ohi dena.

Software garapen proiektu batean sartzerakoan, garatzaile berriek domeinura ohitzeko ikasketa denbora bat behar dute, zeinetan ezin duten haien gaitasunen osotasuna eskaini. Proiektuak Software Produktu-Lerro bezala garatzerakoan, ikasketa denbora hori asko handitzen da domeinua osatzen duten ezaugarriak kodean sakabanatuak geratzen direlako eta SPL-ak inplementatzeak berezko ezagutza eskatzen duelako. Horregatik, domeinu honetan hasiberriak diren garatzaileak, *incomer*-ak, ikasketa prozesuan laguntzea guztiz beharrezkoa da.

Gradu Amaierako Lan honek ikasketa prozesu horretan lagunduko duen aplikazio baten prototipoa garatzea du helburu, aplikazioaren bitartez *incomer*-ak zehaztutako SPL-aren domeinua eta funtzionamendua ulertzeko gai izan dadin. Horretarako bi modulu desberdinetan banatutako arkitektura erabiliko da: lehendabizikoak pure::variants softwarearekin egindako SPL-ak hartu eta informazioa datu-base batera erauziko du; bigarrenak datu-

baseko informazioa hartu eta SPL-a azaltzeko baliabideak sortuko ditu. Azkenengo modulu hori izango da *incomer*-arekin elkarrekintza egiteaz arduratuko dena.

2. KAPITULUA

Proiektuaren kudeaketa-plana

Memoriaren atal honetan egin beharreko proiektua arrakastaz amaitzeko sortutako kudeaketa-plana azaltzen da. Hasteko, proiektuaren testuingurua azalduko da; jarraitzeko honen helburuak eta hauekin lotutako irismena aurkeztuko dira eta, amaitzeko, helburuak betetze aldera sortutako atazen plangintza.

2.1 Aurrekariak

Gradu Amaierako Lan hau Informatika Fakultateko Onekin ikerkuntza taldearen proposamenaren eskutatik jaio da, Software Produktu-Lerroen ulermena eta ikasketa azkartzeko helburuarekin (2.2 atalean zabalago azalduko da).

Proiektuan garatutako produktuaren probetarako erabiliko den SPL-a, *WacLine* izenekoa, anotazioetan oinarritutako pure::variants softwarearekin inplementatua dago, Onekin taldeko Haritz Medina ikerlariak sortua.

Pure::variants softwarearekin egindako SPL-ek fitxategi desberdinetan jasotzen dute haien konfigurazioa (4.1 atalean luzatuko da informazioa). SPL hauek kontzeptu amankomun asko dituzten SPL-aren ideia orokorrarekin, fitxategi batean SPL-aren domeinuko ezaugarriak biltzen baititu eta beste batzuetan konfigurazioak; konfigurazio batean ezaugarrietatik zeintzuk azalduko diren amaierako produktuan erabakitzen da. Hala ere, badira ezohikoak diren aukerak pure::variants-ekin egindako SPL-tan.

2.2 Helburua

Gradu Amaierako Lan honen helburu nagusia garatzaile berriei SPL-etan oinarritutako proiektuak azkarrago ikasteko aplikazio bat garatzea da, SPL-ak anotazioetan oinarritutako pure::variants softwarearekin inplementatuta daudenean.

Aplikazio honek SPL-en gaineko bisualizazioak sortuko ditu, harekin lotutako kontzeptuak errazago uler daitezen. Horretarako, aplikazioa bi modulu desberdinetan banatuko da.

Batetik, pure::variants SPL-a minatu eta fitxategietatik lortutako informazioa datu-base batean gordeko duen modulua, *SPLMiner* izenekoa. Bestetik, sortutako datu-basetan kontsultak egin eta bisualizazioak sortuko dituen modulua, *InsideSPL* izenekoa.

Arkitektura honetan oinarrituta, Gradu Amaierako Lan honen helburu zehatzak aipatuko dira orain:

- SPL-aren inguruko datuak gordetzeko **datu-base baten diseinua**. Datu-base hau pure::variants proiektu batekin beteko da eta, ondoren, SPL-en bisualizazioak sortzeko erabiliko da. Nahiz eta probetarako erabiliko den SPL-a, *WacLine*, anotazioetan oinarritutako pure::variants softwarearekin inplementatuta dagoen, datubasearen diseinuak SPL-en inguruko informazio orokorra gordetzea ahalbidetuko du, erabilitako software zehatzetik aldendu eta independentzia puntu bat eskainiz.
- Pure::variants kodetik lortutako informazioa DBan gordetzeko **meatzaritza modulua**. Modulu honek sarrera bezala pure::variants-ekin egindako proiektu bat izango du, Git biltegi batean gordetakoa, eta irteera bezala aurreko puntuan definitutako diseinua jarraitzen duen DB osatu bat. Meatzaritza modulua Onekin taldeko Raul Medeiros ikerlariak garatutako *FeatureCloud* aplikazioko meatzaritza moduluan inspiratuko da, horretatik erabilgarria dena hartuz eta garatu beharreko moduluan txertatuz.
- SPL-ak ulertzen laguntzeko **bisualizazio modulua**. Modulu honen bidez garatzaile berriek SPL-a modu inkrementalean ezagutzeko aukera izango dute, informazio erabilgarria era ordenatuan eskainiz. Bisualizazio moduluak SPL-en inguruko hainbat ikuspegi eskainiko ditu:
  - Mapa kontzeptuala. SPL-aren kodearekin batera, *CMaps* tresnarekin sortutako mapa kontzeptuala egongo da, SPL-aren kontzeptuak eta haien arteko loturak

jasoko dituena. Meatzaritza prozesuan fitxategi honetatik erauzitako informazioa erabiliz bisualizazio desberdinak sortuko dira SPL-aren ideia orokorra ulertu ahal izateko.

- Produktu ikuspegia. Bisualizazio hauetan SPL-aren produktu (*Variant Model*) desberdinetako datuak aurkeztuko dira, hauen *feature* kopurua edota produktuen arteko konparaketa eginez.
- Arkitektura ikuspegia. Bisualizazio hauekin SPL-aren osagaiak eta hauen arteko harremanak zeintzuk diren erakutsi nahi da. Produktuaren ikuspegian ez bezala, hemen SPL-ko kode-fitxategiak izango dira bisualizazioen erdigunea.

2.3 Irismena

Lan-eremuan gertatzen den moduan, proiektuaren irismen osoa ez dago hasieratik definituta eta horregatik modu iteratiboan garatuko da, denboraren poderioz egin beharreko atazak argi izan arte. Hala ere, helburuen ataletan (2.2 atala) bildutakoa kontuan hartuta, irismenaren mailaketa hau kontuan hartuko da hasieran:

1. Prototipo funtzionala: datu-basearen diseinua eta meatzaritza moduluaren inplementazio osoa, baita bisualizazio moduluaren "produktu ikuspegia". Ikuspegi honen bidez garatzaile berriek SPL-tik eratorritako produktuak aztertu ahalko dituzte eta haien arteko konparaketak egin bi produktuen zati amankomunak eta desberdintasunak ikusteko, adibidez.

2. Bisualizazio moduluaren hedapena: "mapa kontzeptuala", SPL-a osatzen duten ideiak eta kontzeptuak aztertu eta sakontzeko aukera ematen duena.

3. Bisualizazio moduluaren hedapena: "arkitektura ikuspegia". Honetan, SPL-aren arkitekturaren inguruko informazio desberdina emango zaio garatzaileari, hala nola zein da SPL-aren arkitektura orokorra, zein fitxategik osatzen duten SPL-a, zein ezaugarrik (*feature*) duten harremana fitxategi zehatz batekin...

4. *FeatureCloud* aplikazioarekin integrazioa. *FeatureCloud* Onekin ikerkuntza taldean garatutako web aplikazioa da SPL-en ezaugarriak hodei batean aurkezten dituena. Hodeiak garapen prozesuaren bi `git commit`-en artean ezaugarriek jasandako aldaketak erakusten ditu koloreak eta tamainak erabiliz. Jasandako aldaketez aparte,

ezaugarriek haien artean zenbat kode partekatzen duten ikus daiteke hodeian banatzen diren puntuaren arabera.

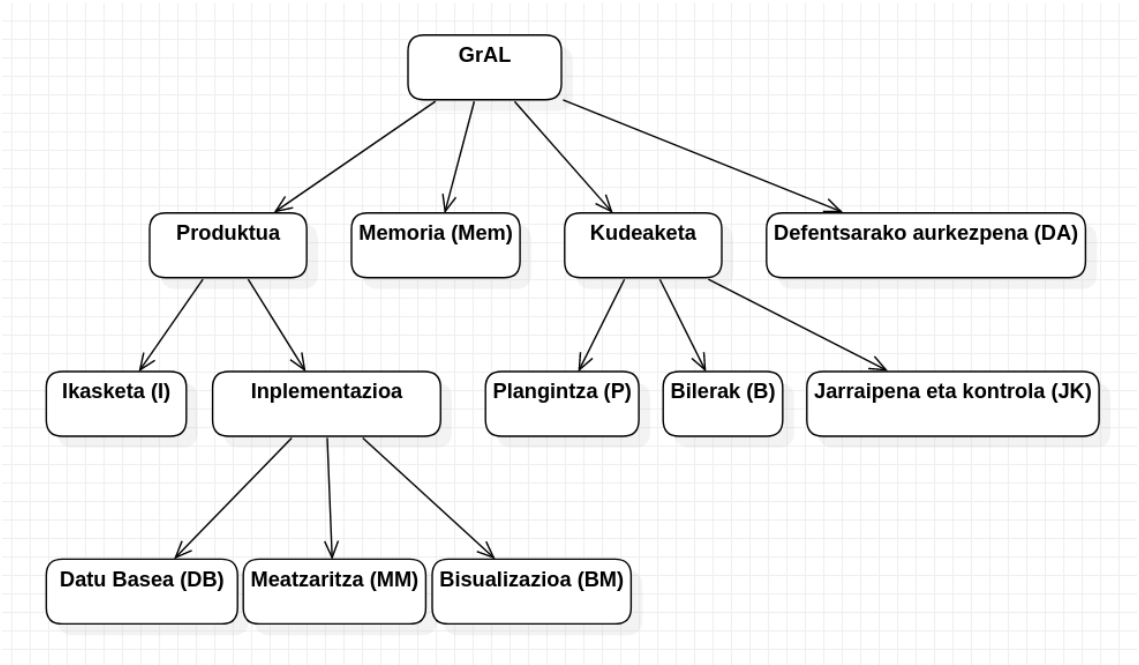
Lehenengo iterazioa amaituta dagoenean proiektuaren irismen minimoa beteta egongo da eta sortutako aplikazioa funtzionala izango da. Hala ere, hurrengo iterazioek aplikazioari, eta batez ere garatzaile berriei eskainitako laguntzari, osotasuna emango diote.

2.4 Plangintza

Atal honetan burututako Gradu Amaierako Lanaren plangintza jasotzen da. Hasteko, egin beharrekoak lan-paketeetan deskonposatu da (2.1 irudia), gero lan-pakete bakoitzak biltzen dituen atazak aurkeztu eta azkenik, ataza bakoitzari dedikatuko zaion denbora balioetsi.

2.4.1 Lanaren Deskonposaketa Egitura (LDE)

Proiektu honen lana multzo desberdinetan deskonposatu da, 2.1 irudian ikus daiteken moduan.



2.1 Irudia: LDE diagrama

Ikasketa (I) lan-paketeak proiektuaren garapenerako beharrezkoak diren ezagutzen eskuraketa biltzen du.

Datu-basea (DB) lan-paketeak sortuko den aplikazioaren DBarekin erlazionatutako atazak hartzen ditu barne.

Meatzaritza (MM) lan-paketea meatzaritza moduluaren garapenarekin zerikusia duten atazen bilduma da.

Bisualizazioa (BM) lan-paketean bisualizazio moduluaren zereginak azaltzen dira. Lanpakete hau izango da irismenean (2.3 atala) azaldutako iterazioak jasango dituen pakete nagusia.

Memoria (Mem) lan-paketeak memoria hau garatzeko beharrezko atazak ditu barne.

Defentsarako aurkezpena (DA) lan-paketeak garatutako proiektua epaimahai baten aurrean aurkezteko beharrezko eginkizunak jasotzen ditu.

Plangintza (P) lan-paketeak proiektuaren plangintzarekin lotutako eginbeharrak jasotzen ditu.

Bilerak (B) lan-paketeak proiektua aurrera eramateko zuzendariarekin eta ikerkuntza taldearekin egindako bilerak hartzen ditu barne.

Jarraipena eta kontrola (JK) lan-paketeak proiektuaren garapen egokia bermatuko duten atazak ditu barne, hala nola plangintzaren jarraipena eta irismenaren kontrola.

2.4.2 Atazak

Ikasketa (I)

1. pure::variants teknologiaren ikasketa, sintaxi espezifikoa ezagutu eta oinarrizko SPLak garatzeko modukoa.
2. *FeatureCloud* aplikaziotik jasotako meatzaritza moduluaren funtzionamendua ulertzea, horretatik ideiak atera ahal izateko.
3. D3.js softwarearen funtsezko erabilera lantzea, bisualizazio modulua inplementatzeko.

Datu-basea (DB)

1. SPL-en domeinuko elementuen eta haien arteko erlazioen ulermena.
2. Datu-basearen diseinua.
3. Datu-basea inplementatzea eta probatzea.

Meatzaritza (MM)

1. Jasotako *FeatureCloud* aplikazioak erabilitako meatzaritza moduluaren azterketa eta ulermena.
2. Meatzaritza modulu berriaren diseinua.
3. Meatzaritza moduluaren inplementazioa.
4. Meatzaritza moduluaren probak, sortutako DBa erabiliz.

Bisualizazioa (BM)

1. Lehen hurbliketa eta erabakiak
2. Bisualizazio moduluaren hasierako diseinua.
3. Bisualizazio moduluaren hasierako inplementazioa.
4. Bisualizazio moduluaren garapen iteratiboa.

Memoria (Mem)

1. Memoriaren idazketa eta berrikuspen iteratiboa.
2. Azkenengo berrikuspena.

Defentsarako aurkezpena (DA)

1. Posterraren prestaketa.
2. Defentsarako aurkezpena osatzea.

Plangintza (P)

1. Betebeharren identifikazioa, hasierako erabakiak hartzea eta zalantzen ebazpena.

2. Hasierako plangintza, orduen estimazioa osatzeko.

3. Plangintzaren eguneraketa, beharrezkoa izanez gero.

Bilerak (B)

1. Komunikazio sistema adostea.

2. Bilerak modu periodikoan egitea.

Jarraipena eta Kontrola (JK)

1. Proiektuaren garapenari buruzko informazio garrantzitsua jasotzea.

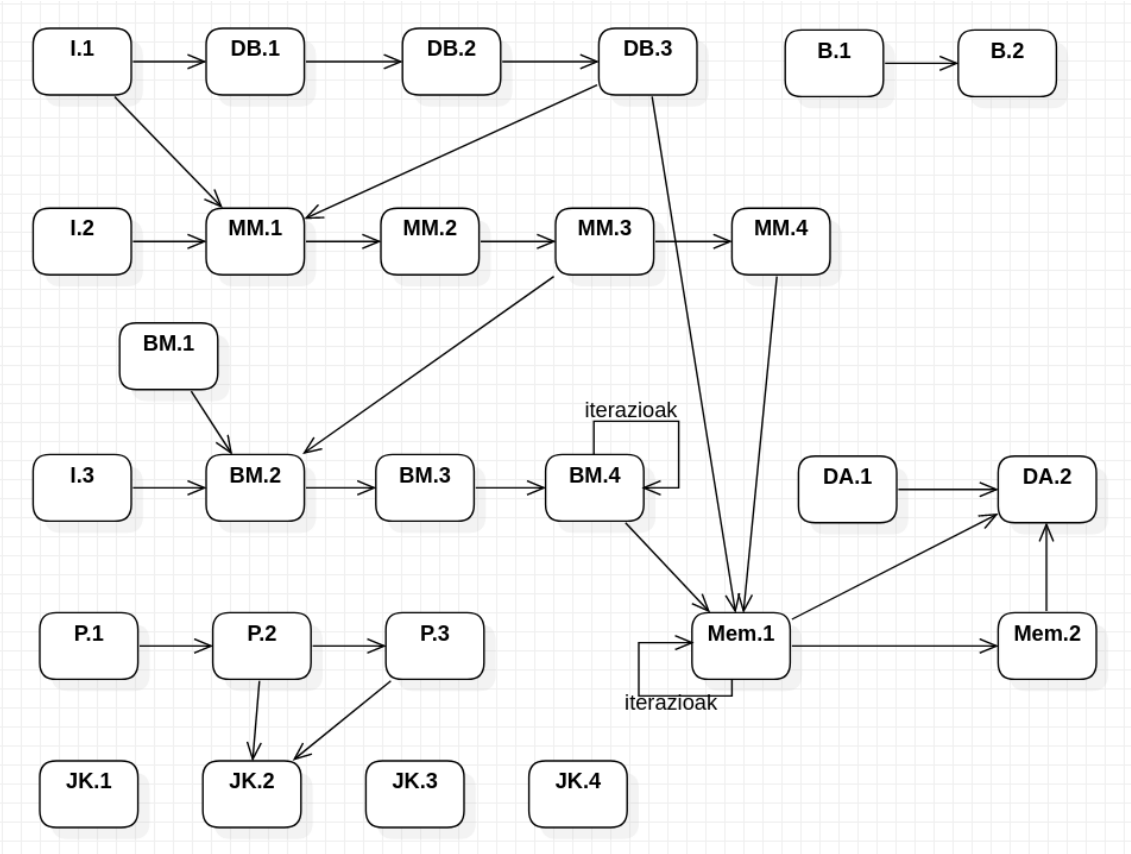
2. Plangintzaren kontrola, desbiderapenak antzeman eta saihesteko.

3. Sortutako arriskuen kudeaketa.

4. 2.5.2 azpiatalean definitutako kalitate ekintzak burutzea.

2.4.3 Atazen arteko menpekotasunak

Aurreko atalean aipatutako atazak ez dira modu linealean osatzen. Haien arteko menpekotasunak 2.2 irudian agertzen dira, eta hura aztertuta oso argi geratzen da menpekotasun nagusiena produktuaren inplementazioaren atalean ematen dela, datu-basearen diseinutik hasi eta bisualizazio moduluarekin amaitu behar baita, ordenean aldaketa handirik onartu gabe.



2.2 Irudia: Atazen arteko menpekotasunak.

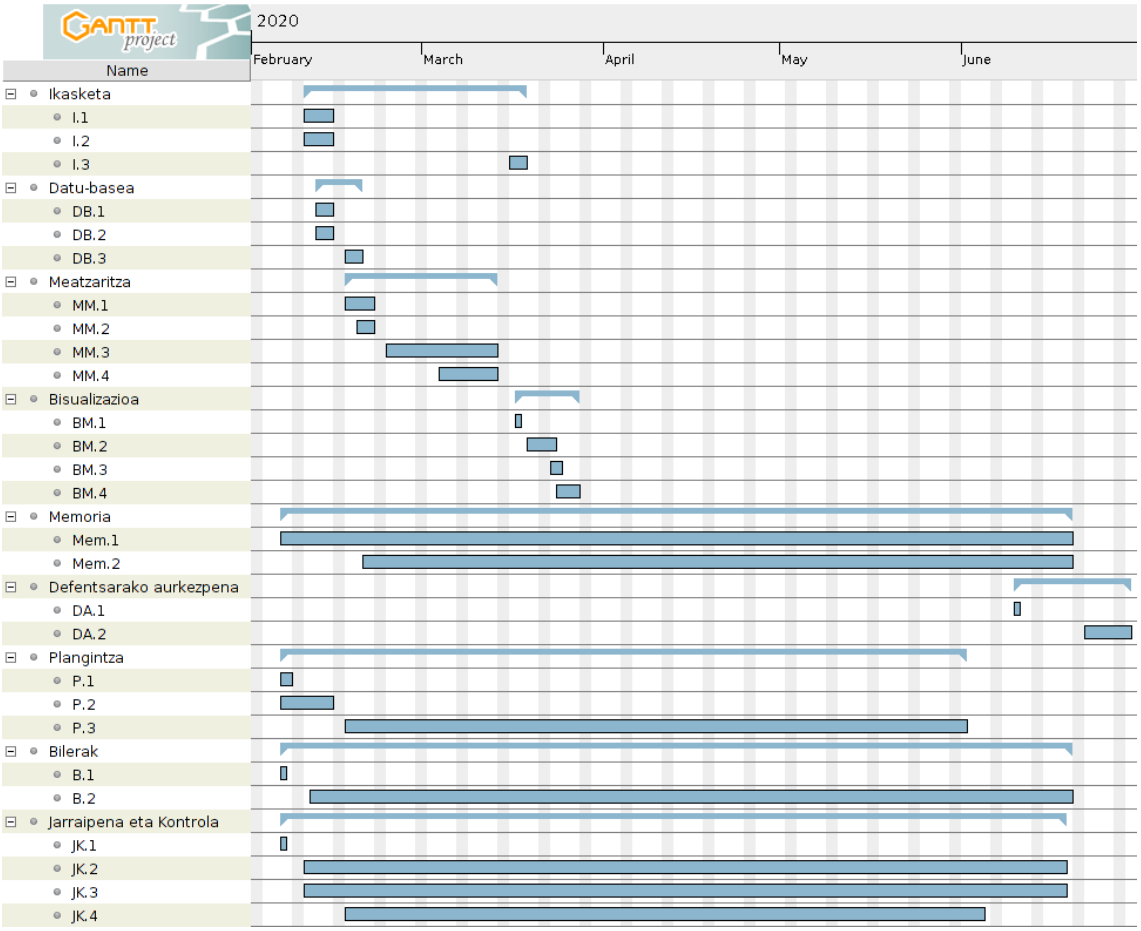
2.4.4 Atazen iraupenaren balioespena

Atazen arteko menpekotasunak kontuan hartuta, egin beharreko zeregin bakoitzaren iraupena balioetsi eta 2.1 taulan bildu da informazioa. Zehaztutako datak eta dedikatu beharreko orduak malguak izan daitezkeen arren, hauek erabiliko dira proiektuaren amaieran pairatutako desbiderapenak aztertu ahal izateko.

Atazaren deskribapena	Balioespena	Hasiera-data	Amaiera-data	Iraupena
Ikasketa	30 ord.	10/02	13/03	8 egun
I.1	10 ord.	10/02	14/02	5 egun
I.2	10 ord.	10/02	14/02	5 egun
I.3	10 ord.	16/03	18/03	3 egun
Datu-basea	20 ord.	12/02	19/02	6 egun
DB.1	5 ord.	12/02	14/02	3 egun
DB.2	5 ord.	12/02	14/02	3 egun
DB.3	10 ord.	17/02	19/02	3 egun
Meatzaritza	70 ord.	17/02	13/03	25 egun
MM.1	10 ord.	17/02	21/02	5 egun
MM.2	10 ord.	19/02	21/02	3 egun
MM.3	40 ord.	24/02	13/03	18 egun
MM.4	10 ord.	04/03	13/03	10 egun
Bisualizazioa	68 ord.	17/03	29/05	73 egun
BM.1	3 ord.	17/03	17/03	Egun 1
BM.2	5 ord.	19/03	23/03	3 egun
BM.3	40 ord.	23/03	24/04	32 egun
BM.4	20 ord.	24/04	29/05	35 egun
Memoria	82 ord.	06/02	21/06	136 egun
Mem.1	80 ord.	06/02	19/06	134 egun
Mem.2	2 ord.	20/06	21/06	Egun 1
Defentsarako aurkezpena	8 ord.	10/06	29/06	8 egun
DA.1	3 ord.	10/06	10/06	Egun 1
DA.2	5 ord.	21/06	29/06	7 egun
Plangintza	15 ord.	06/02	14/02	7 egun
P.1	2 ord.	06/02	07/02	2 egun
P.2	10 ord.	06/02	14/02	7 egun
P.3	3 ord.	-	-	-
Bilerak	30 ord.	11/02	20/06	130 egun
B.1	-	-	-	-
B.2	30 ord.	11/02	20/06	130 egun
Jarraipena eta Kontrola	-	06/02	05/06	120 egun
JK.1	2 ord.	06/02	06/02	Egun 1
JK.2	-	10/02	20/06	131 egun
JK.3	-	10/02	20/06	131 egun
JK.4	-	17/02	5/06	109 egun
Osotara	325 ord.	06/02	10/07	155 egun

2.1 Taula: Atazen iraupenaren balioespena

2.1 taulako informazioa modu bisualean ikusteko, hurrengo 2.3 irudiko Gantt diagrama ikusi.



2.3 Irudia: Gantt diagrama.

2.5 Arriskuak eta kalitatea

2.5.1 Arriskuen analisi eta kudeaketa

Edozein proiektu garatzerako orduan arazoak izatea kontuan hartu behar den zerbait da. Atal honetan proiektuaren garapenean gerta daitezken arriskuen azterketa egingo da. Horretarako, arriskuak zerrendatu eta bakoitzari soluzio posible bat emango zaio, proiektuan izango luketen eraginaren arabera.

Arriskuak

- **Pure::variants softwarearen bertsio aldaketa.** Hasieratik aipatu den moduan, proiektu honekin sortutako aplikazioak pure::variants softwarearekin egindako SPL-ak erabiltzen ditu. Software honek bertsio aldaketa egiteak, beraz, zuzeneko eragin handia izango luke proiektu honetan, aldaketa hauek meatzaritza moduluan minatutako fitxategien formatua aldatuko balituzke. Arazo honen aurrean har daitekeen neurri bakarra meatzaritza moduluaren inplementazioa egiaztatzea eta egin beharreko konponketak egitea litzateke. Egoera hau kontuan hartuta, baina baita softwarea modu egituratu batean sortzeko, meatzaritza moduluko funtzionalitateen arteko menpekotasunak murriztea eta hauek taldekatzea erabaki da (*FeatureModel*-arekin harremana duten funtzionalitateak eta *FamilyModel*-arekin dutenak fitxategi desberdinetan idaztea, adibidez). Modu honetan, egin beharreko aldaketak fitxategi zehatz batzuetara mugatuta egongo dira.

- **Irismenaren aldaketa.** Proiektu honek oinarrizko helburu definitua duen arren, ez da irismen osoa hasieratik definitu eta gerta liteke bat-batean funtzionalitateren bat gehitu edo kendu nahi izatea sortutako aplikazioa eraginkorragoa izan dadin. Batez ere bisualizazio modulua da aldaketa gehien jasan ditzakeen zatia. Arriskua gertatzeko probabilitatea oso altua da eta, kasu okerrenean, plangintzaren berregituraketa egin beharko da, ataza honek (P.3 ataza, 2.1 taulan) dedikatuak dituen orduak erabiliz.

