Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

▪ Gradu Amaierako Proiektua ▪

# GitLine tresnaren zabaltzea: software produktu-lerroen kudeaketa.

---

Eider Irigoyen Garcia

2015 - ekaina

informatika fakultatea
facultad de informática

# Laburpena

Gradu amaierako proiektu hau software produktu-lerroen testuinguruan kokatzen da, GitLine aplikazioan funtzionalitate batzuk gehituz. GitLine, Onekin ikerketa taldean [13] garatua, GitHub-en funtzionalitate gehigarriak eskaintzen dituen Firefox-eko plugin bat da. Plugin honek SPL produktu eraikitzaileari laguntza eskaintzen dio ezaugarri biltegiak eta produktu biltegiak sortu eta beraien arteko sinkronizazioa mantentzeko. Sinkronizazio hori hobetzeko, ikasleak hiru funtzionalitate berri inplementatu ditu.

GitLine-en inplementatuta dagoen *productFork* funtzionalitateari, hau da, ezaugarri biltegi batetik aukeratutako ezaugarriekin produktu biltegia sortzeko funtzionalitateari, hobekuntza nabarmen bat gehitu dio ikasleak: baliozkoak diren produktu biltegiak bakarrik sortzen direla ziurtatzen du (*balioztaketa*). Hobekuntza honi esker, produktu biltegiak erabat sinkronizatuta daude ezaugarri biltegiarekin.

Ezaugarri biltegi baten bizi-zikloa handitzeko, ikasleak ezaugarri biltegian ezaugarri berriak txertatzeko aukera inplementatu du (*ezaugarria txertatu*). Gainera, hasieran aipatutako ezaugarri biltegi eta produktu biltegien arteko sinkronizazio hori bermatzeko, ezaugarri berriak txertatzerakoan, ezaugarri berri horiek sortuta dauden produktu biltegietara hedatzeko aukera inplementatu du ere (*ezaugarria hedatu*).

# Gaien Aurkibidea

# Irudi, Taulen eta Grafikoen zerrenda

# 1. KAPITULUA

# Sarrera

Orain arte, softwaregintzan lan egiten duten enpresetan banaka egin dituzte produktuak, eta sistema osoak zerotik hasita garatu dituzte, baina hori, noski, ez da errentagarria. Orain iristen ari da industrializazioa software-mundura: teknologiaren, metodologiaren eta diseinu-tresnen aurrerapausoei esker, berrerabiltzea posible da. Software produktu-lerroak (SPL) dira gakoa. SPL edo software produktu-lerroak, antzekoak diren software sistemen bilduma bat eraikitzea dute helburu gisa, eta software artefaktuen multzo partekatu batetik abiatzen dira, produkziorako bide komunak erabiliz. Industrializazio arlo honetan, Leticia Montalvillo ikertzailea GitLine softwarea garatzen ari da GitHub-en [5] oinarrituta. Proiektu honen helburua, hain zuzen ere, GitLine-ren garapenean laguntzea da, beharrezkoak diren funtzionalitate batzuk inplementatuz.

5. kapituluan, proiektu honetan egin den garapenaren lehen aurkezpena egiten da, hurrengo kapituluetarako sarrera moduan balio dezan. Gainera, garatutako funtzionalitateak azaltzen diren kapituluetan (6 eta 7 kapituluak) garapenaren inguruko erronkak edota izandako zailtasunak laburbiltzen saiatu da. Bukatzeko, 11. kapituluan proiektu osoaren ondorioak bildu dira, baita etorkizuneko lanak ere.

# 2. KAPITULUA

# Proiektuaren helburu-dokumentua

Proiektuaren helburu dokumentuan, proiektuaren irismena, proiektuan zehar burutuko diren atazak, denboraren plangintza, interesatuak, kalitatearen kudeaketa eta arriskuen kontrola azalduko dira.

## 2.1 Irismena

Irismenaren atala hiru zatitan banatuta dago. Lehenengoan, proiektuaren helburuak zehazten dira, bigarrenean, lanaren deskonposaketa egitura eta azkenekoan, helburuak bete ahal izateko burutu beharko diren atazen zehaztapena eta deskribapena.