- **Ikasketa desbiderapena.** Proiektuan erabilitako teknologia batzuk ezagunak izan arren (Java, HTML, JavaScript...) badaude ikasi beharreko beste batzuk, batez ere bisualizazio modulua garatzeko ez baita teknologia zehatz bat ezarri, eta hauek ikastea balioetsitakoa baino denbora gehiago eska dezake. Ikasketan denbora gehiegi ez dedikatzeko, plangintzan balioetsitako denbora erabiliko da soilik, teknologia hauetan gehienetan ez baita sakondu behar, baizik eta oinarrizko ezagutza lortu. Dena dela, ikasketa denborak balioetsitakoa baino gehiago luzatuz gero, irismenarena maila minimora (2.3 atalean definitua) murriztuko litzateke.

- **Datu-basearen diseinu okerra egitea.** Garatutako aplikazioaren lehen pausoa datubasearen diseinua egitea da, gero honen gainean meatzaritza modulua inplementatu eta azkenik bisualizazioa gauzatzeko. Datu-basearen diseinu okerrak aplikazio guztia berregituratu behar izatea izango luke ondoriotzat. Beraz, DB-aren diseinuaren inguruko zalantzak proiektuko zuzendariekin eta Onekin ikerkuntza taldeko beste

kideekin komentatuko dira hasieratik, diseinu honen gainean inplementazioa hasi baino lehen.

- **Datuen galera.** Informazio sistema guztietan bezala, ezusteko arazoen ondorioz datuen galera izatea gerta liteke, probabilitate altuarekin gainera. Oztopo honi aurre egiteko periodikoki egindako aldaketak *Git* biltegi batean gordeko dira, interneten bidez atzigarri egongo dena *GitHub* plataforman. Modu honetan, nahiz eta uneko datuen galera saihestea ez den lortzen, arazoa modu eraginkorrean pairatzea lortzen da.

2.5.2 Kalitatearen kudeaketa

Gradu Amaierako Lan honen emaitza garatzaile berriek Software Produktu-Lerroak erraz ikasteko aplikazioa izango da. Aplikazioak kalitate ona du jarritako helburuak betetzea lortzen badu, eta kalitate txarra duela bestela. Hala ere, kalitatea neurtzea konplexua denez, atal honetan hura neurtzeko burutuko diren ekintza batzuk definituko dira.

Ekintzak

- **Kodearen gaineko barne probak.** Errorerik gabeko kodea, kalitate oneko kodea dela esan dezakegu. Horregatik, produktuaren inplementazioko fase bakoitzean (datu-basea, meatzaritza eta bisualizazio moduluak) proba batzuk egingo dira zuzentasuna bermatzeko. Proba hauek pasatzea beharrezkoa izango da inplementazioko hurrengo fasera pasatzeko.

- **Kodearen gaineko kanpo probak.** Norberak idatzitako kodean erroreak identifikatzea zailagoa izan daitekeenez, behin barne probak pasata, gaiaren inguruko ezagutza duen garatzaile batek auditoretza bat egingo dio zailak diren erroreak aurkitzeko asmoz.

- **Produktuaren erakusketa probak.** Produktua amaitutzat ematen denean, Software Produktu-Lerroen inguruko ezagutza-maila desberdina duten pertsonei aurkeztuko zaie, hauek erabil eta ebalua dezaten. Puntu honetan proposatutako aldaketak azalekoak izango dira izan jada produktua osatua dagoelako, baina azken erabiltzaileen iritzia oso erabilgarria izan daiteke etorkizunera begira aldaketak txertatzeko.

2.6 Informazio eta komunikazio sistemak

2.6.1 Informazio sistema

Proiektua burutzeko beharrezko informazio guztia egilearen ordenagailu eramangarrian gordeta egongo da, unibertsitateko datuei esleitutako atalean. Honela informazioa ez da ingurune geografiko bakarrera, unibertsitatera esaterako, lotuta egongo.

Gainera, arriskuen analisian aipatu den moduan (2.5.1 azpiatala), egunero egindako aldaketak github plataformako git biltegi batera igoko dira, datuen bat-bateko galera saihesteko. Modu berean, git biltegi honek interneten bidez informazio eguneratua atzitzeko aukera ere eskaintzen du ordenagailu eramangarria izan gabe.

Informazio sistemaren egitura

- **Baliabideak.** Karpeta honetan Gradu Amaierako Lana burutzeko erabilgarriak izan daitezken, baina bere parte ez diren materialak gordeko dira.
  - **GrAL-ak.** Karpeta honetan beste ikasle batzuen GrAL-en memoriak gordeko dira, gidalerro bezala erabiltzeko.
  - **pure::variants.** Karpeta honetan pure::variants teknologiaren inguruko fitxategiak gordeko dira, hala nola erabiltzaile gidak, probarako SPL-ak...
- **Diagramak.** Karpeta honetan proiektuan garatutako aplikazioaren diagrama desberdinak gordeko dira. Adb.: datu-basearen diseinua, klase-diagramak, fluxu-diagramak...
- **eclipse.** Karpeta hau eclipsen Java proiektuak garatzeko *workspace* bezala erabiliko da.
  - ***Git2SPLdb*.** Proiektu hau *FeatureCloud* aplikazioaren meatzaritza modulua da, Onekin taldeko Raul Medeiros ikerlariak garatua.
  - ***SPLMiner*.** Proiektu hau Gradu Amaierako Lan honen minatze modulua da.
  - ***Weather Station Example*.** pure::variants-en egileek garatutako Software Produktu-Lerro hau SPLMiner proiektua probatzeko erabiliko den SPL-a da, hasierako inplementazioetan.
- **eclipse_spring.** Karpeta hau eclipsen Spring proiektuak garatzeko *workspace* bezala erabiliko da.

  - ***FeatureCloud*.** Proiektu honek izen bereko aplikazioaren web interfazea jasotzen du bere barne.
  - ***InsideSPL*.** Proiektu hau Gradu Amaierako Lan honen bisualizazio modulua da.

- **tex.** Karpeta honetan memoria hau idazteko erabilitako fitxategi guztiak gordetzen dira.

2.6.2 Komunikazio sistema

Gradu Amaierako Lan honen zuzendariekin komunikazio laburrak gauzatzeko unibertsitateko posta erabiliko da eta komunikazio luzeagoetarako aurrez-aurreko bilerak, hauek egiteko estimatutako denboraren barruan.

Ikerkuntza taldeko kideekin harremantzeko zuzenean Ingenieritza Informatikoko Fakultateak Onekin ikerkuntza taldeari esleitutako laborategia erabiliko da, hitzorduak aldez aurretik adostuz.

2.7 Interesatuak

Proiektua arrakastaz amaitzeko interesatu nagusia lan honen egilea izango da, bera izango baita lana defendatu eta kalifikazioa jasoko duena. Ikaslearen ardura izango da proiektua zuzen bukatzeko egin beharreko atazak burutzea.

Informatika Fakultateko Onekin ikerkuntza taldea, bezeroaren papera jokatzen duena proiektu honetan, interesatua izango da ere, proiektuaren emaitza SPL-en inguruko ezagutza zabaltzeko erabiltzeko asmoa baitu. Proiektuaren helburuak eta irismena finkatzea izango dira Onekin taldearen ardurak.

Gradu Amaierako Lan honen zuzendariek, Arantza Irastorza eta Maider Azanzak, proiektuaren bilakaera gidatzeko eta horretarako laguntza emateko ardura dutenez, interesatu nagusiak izango dira ere.

Amaitzeko, proiektuaren defentsa ebaluatuko duen epaimahaia bigarren mailako interesatua izango da. Epaimahaiak unibertsitateak definitutako gidalerroak jarraituz proiektua kalifikatzeko ardura izango du.

3. KAPITULUA

Teknologiak

Proiektu hau garatzeko teknologia desberdin asko erabili dira, bai *pure::variants* softwarearekin garatutako Software Produktu-Lerroak minatzeko moduluan, baita bisualizazio moduluan ere. Atal honetan teknologia horien guztien zerrendaketa egingo da eta bakoitzaren erabilera justifikatu.

Gradu Amaierako Lan honen emaitza bi modulu independente direnez teknologiak hiru azpiataletan banatuko dira: teknologia amankomunak, meatzaritza modulukoak eta bisualizazio modulukoak.

3.1 Teknologia amankomunak

Atal honetan bi moduluetan erabilitako teknologiak zerrendatuko dira.

- ***Java*** [1]: helburu orokorreko programazio lengoaia da, klasetan oinarritutakoa eta objektuei zuzendutakoa.

  Proiektua garatzeko Java erabiltzea erabaki da *FeatureCloud* eta *Git2SPLdb* aplikazioak programazio lengoai honekin sortu direlako. Aplikazio hauen integrazioa helburu nagusia ez den arren (2.3 atalean azaltzen den moduan), etorkizunera begira bateragarritasun arazoak txikitze aldera hartu da erabakia.

  Honekin batera, esan beharra dago Java izan dela unibertsitateko ikasturte hauetan gehien erabili den programazio lengoaia aplikazio desberdinak sortzerako orduan

eta maila tekniko altuagoa duela proiektu honen garatzaileak programazio lengoai honetan beste batzuetan baino.

- ***Maven*** [2]: Java proiektuen kudeaketa eta sorkuntzarako erabiltzen den tresna da. 2002 urtean plazaratu zen proiektu independente moduan, 2004. urtean *Apache* fundazioak bereganatu zuen arte.

  Tresna honekin proiektuen menpekotasunak modu errazean kudeatu daitezke. Izan ere, *Maven*ek erabiltzen duen konfigurazio fitxategian lerro batzuk besterik ez dira idatzi behar tresnak automatikoki beharrezko *plugin*-ak deskargatzeko.

  Egun, *Maven*en alternatiba gertuena *Gradle* izeneko tresna da. *Gradle Maven*ek dituen arazo edo zailtasunak errazteko sortu dela esaten da [3]. Hala ere, proiektu hau garatzeko *Maven* erabiltzea erabaki da berriz ere bateragarritasun arazoak saiheste aldera.

- ***MySQL*** [4]: datu-base erlazionalen kudeaketarako sortutako sistema da, mundu mailan ezagunenetarikoa [5], *Oracle* sistemaren ondoren.

  Datu-base, eta ondorioz haien kudeaketa sistemak, mota desberdin asko daude; *Java*n inplementatutako aplikazio batentzat zentzuzkoa litzateke objektuei zuzendutako DB bat erabiltzea, *ObjectDB*, esaterako.

  Proiektu hau garatzeko, aldiz, DB erlazionala erabiltzea erabaki da diseinu fasean meatzaritza modulutik erauzitako datuak aplikazio mota desberdinetarako eskuragarri egotea interesgarritzat jo delako, *MySQL* erabilera orokorreko DB izanda.

- ***Git*** [6]: bertsio-kontrolerako garatutako softwarea da. Bertsio-kontrol sistemek garapen prozesuan dauden proiektuetan egindako aldaketen aztarna jarraitzea ahalbidetzen die garatzaileei.

  Argi dago software mota hauen merkatuan alternatiba ugari daudela [7], baina Git da egun erabiliena [8]; gainera unibertsitatean hainbat proiektu garazeko erabilitakoa izan da. Horregatik guztiagatik izan da hautatua Gradu Amaierako Lana eta bere produktuen bertsio kontrola egiteko.

3.2 Meatzaritza moduluko teknologiak

Atal honetan soilik meatzaritza modulua inplementatzeko erabilitako teknologiak zerrendatzen dira. Zerrenda honetatik kanpo geratzen da *pure::variants*, minatutako SPL-ak

software honekin egin diren arren, garapen prozesuan hura erabiltzea beharrezkoa ez zelako.

- ***Eclipse*** [9]: kode irekiko garapen ingurune integratua (*IDE*, ingeleseko sigletan) da, hainbat programazio lengoai desberdinetako proiektuak sortzea ahalbidetzen duena. *IDE* desberdinak aurki daitezkeen arren, meatzaritza modulua *Eclipse*n garatzea erabaki da produktu finalean eraginik ez duelako eta garatzailea erabiltzera ohituta dagoen garapen ingurunea delako.

- ***JGit*** [10]: *Java* programazio lengoaian inplementatutako *Git* kontrol sistemaren aldaera da. *Eclipse* inguruneako eskuragarri dagoen *plugin* honek *Git* biltegiak irakurri, idatzi eta kudeatzea ahalbidetzen du.

  Garatutako moduluak minatu beharrezko Software Produktu-Lerroak *Git* biltegietan antolatuta zeudenez eta inplementazioa *Java*n egin denez, argi dago JGit dela helburuak betetzeko liburutegi zuzena.

3.3 Bisualizazio moduluko teknologiak

Atal honetan bisualizazio moduluan erabilitako teknologiak azaltzen dira. Modulu hau web aplikazio bat denez, aurreko ataletan baino askoz teknologia gehiago agertzen dira.

- ***Spring Framework*, *Spring Boot* eta *STS***:

  Spring [11] *Java* programazio lengoaiarako sortutako aplikazio garapen ingurunea da. Honen bidez, web aplikazioak egitea posible da, beste aukera askoren artean.

  Spring Boot [12] *Spring* proiektuak hasieratzeko erreminta da, konfigurazio minimo bat zehaztuta martxarako prest dagoen proiektua sortzen duena.

  Spring Tool Suite (STS) Spring aplikazioak gartzeko *Eclipse* motako *IDE* bat da. Honetan beharrezko oinarrizko dependentzia guztiak instalatuta daude jada eta martxan jartzeko baliabide azkarrena da.

  Web aplikazioa garatzeko *Spring* erabiltzea erabaki da *FeatureCloud* aplikazioa teknologia honekin sortu zelako eta, aurretik aipatu den moduan, etorkizunera begira integrazio arazoak ekidin nahi zirelako.

  Gainera, proiektu honen garatzaileak aurreko esperientzia zeukan teknologia honekin eta honek asko erraztu du web aplikazioaren inplementazioa.

Honekin guztiarekin batera, gaur egun *Spring* web aplikazioetarako garapen ingurune nagusia dela kontuan hartu beharra dago ere [13].

- ***Tomcat*** [14]: *Apache* kode irekiko HTTP zerbitzariaren luzapena da, *Java* kodea exekutatzeko ahalmena duena. Honen bidez *Spring* web aplikazioen gisako zerbitzuak interneten atzigarri utz daitezke.

- ***Tiles*** [15]: *Apache* fundazioak sortutako txantiloietan oinarritutako garapen ingurunea (*template framework*) da, web aplikazioetako interfazeen garapena errazteko xedea duena. Teknologia honek orrialde zatiak definitzea ahalbidetzen du, geroago txantiloiak sortu eta hauek berrerabiltzeko. Honen ondorioz, garatzaileak idatzi beharreko kode errepikatua asko murrizten da.

  Nahiz eta 2018. urtearen amaieran proiektu hau garapen aktibotik baztertua izan zen [16], oraindik deskargarako eskuragarri dago eta erabiltzea posible da.

  *Tiles*en egoera zein den jakin arren, proiektu honetan erabiltzea erabaki da garatzaileak teknologia honekin aurreko esperientzia duelako eta enpresetan erabiltzen jarraitzen den teknologia delako.

- ***JavaScript* eta *jQuery***:

  JavaScript [17] goi mailako programazio lengoaia da, gehienetan bezero aldeko kodea exekutatzeko erabilia. *jQuery* [18] *JavaScript*en liburutegietako bat da, hainbat eragiketa errazteko sortua.

  Egun, web orrialde ia-guztietan erabiltzen diren teknologiak dira hauek eta proiektu honetan garatutako aplikazioa ez da salbuespena.

- ***Bootstrap*** [19]: doako eta kode irekiko CSS garapen ingurunea da, aurredefinitutako interfaze-elementu asko eskaintzen dituena eta moldagarria eta mugikorretarako lehentasuna[1] izateko helburua duena.

  Teknologia hau erabiltzea erabaki da garatzaileak aurreko esperientzia asko duelako *Bootstrap*ekin eta ondorioz erabiltzaileen interfazeak azkarrago sortuko zirelako.

- ***Bootstrap tour*** [20]: web aplikazioen funtzionalitateak azaltzeko modu intuitiboa da, aplikazioko orrialde desberdinetan zehar azalpen *tour*-ak egitea ahalbidetzen baitu.

[1] *Responsive and movile-first*

Funtzionatzeko *jQuery* erabiltzen du eta eskuragarri izatekotan, *Bootstrap* ere; hala ere, berezko CSS estiloak erabiltzea posible da bateragarritasun arazoak izanez gero.

Aukera desberdinen artean *Bootstrap tour* erabiltzea erabaki da konfiguratzeko erraza delako eta sektorean ezaguna delako [21].

- ***jsTree*** [22]: kode irekiko *jQuery plugin* bat da zuhaitz diagrama interaktiboak sortzen dituena. HTML eta JSON iturrietako informazioa onartzen du sarrera moduan eta luzapen asko eskaintzen ditu bere baitan.

  Alternatiba ugari aztertu ondoren [23], teknologia hau hautatu da proiekturako beharrezko funtzionalitate guztiak eskaintzen dituelako eta aldi berean konfiguratzeko erraza delako.

- ***MomentJS*** [24]: denbora-tarteak hizkuntza anitzetan (euskara barne) adierazteko *Javascript* liburutegia da.

  Garatutako web aplikazioa eleanitza denez eta liburutegi honek beharrezko hizkuntza guztiak dituenez egokitzat hartu da proiektuan erabiltzeko.

- ***AnyChart*** [25]: grafikoak sortzeko *JavaScript* liburutegia da. Teknologia honen bidez bisualizazio moduluko grafiko gehienak egin dira.

  Teknologia honen aukeraketa B eranskinaren amaieran aurki daiteke.

- ***GoJS*** [26]: grafikoak sortzeko beste *JavaScript* liburutegi bat da. Honekin kontzeptu mapak eta zuhaitz diagramak sortu dira, aurreko liburutegiaren gabeziak osatzeko.

- ***Font Awesome*** [27]: ikonoak web aplikazioetan txertatzeko erreminta da, erabiltzeko oso erraza dena. Teknologia hau erabili da beste alternatiba batzuen aurrean ikonoak txertatzeaz aparte hauen gainean aldaketak egitea ere posible delako.

4. KAPITULUA

Aurrekariak

Gradu Amaierako Lan hau testuinguruan kokatzeko beharrezkoa da ulertzea alde batetik, zer den Software Produktu-Lerro bat eta bestetik, Onekin ikerkuntza taldeko *WacLine* SPL-aren domeinua. Memoriaren zati honetan gai horien inguruko azalpenak jasotzen dira.

4.1 Software Produktu-Lerroak

Softwarearen garapena paradigma desberdinen baitan egin daiteke; ohikoena emaitzatzat software produktu bakarra sortzen duen prozesua da. Produktuaren garapena amaitutzat ematen denean, alde batera utzi eta hurrengo produktuaren garapenera pasatzen da.

Baina, zer gertatzen da hurrengo produktua aurrekoaren oso antzekoa denean? Behar bada hainbat produktuk domeinu berdina izan dezakete, edota funtzionalitate paretsuak.

Software produktu bakarraren garapenean egoera hau ematen denean *copy-paste estrategia*[1] sartzen da jokoan, aurreko jakintzak eta teknologiak *'berrerabiltzeko'* asmoarekin. Hala ere, estrategia honek arazo asko ekar ditzake software produktu horien bizi-zikloan: hainbat produktutan kopiatutako funtzionalitate bat aldatu behar izatea, eredutzat erabilitako produktuak ezkutuko akatsak izatea, produktutik produkturako kodearen degradazioa...

[1] Aurretik egindako produktuak kopiatu eta eredutzat erabiltzen dituen estrategia ez oso gomendagarria, baina software garapenaren komunitatean nahiko orokortua.

Egoera honetan, argi dago antzekoak diren produktuak sortzeko paradigma desberdina erabili behar dela, antzematen diren arazo hauek kontuan hartzen dituena hain zuzen. Erantzuna: Software Produktu-Lerroak.

Software Produktu-Lerro bakoitza amankomuneko nukleo teknologikoa partekatzen duten softwareen familia bat da, aurrez definituriko luzapen eta aldakortasunak dituena. SPL-ek domeinu zehatzei ematen diete erantzuna eta sortutako produktu bakoitzak domeinu horretako berezitasun batzuk izango ditu [28].

SPL-ak sortzeko hainbat gauza izan behar dira buruan:

- **Domeinua.** SPL-aren irteeran aterako diren produktuak domeinu berekoak izango dira, ez baitu zentzu handirik SPL bat erabiltzea domeinu desberdinetako produktuak sortzeko.
- **Ezaugarriak.** Ezaugarri bat funtzionalitate edo eginkizun batekin lotuta dagoen kontzeptua da. Ezaugarri zerrenda domeinuari begira ateratzen da, ez garatu nahi den produktu bakar bati begira. Ideia honekin sartzen da aldakortasuna SPL-en garapenean.

  Ezaugarriek aldakortasun mota desberdinak dituzte (derrigorrezkoak, hautazkoak, alternatiboak... izan daitezke) eta beste ezaugarriekiko senidetasun- eta dependentzia-erlazioak, *WacLine*-ren ezaugarri diagraman (4.2 irudia) ikusi daitekeen moduan.
- **Produktu edo aldaerak.** Hauek SPL-aren emaitzak dira, zati amankomun eta hautazkoak dituzten software produktuak. Aldaera bakoitza ezaugarrien hautaketa zehatz baten ondorioz sortzen da.

Ongi inplementatutako SPL batek onurak dakartza produktu anitzen kostu ekonomiko eta lan karga aldetik [28]. Modu kontrajarrian, kontuan hartu behar da SPL-en paradigma oso konplexua izan daitekeela hasiera batean, ezaugarri batekin lotutako funtzionalitateak kode-fitxategi desberdinetan banatuak gera daitezkeelako. Izan ere, ikasketa prozesu horretan laguntzeko egin da Gradu Amaierako Lan hau.

Pure::variants

SPL-ak sortzeko teknologia eta metodologia ugari daude. Pure::variants kode-anotazioetan oinarritutako SPL garapen tresna da, programazio lengoaiaren independentea dena. Kode-

anotazioak kodean idatzitako oharrak dira, sintaxi zehatz bat jarraitzen dutenak eta produktuaren sorketa fasean erabili ondoren kodetik automatikoki desagertzen direnak. Pure::variants-ekin egindako SPL-ek hurrengo kontzeptu nagusiak dituzte:

- ***Feature Model(s).*** Fitxategi honetan (edo hauetan) sortuko den SPL-aren ezaugarriak (*feature*, ingelesez) jasotzen dira, baita hauen arteko senidetasun-erlazioak, aldakortasun mota eta dependentzia (adb.: E.1 eranskineko atalean).

- ***Family Model.*** Fitxategi honetan domeinuko funtzionalitateak inplementatzen dituzten kode-fitxategiak jasotzen dira, baita hauen arteko senidetasun-erlazioak, aldakortasun mota eta dependentziak. *Feature Model*-aren ideia bera da, baina kode-fitxategiekin (adb.: E.2 eranskineko atalean).

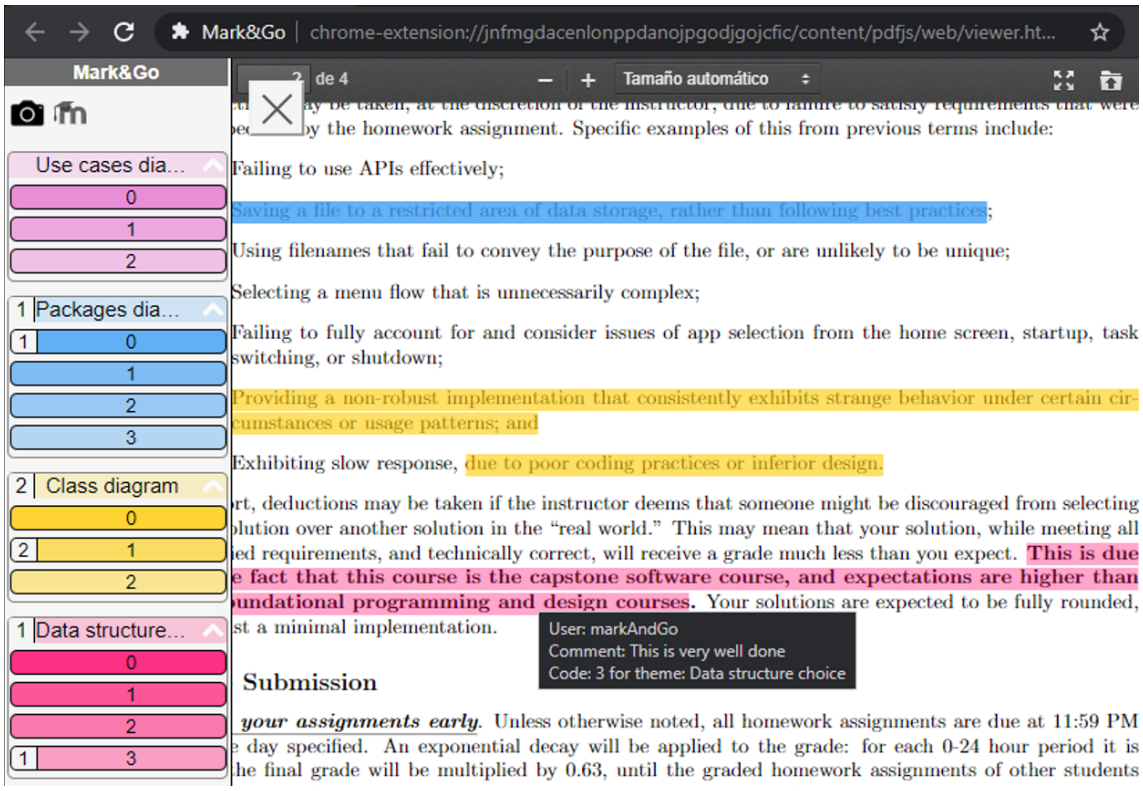
- ***Variant Models.*** Fitxategi hauek produktu desberdinen konfigurazioa biltzen dute, hau da, hautagai dauden ezaugarri eta kode-fitxategi guztietatik zeintzuk diren amaierako produktuan azalduko direnak eta zeintzuk ez (adb.: E.3 eranskineko atalean).

Jada aipatu den moduan, ikus daiteke nola pure::variants softwarearekin egindako SPL-ek kontzeptu amankomun asko dituzten SPL-aren ideia orokorrarekin, baina *Family Model*-aren eta kode-fitxategiak hautatzearen kontzeptua ez da ohiko SPL batean ikusten eta pure::variants-ek eskaintzen duen aukeretako bat da.