### 2.1.1 Proiektuaren helburuak

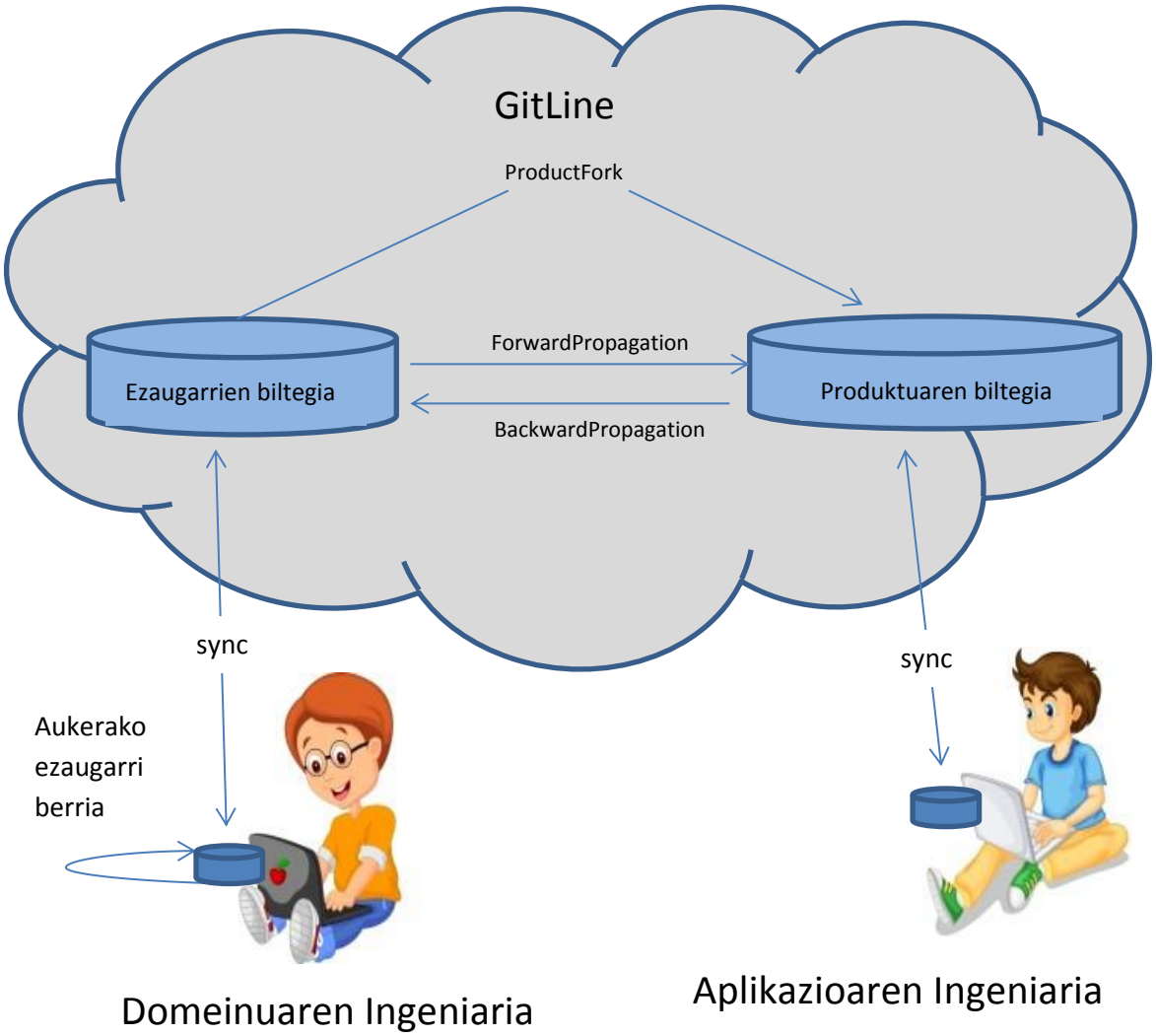
Proiektu honetan landuko den softwarea, GitLine [8], bi teknologien nahasketa dela esan daiteke: FeatureIDE [3] eta GitHub [5].