4.2 WacLine

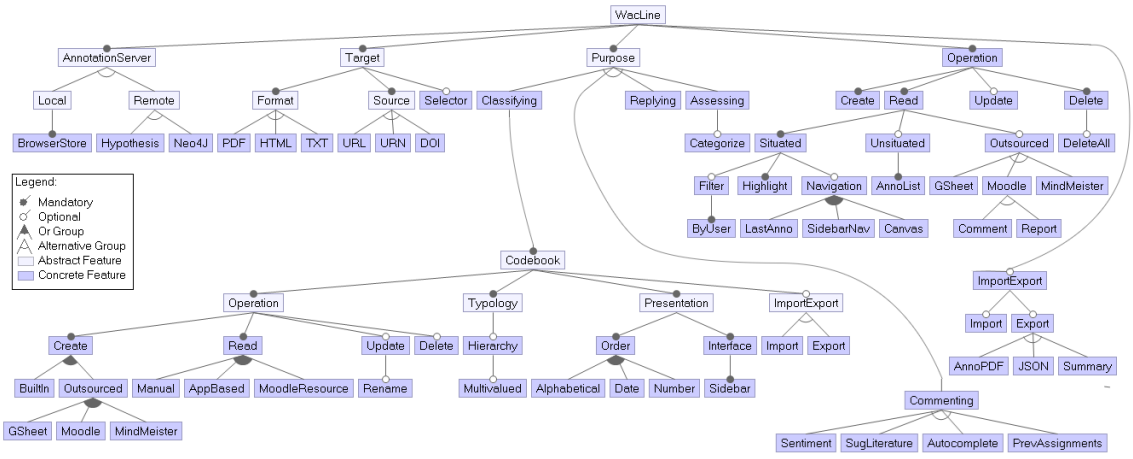
WacLine [29] Donostiako Ingeniaritza Informatikoko Fakultateko Onekin ikerkuntza taldeak garatutako Software Produktu-Lerroa da, pure::variants softwarea erabilita. SPL hau proiektu honen meatzaritza moduluko sarrera moduan erabili da.

WacLine web anotazioen domeinua lantzen du, produktu moduan *Chromium* motako nabigatzaileetarako luzapenak diren anotazio bezeroak sortuz. Web anotazio bat dokumentu edo web orrialde baten gainean egindako testuzko ohar edo marka bat da. Ikusi 4.1 irudia web anotazioen adibide moduan.



4.1 Irudia: MarkAndGo produktuaren pantaila argazkia.

Atal honen helburua ez da *WacLine* SPL-a zehaztasun guztiekin azaltzea, ez baita proiektu honetan garatutako zerbait, baina memoriaren egileak interesgarritzat jo du *WacLine*-ren domeinuko ezaugarriak biltzen dituen diagrama aurkeztea:



4.2 Irudia: *WacLine*-ren ezaugarri diagrama.

Ezaugarri diagrama honetan ikus daitekeenez, *WacLine*-ko produktu bakoitzak hainbat

ezaugarri derrigorrezko izango ditu (diagraman puntu belzdun marraz markatutakoak), hau da, produktu guztiek ezaugarri horiek izango dituzte beti. Beste ezaugarri batzuk, ordea, hautazkoak, haien artean ordezkoak edo haien artean osagarriak dira (puntu zuridun marraz, arku zuriaz edo arku beltzaz markatutakoak, hurrenez hurren); ezaugarri hauek garatzaileak erabakitako konfigurazioaren arabera azalduko dira.

SPL honen tamainaren ideia orokorra egiteko, aurkeztutako ezaugarri diagramatik kalkulatu daiteke *WacLine*-k 5.77*10^13 produktu desberdin sor ditzakeela, konfigurazio konbinaketa guztiak erabiliz eta (irudian azaltzen ez diren) SPL-ak dituen 21 murrizketak errespetatuz.

5. KAPITULUA

Betekizunen bilketa

Gradu Amaierako Lan honek ez du irismen aurredefinitu bat izan hasieratik (2.3 atala) eta honen ondorioz betekizunen bilketa ez da soilik proiektuaren hasierako fasean egin.

Hasieran bildutako betekizunak arkitektura aldetik jarraitu behar den egitura eta bisualizazio moduluaren azaleko aurreikuspenak izan dira (A eranskina). Bestelako betekizunak proiektuaren garapen iteratiboarekin batera jasotzen joan dira, baina atal honetan betekizun guztiak zerrendatuko dira kronologia kontuan hartu gabe.

Aplikazioa diseinatu aurretik ere, beti presente izan da honen *'erabiltzaile prototipoa'* edo *'helburu-erabiltzailea'* SPL-en garapen mundura gerturatzen den pertsona berria dela (ingeleseko *incomer* terminoarekin adierazten dena memorian) eta planteatutako funtzionalitate guztiak SPL-en domeinua ulertzen laguntzeko pentsatuta daude.

5.1 Oinarrizko funtzionalitateak: bisualizazioa

Proiektuaren bi moduluetatik, meatzaritza moduluak SPL-aren informazioa erauzteko ardura du eta ondorioz garatutako aplikazioaren garuna dela esan daiteke. Hala ere, aplikazioaren erabiltzailearentzat modulu honek ez du garrantzi handirik, elkarrekintza bisualizazio moduluarekin egingo baitu. Honen ondorioz, proiektuaren betekizun gehienak bisualizazioaren atalean kokatzen dira.

Aplikazioaren oinarrizko funtzionalitateak erabiltzaile istorioen teknika erabiliz erauzi dira, hau da, *incomer* hipotetikoek izango dituzten kezka edo galderetan garrantzia jarriz

eta ez aplikazioak inplementatu behar dituen funtzio zehatz batzuetan. Modu honetan, atentzioa beti *incomer*-ean dago [30].

Bisualizazioak ikuspegi desberdin asko izan ditzakeen arren, proiektu honetan hiru bisualizazio desberdin jaso dira:

Produktuekin lotutakoak

Ikuspegi honek SPL-en produktuen inguruko informazioa biltzen du. Produktu bat SPL-aren osagaien konfigurazio zehatz bat da, funtzionalitate batzuk eskaini eta beste batzuk alde batera utziko dituena. Hauek dira *incomer*-ak produktuekin lotuta izan ditzakeen galderak:

- Zenbat produktu ditu SPL-ak?
- SPL-aren produktuetatik, zein da handiena/konplexuena eta zein txikiena/sinpleena?
- Produktu bat hartuta, zenbat ezaugarri ditu? Horietatik, zenbat dira hautazkoak eta zenbat derrigorrezkoak?
- Produktu bat hartuta, SPL-aren zenbat kode erabiltzen du?
- Zein da ezaugarri baten eragina/pisua produktu batean?
- Zein da hautatutako produktu baten konfigurazioa? Hau da, ze ezaugarri hautatu ditu eta zeintzuk ez?
- Zeintzuk dira hautatutako bi produkturen arteko antzekotasunak/desberdintasunak?

Ezaugarriekin lotutakoak

Ikuspegi honek SPL-en ezaugarrien inguruko informazioa biltzen du. Ezaugarri bat SPL-aren funtzionalitate batekin edo gehiagorekin egongo da lotuta (gehienetan) eta ezaugarriak ezaugarri diagrametan biltzen dira ,4.1 atalean azaltzen den moduan. Hauek dira *incomer*-ak ezaugarriekin lotuta izan ditzakeen galderak:

- Zenbat ezaugarri ditu SPL-ak?

- Zein da ezaugarrien arteko harremana?
- Ezaugarri bat hautatuta, zein da ezaugarri horren aldakortasun mota? Eta atributurik al du?
- Ezaugarri bat hautatuta, SPL-aren ze aldaketa puntuetan (*variation point*) du eragina?

Kontzeptu mapekin lotutakoak

Azkenik, ikuspegi honetan kontzeptu mapekin zerikusia duen informazioa jasotzen da. Kontzeptu mapak, orokorrean, informazioa aurkezteko diagramak dira, eta gai bati buruzko ikuspuntu zehatza irudikatzeko erabiltzen dira, kontzeptuak erlazionatuz eta hierarkikoki antolatuz. Hiru elementuk osatzen dituzte mapa kontzeptualak: kontzeptuak, horiek lotzeko geziek, eta lotura-hitzak. Azkeneko horiek kontzeptuen arteko erlazioak azaltzen dituzte.

Aplikazio honetan kontzeptu mapak erabiliko dira SPL-en domeinua azaltzeko, ezaugarri diagramaren ezaugarrien eta kontzeptu maparen kontzeptuen arteko lotura eginez.

Kontzeptu mapen ikuspegiaren bidez kezka hauek erantzun nahi dira:

- Zein dira SPL-aren kontzeptu mapak?
- Zeintzuk dira ezaugarri batek lotuta dituen kontzeptuak? Eta baliabideak?
- Zeintzuk dira kontzeptu batek lotuta dituen ezaugarriak? Eta baliabideak?
- SPL-aren produktuetatik, zenbat kontzeptu betetzen ditu bakoitzak?

5.2 Arkitektura

Aplikazioaren funtzionalitateei dagozkien betekizunez gain, bezeroak bere arkitekturaren gaineko betekizunak ere ezarri ditu:

Alde batetik, meatzaritza modulua egongo da SPL-ak minatu eta horietatik informazio guztia aterako duena, ondoren datu-base batean gordetzeko. Meatzaritza modulua *Java* proiektu bat izango da, bere barne dituen funtzionalitateak ahal bezain isolatuak izango dituena. Isolamendu maila altuak etorkizunera begira izan dezakeen ibilbidean lagun

dezake; *FeatureCloud* aplikazioaren integrazioan, adibidez. Minatu beharreko SPL-a Git biltegi batetik lortu beharko da.

Bestetik, bisualizazio modulua egongo da. Modulu honek aurreko datu-baseko informazioa erabiliko du hurrengo puntuan azalduko diren betekizunak betetzeko. Modulu hau web aplikazio bat izango da, *Spring* eta *Java* erabiliz garatuko dena.

Aipatutako datu-baseak SPL-en domeinuko elementuen informazioa gordetzeko prest dagoen eskema bat jarraituz sortu beharko da, domeinua posible den orokorrena izanda. Hau da, domeinua ez da pure::variants teknologiaren baitan egingo, baizik eta SPL-aren ideia orokorraren baitan, ahal den neurrian. Honela, etorkizunean beste teknologia batekin sortutako SPL-en informazioa bisualizatu nahi bada, meatzaritza modulu berria besterik ez da garatu behar, datu-basea eta bisualizazio modulua mantenduz.

Honekin guztiarekin batera, mapa kontzeptualen domeinua ere landuko da, SPL-aren kontzeptuak azaltzeko mapa kontzeptualak erabiliko direlako. Ondorioz meatzaritza modulua eta datu-basea informazio mota hori lortu eta erauzteko prestatu beharko dira ere.

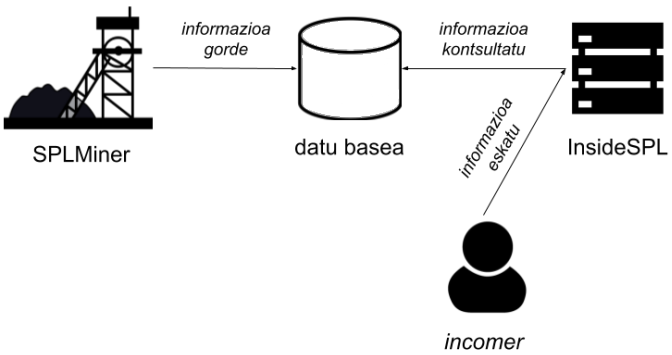
6. KAPITULUA

Soluzioaren diseinua

Proiektuaren betekizunak jaso eta gero (5 atala), kapitulu honetan proposatutako softwarearen diseinua azalduko da. Hasteko, garatutako aplikazioaren arkitektura aurkeztu eta azalduko da; ondoren, erabilitako domeinuen klase-diagrama.

6.1 Arkitektura

Aurreko kapituluetan aipatu den moduan, egindako osagaiek isolamendu maila altua mantendu behar dute etorkizunera begirako lanak errazteko; beraz aplikazioa modulu desberdinetan eta modulu bakoitzaren funtzionalitateak talde desberdinetan garatu dira. Hau hobeto ulertzeko 6.1 irudia ikus daiteke:



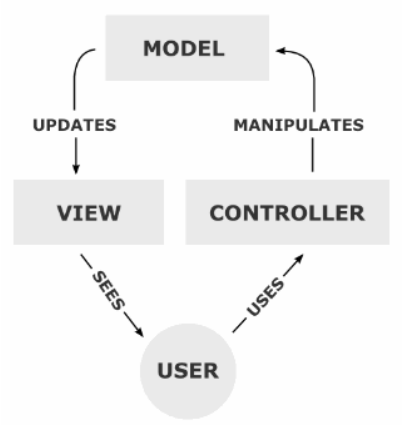
6.1 Irudia: Proiektuaren arkitekturaren diseinua.

Ezkerretik eskuinera hasita, meatzaritza modulua ikus dezakegu, *SPLMiner*. Meatzaritza modulua *Java*-n oinarritutako proiektu bat da, XML fitxategietako informazioa interpretatu eta SQL adierazpenetan itzultzen duena. Honetan funtzionalitate desberdinak garatzen dituzten klaseak daude: alde batetik, pure::variants fitxategiak hartu eta horietatik informazioa erauzten duten meatzariak; bestetik, erauzitako informazioa SQL sintaxia eta sortutako datu-basearen eskema jarraitzen duten adierazpenetan itzultzen duten klaseak. Bi klase mota hauek ondoren azalduko den SPL domeinuaren adar nagusien sailkapena jarraitzen dute: *FamilyModel*, *FeatureModel* eta *VariantModel*.

Meatzaritza moduluaren egitura honen ondorioz, SPL-ak pure::variants ez den beste software batekin inplementatuta egongo balitz soilik meatzari klase desberdin bat sortu beharko litzateke (SQL itzulpena egiten duten klaseak mantenduz) edo erauzitako informazioa beste datu-base mota batean gorde nahi izanez gero, sintaxi hori errespetatzen duen klasea besterik ez.

Arkitektura honen erdiko elementua, meatzaritza moduluak lortutako informazioa gordetzeko ardura duena, SQL datu-basea da. Datu-base honen eskemak 6.3 irudiko klase-diagrama jarraitzen du eta bere eginkizun nagusia bisualizazio moduluari informazioa eskaintzea da. Klase-diagrama eta datu-basearen eskema SPL-aren ideia orokorra adierazten saiatzen dira, etorkizunean meatzaritza edo bisualizazio modulu desberdinek informazioa datu-base berdinarekin kudeatzea errazteko.

Arkitekturaren zehaztapenekin amaitzeko bisualizazio modulua falta da, *InsideSPL*. Bisualizazio modulua web aplikazio bat da, *Spring* teknologiarekin garatutakoa. Web aplikazioak bezero-zerbitzari eredua eta MVC (Model-View-Controller) arkitektura patroia erabiltzen ditu, azkenengo hau hurrengo irudian ikus daitekeena:



6.2 Irudia: MVC (*Model-View-Controller*, ingelesezko sigletan) arkitektura patroia.

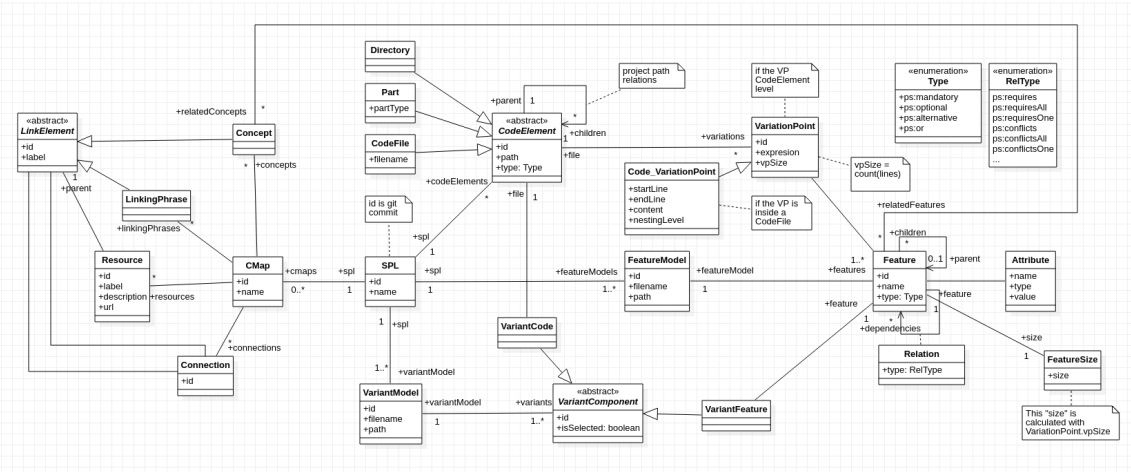
Arkitektura mota honetan bezeroa, *incomer*-a, nabigatzailea erabiliz zerbitzariaren *Controller*-arekin komunikatuko da. *Controller* honek eskatutako informazioa lortuko du *Model*-etik eta azkenik informazio hori jasotzen duen *View*-a eskainiko dio erabiltzaileari.

MVC arkitekturarekin batera hiru geruzako arkitekturaren banaketa erabili da ere, hau da, aurkezpen, negozio logika eta datuen geruzak. Aurkezpen geruza *View*-ean kokatu ondoren eta *Controller*-a erabiltzailearekin komunikatzeko funtzioa izanda, negozio logikaren geruza eta datuen geruza *Model*-aren parte izango dira. *Model*-a izango da, beraz, jarraian azalduko den klase-diagramako klaseak kudeatuko dituen elementua.

6.2 Klase-diagrama

Klase-diagramak bi domeinu desberdin jasotzen ditu: SPL-aren domeinua eta kontzeptu mapen domeinua. Nahiz eta haien artean lotuta egon, oso desberdinak diren domeinuak dira. Horregatik, hasteko klase-diagrama osoa aurkeztuko da eta ondoren domeinu bakoitza zehatzago azalduko da.

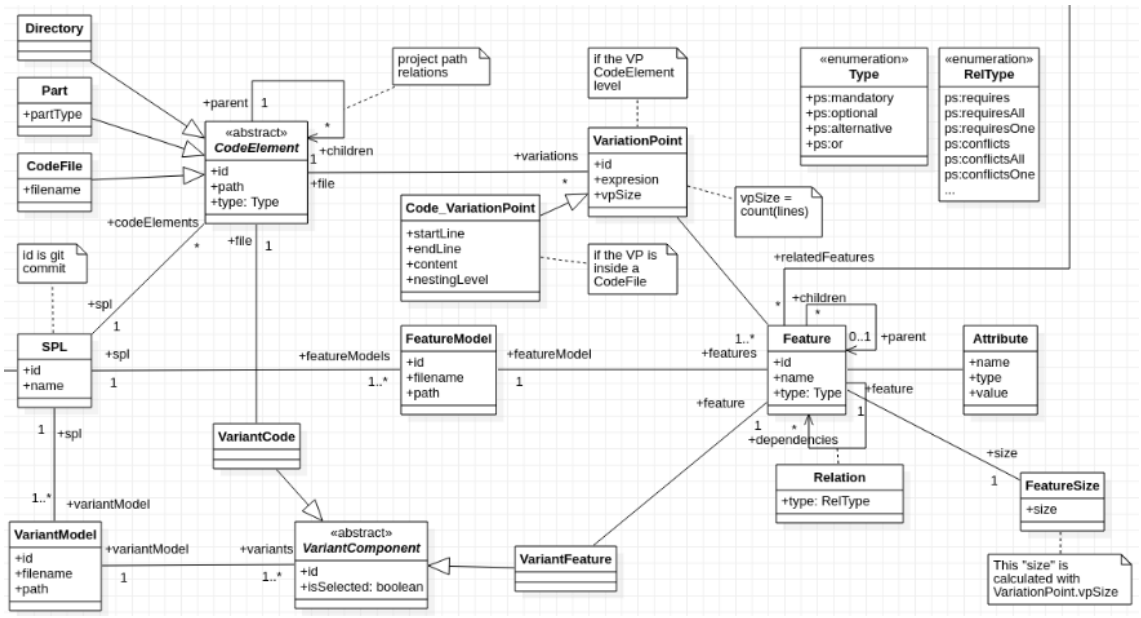
Klase-diagrama hau da datu-baseak jarraitu duena taulen eskema egiteko baita meatzaritza eta bisualizazio moduluek informazioa irudikatzeko jarraitu dutena ere.



6.3 Irudia: Klase-diagrama osoa.

6.2.1 SPL-aren domeinua

Software Produktu-Lerroen domeinua aplikazioaren ardatza da eta 6.4 irudian ikus daiteke:



6.4 Irudia: Klase-diagrama: SPL-aren domeinua.

Aurretik azaldu den moduan SPL-en domeinua ahalik eta orokorren egiten saiatu da teknologia desberdinekin garatutako SPL-ak eredu honekin bat etortzeko. Honen ondorioz, pure::variants-en parte den baina klase-diagraman agertzen **ez** den elementu bat dago, *FamilyModel*-a. *FamilyModel*-ak pure::variants teknologiarekin garatutako SPL-en funtzionalitateak inplementatzen dituzten kode-fitxategiak, karpetak eta bestelako egiturak gordetzen dituzten fitxategiak dira (4.1 atala).

6.4 irudian agertzen diren klaseak hobeto ulertu ahal izateko, azalpen batzuk:

SPL klasea Software Produktu-Lerro baten informazio guztia jasoko duen klasea izango da, SPL-a *Git* biltegi batetik minatutako informazioa izanda. **SPL**-ak **FeatureModel**-ak, **VariantModel**-ak eta **CodeElement**-ak izango ditu.

FeatureModel batek hainbat **Feature** (ezaugarri) izango ditu eta hauetako bakoitzak aldakortasun mota (**Type** enumerazioa) eta beste **Feature** batzuekin senidetasunak. **Feature**-ek **Attribute**-ak izan ditzakete.

CodeElement klasea **CodeFile**, **Directory** eta **Part**[1] klaseen abstrakzioa da eta aurretik aipatutako **FamilyModel**-ak jasotzen zituen elementuak adierazten ditu. Nahiz eta hauetako elementu bakoitzak desberdintasun gutxi izan, kontuan hartzeko klaseak dira **VariationPoint**-ak (aldaketa puntuak) sartzen diren momentuan.

[1] *Sekzio* bezala euskaratuta, pure::variants-en aurkitutako elementua da, SPL-en direktorio eta fitxategien tartean aurkitzen dena

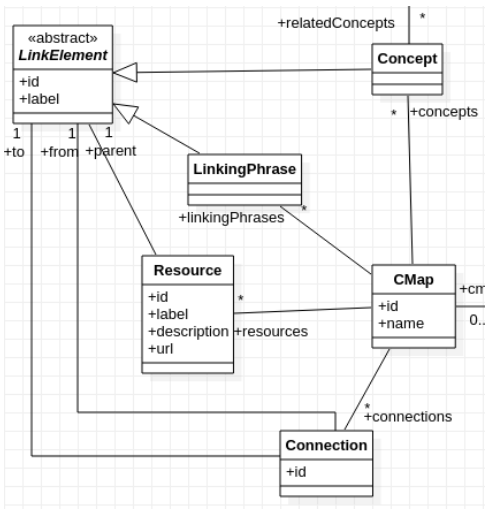
VariantModel-ak produktu desberdinen konfigurazioak dira eta konfigurazio hauetan **VariantComponent**-ak hautatu edo baztertuko dira. **VariantComponent** batek **Feature** bati egiten badio erreferentzia, **VariantFeature** bezala espezializatuko da, eta **CodeElement** bati egiten badio erreferentzia, orduan **VariantCode** izango da. Kode elementuek aldakortasun mota izatea (eta horren ondorioz hautatu edo ez hautatu ahal izatea) pure::variants-en berezitasuna da, normalean kode-fitxategien egitura estatikoa baita.

Domeinu honekin amaitzeko **VariationPoint**-ak falta dira eta hauek SPL-en potentzialaren erroa dira, konfigurazioetan oinarritutako produktuen aldakortasunak inplementatzen baitituzte. **VariationPoint** bakoitza espresio baten bidez adierazten da (boolearra, normalean) eta espresio hori **Feature** batzuekin egongo da lotuta. Bestetik, **VariantPoint**-a **CodeElement** bati (bakarrari) eragingo dio, honetan aldakortasunak sortuz. **CodeElement**-a **CodeFile** bat bada, orduan **Code_VariationPoint** espezializazioa sortuko da. Espezializazio hau testu fitxategietan bakarrik da erabilgarria gordetzen duen informazioa edukia delako, hain zuzen.

Ohar moduan, SPL-en aldakortasun gehiena **CodeFile**-ekin lotuta egon ohi da (*WacLine*-n %90,6-a).

6.2.2 Kontzeptu mapen domeinua

SPL-aren domeinuaren **Feature**-ekin lotuta, 6.5 irudian kontzeptu mapen domeinua ikus daiteke:



6.5 Irudia: Klase-diagrama: Kontzeptu mapen domeinua.

Nahiz eta kontzeptu mapak egiteko erabilitako teknologia *CMaps* izan den, domeinu honetan kontzeptu mapek dituzten klase orokorrak aurkezten dira.

CMap klaseak kontzeptu maparen informazio guztia jasoko du. Kontzeptu mapak **LinkElement**-ak (konexioetan parte hartzen duten elementuak) ditu, **LinkingPhrase** (lotura-esaldi) edo **Concept** (kontzeptu) izan daitezkeela. **Connection** (lotura) bakoitza bi **LinkElement** lotuko ditu (*CMap*-en kasuan **Concept** eta **LinkingPhrase** bana). **LinkElement**-ek **Resource**-ak (baliabideak) izan ditzakete lotuta.

Aurreko ataleko domeinuarekin lotura **CMap** eta **Concept** klaseek egiten dute. **SPL** bakoitzak bere domeinua azaltzen duten hainbat **CMap** izan ditzake eta kontzeptu maparen **Concept** bakoitzak SPL-aren zenbait **Feature** azal ditzake.

7. KAPITULUA

Soluzioaren garapena

Kapitulu honetan, proposatutako diseinua garatzerakoan hartutako erabakiak, funtzionalitateen ezaugarriak eta izandako arazoak azalduko dira. Kapitulu hau bi ataletan zatituko da: meatzaritza moduluaren garapena, batetik; eta bisualizazio moduluarena, bestetik. Izan ere, nahiz eta bisualizazio moduluak meatzaritza moduluak sortutako informazioaren menpekoa izan, garapen prozesuak era independentean eman dira. Atal bakoitzean kode zati esanguratsuak azalduko dira.

7.1 *SPLMiner*: meatzaritza modulua

Modulu honen ardura pure::variants softwarearekin egindako SPL-en informazioa SQL datu-base batera iraultzea izanda, bi funtzionalitate mota desberdin inplementatzen duten klaseak aurki daitezke: meatzariak eta SQL adierazpenen itzultzaileak. Erabaki hau aplikazioaren diseinu fasean eman zen eta justifikazioa 6.1 atalean aurki daiteke.

Bi klase mota hauetaz aparte, konfigurazioa jaso eta funtzionalitate guztiak era egokian erabiltzen dituen SPLMiner aplikazioaren klase nagusia, exekutagarria, dago. *SPLMiner* martxan jartzeko behar diren datuak eta bere funtzionamenduaren azalpen laburra atal honen amaieran emango da.

Ez da ahaztu behar SPL-aren domeinua lantzeaz aparte kontzeptu mapen domeinua ere landu dela proiektu honetan eta ondorioz *CMap* softwarearekin sortutako XML fitxate-

giak prozesatzeko ere meatzariak eta SQL adierazpenen sortzaileak daudela modulu honetan.

7.1.1 Meatzariak

Meatzari baten ardura XML fitxategiak irakurri eta informazioa jasotzen duten *Java* klaseak sortzea da. Pure::variants-ek hiru XML fitxategi mota sortzen dituenez (*FamilyModel*, *FeatureModel* eta *VariantModel*), hiru meatzari desberdin egongo dira fitxategi hauetatik informazioa erauziko dutenak. Hauekin batera testuzko fitxategiak (kode fitxategiak gehienetan) pure::variants-en anotazioen bila minatuko duen beste meatzari bat eta CMap softwareak sortutako fitxategiak minatzen dituen meatzaria egongo dira.

Meatzari guztiek `mine()` edo `mineAll()` motako funtzio publikoak izango dituzte eta haien bitartez SPL-aren informazioa osatzea lortuko da.

- **FeatureModelMiner.** Meatzari honek SPL-aren ezaugarrien inguruko informazioa minatuko du pure::variants-ek sortutako `.xfm` luzapeneko fitxategiak prozesatuz. Meatzariari deia `mineAll()` funtzioarekin egiten da eta honen eragina aurretik sortutako `spl` elementuaren informazioa eguneratzea izango da.

  Aurrebaldintza: `spl` instantzia sortua bere `FeatureModel`-ekin, baina hauetan ez dago ezaugarrien inguruko informaziorik.

  Postbaldintza: `spl` instantzia eguneratua, `FeatureModel`-ean ezaugarri guztiak eta haien arteko dependentziak gorde dira.

- **FamilyModelMiner.** Meatzari honek SPL-aren fitxategien egituraren inguruko informazioa minatuko du pure::variants-ek sortutako `.ccfm` luzapeneko fitxategiak prozesatuz. Meatzariari deia `mineAll(familyModelPaths)` funtzioarekin egiten da eta honen eragina aurretik sortutako `spl` elementuaren informazioa eguneratzea izango da. `familyModelPaths` atributuak meatzariak minatu beharreko fitxategien kokapenen zerrenda izango da, klase-diagraman ez baitago `FamilyModel` klaserik informazio hori gordetzeko.

  Aurrebaldintza: `spl` instantzia inolako `CodeElement`-ik gabe.

  Postbaldintza: `spl` instantzia eguneratua, `CodeElement` guztiak eta haien arteko dependentziak gordeta.

`CodeElement`-ak `CodeFile` direnean, hauek aldaketa puntuak (*Variation Point*) dituzten fitxategiak direnentz egiaztatzen da. Horretarako funtzio laguntzaile bat erabiltzen da:

```
1  private static List<String> notConsideredVPFiles = Arrays.asList("png", "jpg", "svg","
       properties","gif");
2
3  private static boolean isVPFile(String name) {
4      String[] splitName = name.split("\\.");
5      String format = splitName[splitName.length - 1];
6      if(notConsideredVPFiles.contains(format))
7          return false;
8
9      return true;
10 }
```

Funtzio honek egiazko balioa itzultzen badu, jarraian azalduko den CodeMiner meatzaria erabiliko da.

- **CodeMiner.** Meatzari hau aurreko meatzariaren laguntzailea da, `CodeElement`-ak mota zehatz batzuetako `CodeFile` direnean fitxategiak pure::variants-en anotazioen bila minatzeko ardura duena. Meatzariari deia `extractVPsFromFile(codeFile,fileType)` funtzioarekin egiten da eta honek aldaketa puntuak bilatuko ditu fitxategian. `codeFile` atributuak minatu beharreko fitxategiari egiten dio erreferentzia eta `fileType` fitxategiaren motari, izan ere, fitxategi motaren arabera minatu mota desberdinak aplikatuko baitira.

  Aurrebaldintza: `CodeElement`-a `CodeFile` motakoa da eta aldaketa puntuak izan ditzake.

  Postbaldintza: `CodeFile` prozesatua eta, aldaketa puntuak izanez gero, hauen inguruko informazioa `spl` instantzian gordeta.

CodeMiner-ak pure::variants-en anotazioak bilatuko ditu, baina anotazio zehatz batzuetaz arduratuko da soilik. Anotazio horiek *SPLMiner* klase nagusian definituta daude eta hurrengoak dira:

```
1  public static final String[] VARIATION_POINT_NO_XML_STATEMENTS = { "PV:IFCOND", "PVSCL:
       IFCOND" };
2  public static final String[] VARIATION_POINT_NO_XML_STATEMENTS_END = { "PV:ENDCOND", "
       PVSCL:ENDCOND" };
```

Hauekin batera XML adierazpenak ere prozesatzen dira baina hauek modu desberdinean, horregatik da beharrezkoa `fileType` atributua. Fitxategien sailkapenaren beharra ulertzeko, hona hemen aldaketa puntuak detektatzeko funtzioak XML fitxategietarako eta ez-XML fitxategietarako, hurrenez hurren:

```
// XML fitxategiak
private static boolean hasVPInside(String expresion) {
    // Return true if it has 1 VP or more
    return (expresion.split("pv:condition").length - 1) >= 1;
}
```

```
// ez-XML fitxategiak
private static Pair<String, ArrayList<Feature>> hasVariationStatement(String line) {
    for (String s : MainClass.VARIATION_POINT_NO_XML_STATEMENTS) {
        if (line.contains(s)) {

            // Capture what is between (...)
            // (\(((\w+|\w+\(.*\)|[\s\'\.\,\:\-\>\=\<])+\))
            Pattern p = Pattern.compile("(" + INSIDE_BRACKETS+ ")");
            Matcher m = p.matcher(line);
            if (m.find()) {

                String r = m.group()
                        .replaceAll(" or | OR | and | AND | NOT | \\| | \\&", " ");

                // HAS VARIATION STATEMENT
                Pair<String, String> cleaned = cleanVariationStatement(line, s);
                String expr = cleaned.a;
                String rest = cleaned.b;

                // PROCESS THIS LINE AGAIN
                checkVPString = true;
                vpString = rest;

                return new Pair<String, ArrayList<Feature>>(expr, extractVPsFromStatement(
    r));
            }
        }
    }

    return null;
}
```

Adibide honekin argi geratzen da XML fitxategien minatu prozesua askoz sinpleagoa dela beste testuzko fitxategienenarekin konparatuz.

Meatzari hau moduluaren garapen prozesu osoan egin den konplexuena da, aldake-

ta puntuak identifikatzeko hainbat adierazpen erabiltzeaz aparte habiaratutako adierazpenak ere kontuan hartu behar direlako. Habiaratutako adierazpenak (nested VP, ingelesez) adierazpen bat beste baten barnean azaltzen denean sortzen dira, adibide honetan bezala:

```
// PV:IFCOND('A')
    System.out.println('Feature A is selected');
    // PV:IFCOND('B')
    System.out.println('Feature B too');
    // PV:ENDCOND
// PV:ENDCOND
```

Pure::variants-en sintaxia ulertu gabe, suposa daiteke lehenengo `System.out.println` deia azaltzeko `A` izeneko ezaugarria agertu behar duela nonbait, baina bigarren `System.out.println` deia azaltzeko $A \wedge B$ ezaugarriak agertu behar direla (nahiz eta soilik `B` jarri). Habiaratutako adierazpenen ondorioz ez da soilik momentuko testulerroa prozesatu behar, baizik eta aurreko lerroen informazioa kontuan hartu ere.

Habiaratutako adierazpenez aparte, hauen agerpenaren ordenak garrantzi handia du, lerro berean adierazpen baten amaiera eta beste baten hasiera ager daitezkeelako, hauen esanahia guztiz aldatuz. Adibidez:

```
/* PV:IFCOND('A') */
    String feature = "A";
/* PV:ENDCOND */ /* PV:IFCOND('B') */ String feature2 = "B";
/* PV:ENDCOND */

edo

/* PV:IFCOND('A') */
    String feature = "A";
/* PV:IFCOND('B') */ String feature2 = "B"; /* PV:ENDCOND */
/* PV:ENDCOND */
```

Adibide honetan, lehenengo kasuan `feature` aldagaia `A` ezaugarriaren menpe dago eta `feature2` aldagaia `B` ezaugarriaren menpe. Bigarren kasuan, aldiz, `feature` aldagaia `A` ezaugarriaren menpe dago baina `feature2` aldagaia $A \wedge B$ ezaugarrien menpe. Adierazpenen agerpenak, beraz, sekulako garrantzia izango du prozesamendu zuzen bat egiteko.

- **VariantModelMiner.** Meatzari honek SPL-aren produktuen konfigurazioaren inguruko informazioa minatuko du pure::variants-ek sortutako `.vdm` luzapeneko fitxategiak prozesatuz. Meatzariari deia `mineAll()` funtzioarekin egiten da eta ho-

nen eragina aurretik sortutako `spl` elementuaren informazioa eguneratzea izango da.

> *Aurrebaldintza:* `spl` instantziak `VariantModel`-ak ditu, baina hauetan ez dago konfigurazioaren inguruko informaziorik, soilik fitxategien kokapena eta izena.
>
> *Postbaldintza:* `spl` instantzia eguneratua, `VariantModel`-ean konfigurazioaren informazio guztia gorde da, hautatutako eta hautatu gabeko `VariantCode` eta `VariantFeature`-ak, hain zuzen.

Konfigurazio batean hautagarri dauden ezaugarri eta kode elementu guztiak hautatuak diren edo ez gordeko da, baina pure::variants-ek ez du esplizituki adierazten ezaugarri edo kode fitxategi batzuen ez-hautaketa (hauen gurasoa ez badago hautatua umeak ezin dira hautatuak izan eta ondorioz, fitxategian umeak ez dira aipatzen). Salbuespen hauek kontrolatzeko SPL-aren ezaugarri zein kode elementu guztien zerrenda egiaztatzen da eta tratatu gabeko elementuak ez-hautatu moduan markatzen dira, hurrengo kode zatian ikus daitekeen moduan:

```
1  // Treat unselected Features that doesn't appear on the file
2  for (Feature f : untreatedFeatures) {
3      VariantFeature vf = new VariantFeature(GenericUtils.generateID(), false, vm, f);
4      vm.addVariant(vf);
5      MainClass.getLogger().info("(" + vf.getId() + ") Untreated Feature \"" + vf.getFeature
       () + "\" is unselected");
6  }
7
8  // Treat unselected CodeElements that doesn't appear on the file
9  for (CodeElement ce : untreatedCodeElements) {
10     VariantCode vf = new VariantCode(GenericUtils.generateID(), false, vm, ce);
11     vm.addVariant(vf);
12     MainClass.getLogger().info("(" + vf.getId() + ") Untreated CodeElement \"" + vf.
       getCodeFile() + "\" is unselected");
13 }
```

- **CMapMiner.** Meatzari honek SPL-aren kontzeptuak azaltzen dituen kontzeptu mapen inguruko informazioa minatuko du CMap softwareak sortutako `.cxl` luzapeneko fitxategiak prozesatuz. Meatzariari deia `mineAlone(filepath,relatedSplId)` edo `mineWithConnections(filepath,SPL)` funtzioekin egiten da.

  Lehenengo funtzioak fitxategiko kontzeptuak, geziak eta lotura-hitzak erauzten ditu, baina ez du SPL-aren ezaugarriekin loturarik egiten.

Aurrebaldintza: SPL-ak kontzeptu mapa bat dauka baina honetan ez da SPL-aren ezaugarriekin loturarik adierazten.

Postbaldintza: SPL-an kontzeptu maparen informazioa gorde da.

Bigarren funtzioak, ordea, fitxategiko kontzeptuak, geziak eta lotura-hitzak erauzten ditu eta SPL-aren ezaugarriekin lotura sortu.

Aurrebaldintza: SPL-ak kontzeptu mapa bat dauka eta honetan SPL-aren ezaugarriak azaldu egiten dira, kontzeptu eta ezaugarrien arteko lotura emanez.

Postbaldintza: SPL-an kontzeptu maparen informazioa gorde da eta kontzeptuek SPL-aren ezaugarriekin izango dute lotura.

Azkenengo funtzio honen kasuan, ezaugarri eta kontzeptuen arteko lotura esplizitua behar du izan, hau da, CMap softwarearekin egindako kontzeptu mapan lotutako ezaugarrien izena idatzi egin behar du maparen sortzaileak. Honetarako, CMap-en `short-comment` atributua erabili da eta `"PV: EzaugarriIzena"` sintaxia. Lotura hauek prozesatzeko funtzio lagungarri hau erabili da, SPL-aren ezaugarrien izenak atributuaren balioarekin konparatzen dituena:

```
1  private static ArrayList<Feature> connectLinkElementWithFeatures(String line) {
2
3      if(spl == null)
4          return new ArrayList<>();
5
6      ArrayList<Feature> listfeatures = new ArrayList<>();
7      for (FeatureModel fm : spl.getFeatureModels()) {
8          for (Feature f : fm.getFeatures()) {
9              Pattern p = Pattern.compile("\\b("+f.getName()+")\\b");
10             Matcher m = p.matcher(line);
11             if (m.find()) {
12                 listfeatures.add(f);
13             }
14         }
15     }
16     return listfeatures;
17 }
```

7.1.2 SQL itzultzaileak

Itzultzaile baten ardura SPL-aren informazioa jasotzen duten *Java* klaseetatik SQL adierazpenak sortzea da. Itzultzaileen kasuan ere, pure::variants-en fitxategien sailkapena era-

biltzen da (*FamilyModel*, *FeatureModel* eta *VariantModel*). Hauekin batera klase nagusi bat eta CMap domeinuko klaseen SQL itzulpena egiteko klasea ere aurkituko dira.

Itzultzaile guztiek `generateAllInserts()` motako funtzio publikoa izango dute eta honen bitartez SPL-aren informazioa SQL adierazpenetara itzultzen joango da.

- **MainSql.** Klase honen ardura sortutako adierazpen kudeaketa eta SPL-aren inguruko SQL adierazpenak sortzea da. Adierazpenen kudeaketarako tresna hauek eskaintzen ditu:

```
1 // Insert-ak gordetzeko zerrenda, pribatua
2 private static ArrayList<String> inserts = new ArrayList<>();
3
4 // Zerrenda kudeatzeko metodoak
5 public static void addInsert(String s);
6 public static ArrayList<String> getInserts();
7 public static void emptyInserts();
8 public static void exportToFile(String filepath);
```

- **FeatureModelDB.** Itzultzaile honek ezaugarrien inguruko informazioa SQL adierazpenetan itzuliko du. Itzultzaileari deia `generateAllInserts()` funtzioarekin egiten da eta honen eragina `inserts` zerrendan ezaugarrien SQL adierazpenak batzea izango da.

  Aurrebaldintza: `inserts` zerrendan ez dago ezaugarrien inguruko adierazpenik, soilik SPL-aren ingurukoak.

  Postbaldintza: `inserts` zerrendan ezaugarrien inguruko adierazpen guztiak gorde dira.

- **FamilyModelDB.** Itzultzaile honek kode elementuen inguruko informazioa SQL adierazpenetan itzuliko du. Itzultzaileari deia `generateAllInserts()` funtzioarekin egiten da eta honen eragina `inserts` zerrendan kode elementuen eta aldaketa puntuen (VP) SQL adierazpenak batzea izango da.

  Aurrebaldintza: `inserts` zerrendan ezaugarrien inguruko adierazpenak daude.

  Postbaldintza: `inserts` zerrendan kode elementuen eta aldaketa puntuen inguruko adierazpen guztiak gorde dira, eta hauek ezaugarriekin duten lotura ere.

- **VariantModelDB.** Itzultzaile honek produktuen konfigurazioen inguruko informazioa SQL adierazpenetan itzuliko du. Itzultzaileari deia `generateAllInserts()` funtzioarekin egiten da eta honen eragina `inserts` zerrendan konfigurazio bakoitzean azaltzen diren hautapenak gordetzea izango da.

  Aurrebaldintza: `inserts` zerrendan ezaugarri eta kode elementu guztiak daude.

  Postbaldintza: `inserts` zerrendan produktu bakoitzaren konfigurazioa gorde da, hauetako bakoitzean ezaugarri zein kode elementu bakoitza hautatua denentz adieraziz.

- **CMapDB.** Itzultzaile honek CMap domeinuko SQL adierazpen guztiak sortu eta esportatzeko ardura du, MainSql-ren antzera. Itzultzaileari deia `generateInserts(cmap)` funtzioarekin egiten da eta honek pasatako `cmap` instantziak gordetzen duen informazio guztia `inserts` zerrenda propio batean gordetzen du.

  Aurrebaldintza: `inserts` zerrenda propioa hutsik dago.

  Postbaldintza: `inserts` zerrendan kontzeptu maparen informazio guztia gorde da eta esportatzeko prest dago.

  Behin aurreko funtzioa exekutatuta, `exportToFile(filepath)` funtzioa erabili daiteke SQL adierazpenen zerrenda fitxategi batean gordetzeko.

  Itzultzaile honek ez du aurrekoekin elkarlanik egiten domeinu desberdinetako klaseak kudeatzen dituelako eta isolatuta mantendu nahi delako.

7.1.3 Klase nagusia

Atal honetan konfigurazioa jaso eta funtzionalitate guztiak era egokian erabiltzen dituen SPLMiner aplikazioaren klase nagusia azalduko da. Klase nagusia *Git* biltegi bat irakurtzen saiatuko da eta automatikoki pure::variants-en fitxategiak detektatu. Prozesua amaitzerakoan miatutako SPL-aren informazioa *Git* biltegiaren erro karpetan sortutako SQL fitxategi batean gordeko da.

Klase nagusiak hiru modu desberdinetan egin dezake SPL-aren minatua:

1. **SPL-aren minatu osoa.** *FeatureModel*, *FamilyModel* eta *VariantModel* guztiak.

2. **SPL-aren minatu partziala CMap konexioetarako.** Soilik *FeatureModel* minatzen da ezaugarri eta kontzeptuen arteko lotura egiteko. Gainera ez da SPL-aren SQL fitxategirik sortzen, soilik CMap-ena.

3. **SPL-aren minatu osoa eta CMap konexioak.** Aurreko bi aukerak konbinatu eta SQL fitxategi guztiak sortzen dituen aukera.

Minatze modua, beste hainbat gauzen artean klase nagusiaren konfigurazio atalean zehazten da. Konfigurazio atalak informazio hau jasotzen du:

```
/* SPL konfigurazioa */

// Minatu beharreko SPL-aren .git/ karpetaren kokapena
private static final String SPL_LOCAL_GIT_REPO = "/path/to/git_repo";

// Kode elementuak aurkitzen diren karpeta (normalean '/input')
private static final String SPL_CODE_FOLDER = SPL_LOCAL_GIT_REPO + "/input";

// SPL-aren izena, gero aplikazioan agertuko dena
private static final String SPL_NAME = "WacLine";

// Minatu mota
private static final int MINING_TYPE = 3;


/* CMap konfigurazioa (MINING_TYPE == 2 OR 3) */

// CMap fitxategiak aurkitzen diren karpeta
private static final String CMAPS_FOLDER = "/cmaps";

// Minatu nahi diren CMap-en kokapena, zerrendatuta
private static final String[] CMAPS = {
            SPL_LOCAL_GIT_REPO + CMAPS_FOLDER + "/WebAnnotation.cxl"
        };
```

Behin informazio zuzena zehaztuta, klasea exekutatzearekin aski izango da prozesua hasteko eta automatikoki fitxategi guztiak detektatzeko. Horretarako, *JGit* eta fitxategien luzapenak erabili dira, kode zati honetan ikusten den moduan:

```
// Autodetect all files on specified folder
RevTree tree = lastCommit.getTree();
TreeWalk treeWalk = new TreeWalk(repo);
treeWalk.addTree(tree);
treeWalk.setRecursive(true);

while (treeWalk.next()) {
```

```
8       String path = treeWalk.getPathString();
9
10      if (isFeatureModel(path)) {
11          FeatureModel fm = new FeatureModel(SPL_LOCAL_GIT_REPO + "/" + path);
12          spl.addFeatureModel(fm);
13      } else if (isFamilyModel(path)) {
14          familyModelPaths.add(SPL_LOCAL_GIT_REPO + "/" + path);
15      } else if (isVariantModel(path)) {
16          VariantModel vm = new VariantModel(SPL_LOCAL_GIT_REPO + "/" + path);
17          spl.addVariantModel(vm);
18      }
19
20  }
```

7.2 *InsideSPL*: bisualizazio modulua

Garatutako proiektuaren arkitekturaren azkenengo modulu honek *incomer*-arekin elkarrekintza izateko eta datu-basean gordetako informazioa erakusteko eginkizuna betetzen du. Aurreko kapituluetan aipatu den moduan hiru ikuspegi garatzea erabaki da, modu iteratiboan, SPL-en domeinua azaltzeko:

1. **Produktu ikuspegia.** Ikuspegi honetan SPL-aren produktu desberdinen informazioa erakusten da, hala nola konfigurazioa, kode-estaldura eta produktuen arteko alderaketa.

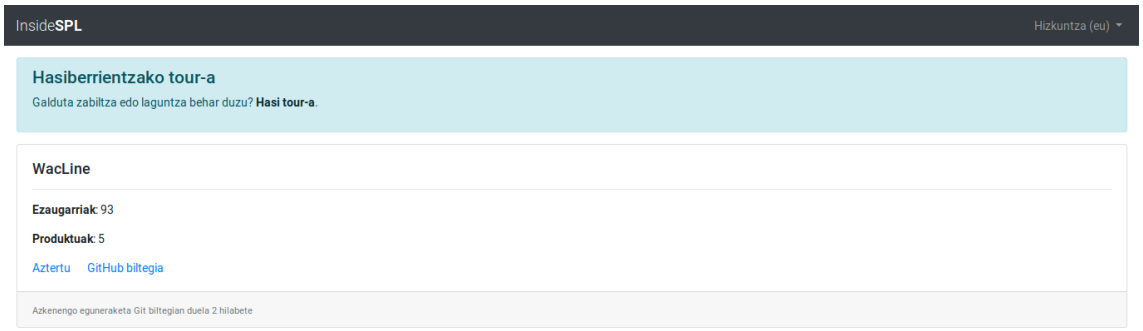
2. **Ezaugarri ikuspegia.** Ikuspegi honek SPL-aren ezaugarri diagramak, ezaugarriak eta hauek aldaketa puntuekin duten harremana erakusten du.

3. **Kontzeptu mapen ikuspegia.** Kotzeptu mapak SPL-aren domeina azaltzeko tresna baliotsua da eta horregatik, azkenengo ikuspegi honek SPL-a azaltzen duten kontzeptu mapak eta mapako kontzeptuek ezaugarriekin duten lotura erakusten ditu.

Hauetako ikuspegi bakoitzak SPL-aren jakintzaren arlo desberdinak estaltzen ditu (5.1 atalean definitutakoak). Funtzionalitate hauek guztiak erabiltzaileak eskuragarri dituen web orrialde desberdinetan azaltzen direnez, atal honetan orrialde horiek aurkeztuko dira eta horietako bakoitzean dauden funtzionalitate desberdinak. Amaitzeko, web aplikazioaren internazionalizazioa (hizkuntza anitzen agerpena) azalduko da.

7.2.1 Hasierako orrialdea

Orrialde honetan datu-basean dauden SPL guztiak zerrendatuko dira *incomer*-ak aztertu nahi duena hautatu dezan (7.1 irudia). Honekin batera aplikazioa erabiltzen berriak diren erabiltzaileentzat *'Hasiberrientzako tour'*-a dago eskuragarri, aplikazioaren funtzionalitate guztiak azaltzen dituena. Behin SPL bat hautatuta, hasierako orria ikuspegi desberdinen zerrenda batean bihurtuko da (7.2 irudia) eta honen bidez ikuspegi batetik besterako aldaketa egin ahal izango da.



7.1 Irudia: Hasierako orria, SPL-ak hautatzeko zerrenda.

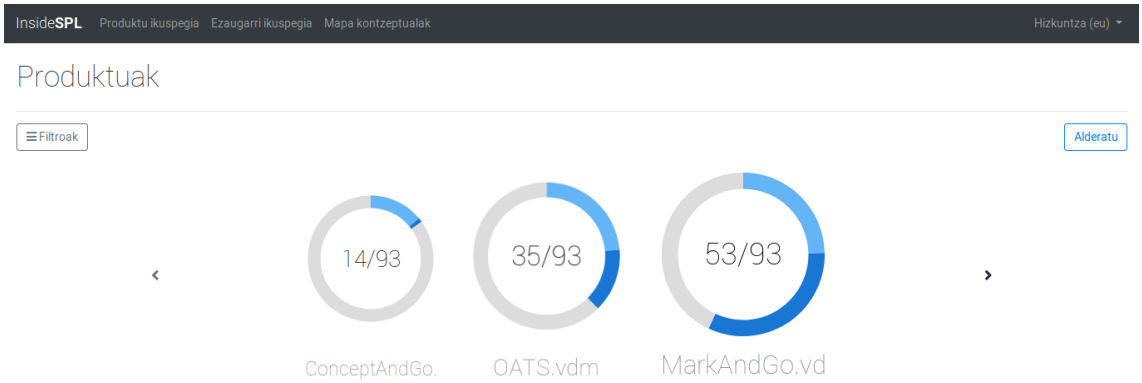
7.2 Irudia: Hasierako orria SPL-a hautatu ondoren, ikuspegien zerrenda.

7.2.2 Produktu ikuspegia: zerrenda

Orrialde honetan SPL-aren produktu desberdinen zerrenda aurkezten da. Produktuak era grafikoak aurkezteko B eranskineko proposamena erabili da. Honetan produktu batek bi grafiko desberdin izango ditu:

- **Grafiko sinplea.** Grafiko honetan produktuak dituen derrigorrezko ezaugarri hautatuak, hautazko ezaugarri hautatuak eta ezaugarri hautatu gabeak azaltzen dira zenbakizko balio soil bezala. Grafiko hauek produktu desberdinen tamainak era intuitiboan alderazteko pentsatuak daude eta zenbakizko erreferentziaz aparte grafikoaren tamainak ere handiagoak edo txikiagoak izango dira (ikusi 7.3 irudia).
- **Grafiko osoa.** Grafiko honetan SPL-a osatzen duten ezaugarri guztiak hierarkikoki azaltzen dira (guraso-ume erlazioa mantenduz, alegia) eta grisa ez diren koloreen bidez ezaugarri hauetako zenbat kode estaltzen den azaldu nahi da. Grafikoaren erdian produktuaren kode-estaldura ehunekotan azaltzen da (ikusi 7.4 irudia).

Bi grafiko hauetatik, produktuen zerrendarako grafiko sinpleak erabili dira eta *incomer*-ak filtro desberdinak erabil ditzake zerrenda horretatik produktu batzuk kanpo uzteko. *'Alderatu'* botoia sakatuz gero, zerrendako bi produktu hautatu eta haien konparaketa erakutsiko duen orrira joateko aukera egongo da.