FeatureIDE Eclipsen oinarritutako kode irekiko lan ingurunea da, Ezaugarriei Bideratutako Software Garapena (FOSD, inglesez) metodologia aplikatzen duena. Eclipse-ko plugin honek FOSD-en atal guztiak eusten ditu: ezaugarri eredua definitu (domeinuaren analisia), kode aktiboak ezaugarrietara mapeatu (domeinuaren inplementazioa), aplikazio berriarentzako konfigurazio berria definitu (betekizunen analisia) eta produktua automatikoki sortzen du (softwarearen gauzatzea).

GitHub-ek softwareko lankidetza-proiektuak gordetzeko balio du. Proiektu bakoitzean adarretan (*branc*h), proiektuaren kopiak egin daitezke bertsio desberdinetan independenteki lan egiteko.

GitLine SPL biltegiekin (repository, inglesez) lan egiteko GitHub-en hedapena da, ezaugarri eta produktu ezberdinen biltegiez osatuta dago. Ezaugarri biltegiak ezaugarrien inplementazioa dauka bere adarretan (*branches*), ezaugarri bakoitzeko adar bat. Produktu biltegiek, GitHub biltegi "normal" baten moduan, produktu bakoitzaren inplementazioa gordetzen dute.

Hedaketarako, hiru funtzio gehitu dira, 1. irudian agertzen direnak: *newOptionalFeature*, *productFork* eta *forwardPropagation*. Domeinuaren ingeniariek eta Aplikazioaren ingeniariek biltegi ezberdinen gainean egiten dute lan. Domeinuaren ingeniariek ezaugarri biltegien gainean egiten dute lan, aldiz, Aplikazioaren ingeniariek produktu biltegien gainean. Sinkronizaziorako, beharrezkoa da produktuaren eratorpena (*productFork*) eta hobetutako ezaugarriak aurreko produktuetara hedatzea (*forwardPropagation*). Bestalde, *productFork* eta *forwardPropagation* Firefox Plugin baten bidez sortuta daude, plugin honen bidez, GitHub hedatzen da SPL biltegien semantika onartzea lortzen da.




1. irudia: GitLine-ren egitura.

Beraz, proiektu honen helburua GitLine izeneko proiektu handiaren zati bat inplementatzea da. Hiru dira inplementatu beharreko funtzionalitateak: *balioztaketa*, *aukerako ezaugarri bat gehitu* eta *derrigorrezko ezaugarri bat gehitu*.

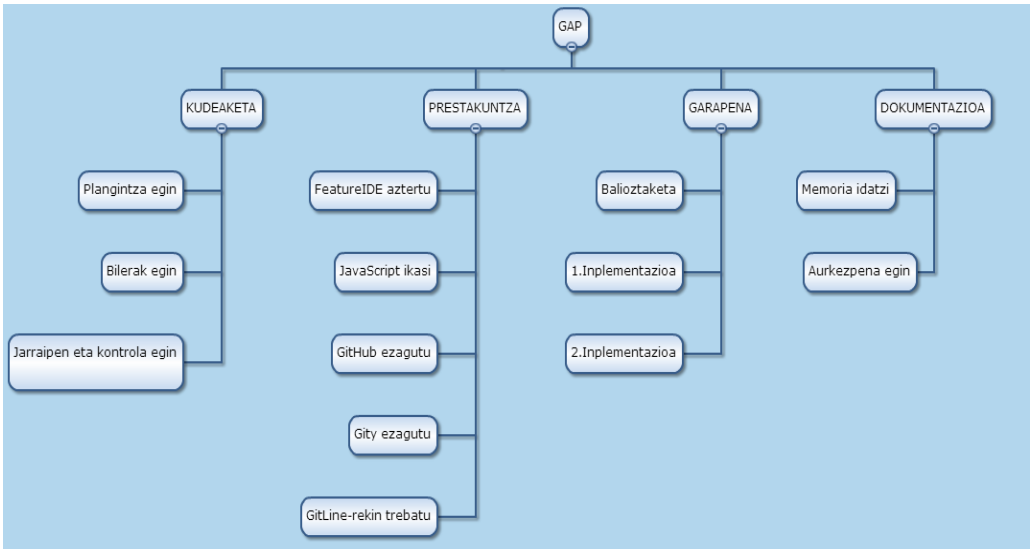
- *balioztaketa*: Sortutako produktu guztiek ezaugarri biltegian zehaztutako baldintza guztiak betetzen dituztela bermatu.
- *aukerakoEzaugarriaGehitu*: ezaugarri ereduan aukerakoa den ezaugarri bat gehitu.
- *derrigorrezkoEzaugarriaGehitu:* ezaugarri ereduan derrigorrezkoa den ezaugarria gehitu, eta aldaketa horren hedapena produktuetara.

ERRONKAK:

- FeatureIDE plugin-a aztertu eta ezaugarrien konfigurazioa balioztatzeko erabiltzen duen inplementazioa ikertu.
  - FeatureIDE-k erabiltzen duen xml sintaxia ikasi: ulertu nola zehazten diren ezaugarrien motak eta horien murriztapenak.
  - Ulertu FeatureIDE-k nola balioztatzen dituen konfigurazioak Sat4j [15] erabiliz.

### 2.1.2 Lanaren Deskonposaketa Egitura (LDE)

Proiektuak lau lan fase nagusi izango ditu: kudeaketa, prestakuntza, garapena eta dokumentazioa. Jarraian, 2. irudian eta 2.1.3 azpiatalean deskribapen zehatzago emango da.




2. irudia: Proiektuaren LDE.

### 2.1.3 Atazak

Atal honetan, proiektua aurrera eramateko burutuko diren atazak zehaztu eta deskribatuko dira. Proiektuaren atazak lau talde nagusietan banatuko dira: kudeaketa, prestakuntza, garapena eta dokumentazioa.

- KUDEAKETA
  - **Plangintza egin**: Proiektuaren oinarriak ondo zehaztu: helburuak, denboraren kudeaketa, mugarri nagusienak…
  - **Bilerak egin**: Proiektu honetan, ikasleaz gain, gutxienez bi interesatu gehiago daude (emangarriaren 2.3 atalean aurkeztuak). Beraz, ezinbestekoa da interesatuekin bilerak egitea proiektua aurrera eramateko.
  - **Jarraipen eta kontrola**: Proiektuaren jarraipen eta kontrola. Plangintzan zehaztutako helburuak eta epeak betetzen direla bermatzeko ezinbesteko prozesua.
- PRESTAKUNTZA
  - **FeatureIDE aztertu**: *FeatureIDE* plugin-a aztertu eta ikasi ezaugarrien (*feature*) balioztatzearen inguruko inplementazioa. Gainera, *FeatureIDE*-k erabiltzen duen *xml* sintaxia eta Sat4j ikasiko dira.
  - **JavaScript menderatu**: *Javascript* lengoaia hobeto menderatu, nagusiki .xml fitxategiekin lan egiteko.
  - **GitHub ezagutu**: *GitHub*-en funtzionamendua ulertu hedapenarekin jarraitu ahal izateko.
  - **Gity ezagutu**: *Gity* ezagutu inplementazioak ondo egiteko.
  - **GitLine-rekin trebatu** : Osatu beharreko softwarea (*GitLine*) ondo menderatu inplementazioak ondo egiteko.
- GARAPENA
  - **Balioztaketa**: GitLine-k konfigurazioak balioztatzea lortu.
  - **1.Inplementazioa**: "*aukuerakoEzaugarriaGehitu*" funtzioa inplementatu.
  - **2.Inplementazioa**: "*derrigorrezkoEzaugarriaGehitu*" funtzioa inplementatu.
- DOKUMENTAZIOA
  - **Memoria idatzi**: Memoria proiektu osoan zehar idatziko da.
  - **Aurkezpena egin**: Produktua eta memoria bukatzerakoan, defentsa egin ahal izateko, 30 minutuko aurkezpena prestatuko da.

## 2.2 Denboraren Plangintza

Behin atazak identifikatu direla, horietako bakoitzak izango duen iraupenaren kalkulu bat egingo da. Ondoren, ataza horien antolaketa egingo da, kronograma batean kokatuz eta bukatzeko proiektuan zehar egingo diren emangarriak zehaztuko dira.