7.3 Irudia: Produktu ikuspegia, produktuen zerrenda.

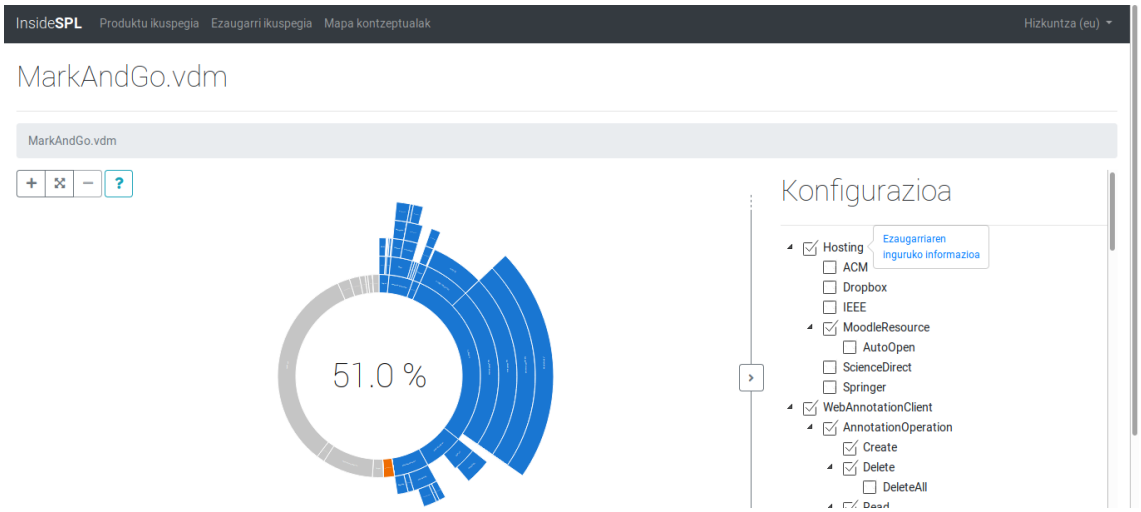
7.3 irudian ikus daiteke produktuak *'sinpleenetik konplexuenera'* zerrendatuta agertzen direla. Hori lortzeko aplikazioko negozio logikan ordenatzeko `ProductComparator` klase bat erabili da:

```
1  public List<VariantModel> orderBySelectedFeaturesSize(List<VariantModel> vm) {
2      vm.sort(new ProductComparator());
3      return vm;
4  }
5
6  private class ProductComparator implements Comparator<VariantModel> {
7      @Override
8      public int compare(VariantModel o1, VariantModel o2) {
```

```
9           return getProductSize(o1) - getProductSize(o2);
10      }
11  }
12
13  public int getProductSize(VariantModel p) {
14      int size = 0;
15      for (VariantComponent vc : p.getVariants()) {
16          if ((vc instanceof VariantFeature) && vc.isSelected())
17              size++;
18      }
19      return size;
20  }
```

7.2.3 Produktu ikuspegia: produktua

7.2.2 puntuko orrialdean produktu bat hautatu ondoren, orrialde honetan grafiko osoa aurkeztuko zaio *incomer*-ari. Kode-estaldura aztertzeko grafikoaz aparte produktuaren konfigurazioa azaltzen duen zuhaitza izango du orrialdearen eskuineko aldean. Konfigurazio atalean SPL-aren ezaugarri guztiak hautatuta edo ez-hautatuta dauden ikusi ahalko da, baita ezaugarri bakoitzaren informazioa kontsultatu (7.2.6 puntuan azalduko dena).



7.4 Irudia: Produktu ikuspegia, produktu baten informazioa.

Irudian ikusten den grafikoa egiteko ezaugarri bakoitzaren tamaina kalkulatu behar izan da eta ondoren tamaina horretatik zenbat kode estaltzen den produktuak duen konfigurazioarekin. Ezaugarrien tamaina lerro kopuruen arabera egin da eta meatzaritza moduluan

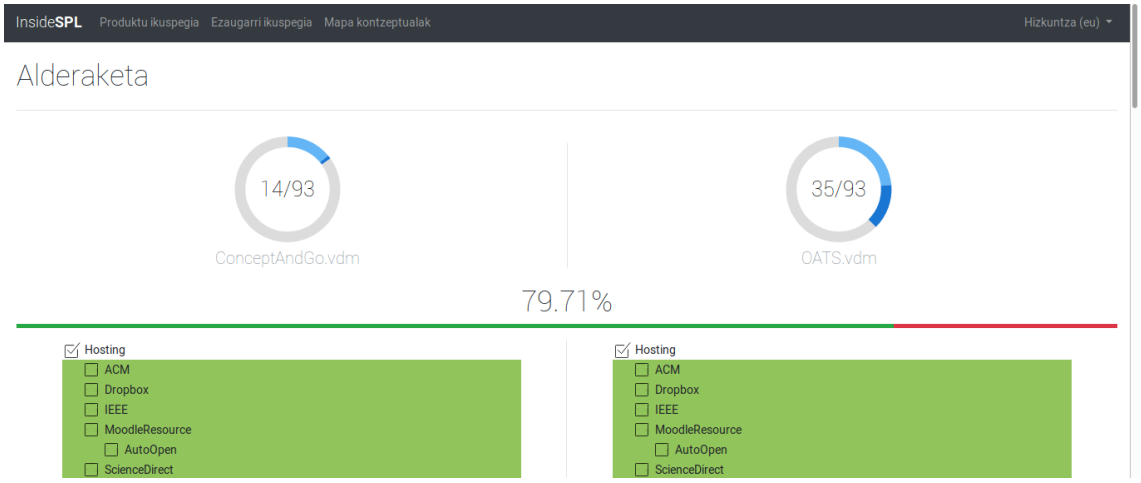
kalkulatutako balioa da, hau da, datu-basetik atzitutako balioa. Produktu baten konfigurazioak estaltzen duen kode kopurua, ordea, web aplikazio honen negozio logikak egin beharreko zerbait da. Horretarako grafikoa sortu ahal izateko funtzio errekurtsibo hau erabili da:

```
public Triplet<String, Integer, Integer> getProductInfoById(String id) {

    unselectedJson = "";
    unselectedSize = 0;

    VariantModel product = getProductById(id);

    if (product != null) {
        List<Feature> rootFeatures = featureBL.getRootFeaturesOfSpl(product.getSpl().getId());

        int unselected = 0;
        int fromTotalSize = 0;

        String resultJson = "[";
        for (Feature root : rootFeatures) {
            Pair<String, Integer> rootR = infoRecursive(product, root, 0);
            resultJson += rootR.getFst();
            unselected += rootR.getSnd();
            fromTotalSize += featureBL.getFeatureSize(root.getId());
        }

        int selectedFeaturesSize = fromTotalSize - unselected;

        // Add unselected part
        resultJson += "{ name: 'unselected', value: " + unselectedSize + ", "
                + "children: ["
                + unselectedJson
                + "]}";

        resultJson += "]";

        resultJson = "[{ id: 'root', name: '" + product.getFilename() + "', children: " +
    resultJson + "}]";

        Triplet<String, Integer, Integer> result = new Triplet<String, Integer, Integer>(
    resultJson,
                selectedFeaturesSize, fromTotalSize);
        return result;
    }

    return null;
}
```

7.2.4 Produktu ikuspegia: konparaketa

Produktu ikuspegiaren azkenengo orrialde honetan bi produkturen arteko konparaketa azaltzen da. Konparaketa orrira iristeko produktuen zerrendatik (7.2.2 puntuko orrialdea) bi produktu hautatu behar dira.

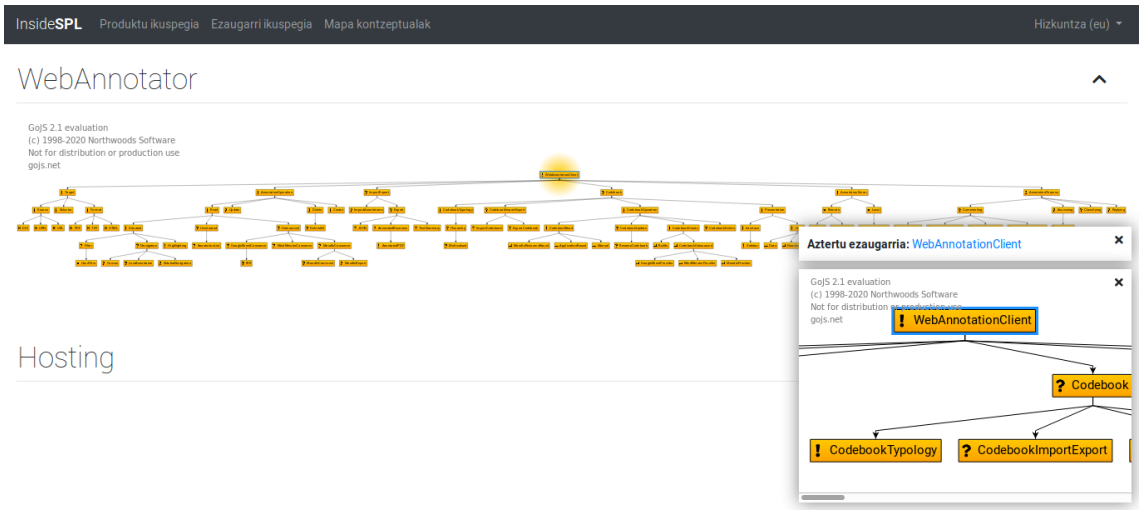
Konparaketa produktuen konfigurazioaren araberakoa da eta kontuan hartzen diren ezaugarriak derrigorrezkoak ez diren guztiak dira. 7.5 irudian ikusten den moduan, bi produktuek ezaugarri baten aurrean hautaketa bera egin badute berdez azalduko da konparaketan eta hautaketa desberdina egin badute gorriz.



7.5 Irudia: Produktu ikuspegia, bi produkturen konparaketa.

7.2.5 Ezaugarri ikuspegia: ezaugarri diagramak

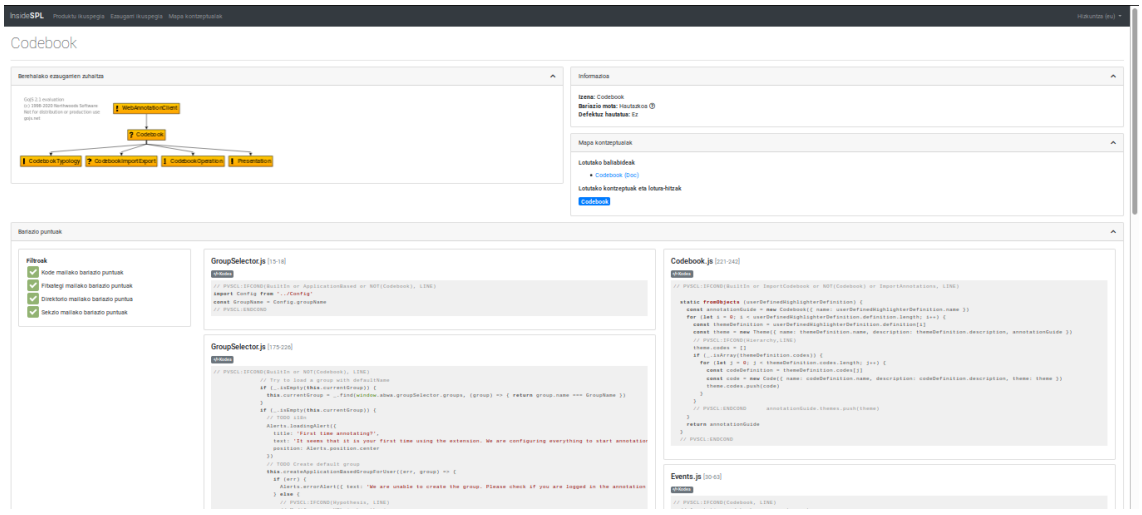
Hasierako orrian ezaugarri ikuspegia hautatuz gero orrialde hau ikusiko du *incomer*-ak. Honetan SPL-a osatzen duten ezaugarri diagramak (normalean bakarra, *WacLine*-n bi daude) azalduko dira. Ezaugarri diagramak oso handiak izan ohi direnez eskuineko aldean zoom leiho bat azalduko da momentuan hautatutako ezaugarria eta bere senideak aurkeztuko dituena. Zoom horren gainean ezaugarriaren informazioa aztertzeko esteka egongo da.



7.6 Irudia: Ezaugarri ikuspegia, ezaugarri diagramak.

7.2.6 Ezaugarri ikuspegia: ezaugarria

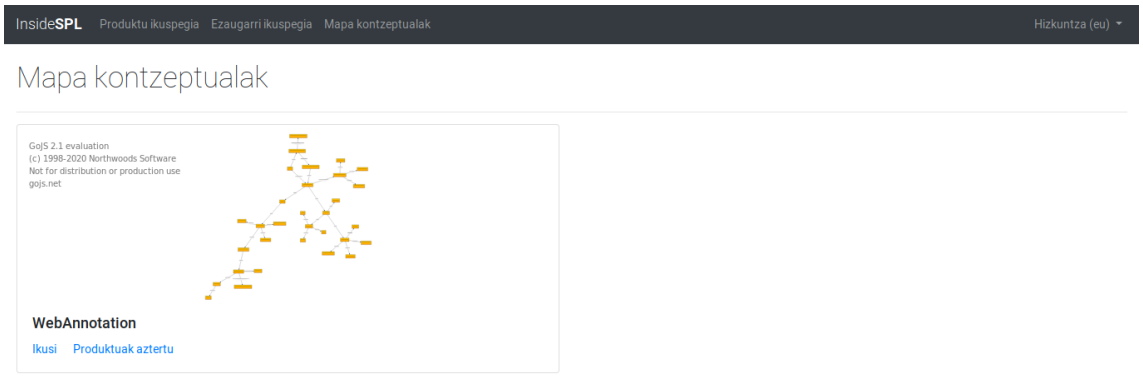
7.2.5 puntuko orrialdean ezaugarri baten informazioa aztertzeko esteka sakatu ondoren orrialde hau ikusiko da. Honetan ezaugarri baten informazio guztia azalduko da: berehalako ezaugarrien zuhaitza (gurasoa eta umeak, izanez gero), ezaugarriaren informazio orokorra, SPL-aren kontzeptu mapen eta ezaugarriaren arteko loturak eta aldaketa puntuak.



7.7 Irudia: Ezaugarri ikuspegia, ezaugarri baten informazioa.

7.2.7 Kontzeptu mapen ikuspegia: zerrenda

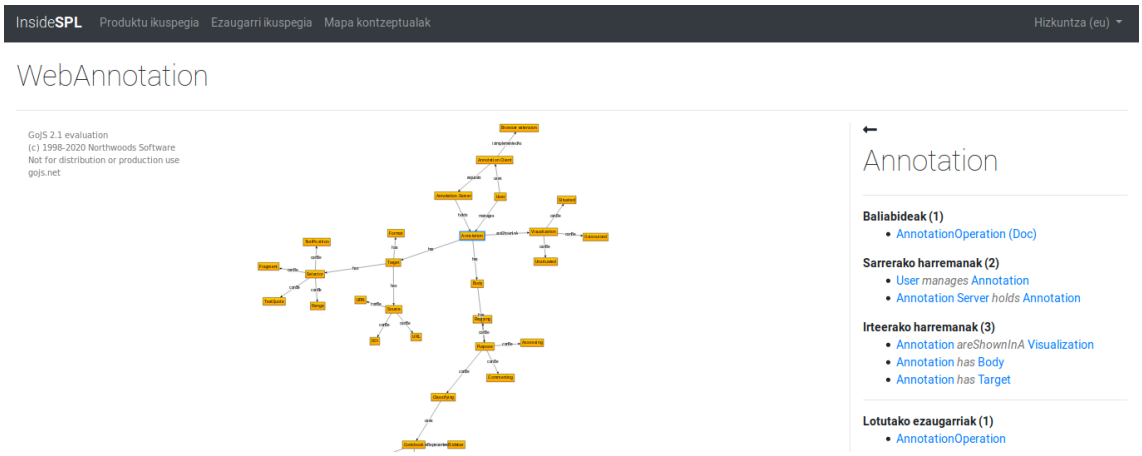
Hasieran aipatutako hiru ikuspegiekin amaitzeko, kontzeptu mapen ikuspegia falta da, zeinetan SPL-aren domeinua azaltzen duten kontzeptu mapak azalduko dira. Orrialde honetan hasierako orrian hautatutako SPL-ak lotuta dituen kontzeptu mapen zerrenda azalduko da, eta bakoitzeko bi aukera: kontzeptu mapa ikusi eta produktuak aztertu. Lehenengo aukerarekin kontzeptu mapa ikusi ahal izango da (7.2.8 puntua) eta bigarrenarekin SPL-aren produktuen kontzeptu-estaldura (7.2.9 puntua).



7.8 Irudia: Kontzeptu mapen ikuspegia, zerrenda.

7.2.8 Kontzeptu mapen ikuspegia: kontzeptu mapa

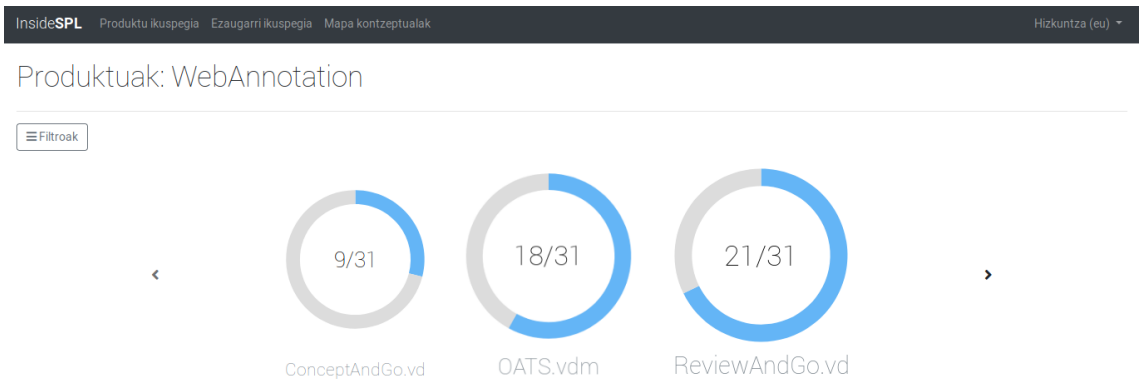
Kontzeptu mapen zerrendan *'ikusi'* hautatuz gero *incomer*-ak kontzeptu mapa horren orrialdera joko du. Honetan kontzeptuen arteko loturak ikusi ahalko ditu eta kontzeptuen gainean sakatuz hauek lotuta dituzten baliabideak eta SPL-aren ezaugarriak aurkeztuko zaizkio pantailaren eskuineko aldean (ikusi 7.9 irudia). Pantailaren zati honetan agertzen diren elementu guztiak estekak dira eta hauen bidez kontzeptu mapan nabigatu, baliabideak kontsultatu eta lotutako ezaugarrien informazioa aztertu ahal izango ditu erabiltzaileak.



7.9 Irudia: Kontzeptu mapen ikuspegia, kontzeptu mapa baten informazioa.

7.2.9 Kontzeptu mapen ikuspegia: produktuen kontzeptu-estaldura

Web aplikazioaren funtzionalitateekin amaitzeko produktuen kontzeptu-estaldura besterik ez da falta. Orrialdea kontzeptu mapen zerrendan *'produktuak aztertu'* aukera sakatzerakoan agertuko da eta honetan 7.2.2 puntuan ikusitako interfazearen antzekoa den orria azalduko da. Kasu honetan *incomer*-ak ikusiko duena produktu bakoitzak hautatutako kontzeptu mapatik zenbat kontzeptu *'betetzen'* dituen eta zenbat ez izango da (7.10 irudia). Kontzeptuak betetzearen ideia kontzeptu-ezaugarri loturatik dator eta produktu batek kontzeptu bat betetzen duela esango da kontzeptuak lotuta dituen ezaugarriak hautatuta baditu bere konfigurazioan produktuak. Honela kontzeptu mapak zein produktu azaltzen duen gehiago eta zein gutxiago azter daiteke.



7.10 Irudia: Kontzeptu mapen ikuspegia, produktuen kontzeptu-estaldura.

7.2.10 Internalizazioa (i18)

Web aplikazio osoa hiru hizkuntza desberdinetan dago eskuragarri: euskara, gaztelera eta ingelesa. Erabaki hau garatzailearen esku hartu da ez baita inolako betekizunetan azaltzen. Etorkizunera begira hizkuntza gehiagoko aplikazio bat nahi izanez gero errazagoa da hasieratik horrela planteatzea eta ez gero kode guztia aldatu behar izatea. Web aplikazioaren internazionalizaziorako (edo lokalizaziorako) *Spring* aplikazioa konfiguratu behar izan da eta horren ostean, hizkuntza anitzetako mezuak azaltzeko *JSP* orrialdeetan kode elementu hauek txertatu:

```
1 <spring:message code="string.code"></spring:message>
```

Etiketa horren `string.code` kodeak `.properties` luzapena duten fitxategietan gordetako esaldiak identifikatzen dituzte eta fitxategi horiek honelako egitura izango dute:

```
1 // messages_es.properties (gaztelerazko mezuak gordetzen dituen fitxategia)
2 string.code = Esto es un ejemplo
```

```
1 // messages_en.properties (ingelesezko mezuak gordetzen dituen fitxategia)
2 string.code = It's an example
```

```
1 // messages_eu.properties (euskarazko mezuak gordetzen dituen fitxategia)
2 string.code = Hau adibide bat da
```

Fitxategi hauetan web aplikazioko mezu guztiak definituz gero eta JSP motako fitxategietan `<spring:message/>` etiketak kokatuta garatutako aplikazioa hizkuntza anitzetakoa izango da. Hizkuntza berri bat sortzeko `messages_XX.properties` fitxategia sortu besterik ez da behar.

Hala ere, aplikazioaren lokalizazio prozesuan arazoak izan dira ere, izan ere, grafikoak *JavaScript* teknologiarekin sortu dira eta hauetan mezuak hizkuntza anitzetan agertzea beharrezkoa da ere, baina *JavaScript* bezero aldeko teknologia da eta JSP orrialdeetako `<spring:message/>` etiketak zerbitzari aldean prozesatzen dira. Arazo honi irtenbidea emateko bi gauza egin dira:

Alde batetik, web aplikazioan momentuko hizkuntzan esaldiak zerbitzatuko dituen esteka bat izango du:

```
@RequestMapping(value="/locales/{springCode}")
@ResponseBody
public String localeResolver(@PathVariable(name = "springCode",required = true) String code) {
    Locale locale = LocaleContextHolder.getLocale();
    code = code.replaceAll("-", ".");
    try {
        // Return string associated to code on the current language
        return messageSource.getMessage(code, null, locale);
    }catch(Exception e) {
        // Not defined code
        return "*" + code + "*";
    }
}
```

Bestetik, *jQuery* eta *AJAX* erabilita aplikazioan definitutako zerbitzuari esaldiak eskatzeko funtzio bat definituko da:

```
function getString(code){
    var tmp = null;
    $.ajax({
        url: "/locales/" + code.replace(/\./g,"-"),
        async: false,
        success: function(result){
            tmp = result;
        }
    });
    return tmp;
}
```

Bi ideia hauek konbinatuta *JavaScript*-en bidez `getString('string.code')` idaztea besterik ez da behar momentuko hizkuntzan `string.code` esaldia lortzeko.

8. KAPITULUA

Proiektuaren erronkak

Kapitulu honetan proiektua proposatu zenetik garapenaren amaieraraino jasan diren erronka nagusienak jasotzen dira. Erronkak zailtasun handieneko atazak izan dira, denbora inbertsio handia behar izan dutenak.

8.1 *SPLMiner*: meatzaritza moduluko erronkak

Atal honetan meatzaritza modulua gartzerakoan izandako erronkak biltzen dira. Meatzaritza moduluak pure::variants-ek automatikoki sortutako konfigurazio fitxategiak eta kontzeptu mapak minatzeko ardura du, baita SPL-a osatzen duen kodea (garatzaileek idatzitakoa) minatzekoa ere. Jarraian azalduko den moduan, kode anotazioak minatzea izan da modulu honen buruhauste handiena.

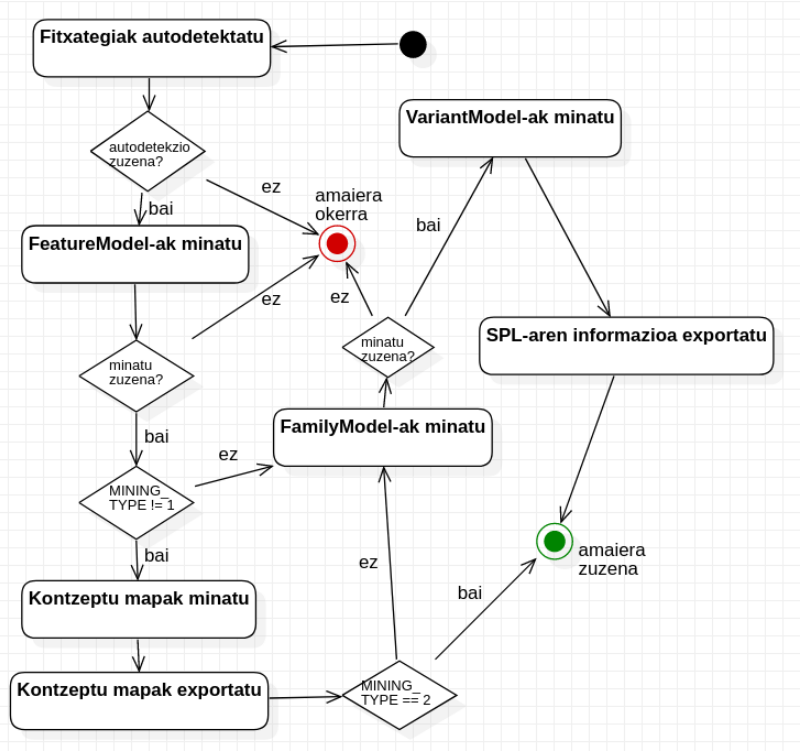
- **Minatu mota desberdinen uztarpena.**

  7.1.3 puntuan azaltzen den konfigurazioan ikus daiteke `MINING_TYPE` parametroa jasotzen dela. Parametro horrek klase nagusiak burutu ditzakeen minatu mota desberdinei egiten die erreferentzia eta, atal horretan atalean azaldu den moduan, hauek dira mota desberdinek burututako prozesuak:

  1. **SPL-aren minatu osoa.** *FeatureModel*, *FamilyModel* eta *VariantModel* guztiak.

2. **SPL-aren minatu partziala CMap konexioetarako.** Soilik *FeatureModel* minatzen da ezaugarri eta kontzeptuen arteko lotura egiteko. Gainera ez da SPL-aren SQL fitxategirik sortzen, soilik CMap-ena.
3. **SPL-aren minatu osoa eta CMap konexioak.** Aurreko bi aukerak konbinatu eta SQL fitxategi guztiak sortzen dituen aukera.

Minatu mota hauen uztarpena hobeto ulertzeko 8.1 irudiko fluxu diagrama ikus daiteke.



8.1 Irudia: Minatu mota desberdinen fluxu diagrama.

Minatu mota desberdinak uztartzea, batez ere CMap eta SPL domeinuak nahasten dituen minatuarentzat, ez da ataza erraza izan meatzari batzuek besteen menpekoak direlako eta batek behar duen informazioa oraindik sortuta ez dagoelako.

- **Aldaketa puntuen minatua.**

  Anotazioetan oinarritutako SPL-en potentziala kode-fitxategietan idatzitako baldintzetan oinarritzen da. Anotazioek aldaketa puntuak sortzen dituzte kodean eta, 6.3 irudiko klase-diagraman ikus daitekeen moduan, kode-fitxategietako aldaketa

puntuetan (`Code_VariationPoint` klasea) informazio hau gordetzen da: kode-elementua (fitxategia), aldaketa puntuaren baldintza (anotazioa), hasierako lerroa, amaierako lerroa, edukia (hasierako lerrotik amaierakora dagoen testu zatia), aldaketa puntuaren tamaina (lerro kopurua) eta habiaratze maila (≥ 0).

Pure::variants-ek anotazio desberdinak eskaintzen ditu fitxategi motaren arabera eta honek minatua asko zailtzen du. Horregatik, proiektuaren irismena mugatzeko hiru anotazio mota hartu dira kontuan: XML fitxategietan agertzen direnak (`pv:condition="A"`), eta bestelako kode-fitxategietan agertzen diren bi anotazio (`PV:IFCOND('A')` eta `PVSCL:IFCOND('A')`). Anotazio hauek minatzea erabaki da ohikoenak direlako eta *WacLine*-n erabiltzen direnak direlako.

Anotazioek sintaxi desberdina duten arren, bestelako kode-fitxategietan agertzen diren anotazioek antza handiagoa dute haien artean XML fitxategietan azaltzen direnekin alderatuz. Horregatik, kode-fitxategiak minatzeaz arduratzen den meatzariak (`CodeMiner`) fitxategiak bi moten arabera sailkatuko ditu: XML eta ez-XML.

XML fitxategiak minatzea, beste fitxategiekin alderatuz, gauza erraza da. Fitxategi hauek minatzeko `PositionalXMLReader` [31] klasearen aldaera bat erabili da. Jasotako klaseak elementu baten hasierako lerroa itzultzeko ahalmena zuen eta aldaketa puntu bat sortzeko informazio gehiago behar denez, garatzaileak aldaketa batzuk egin zituen klase horretan XML elementu baten hasierako lerroa, amaierako lerroa eta bere eduki osoa lortu ahal izateko (klase osoa D eranskinean). `PositionalXMLReader` klasea erabilita, soilik XML elementuak aztertu behar dira `pv:condition` atributuak bilatuz eta habiaraketa kontrolatuz.

XML ez diren fitxategietan, ordea, minatu prozesua asko zailtzen da: anotazioen sintaxi aldakorra kontuan hartu behar da, anotazioen ordena egiaztatu, habiaratutako anotazioak kontrolatu... eta gainera, XML elementuekin ez bezala, testu fitxategiak irakurtzerakoan testuingurua momentuan irakurtzen ari den lerroa besterik ez da. Lerro batean anotazio bat dagoela detektatzea, bere horretan, ez da zaila: `PV:IFCOND` edo `PVSCL:IFCOND` testuaren agerpena egiaztatzea aski izan daiteke horretarako. Arazoa sintaxi konplexuarekin dator, pure::variants-entzat anotazio hauek guztiak zuzenak baitira:

```
1 PV:IFCOND(A)
2 PV:IFCOND('A')
3 PV:IFCOND(A->hasAttribute('b'))
4 PV:IFCOND('A' AND C)
5 PV:IFCOND(A & C)
6 ...
```

Ikus daitekeenez kasu asko kontrolatu behar dira anotazioetan aipatzen diren ezaugarriak lortzeko eta horretarako, buruhauste askoren ondoren, adierazpen erregularrak eta hainbat funtzio lagungarri erabili dira (F.1 eranskineko atalean eskuragarri).

Baian arazoa ez da bakarrik ezaugarriak adierazpenetatik erauztea, honetaz gain lerro berean adierazpen bat baino gehiago (edo adierazpen baten amaiera eta beste baten hasiera) azaltzen den kontrolatu behar da. Horretarako, bi doiketa egin behar izan dira: lehenengoa, lerro bera hainbat aldiz prozesatu beharko dela kontuan hartzea; bigarrena, adierazpenen ordena kontuan hartzea.

Behin aldaketa puntu baten hasierako eta amaierako anotazioa aurkituta, eta aurreko prozesu osoa akatsik gabe burutu dela suposatuz, azkenengo erronka bat geratzen da: bi anotazioen artean irakurritako testua zuzen gordetzea. Ataza honen zailtasuna adibide honekin ikus daiteke:

```
1 testu hau ez da kontuan hartu behar /* PV:IFCOND(A) */ hau hasiera da
2 lerro anitz izan ditzake
3 eta amaiera ondoren /* PV:ENDCOND */ hau ez da kontuan hartu behar
```

Minatu prozesuaren ostean aldaketa puntuak eduki hau izan beharko luke:

```
1 /* PV:IFCOND(A) */ hau hasiera da
2 lerro anitz izan ditzake
3 eta amaiera ondoren /* PV:ENDCOND */
```

Lerro batzuetako adibide batean erraza iruditu daiteke prozesua, baina habiaratutako aldaketa puntuek asko nahasten dute prozesua eta informazio zuzena gordetzeko, aldaketa puntuaren edukia garbitzen duen funtzio hau erabili da:

```
1  private static String clean(String content, String expresion, String line, String
       endStatement) {
2
3      if(! reprocess)
4          content = content.substring(content.indexOf(expresion), content.length() - (line +
         "\n").length());
5      else
6          content = content.substring(content.indexOf(expresion), content.length() -
       endStatement.length() - vpString.length() );
7
8      content += endStatement;
9
10     return content;
```

```
11 }
```

Puntu honetan aurkeztutako ideia guztiak batuz lortu da fitxategi handietako aldaketa puntuak minatzea errorerik detektatu gabe. Hala ere, kodearen test unitateak ez direnez egin baliteke anotazioen sintaxiaren kasuistika guztia ez estaltzea funtzio hauekin.

- **SPL eta kontzeptu mapen arteko loturak.**

  Soluzioaren diseinuan azaldutako klase-diagraman (6.2 atalean) bi domeinu desberdin agertzen dira: SPL-aren domeinua eta kontzeptu mapena. Bi domeinu horiek loturak dituzte haien artean, 6.3 irudian ikus daitekeen moduan. Izan ere, minatutako kontzeptu mapek lotutako SPL-aren nondik norakoak azaltzeko helburua dute.

  SPL-aren ezaugarriek funtzionalitateekin lotura zuzena dutela kontuan hartuta, suposatzekoa da kontzeptu mapa horietan ezaugarriei erreferentzia egiten dieten kontzeptuak azalduko direla, eta hala da. Arazoa kontzeptu eta ezaugarrien arteko lotura egite nahiarekin dator: nola jakin dezake meatzari programa batek `Browser extension` kontzeptuak `WebAnnotationClient` ezaugarriarekin duela lotura? Hori SPL-aren garatzaileen ezagutzan dago eta ezagutza hori meatzariak erabiltzeko, esplizituki adierazita egon beharko da.

  Honetarako, CMap softwarearen `short-comment` atributua erabili da eta `"PV: EzaugarriIzena"` sintaxia. Lotura hauek prozesatzeko funtzio lagungarri bat erabili da, SPL-aren ezaugarrien izenak atributuaren balioarekin konparatzen dituena adierazpen erregularren bitartez (F.2 eranskineko atalean eskuragarri).

  Kontzeptu eta ezaugarrien arteko harremana `M:N` motakoa da, hau da, kontzeptu batek hainbat ezaugarri azal ditzake eta ezaugarri bat zenbait kontzepturen bidez azal daiteke.

- **Osatu gabeko konfigurazioak.**

  Software Produktu-Lerroak produktu bakoitza adierazten duen konfigurazioak dituzte, hauetan hautazko ezaugarrien aukeraketa zehatz bat gordetzen da. Pure::variants softwarearekin egindako SPL-en konfigurazioak *VariantModel*-ak dira, eta ezaugarrien aukeraketaz aparte kode-elementuen aukeraketa ere gordetzen da (*FamilyModel*-ean azaltzen diren elementuak).

  Pure::variants-ek sortutako beste fitxategietan bezala, SPL-aren elementu guztiak hautatuak izan diren edo ez aipatzea espero liteke, baina konfigurazioetan badira

azaltzen ez diren elementuak: elementu hauen gurasoa ez badaude hautatuak umeak ezin dira hautatuak izan eta ondorioz ez dira fitxategian azaltzen.

Informazio falta hori ezin da begi bistara kontrolatu, SPL batek ezaugarri eta kode-elementu asko izan ditzakeelako[1], eta informazio faltaren eragina SPL handiak minatzen hasterakoan azaldu zen, emaitzak aztertu ondoren. Arazoa konpontzeko, 7.1.1 puntuan aipatu den moduan, *VariantModel*-ak minatzeko ardura zuen meatzariari (`VariantModelMiner`) kode zati bat gehitu zitzaion, SPL-aren ezaugarri eta kode-elementuak tratatuak zeuden edo ez egiaztatzeko eta tratatu gabekoak gehitzeko (F.3 eranskineko atalean eskuragarri). Kode zati horrekin lortzen da konfigurazio guztiek ezaugarri eta kode-elementu guztien hautaketaren inguruko informazio esplizitua izatea.

8.2 *InsideSPL*: bisualizazio moduluko erronkak

Kapitulu honen azken atalean bisualizazio modulua garatzerakoan izandako zailtasunak barnebiltzen dira. Bisualizazio modulua *incomer*-ak SPL-aren ezagutza zabaltzeko erabiliko duen tresna izanda, sortutako bisualizazioak sinple eta ulergarriak izatea saiatu da. Hala ere, hurrengo zerrendan aipatuko den moduan, izan dira informazio asko azaltzeko erabili diren grafikoak, hauek sortzeko prozesua nahiko konplexua bihurtuz.

- **Hizkuntza anitzen integrazioa.**

  Proiektuaren baitan web aplikazio bat garatu behar zela eta hura *Spring* teknologiarekin inplementatuko zela erabaki zen momentutik, proiektu honen garatzaileak argi izan zuen web aplikazioa hizkuntza anitzetan eskainiko zuela. Erabaki hori ez zen betekizunen bilketan jasotzen, baina garatzailearen aurreko esperientziek ikasketa argi bat utzi zioten gai honen inguruan: hobe da hasieratik hizkuntza anitzen integrazioa egitea, ez egitea eta etorkizunean dena aldatu behar izatea baino.

  Hartutako erabakiak denbora inbertsio handia eskatzen du (hizkuntza bakar batekin konparatuz), web aplikazioan azaltzen diren termino eta azalpen guztiak hainbat hizkuntzatara itzuli behar direlako. Hala ere, ezin da egindako denborako inbertsioa alderatu etorkizunean hizkuntza berri bat integratzea erabakiz gero aurreztuko den denborarekin.

[1] *WacLine* 1101 elementuz osatuta dago, 93 ezaugarri eta 1008 kode-elementu. Hauetako bakoitzaren hautaketaren informazioa gordetzen da konfigurazio fitxategi batean eta SPL batek konfigurazio (produktu) desberdin asko ditu.

Hala ere, soluzioaren garapenean azaldu den moduan (7.2.10 atala), hizkuntzaren araberako testuaren ebaluazioa zerbitzari aldean (bezeroak jaso baino lehen) egiten da eta badira bezero aldean ebatzi beharreko testu kodeak. Arazo hori konpontzeko bi gauza egin dira: batetik, web aplikazioan momentuko hizkuntzan esaldiak zerbitzatuko dituen zerbitzua sortu (`/locales/string.code` estekan atzigarri dagoena, `string.code` lortu nahi den esaldiaren kodea izanda); eta bestetik, *jQuery* eta *AJAX* erabili aplikazioan definitutako zerbitzuari esaldiak eskatzeko.

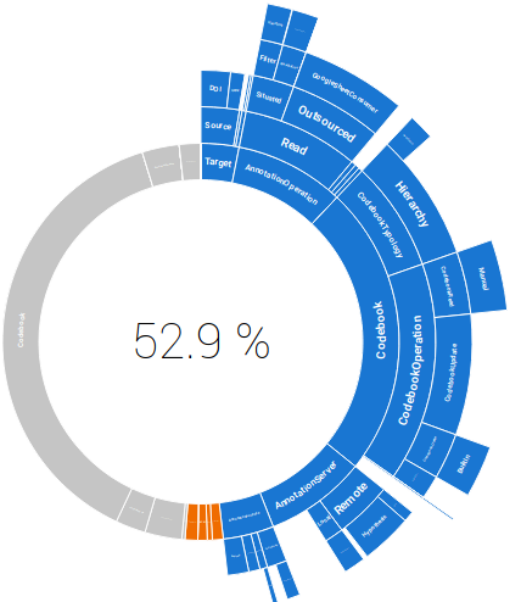
- **Produktuaren grafiko osoa.**

Kapitulu honekin amaitzeko, bisualizazio moduluaren funtzionalitateak inplementatzerakoan egindako grafiko konplexuena azaldu nahi da. Grafiko hau 7.2.3 puntuan azaldutako web orrian azaltzen da eta bere helburua produktu baten kode-estaldura adieraztea da. Gogoratzearren, kode-estaldura produktu batek bere konfigurazioan hautatutako ezaugarrien ondorioz erabiltzen duen baldintzazko kodea (aldaketa puntuek baldintzatuta) da.

Produktu baten kode-estaldura kalkulatzeko hainbat gauza hartu behar dira aintzat: zein dira hautatutako ezaugarriak, zein dira hautatu gabekoak, zein da ezaugarri bakoitzaren tamaina[2], ezaugarrien arteko harremanak (hierarkia)...

Gauza horiek guztiak kontuan hartuta, grafikoan bi atal nagusi desberdintzen dira: hautatu gabeko ezaugarriek estaltzen duten kodea (8.2 irudian, grisez) eta hautatutako ezaugarriek estaltzen dutena (8.2 irudian, kolorez).

[2] 6.4 irudian, `Feature` klaseak lotuta duen `FeatureSize` klasea



8.2 Irudia: Produktu baten grafiko osoa.

Hautatu gabeko ezaugarrien kode-estaldura adierazteko lehenengo mailako ezaugarriak besterik ez dira azaltzen (`Codebook` adibidez, zabalena) eta hauen tamaina haietatik zintzilik dauden ezaugarri ez-hautatu guztien batura da. Hautatutako ezaugarrien kode-estaldura adierazteko hierarkia bisuala erabili da, ezaugarrien arteko harremanak ikusi ahal izateko, baita ezaugarri batek bere gurasoaren zenbatekoa estaltzen duen jakin ahal izateko. Grafikoan azaltzen den informazio guztia laburtzeko, grafikoaren erdian kode-estaldura ehunekotan azaltzen da.

Grafikoa sortu ahal izateko erabilitako funtzioak F.4 eranskineko atalean ikus daitezke.

9. KAPITULUA

Jarraipen eta kontrola

Kapitulu honetan Gradu Amaierako Lanaren jarraipena, proiektuak jasandako desbiderapenak eta plangintzan planteatutako kontrol atazen azalpena emango da. Kapitulu honen muinean sartu aurretik ohorezko aipamena eman behar zaio ikasle zein irakasleen egunerokoan eragin ikaragarria izan duen kudeatu gabeko arazo handienari: osasun egoera globala.

9.1 Irismenaren desbiderapena

Memoria honek deskribatzen duen proiektuaren diseinuarekin hasi baino lehen irismena ez zegoen guztiz definitua, 2.3 atalean azaldu den moduan irismena modu iteratiboan definitzen joan delako. Honen ondorioz, plangintzan azaldutakoarekin konparatuz desbiderapenak egon dira, hurrengo zerrendan ikusten diren puntuak izanda garapenaren amaierako faseetan definitutako irismena:

1. Prototipo funtzionala: datu-basearen diseinua eta meatzaritza moduluaren inplementazio osoa, baita bisualizazio moduluaren *'produktu ikuspegia'*. Ikuspegi honen bidez garatzaile berriek SPL-tik eratorritako produktuak aztertu ahalko dituzte eta haien arteko alderaketa egin bi produktuen zati amankomunak eta desberdintasunak ikusteko, adibidez.

   Puntu hau plangintzaren irismenean definituta zegoen eta ez du desbiderapenik izan.

2. Bisualizazio moduluaren hedapena: *'ezaugarri ikuspegia'*. Honetan, SPL-aren ezaugarrien inguruko informazio desberdina emango zaio erabiltzaileari, hala nola zein diren SPL-aren ezaugarri-diagramak, ezaugarri batek ze aldaketa puntuetan duen eragina, eta abar.

   Produktuaren garapen iteratiboarekin definitutako irismeneko puntu berria da, bisualizazio moduluaren *'kontzeptu mapen ikuspegia'* garatzeko beharrezkoa.

3. Meatzaritza moduluaren hedapena: garatutako modulua CMap softwarearekin sortutako kontzeptu mapak minatzeko aukera emateko eguneratu.

   Nahiz eta hasierako irismenaren definizioan kontzeptu mapen inguruko ideiak azaltzen ziren, garapenaren lehenengo faseetan ez zen kontuan hartu eta meatzaritza modulua eguneratu behar izan da kontzeptu mapak minatzeko.

4. Bisualizazio moduluaren hedapena: *'kontzeptu mapen ikuspegia'*, SPL-aren domeinua azaltzen duten kontzeptuak aztertzeko aukera ematen duen ikuspegia, baita kontzeptuek ezaugarriekin duten lotura ikustekoa.

   Plangintzan definituta zegoen arren, hedapen hau garatu aurretik *'ezaugarri ikuspegia'* eta meatzaritza moduluaren hedapena garatu behar izan da, kontzeptu eta ezaugarrien arteko loturak adierazi ahal izateko.

9.2 Plangintzaren desbiderapena

Atal honetan burututako Gradu Amaierako Lanaren plangintzaren desbiderapena jasotzen da. Horretarako ataza bakoitzari dedikatuko zaion denbora, hasieran balioetsitakoa eta bi balio horien arteko desbiderapena aurkeztuko dira (9.1 taula).

Atazaren deskribapena	Balioespena	Dedikazio erreala	Desbiderapena
Ikasketa	30 ord.	25 ord.	-5 ord.
I.1	10 ord.	11 ord.	+1 ord.
I.2	10 ord.	2 ord.	-8 ord.
I.3	10 ord.	12 ord.	+2 ord.
Datu-basea	20 ord.	21 ord.	+1 ord.
DB.1	5 ord.	12 ord.	+7 ord.
DB.2	5 ord.	4 ord.	-1 ord.
DB.3	10 ord.	5 ord.	-5 ord.
Meatzaritza	70 ord.	72 ord.	+2 ord.
MM.1	10 ord.	2 ord.	-8 ord.
MM.2	10 ord.	5 ord.	-5 ord.
MM.3	40 ord.	50 ord.	+10 ord.
MM.4	10 ord.	15 ord.	+5 ord.
Bisualizazioa	68 ord.	106 ord.	+38 ord.
BM.1	3 ord.	3 ord.	0
BM.2	5 ord.	3 ord.	-2 ord.
BM.3	40 ord.	60 ord.	+20 ord.
BM.4	20 ord.	40 ord.	+20 ord.
Memoria	82 ord.	72 ord.	-10 ord.
Mem.1	80 ord.	70 ord.	-10 ord.
Mem.2	2 ord.	2 ord.	0
Defentsarako aurkezpena	8 ord.	6 ord.	-2 ord.
DA.1	3 ord.	Ord. 1	-2 ord.
DA.2	5 ord.	5 ord.	0
Plangintza	15 ord.	14 ord	-1 ord.
P.1	2 ord.	2 ord.	0
P.2	10 ord.	12 ord.	+2 ord.
P.3	3 ord.	0 ord.	-3 ord.
Bilerak	30 ord.	11 ord.	-19 ord.
B.1	-	Ord. 1	+1 ord.
B.2	30 ord.	10 ord.	-20 ord.
Jarraipena eta Kontrola	-	12 ord.	+12 ord.
JK.1	2 ord.	3 ord.	+1 ord.
JK.2	-	2 ord.	+2 ord.
JK.3	-	5 ord.	+5 ord.
JK.4	-	2 ord.	+2 ord.
Osotara	323 ord.	339 ord.	+16 ord.

9.1 Taula: Atazen iraupenaren desbiderapena

Aurreko taulan ikus daiteke Gradu Amaierako lana bere osotasunean hartuta ez dela desbiderapen handirik egon (16 ordukoa), baina argi dago garapenaren atalak aurreikusitakoa

baino askoz denbora gehiago eskatu duela, bereziki bisualizazio moduluak.

Nahiz eta desbiderapen txikia izan (ordu batekoa) aipagarria da komunikazio sistema adosteko atazak (B.1) ez zuela denborarik aurreikusita tribiala zelako, baina ezusteko osasun egoera globala dela eta komunikazio sistema eta bileren kudeaketa goitik behera aldatu behar izan da (9.4 puntuan azalduko den moduan).

9.3 Arriskuak eta kalitatea

9.3.1 Arriskuen kudeaketa

Proiektua hasi aurretik arrisku batzuk zerrendatu ziren (2.5.1 puntua) eta atal honetan arrisku horien kudeaketa nolakoa izan den azalduko da.

Batetik, aipatu beharra dago arriskuen zerrendan ez zela bizi izan dugun osasun egoera globala azaltzen, plangintza sortzerako momentuan ez baitzen arrisku erreal bezala ikusten. Hala ere, argi dago proiektuaren garapenean eta inplikatu guztien bizitzan eragina izan duela eta modu batean edo bestean desbiderapenak sortu dituela.

Bestetik, proiektuak aipatutako zerrendako arrisku gehienak jasan dituela onartu behar da: irismenaren aldaketa, datu-basearen diseinu okerra egitea eta datuen galera. Arazo hauek ez badute proiektuan eragin handirik izan, definitutako arriskuen kudeaketa estrategiak erabili direlako izan da. Datuen galera, adibidez, hirutan gertatu da eta interneten atzigarri dagoen *GitHub* biltegiari esker ordu askotako arazoa izan litekeena minutu gutxitan konpondu da.

9.3.2 Kalitatearen kudeaketa

Arriskuen modu berean, kalitatearen kontrola egiteko ekintza batzuk definitu ziren hasieran (2.5.2 puntua). Hemen kalitatearen kudeaketa nola egin den (eta ez den egin) azalduko da.

Ekintzak

- **Kodearen gaineko barne probak.** Hasieran, probak modu estandarizatu batean egitea (*JUnit*, adibidez) planteatzen zen eta garatzaileak hainbat test prestatu zituen

horretarako, baina proiektuaren garapen iteratiboarekin batera test horiek garatzeko denbora mugatuz joan zen eta amaieran ez da inolako test unitaterik egin. Horrek, ordea, ez du esan nahi funtzionalitateak ez direnik probatu; test unitateen ordez garatzailearen eskuzko probak erabili dira funtzionalitateen kalitate egokia aztertzeko, denbora falta dela eta. Argi dago pertsona batek egindako probak kasu guztiak kontrolatzea oso zaila dela, baina garapen prozesua arintzeko estrategia aproposena izan da.

- **Kodearen gaineko kanpo probak.** Aurreko proben antzera, kanpo probak ez dira dokumentu tekniko zehatzetan jaso, baina produktua inplementatu ahala GrAL-aren tutoreek eta Onekin ikerkuntza taldeko beste interesatuek funtzionalitateen oniritzia ematen joan dira, ikusitako akatsak komunikatzen zituzten bitartean.

- **Produktuaren erakusketa probak.** Kalitatearen kontrolerako puntu hau *Google Forms* plataformaren bitartez egindako galdetegi baten bidez egin da eta C eranskinean aztertzen dira emaitzak.

9.4 Komunikazio sistemak

Plangintzaren desbiderapenean aipatu den moduan komunikazio sistemak ezusteko desbiderapena izan du: komunikazio gehiena fisikoki egiteaz, dena birtualki egitera behartuak izatera. Ez dago zalantzarik 2020 urte hasieran ez zela maila globaleko osasun alerta bat aurreikusten eta honen eraginez proiektuak komunikazio sistema desegoki bat planteatu izan du.

Bat-bateko egoera honen aurrean komunikazio sistema aldatu behar izan da eta komunikazio gehienetarako EHU-ko posta erabili da. Soilik azalpenak behar zirenean edo kontrol bilerak egiteko egin dira bilera presentzialak eta unibertsitateak eskainitako *Blackboard Collaborate*-ren bidez burutu dira.

Hala ere, osasun egoera honek espero zitekeenaren kontrako eragina izan du proiektuaren desbiderapenean, posta gehiago erabiliz askoz bilera gutxiago egin baitira eta aurreikusitako orduak baino 20 ordu gutxiago erabili dira bileretarako.

10. KAPITULUA

Ondorioak

Proiektu baten bilakaeran ikasketak daude beti, neurri batean edo bestean, eta ikasketa horiek hurrengo proiektuak modu eraginkorragoan burutzea ahalbidetzen dute. Memoriaren azkenengo kapitulu honetan, Gradu Amaierako Lanaren emaitzaren inguruan hitz egingo da, emaitzaren balorazio bat eginez eta garapen prozesuan ikasitako lezioak jasoz. Amaitzeko, garatutako prototipoak etorkizunean izan dezaken ibilbidea aipatuko da.

10.1 Lortutako emaitza

Proiektuaren helburu nagusia *"garatzaile berriei SPL-etan oinarritutako proiektuak azkarrago ikasteko aplikazio bat garatzea"* (2.2 ataletik aterata) izanda, esan daiteke proiektuaren emaitza aproposa dela. Garatzaileak gustura burutu ditu planteatutako betebeharrek suposatu dituzten erronkak eta garatutako soluzioarekin pozik dago. Gainera, prototipoaren garapenaren bitartean Onekin ikerkuntza taldeko hainbat interesatuk emandako iritzien eta prototipoaren azkenengo bertsioaren onarpenarengatik, prototipoak interesatu nagusiak asebete dituela interpreta daiteke.

10.2 Ikasitako lezioak

Proiektu bat amaitzerakoan, honekin ikasitako lezioak errepasatzea komeni da, etorkizunean akats berdinak ez errepikatzeko xedearekin. Atal honetan proiektuaren ondorio

pertsonal esanguratsuen zerrenda egin da:

- **Diseinuaren garrantzia.** Programatzen ikasterakoan, batez ere unibertsitateko lehenengo urteetan, softwarearen diseinu fasea irakasleek aipatzen duten baina inork ikusten (eta egiteko gogorik) ez duen fasea da. Jarrera hori bateragarria izan daiteke proiektu txikiekin, erabaki okerrek eragin txikia izango baitu; proiektu handiekin, ordea, diseinu fasea saltatzea proiektuaren bizi-zikloan atzerako kontaketa bat jartzearen modukoa da, ezin da jakin noiz iritsiko den zerora, baina iristen denean (eta iritsiko da) proiektu osoa berregituratu beharko da. Horregatik, proiektu honetan diseinuan eta garatutako moduluen independentzian garrantzi handia jarri da eta arazoak sortu direnean erabaki zuzena izan dela argi geratu da.

- **Bertsioen kontrola.** Proiektua hasi aurretik garatzaileak bertsioen kontrola eramatea garrantzitsua zela uste bazuen, proiektuaren amaieran argiago geratu da oraindik. Garapen prozesuan hiru aldiz galdu dira datuak (proiektu osoa, bai...) eta portaera zuzena zuten funtzionalitate batzuek emaitza okerrak eman dituzte bat-batean. Bertsioen kontrolari esker arazo hauek minutu gutxietako buruhausteak bihurtu dira.

- **Anotazioen konplexutasuna.** Software batek kodean anotazioak egitea ahalbidetzen duenean, baina sintaxi malgua duenean, garatzailearen ohiturak azaltzen dira anotazioetan: letra xeheak edo larriak erabiltzea, iruzkinak idazteko marka desberdinak erabiltzea (*Java*-n `//` edo `/* */`), lerro-saltoak modu batean edo bestean erabiltzea... Egia esan, gauzak asko erraztuko lituzkeen arren, ezin da espero munduko garatzaile guztiek anotazioak modu berean egitea. Ondorioz, anotazioen minatuari denbora asko inbertitzea eta kasu asko kontuan hartzea egokitu zaio garatzaileari, adierazpen erregularren zale bilakatuz.

- **Ikuspegi mugatua.** Norberak egindako lana argiena dela dirudi beti, norberaren ikuspuntutik; beste batena ordea ez da oso ongi ulertzen edo alor desberdinetan sakontzeko laguntza behar dela dirudi. Oro har proiektu gehienak, eta bereziki prototipo hau, beste pertsonek erabiltzeko garatzen dira eta C eranskinean egindako kalitate atazaren ondorioz garatzaileak ikusi ahal izan du bere ikuspegia mugatua dela eta kanpoko pertsonen iritzia jasotzeak oso mesedegarria izan daitekeela prototipoaren bizi-zikloan.

- **Garatzaile komunitatearen laguntza.** Proiektuaren etapa guztietan izan ditu zalantzak garatzaileak: *'nola berreskuratzen dut Git-en aurreko bertsioa?'*, *'klasea*

abstraktua izan beharko luke?', *'nola gordetzen da hau datu-basean?'*, ... Zalantzak izatea normala da, nolanahi ere; zalantzak ebaztearen erronka nagusia **non** ebazten diren da, **zeinekin** fidatu daitekeen, alegia. Proiektu honetan asko erabili dira Interneten atzigarri dauden garatzaileentzako foroak eta web orriak (StackOverflow eta Baeldung, esaterako) eta arazoen aurrean baliabide zuzenak erabiltzea garrantzitsutzat jotzen du garatzaileak. Hala ere, Interneten ez daude erantzun guztiak eta norberaren irtenbideak bilatzeko gaitasuna sustatzea ere behar-beharrezkoa da, proiektu honetan hainbatetan egin den bezala.

- **Aukeraketen garrantzia.** Teknologia guztiek ordezkoak dituzte beti, antzeko funtzionalitateekin, erabiltzeko errazagoa edo konplexuagoa izan. Proiektua garatzeko teknologia desberdin asko erabili dira (*Java, Spring, MySQL, JavaSript, AnyChart, jsTree, JGit, Bootstrap...*) eta hauek ez dira ausaz hautatu, atzetik beti arrazoiak egon dira. Teknologia okerra hautatzeak proiektuaren garapena asko atzeratzea eragin dezake eta horregatik garatzaileak teknologien aukeraketari behar duen garrantzia eman nahi izan dio.

10.3 Etorkizuneko aukerak

Memoria honetan zehar asko hitz egin da prototipoaren etorkizunera begira hartutako erabakiez eta atal honetan aplikazioak izan dezakeen ibilbidea jaso nahi izan da:

- ***FeatureCloud* aplikazioarekin integrazioa**. Garatutako prototipoa Informatika Fakultateko Onekin taldearentzat egin da, egun SPL-en munduan aplikazio desberdinak sortzen eta garatzaile berrien zailtasunak aztertzen dihardutela. *FeatureCloud* ikerkuntza taldearen aplikazioetako bat da SPL-aren garapen prozesuaren bi `git commit` alderatzea ahalbidetzen duena eta egindako prototipoaren egitura oso antzekoa duena (meatzaritza modulua, datu-basea eta bisualizazio modulua). Prototipoaren ibilbideetako bat, beraz, aplikazio horrekin integrazioa izan ahalko litzateke, maila desberdinetan: datu-base amankomuna erabiltzea, bisualizazio moduluak elkartzea, meatzaritza modulu bakarra sortzea...

- **Pure::variants-en anotazio mota desberdinak erabiltzea.** Aurretik aipatu den moduan (8.1 atala, *'Aldaketa puntuen minatua'* puntuan) *SPLMiner* moduluak anotazio mota mugatuak hartzen ditu kontuan, anotazioak fitxategien araberakoak izan-

da. Pure::variants-ek eskaintzen dituen anotazio gehiago kontuan hartzea izango litzateke meatzaritza moduluaren hobekuntza posiblea.

- **Anotazioetan oinarritutako beste teknologiak erabiltzea.** Meatzaritza modulua pure::variants-ekin egindako SPL-ak minatzeko prestatuta dago, baina datu-basea eta bisualizazio modulua ez dira teknologia horren menpeko. Horregatik, posible da beste meatzaritza moduluak inplementatzea teknologia desberdinekin garatutako SPL-ak datu-base berean gorde eta *InsideSPL* web aplikazioarekin bisualizatu ahal izateko.

- **Bisualizazio moduluan ikuspegi gehiago integratzea.** Proiektuaren hasieratik irismena moldatzen joan da, momentuko egoera eta garatu nahi ziren funtzionalitateen arabera. Denbora mugatuaren eraginez, irismena 9.1 puntuan aipatutakoa izan da. Hala ere, horrek ez du esan nahi *InsideSPL* amaituta dagoenik, are gutxiago, bisualizazio moduluak nahi bezain beste ikuspegi integratu baititzake. Plangintzan aipatutako *'arkitektura ikuspegia'* edo aldaketa puntuetan ardatza jartzen duen ikuspegia izan daitezke ikuspegi berrien integrazioaren adibide.

- **Datuak kontsultatzeko beste aplikazioak.** Bisualizazio modulua pure::variants teknologiaren menpeko ez den moduan, meatzaritza moduluak sortutako datuak ez daude *InsideSPL* web aplikaziora zuzenduta. Datu-basean gordetako datuak SPL-aren ezagutza orokorra gordetzen dute eta informazio hori beste mota batzuetarako erabilgarria izan daiteke, hala nola, SPL-en gainean estatistikak ateratzen dituzten aplikazioak, SPL-aren konplexutasuna aztertzen duten aplikazioak, etab.

Eranskinak

A. ERANSKINA

Gradu Amaierako Lanaren hasierako proposamena

Desarrollo de un asistente para desarrolladores noveles en la tarea de entender una Línea de Producto Software

---- Iosu Salaberri ----

2020-02-04

Los **objetivos principales** serán los siguientes:

- .... Realizar el **diseño de la "BD de minado"** teniendo en cuenta las configuraciones de los productos. Para ello:
  - Comprobar cómo se guarda la descripción de un producto en *pureVariants* (ficheros xml de feature model, family model y variant model).
  - Utilizar la BD de *FeatureCloud*, el mapa conceptual de las SPLs, la lista de consultas para un newcomer.
- .... **Modificar el módulo de minado** de código adecuándolo al nuevo esquema de la BD, para que tenga en cuenta las configuraciones de los productos.
- .... **Modificar el módulo de visualización**, a decidir si uno de los siguientes, "en profundidad" o los tres en modo "incremental" a un nivel más superficial:
- .... **Modificar el módulo de visualización**, de forma que se muestren aspectos relacionados con **productos** (configuraciones):
  - Show in a diagram (e.g. Venn diagrams) what the SPL products are
    - comparing their size, the intersection among them
- .... **Modificar el módulo de visualización**, de forma que se muestren aspectos relacionados con el **mapa conceptual**:
  - Mapa conceptual creado con *CMap tool*, almacenado en fichero *.cxl
- .... **Modificar el módulo de visualización**, de forma que se muestren aspectos relacionados con la **arquitectura**:
  - Show in a diagram what the architecture of the SPL is
  - Show in a diagram what the components in the architecture of the SPL are
  - What are the files in the architecture of the SPL?
  - Show in a diagram what part of the SPL architecture corresponds to this specific product (e.g. specific components in different color)
  - Show in a diagram what the components in the architecture of this specific product are
  - From the whole set of components, show in a diagram what the ones in this specific product are (e.g. specific components in different color)
  - What are the files in the architecture of this specific product?
  - From the whole set of files, show in a diagram what the ones in this specific product are (e.g. specific files in different color)

Objetivo secundario:

- .... Integrar la aplicación de minado con la aplicación de visualización mediante una REST-API (dando la posibilidad de filtrar por punto de commit).

B. ERANSKINA

Bisualizazio moduluaren hasierako proposamena eta teknologia aukeraketa

Bisualizazio modulua

A) Produktu ikuspegia

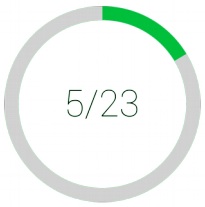
Bisualizazio hauetan SPL-aren produktu (*Variant Model*) desberdinetako datuak aurkeztuko dira, hauen *feature* kopurua edota produktuen arteko konparaketa eginez. Bisualizazioekin galdera desberdinak erantzungo dira, hala nola:

1. Produktu bat hartuta, bere ezaugarri kopurua erakutsi
2. Produktu bat hartuta, horrek posible duen funtzionalitateen zenbateko ehunekoa duen erakutsi
3. Produktu bat hartuta, bere konfigurazioa erakutsi
4. Bi produktu hartuta, haien harteko konparaketa egin

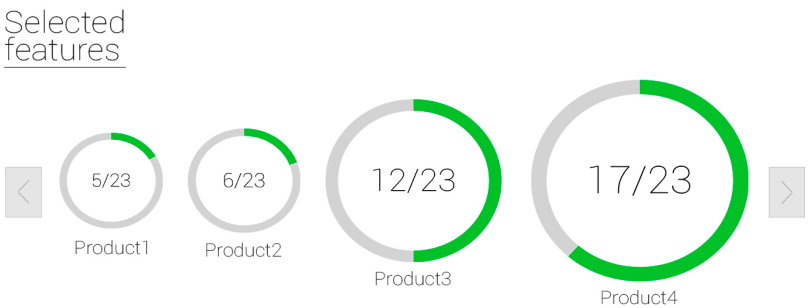
Proposamenarekin hasteko, bisualizazioen diseinua bera aurkeztuko da eta ondoren hauek sortzeko erabil daitekeen teknologiaren aukeraketa.

Diseinua

Produktu ikuspegia hautatzerakoan, SPL-ak dituen produktu desberdin guztien aurkezpenarekin hasiko nintzateke, hau da, SPL-aren *VariantModel* bakoitzaren errepresentazio bisual batekin. Produktu baten errepresentazio bisuala hautagarri dauden ezaugarrietatik zenbat hautatu dituen erakustea izan liteke, honekin 1. galdera erantzuten. Adibidez:

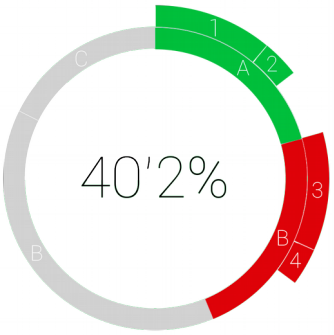


Modu honetan erabiltzaileak produktuaren konplexutasunaren ideia azkar hartuko du, geroz eta ezaugarri gutxiago hautatuta geroz eta sinpleagoa izango baita, eta alderantziz. Produkutaren bisualizazio honetan oinarrituta, produktu ikuspegiaren lehenengo pantaila produktu guziten aurkezpena izango da, hauek konplexutasunaren (hautatutako ezaugarri kopuruen) arabera ordenatuta:



Produktu sinpleenak hasieran azalduko dira eta konplexuenak amaieran, erabiltzailea produktu sinpleetan sakontzen doan heinean errazago ulertuko baitu produktu konplexuena.

Produktu zehatz bat aukeratzerakoan (click egiterakoan), honen informazio zehatza aurkeztuko zaio beste pantaila batean erabiltzaileari:



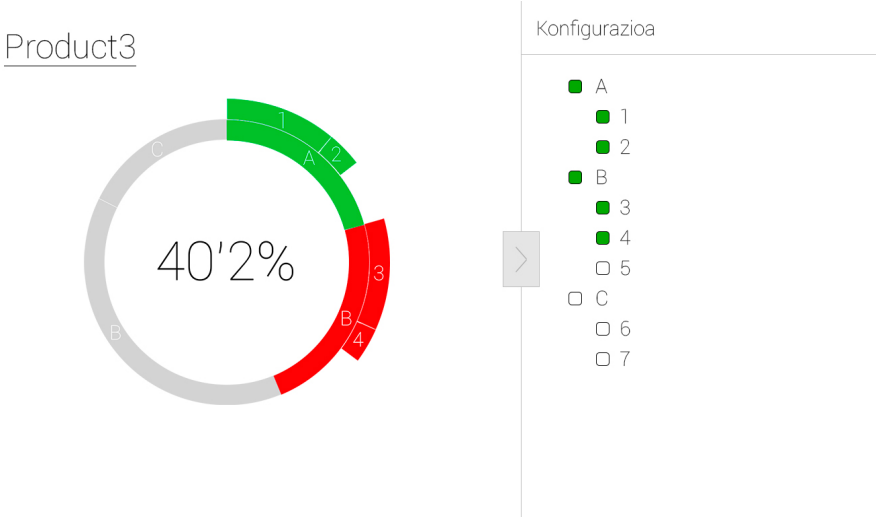
Grafiko honek 2. galderari erantzuten dio. Ehunekoa kalkulatzeko *Variant Point* bakoitzaren pisua kalkulatu behar da eta horretarako kode-fitxategietan agertzen diren *VP*-en karaktereak zenbatuko dira. Hau da, *VP* batek pisu handiagoa izango du geroz eta baldintzazko testu gehiago izanda. Adibidez:

```
// PV:IFCOND(hasFeature('A'))
    int kont = 0;
    for(int i = 0; i < 10; i++){
        kont++;
        // PV:IFCOND(hasFeature('B'))
        if(i % 2 == 0) kont--;
        // PV:ENDCOND
    }
// PV:ENDCOND
```

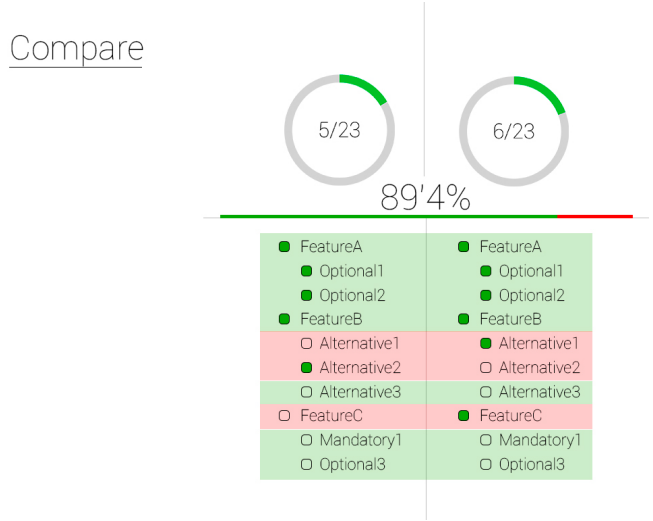
Kode zati honetan A ezaugarriaren pisua 199 izango da eta B ezaugarriarena 80.

Grafikoan, gainera, ezaugarrien eta haien umeen arteko erlazioa ere azaltzen da erabiltzaileak ikus dezan A ezaugarria hautatzeak, esaterako, baldintzatzen duen kode kopurua.

Grafikoa hurrengo pantailan agertuko litzateke, produktuaren konfigurazioa erakusten duen atalarekin (3. galderari erantzunez) batera:



Hasieran planteatutako galdera guztiei erantzuna ematen amaitzeko, bi produkturen arteko konparaketa egitea faltako litzateke. Horretarako, produktuen konfigurazioak aztertzea da ulergarriena, produktu batek dituen eta besteak ez dituen ezaugarriak modu bisualean ikusteko. Gainera, bi produktuen arteko antzekotasuna ehunekotan adieraziko da, produktuek amankomunean dituzten *Variation Point*-en pisuak kontuan hartuta. Azkenengo pantaila, beraz:



Teknologia aukeraketa

Aurkeztutako bisualizazioak egiteko eskuragarri dauden teknologia desberdinak ikertu dira, aztertutako teknologiak D3.js, Highcharts, Plotly.js eta AnyChart izanda.

Ebaluatu diren irizpideak hauek izan dira:

1. Planteatutako grafikoak sortu ahal izatea.
2. Grafikoak sortzeko erraztasuna.
3. Datuak modu egituratuan emateko aukera.

Eta informazioa taula honetan jaso da:

	D3.js	Highcharts	Plotly.js	AnyChart
1.	Posible da grafiko hauek sortzea eta aldaera asko egitea posible da.	Donut motako grafikoak sortzea posible bada ere, beste grafiko batzuk sortzea ez da posible.	Planteatutako grafikoen antzeko grafikoak sortzea posible da, baina ez dira guztiz igualak.	Planteatutako grafikoak sortzea posible da.
2.	Nahiz eta grafiko batzuk sortzea oso erraza den, diseinu honetarako planteatu grafikoak sortzea ez da batere erraza.	Grafikoak sortzeko informazio eta konfigurazio asko bete behar da. Hala ere, interneteko adibideak kopiatzea ere posible da.	Grafikoak sortzea erraza da, soilik datuak eta diseinua adierazten duten konfigurazioak pasa behar zaizkio.	Grafikoak sortu eta konfiguratzea erraza da, nahiz eta ez dituen konfigurazio aukera asko.
3.	Datuak modu hierarkikoan aurkezten dira, JSON formatuan. Formatua oso egokia da eta hierarkiak elementuen arteko guraso-ume harremanak modu naturalean aurkeztea errazten du.	Datuak JSON objektuez osatutako array batean aurkezten dira, elementu bakoitza JSON objektu bat izanda. Hierarkia objektuen IDekin sor daitekeen arren ez da ulertzeko erraza datuak aztertzerakoan.	Informazioa array desberdinetan pasa behar zaizkio, IDak, izenak, balioak... bisualki lotzea zaila eginda.	Datuak modu hierarkikoan eta JSON formatuan aurkezten dira. Formatu egokia eta hierarkiak elementuen arteko guraso-ume harremanak modu naturalean aurkeztea errazten du.

Taulako informazioan oinarrituz AnyChart teknologia erabiltzea planteatzen da, oso modernoa ez den arren planteatutako diseinuak sortzea posible egiteko aski baita.

C. ERANSKINA

Produktuaren erakusketa proba

Kalitatearen kontrola egiteko atazen artean, prototipoaren erakusketa proba bat egitea planteatu zen. Honen helburua amaierako erabiltzaile hipotetikoek aplikazioa erabiltzea eta haien sentsazioak jasotzea zen. Kalitate ataza hau SPL-en inguruko ezagutza maila desberdinak dituzten pertsonekin egin da eta hasieran modu fisikoan egiteko asmoa zegoen arren, egungo osasun egoera dela eta atazaren planteamendua aldatu behar izan da.

Elkarretaratze ezintasuna saiheste aldera, prototipoaren ebaluazioa galdera batzuen bitartez egin da eta hauen erantzunak *Google Forms* plataformaren bidez jaso. Galdera bakoitza 5.1 atalean zerrendatutako SPL-aren ezagutzaren puntu desberdinetan kokatzen da.

Galdetegia 12 galdera-blokez osatuta dago eta blokeek egitura hau jarraitzen dute:

- **Ezagutzarekin lotutako galdera(k).**
- **Erabilerraztasuna.** Erabiltzailearen sentsazioak jasotzeko galdera, erantzuna erraz aurkitu duenentz jakiteko helburuarekin.
- **Denbora.** Galderei erantzuna aurkitzeko erabiltzaileak behar izan duen gutxi gora beherako denbora.

Hauek dira galdetegian erabiltzaileei egindako galderak:

1. Zenbat produktu ditu *WacLine*-k?
2. Zenbat ezaugarri ditu SPL-ak?

3. Zein da produktu txikiena?
4. Zenbat hautazko ezaugarri ditu *'ConceptAndGo'* produktuak?
5. Zein da *'OATS'* produktuaren kode estaldura?
6. *'OATS'* produktuan, zein da *'Codebook'* ezaugarriaren eragina kodean, ehunekotan?
7. Zein da produktu txikienaren eta handienaren arteko antzekotasuna, ehunekotan?
8. Zenbat ezaugarri diagrama ditu *WacLine*-k?
9. Zein da *'Codebook'* ezaugarriaren aldakortasun mota?

   Zenbat ezaugarri-ume ditu *'Codebook'*-ek?

   Zenbat fitxategi **desberdinetan** du eragina *'Codebook'* ezaugarriak? Hau da, zenbat kode mailako aldaketa puntutan du eragina, fitxategi desberdinekoak izanda?
10. Idatzi *'Dropbox'* ezaugarriaren informazioa kontsultatzeko esteka
11. Zein da *'WebAnnotation'* kontzeptu maparen kontzeptuak gehien estaltzen dituen produktua?
12. Ze ezaugarrirekin ez du lotura *'visualization'* kontzeptuak?

    Zenbat baliabide ditu *'visualization'* kontzeptuak?

C.1 Parte hartzea

Galdetegian parte hartzeko SPL-en ezagutzaren inguruko hiru profil desberdin bilatzen ziren:

A. Zero ezagutza. SPL bat zer den ez dakien pertsona, informatikako ezagutzak izanda edo ez. Pertsona hauek ez dira garatutako aplikazioaren helburu-erabiltzaileak, baina SPL-en inguruko ezagutzarik izan gabe galderak erantzuteko gai izateak aplikazioak informazioa modu eraginkorrean azaltzen duela adieraz dezake.

B. Ezagutza gutxi. SPL-en inguruan noizbait entzun duen pertsona, software garatzailea edo ikaslea ziurrenik. Pertsona hauek aplikazioaren helburu-erabiltzaileak dira, SPL-aren munduan sartzen diren *incomer*-en ordezkoak.

C. **Ezagutza asko.** SPL-ekin (noiz edo noiz) aritzen den garatzailea, SPL-aren oinarrizko kontzeptu eta funtzionamendua ulertzen duena.

Momentuko osasun egoeraren ondorioz kalitate ataza hau birplanteatu behar izan da eta Interneten bidezko galdetegi bat erabili. Ondorioetan azalduko den moduan, birplanteaketa honek hainbat arazo suposatu ditu eta galdetegi honen parte hartzea oso eskasa izan da: bost pertsonek bete dute galdetegia. Pertsona horietatik hiru A profilekoak (zero ezagutza) eta bi B profilekoak (ezagutza gutxi) izan dira.

Nahiz eta pertsona gutxiren erantzuna jaso den, erantzun hauek ondorio baliotsuak ateratzeko erabili dira eta C.1 taulan erantzunen inguruko hainbat informazio ikus daiteke:

Profila	**Erantzun denbora (seg.)**			**Asmatze-tasa (%)**	**Erraztasuna***
	minimoa	**maximoa**	**bataz bestekoa**		
A	3 s	120 s	29’63 s	% 83’3	2’8
B	1 s	60 s	14’33 s	% 91’7	1’9
Guztiak	1 s	120 s	24,53 s	% 86’1	2’5

C.1 Taula: Galdetegiaren erantzunen laburpena.

** Erraztasunaren pertzepzioa (subjektiboa): [1-5] eskala batean, 1 oso erraza eta 5 oso zaila izanda.*

C.2 Ondorioak

Elkarretaratze ezintasunak arazo asko suposatu dizkio kalitate ataza honi galdetegiaren bidez egindako galderak erantzuteko web aplikazioa atzitu behar zelako. Web aplikazioa atzitzeko proiektuaren garatzaileak aplikazioa interneten atzigarri jarri behar izan zuen, baina etxeko konexioa eta baliabide mugatuen ondorioz ez da erraza izan. Honek, nahi zirenak baino erantzun gutxiago jasotzea eragin du. Hala ere, jasotako erantzun eskasek ondorio orokor batzuk ateratzea ahalbidetu dute:

- **Hasiberrientzako tour-ak aplikazioa ulertzen laguntzen du.** Galdetegia hasi aurretik tour-a erabiltzeko aukera eskaini da, honen bidez aplikazioaren funtzionalitate guztiak azaltzen baitira. Tour-a erabili duten erabiltzaileek *’erantzunak aurkitzen*

laguntzen duela' uste dute, oro har. Onartu beharra dago, haatik, erantzuna aurkitzeko denboretan ez dela desberdintasun handirik topatu, baina honek beste hainbat faktoreren menpe egon daiteke (erabiltzaileak teknologiekin duen elkarrekintza, esaterako).

- **Aplikazioa ez da edonorentzat.** Egindako aplikazioak ez du inoiz edozein erabiltzailerentzat izateko asmoa izan, Software Produktu-Lerroetan berria den garatzailearentzat izateko planteatu da hasieratik. Erabiltzaile honek ezaugarri batzuk dituela suposa daiteke: teknologiarekin ongi moldatzen da, informatikarekin eta softwarearen garapenarekin lotutako ezagutzak ditu, terminologia teknikoa ulertzeko gaitasuna du... Argi dago garatzaile desberdin asko daudela munduan eta bakoitzak gaitasun desberdinak izango ditu, baina orokorrean, software garatzaile batek edo abokatu batek ez dute profil antzekoa izango, teknologiaren menperatzearen ikuspuntutik.

- **Kontzeptu eta terminologia nahasgarriak daude.** Galdera gehienetan erantzun zuzenak jaso diren arren, batzuetan erabiltzaileek galderaren interpretazio okerra egin dute edo ezin izan dute erantzun zuzena topatu. Hau kontuan hartzekoa da, erabiltzaile berriek arazo berarekin topa daitezkeelako. Terminologia, gainera, nahasgarria izan daiteke aplikazioa hizkuntza anitzetan kontsultatu daitekeelako eta SPL-aren terminologia euskaraz guztiz finkatua ez dagoelako.

- **Azalpenak inoiz ez daude soberan.** Produktu bat garatzen denean garatzaileen ikuspegia baldintzatuta dago, izan ere, norberak egindako funtzionalitateek argiak dirudite beti. Erabiltzaile berri baten ikuspuntutik, ordea, funtzionalitate berri bakoitzak ahalegina eta ulermena eskatzen du eta prozesu horretan azalpenak eskaintzeak asko lagun dezake. Hasiberrientzako tour-a honetara zuzenduta egon arren, funtzionalitate batzuek laguntza gehiago behar dutela dirudi.

- **Erantzun okerrak aplikazioaren hutsegiteak dira.** Galdetegiko galderak gaizki erantzutea faktore askoren menpe egon daiteke (erabiltzailearen atentzioa, ulermen maila, aplikazioaren erabilerraztasuna...), baina argi dagoena da erantzun oker batek SPL-aren ulermen okerra adierazten duela. Aplikazioaren helburua SPL-en ikasketa prozesuan laguntzea bada, erantzun okerrek aplikazioak hobetu beharreko atalak islatuko dituzte.

D. ERANSKINA

PositionalXMLReader.java

```
1  // PositionalXMLReader.java
2  // Based on StackOverflow response:
3  //      https://stackoverflow.com/questions/4915422/get-line-number-from-xml-node-java
       /4918080#4918080
4
5  import java.io.IOException;
6  import java.io.InputStream;
7  import java.util.Stack;
8
9  import javax.xml.parsers.DocumentBuilder;
10 import javax.xml.parsers.DocumentBuilderFactory;
11 import javax.xml.parsers.ParserConfigurationException;
12 import javax.xml.parsers.SAXParser;
13 import javax.xml.parsers.SAXParserFactory;
14
15 import org.w3c.dom.Document;
16 import org.w3c.dom.Element;
17 import org.w3c.dom.Node;
18 import org.w3c.dom.NodeList;
19 import org.xml.sax.Attributes;
20 import org.xml.sax.Locator;
21 import org.xml.sax.SAXException;
22 import org.xml.sax.helpers.DefaultHandler;
23
24 public class PositionalXMLReader {
25     final static String START_LINE_NUMBER_KEY_NAME = "startLineNumber";
26     final static String END_LINE_NUMBER_KEY_NAME = "endLineNumber";
27     final static String LINE_CONTENT_KEY_NAME = "lineTextContent";
28
29     public static Document readXML(final InputStream is) throws IOException, SAXException {
30         final Document doc;
```

```
31          SAXParser parser;
32          try {
33              final SAXParserFactory factory = SAXParserFactory.newInstance();
34              parser = factory.newSAXParser();
35              final DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance()
    ;
36              final DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
37              doc = docBuilder.newDocument();
38          } catch (final ParserConfigurationException e) {
39              throw new RuntimeException("Can't create SAX parser / DOM builder.", e);
40          }
41
42          final Stack<Element> elementStack = new Stack<Element>();
43          final StringBuilder textBuffer = new StringBuilder();
44          final DefaultHandler handler = new DefaultHandler() {
45              private Locator locator;
46
47              @Override
48              public void setDocumentLocator(final Locator locator) {
49                  this.locator = locator; // Save the locator, so that it can be used later for
    line tracking when traversing nodes.
50              }
51
52              @Override
53              public void startElement(final String uri, final String localName, final String qName
    , final Attributes attributes)
54                      throws SAXException {
55                  addTextIfNeeded();
56                  String elString = "<" + qName ;
57                  final Element el = doc.createElement(qName);
58                  for (int i = 0; i < attributes.getLength(); i++) {
59                      elString += " " + attributes.getQName(i) + "=\"" + attributes.getValue(i) +
    "\"";
60                      el.setAttribute(attributes.getQName(i), attributes.getValue(i));
61                  }
62
63                  elString += ">";
64                  el.setUserData(START_LINE_NUMBER_KEY_NAME, String.valueOf(this.locator.
    getLineNumber()), null);
65                  el.setUserData(LINE_CONTENT_KEY_NAME, elString, null);
66                  elementStack.push(el);
67              }
68
69              @Override
70              public void endElement(final String uri, final String localName, final String qName)
    {
71                  addTextIfNeeded();
72                  final Element closedEl = elementStack.pop();
73                  closedEl.setUserData(END_LINE_NUMBER_KEY_NAME, String.valueOf(this.locator.
    getLineNumber()), null);
74
75                  String nodeElementsString = "";
```

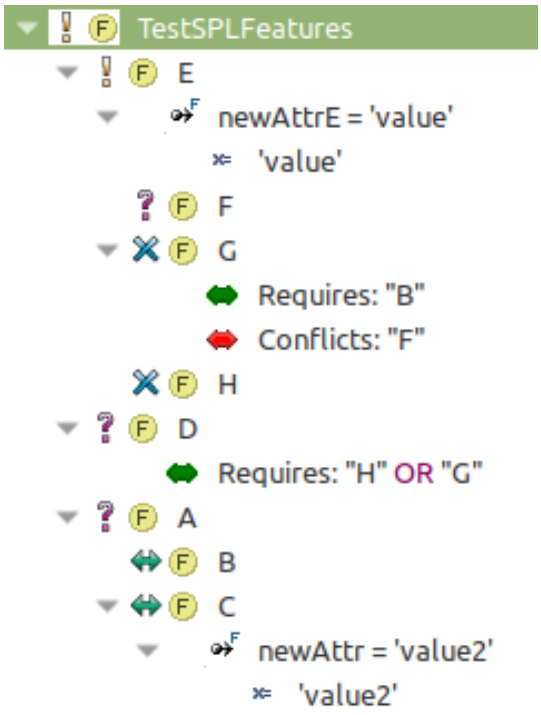
```
76                  NodeList nl = closedEl.getChildNodes();
77
78                  for(int i = 0; i<nl.getLength();i++) {
79                      Node n = nl.item(i);
80                      if(n.getNodeType() == Node.ELEMENT_NODE) {
81                          Element e = (Element) n;
82                          nodeElementsString += e.getUserData(LINE_CONTENT_KEY_NAME);
83                      }else {
84                          nodeElementsString += n.getTextContent().replaceAll("\\s+"," ");
85                      }
86                  }
87
88                  String elString = (String) closedEl.getUserData(LINE_CONTENT_KEY_NAME);
89                  elString += nodeElementsString;
90                  elString += "</" + closedEl.getNodeName() + ">";
91
92                  closedEl.setUserData(LINE_CONTENT_KEY_NAME, elString, null);
93
94                  if (elementStack.isEmpty()) { // Is this the root element?
95                      doc.appendChild(closedEl);
96                  } else {
97                      final Element parentEl = elementStack.peek();
98                      parentEl.appendChild(closedEl);
99                  }
100             }
101
102             @Override
103             public void characters(final char ch[], final int start, final int length) throws
    SAXException {
104                 textBuffer.append(ch, start, length);
105             }
106
107             // Outputs text accumulated under the current node
108             private void addTextIfNeeded() {
109                 if (textBuffer.length() > 0) {
110                     final Element el = elementStack.peek();
111                     final Node textNode = doc.createTextNode(textBuffer.toString());
112                     el.appendChild(textNode);
113                     textBuffer.delete(0, textBuffer.length());
114                 }
115             }
116         };
117
118         parser.parse(is, handler);
119
120         return doc;
121     }
122 }
```

E. ERANSKINA

Pure::variants-ekin egindako SPL baten adibidea

Eranskin honetan pure::variants softwarearekin egindako SPL baten adibidea aurki daiteke. SPL hau oso sinplea da eta pure::variants-en oinarrizko fitxategiak aztertzeko erabili da.

E.1 *FeatureModel*



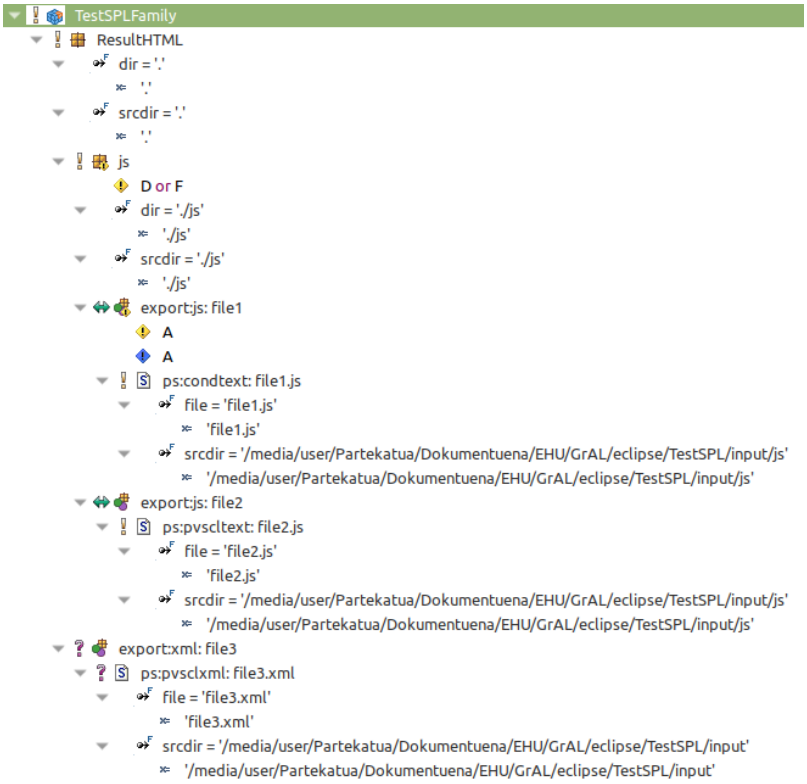
E.1 Irudia: SPL-aren *FeatureModel*-a, *Eclipse*-n ikusita.

Ezaugarri baten errepresentazioa *FeatureModel*-aren fitxategian:

```
<cm:element cm:id="iExQIbH1WHvtTzEb-" cm:name="TestSPLFeatures" cm:type="ps:feature" cm:class="ps
    :feature" cm:default="on">
  <cm:relations cm:id="i0UrD5GZTsGvoQkC_" cm:class="ps:children">
    <cm:relation cm:id="inGa4Ub75ujoS_2T8" cm:type="ps:mandatory" cm:range="n">
      <cm:target cm:id="iVqtFApKN8b_xObLc">./iUXDhVMWji1bTeAfr</cm:target>
    </cm:relation>
    <cm:relation cm:id="ixZRPnAxgZm4rldRj" cm:type="ps:optional" cm:range="[0,n]">
      <cm:target cm:id="iSTUgFuXKKx3dSqbE">./i2nxwqLB3hmD7AWyl</cm:target>
      <cm:target cm:id="iLUXgI2JnDXTHMNg9">./i8kWmWfyOodrTOOp3</cm:target>
    </cm:relation>
  </cm:relations>
  <cm:properties>
    <cm:property cm:id="iSfBNstWaQL7IKldF" cm:type="ps:string" cm:fixed="true" cm:readonly="true"
     cm:name="ps:Source">
      <cm:constant cm:id="iLVyDMSzlPYoIuoX8" cm:type="ps:string">user</cm:constant>
    </cm:property>
    <cm:property cm:id="iXi1eNgOgwWQGCPgz" cm:type="ps:datetime" cm:fixed="true" cm:readonly="
     true" cm:name="ps:Created">
```

```
16        <cm:constant cm:id="iN1ZSe_WAD0pX3LjH" cm:type="ps:datetime">2020-02-26T13:44:23.297Z</cm:
      constant>
17      </cm:property>
18      <cm:property cm:id="i4euGdjqTQV7zfpYH" cm:type="ps:string" cm:fixed="true" cm:readonly="true"
        cm:name="ps:ChangedBy">
19        <cm:constant cm:id="ixpnVaxeviphHEeqD" cm:type="ps:string">user</cm:constant>
20      </cm:property>
21      <cm:property cm:id="inGOvYy6YTnyNWrfA" cm:type="ps:datetime" cm:fixed="true" cm:readonly="
      true" cm:name="ps:Changed">
22        <cm:constant cm:id="ieAYaQrrOkQzawV28" cm:type="ps:datetime">2020-06-16T17:11:33.426Z</cm:
      constant>
23      </cm:property>
24    </cm:properties>
25  </cm:element>
```

E.2 *FamilyModel*



E.2 Irudia: SPL-aren *FamilyModel*-a, *Eclipse*-n ikusita.

Kode-elementu baten errepresentazioa *FamilyModel*-aren fitxategian:

```
<cm:element cm:id="iFQjytnZfMwu5_eIr" cm:name="TestSPLFamily" cm:type="ps:family" cm:class="ps:family" cm:default="on">
  <cm:relations cm:id="iZhd_w_lgv9m-rkww" cm:class="ps:children">
    <cm:relation cm:id="is58hDvSkFshJCj_h" cm:type="ps:mandatory" cm:range="n">
      <cm:target cm:id="iFCdwRJGeLIEM-6eA">./idcPJ83T6d_YlAQnU</cm:target>
    </cm:relation>
  </cm:relations>
  <cm:properties>
    <cm:property cm:id="iwyEA6oacE9B6uyIy" cm:type="ps:string" cm:fixed="true" cm:readonly="true" cm:name="ps:Source">
      <cm:constant cm:id="ig-MZ5YhFHNFYRKTU" cm:type="ps:string">user</cm:constant>
    </cm:property>
    <cm:property cm:id="ikVGH_cjnA41s7Zxm" cm:type="ps:datetime" cm:fixed="true" cm:readonly="true" cm:name="ps:Created">
      <cm:constant cm:id="ishdV-sAcdX9Of7QN" cm:type="ps:datetime">2020-02-26T13:44:23.506Z</cm:constant>
    </cm:property>
    <cm:property cm:id="iJxLFzwE5y4cXvk-d" cm:type="ps:string" cm:fixed="true" cm:readonly="true" cm:name="ps:ChangedBy">
      <cm:constant cm:id="iuKG98E4SepOZUHyw" cm:type="ps:string">user</cm:constant>
    </cm:property>
    <cm:property cm:id="ifN_ThcfWOh7EkAXM" cm:type="ps:datetime" cm:fixed="true" cm:readonly="true" cm:name="ps:Changed">
      <cm:constant cm:id="inBYZRVQiiht6-TBX" cm:type="ps:datetime">2020-02-26T14:02:57.162Z</cm:constant>
    </cm:property>
  </cm:properties>
</cm:element>
```

E.3 *VariantModel*

E.3 Irudia: SPL-aren *VariantModel* baten bi atalak (ezaugarrien eta kode-elementuen hautaketa), *Eclipse*-n ikusita.

SPL-aren elementu baten (ezaugarri edo kode-elementu) errepresentazioa *VariantModel*-aren fitxategian:

```
<cm:element cm:id="isr05U9pGrh5FJSzc" cm:type="ps:selected" cm:class="ps:selection">
  <cm:relations cm:id="iq78NgCk3DPafzy00" cm:class="ps:references">
    <cm:relation cm:id="iUz7WxkK5KYgWRC7y" cm:type="ps:references">
      <cm:target cm:id="imW3w042P9pQpvUnO">i7F5ZxAqBG-VNU4CJ/iExQIbH1WHvtTzEb-</cm:target>
    </cm:relation>
  </cm:relations>
  <cm:properties>
    <cm:property cm:id="iLpOLpAEqu6NOdk7z" cm:type="ps:string" cm:invisible="true" cm:name="ps:
     selector">
      <cm:constant cm:id="i2bUlN9g3CfI4Ya9K" cm:type="ps:string">ps:auto</cm:constant>
    </cm:property>
  </cm:properties>
</cm:element>
```

F. ERANSKINA

Proiektuaren erronkak: kodea

F.1 Aldaketa puntuen minatua

```
/* REGEX */
private final static String INSIDE_BRACKETS = "\\(\\w+|\\w+\\(.*\\)|[\\s
    \\'\\.\\,\\:\\-\\>\\=\\<])+\\)";
private final static String COMMENT = "\\/[\\/\\*]";

private static Pair<String, ArrayList<Feature>> hasVariationStatement(String line) {
    for (String s : MainClass.VARIATION_POINT_NO_XML_STATEMENTS) {
        if (line.contains(s)) {

            // Capture what is between (...)
            // (\(\w+|\w+\(.*\)|[\s\'\.\,\:\-\>\=\<])+\))
            Pattern p = Pattern.compile("(" + INSIDE_BRACKETS+ ")");
            Matcher m = p.matcher(line);
            if (m.find()) {

                String r = m.group()
                        .replaceAll(" or | OR | and | AND | NOT | \\| | \\&", " ");

                // HAS VARIATION STATEMENT
                Pair<String, String> cleaned = cleanVariationStatement(line, s);
                String expr = cleaned.a;
                String rest = cleaned.b;

                // PROCESS THIS LINE AGAIN
                checkVPString = true;
                vpString = rest;

```

```
                return new Pair<String, ArrayList<Feature>>(expr, extractVPsFromStatement(r));
            }
        }
    }

    return null;
}

private static Pair<String, String> cleanVariationStatement(String line, String statement) {
    String r1; // Expresion
    String r2; // Rest

    String startRegex = "[^(" + statement + ")]*";

    // .*(\/[\/\*]\s*PVSCL:IFCOND\((\w+|\w+\(.*\)|[\s\'\.\,\:\-\>\=\<])+\)(\s*\*\/)?\S*)(.*)
    Pattern p1 = Pattern.compile(
            startRegex + "(" + COMMENT + "\\s*" + statement + INSIDE_BRACKETS + "(\\s*\\*\\/)?\\S
    *)(.*)");
    Matcher m1 = p1.matcher(line);

    Pattern p2 = Pattern.compile(startRegex + "(\\s*" + statement + INSIDE_BRACKETS + "(\\s
    *\\*\\/)?\\S*)(.*)");
    Matcher m2 = p2.matcher(line);

    if (m1.find()) {
        // Contains "//" or "/*"
        r1 = m1.group(1);
        r2 = m1.group(4);
    } else if (m2.find()) {
        // Doesn't contain comment marks
        r1 = m2.group(1);
        r2 = m2.group(4);
    } else {
        // Failsafe
        r1 = line;
        r2 = "";
    }

    return new Pair<String, String>(r1, r2);
}
```

F.2 SPL eta kontzeptu mapen arteko loturak

```
private static ArrayList<Feature> connectLinkElementWithFeatures(String line) {

    if(spl == null)
        return new ArrayList<>();
```

```
5
6       ArrayList<Feature> listfeatures = new ArrayList<>();
7       for (FeatureModel fm : spl.getFeatureModels()) {
8           for (Feature f : fm.getFeatures()) {
9               Pattern p = Pattern.compile("\\b("+f.getName()+")\\b");
10              Matcher m = p.matcher(line);
11              if (m.find()) {
12                  listfeatures.add(f);
13              }
14          }
15      }
16      return listfeatures;
17  }
```

F.3 Osatu gabeko konfigurazioa

```
1   // Remove treated Features and CodeElements from untreated list
2   for (VariantComponent vc : vm.getVariants()) {
3       if (vc instanceof VariantFeature) {
4           VariantFeature vf = (VariantFeature) vc;
5           untreatedFeatures.remove(vf.getFeature());
6       } else if (vc instanceof VariantCode) {
7           VariantCode vf = (VariantCode) vc;
8           untreatedCodeElements.remove(vf.getCodeFile());
9       }
10  }
11
12  // Treat unselected Features that doesn't appear on the file
13  for (Feature f : untreatedFeatures) {
14      VariantFeature vf = new VariantFeature(GenericUtils.generateID(), false, vm, f);
15      vm.addVariant(vf);
16  }
17
18  // Treat unselected CodeElements that doesn't appear on the file
19  for (CodeElement ce : untreatedCodeElements) {
20      VariantCode vf = new VariantCode(GenericUtils.generateID(), false, vm, ce);
21      vm.addVariant(vf);
22  }
```

F.4 Produktuaren grafiko osoa

```
1   private String unselectedJson;
2   private int unselectedSize;
```

```
3
4   @Override
5   public Triplet<String, Integer, Integer> getProductInfoById(String id) {
6
7       unselectedJson = "";
8       unselectedSize = 0;
9
10      VariantModel product = getProductById(id);
11
12      if (product != null) {
13          List<Feature> rootFeatures = featureBL.getRootFeaturesOfSpl(product.getSpl().getId());
14
15          int unselected = 0;
16          int fromTotalSize = 0;
17
18          String resultJson = "[";
19          for (Feature root : rootFeatures) {
20              Pair<String, Integer> rootR = infoRecursive(product, root, 0);
21              resultJson += rootR.getFst();
22              unselected += rootR.getSnd();
23              fromTotalSize += featureBL.getFeatureSize(root.getId());
24          }
25
26          int selectedFeaturesSize = fromTotalSize - unselected;
27
28          // Add unselected part
29          resultJson += "{ name: 'unselected', value: " + unselectedSize + ", "
30                  + "children: ["
31                  + unselectedJson
32                  + "]}";
33
34          resultJson += "]";
35
36          resultJson = "[{ id: 'root', name: '" + product.getFilename() + "', children: " +
        resultJson + "}]";
37
38          Triplet<String, Integer, Integer> result = new Triplet<String, Integer, Integer>(
        resultJson,
39                  selectedFeaturesSize, fromTotalSize);
40          return result;
41      }
42
43      return null;
44  }
45
46  private Pair<String, Integer> infoRecursive(VariantModel vm, Feature actual, int level) {
47
48      int size = featureBL.getFeatureSize(actual.getId());
49
50      int unselected;
51      int selected;
52
```

```
53      String children = "[";
54
55      // Is feature selected?
56      if(isFeatureSelected(vm, actual)) {
57          // Yes:
58          //  unselected = 0 + sum(child.unselected);
59          //  children = [...];
60
61          unselected = 0;
62
63          List<Feature> childs = featureBL.getChildFeaturesOfFeature(actual.getId());
64
65
66
67          for (Feature child : childs) {
68              Pair<String, Integer> childR = infoRecursive(vm, child, level + 1);
69              children += childR.getFst();
70              unselected += childR.getSnd();
71          }
72
73      }else {
74          // No:
75          //  unselected = size;
76          //  children = [];
77
78          unselected = size;
79      }
80
81      children += "]";
82
83      selected = (size - unselected) >= 0 ? (size - unselected) : 0 ;
84
85      // Fill info
86
87      String json = "";
88
89      if (level == 1) {
90
91          // Selected part
92          json = "{id: '" + actual.getId() + "', "
93                  + "name: '" + actual.getName() + "', "
94                  + "value: " + selected + ", "
95                  + "children: " + children + "},";
96
97          // Unselected part (keep it apart)
98          unselectedJson += "{id: '" + actual.getId() + "', "
99                  + "name: '" + actual.getName() + "', "
100                 + "value: " + unselected + ","
101                 + "normal: {fill: '#c5c5c5'}},";
102
103         unselectedSize += unselected;
104
```

```
105     } else {
106         // Only selected part
107         if(isFeatureSelected(vm, actual)) {
108             json = "{id: '" + actual.getId() + "', "
109                     + "name: '" + actual.getName() + "', "
110                     + "value: " + selected + ", "
111                     + "children: " + children + "},";
112         }
113     }
114 
115     Pair<String, Integer> result = new Pair<String, Integer>(json, unselected);
116 
117     return result;
118 
119 }
```

Bibliografia

[1] O. Corporation, "Java se development kit 8." https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html, 2014.

[2] A. S. Foundation, "https://maven.apache.org." https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html, 2019.

[3] J. Sonoo Jaiswal, "Gradle vs maven." https://db-engines.com/en/ranking.

[4] O. Corporation, "Mysql: the world's most popular open source database." https://www.mysql.com, 2020.

[5] S. IT, "Db-engines ranking by popularity." https://db-engines.com/en/ranking, May 2020.

[6] L. Torvalds, "git –local-branching-on-the-cheap." https://git-scm.com/, 2005.

[7] AlternativeTo, "Git alternatives." https://alternativeto.net/software/git/.

[8] Slant, "What are the best version control systems?." https://www.slant.co/topics/370/~best-version-control-systems, May 2020.

[9] I. Eclipse Foundation, "Eclipse: The platform for open innovation and collaboration." https://www.eclipse.org/.

[10] I. Eclipse Foundation, "Jgit: an implementation of the git version control system in pure java." https://www.eclipse.org/jgit/.

[11] I. VMware, "Spring makes programming java quicker, easier, and safer for everybody.." https://spring.io/.

[12] I. VMware, "Spring boot makes it easy to create stand-alone, production-grade spring based applications that you can 'just run'.." https://spring.io/projects/spring-boot#overview.

[13] I. Rollbar, "Most popular java web frameworks in 2019." https://rollbar.com/blog/most-popular-java-web-frameworks/, 2019.

[14] A. S. Foundation, "Apache tomcat." https://tomcat.apache.org/, 1999.

[15] A. S. Foundation, "Apache tiles." https://tiles.apache.org/index.html.

[16] A. S. Foundation, "Apache tiles moves to attic." https://attic.apache.org/projects/tiles.html, December 2018.

[17] B. Eich, "Javascript: high-level programing language." https://www.javascript.com, December 1995.

[18] J. Resig, "jquery: write less, do more." https://jquery.com/, August 2006.

[19] J. T. Mark Otto, "Build fast, responsive sites with bootstrap." https://getbootstrap.com/, August 2011.

[20] U. Sossou, "Bootstrap tour: The easiest way to show people how to use your website." https://bootstraptour.com/, July 2012.

[21] O. C. W. Carlos Delgado, "Top 10: Best tour (website guide) javascript and jquery plugins." https://bootstraptour.com/, January 2019.

[22] I. Bozhanov, "jstree: jquery tree plugin." https://bootstraptour.com/, October 2009.

[23] jQueryScript.Net, "Free jquery tree view plugins." https://www.jqueryscript.net/tags.php?/tree%20view/.

[24] T. Wood, "Momentjs: parse, validate, manipulate, and display dates and times in javascript." https://momentjs.com/, March 2011.

[25] AnyChart, "Javascript charts designed to be embedded and integrated." https://www.anychart.com/products/anychart/overview/.

[26] N. Software, "Gojs: Interactive javascript diagrams for the web." https://gojs.net/latest/index.html.

[27] I. Fonticons, “Font awesome: the web’s most popular icon set and toolkit.” https://fontawesome.com/.

[28] L.Ñ. Paul Clements, “Software product lines: Practices and patterns,” August 2001.

[29] H. M. Onekin, “Wacline.” https://onekin.github.io/WacLine/, 2019.

[30] A. Max Rehkopf, “istorias de usuario con ejemplos y plantilla,”

[31] S. @priomsrb, “Get line number from xml node - java.” https://stackoverflow.com/questions/4915422/get-line-number-from-xml-node-java/4918080#4918080, 2011.