### 2.2.1 Dedikazio taula

1. taulan ataza bakoitzari esleitutako ordu kopurua ikus daiteke. Proiektua 335 orduetan bukatzea balioetsita dago, falta diren 25 orduak erreserba moduan utziko dira arazoei aurre egin ahal izateko.

Prestakuntza atalean, denbora gehien hartuko duten atazak *FeatureIDE* aztertu, *Gity* ezagutu eta *GitLine-rekin* trebatu dira, teknologia ezezagunenak direlako. Besteen kasuan, ezagutza altuagoa da eta denbora gutxiago aurreikusten da.

Garapen atalean, ordu kopuru altuena *balioztaketa* funtzionalitatean aurreikusten da, konplexuena delako eta GitLine-rekin izango den trebetasun maila besteak inplementatzerako garaian baino baxuagoa izango delako.

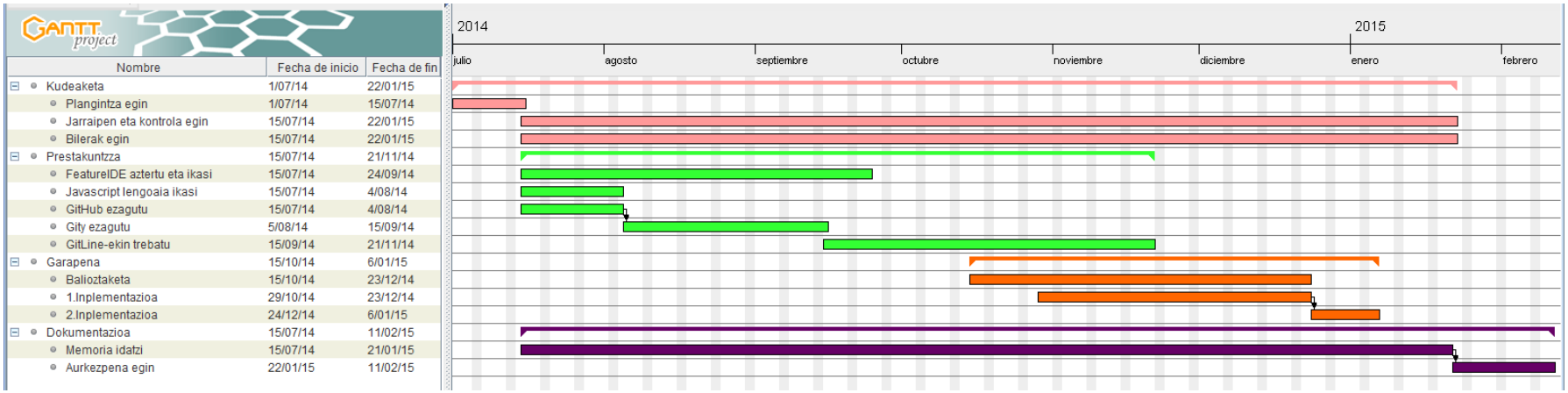
| ATAZAK | ORDUAK | |
|---|---|---|
| Kudeaketa | | 25 |
| -Plangintza egin | 5 | |
| -Bilerak egin | 10 | |
| -Jarraipen eta kontrola egin | 10 | |
| Prestakuntza | | 150 |
| -FeatureIDE aztertu | 50 | |
| -JavaScript ikasi | 20 | |
| -GitHub ezagutu | 10 | |
| -Gity ezagutu | 30 | |
| -GitLine-rekin trebatu | 40 | |
| Garapena | | 100 |
| -Balioztaketa | 50 | |
| -1.Inplementazioa | 30 | |
| -2.Inplementazioa | 20 | |
| Dokumentazioa | | 60 |
| -Memoria idatzi | 50 | |
| -Aurkezpena prestatu | 10 | |
| | GUZTIRA: 335 | |

1. taula: Ataza bakoitzari kalkulatutako iraupena orduetan.

### 2.2.2 Kronograma

3. irudian proiektuan zehar jarraituko den kronograma ikus daiteke. Egingo den lehenengo gauza plangintza izango da, ondoren, prestakuntza ataleko atazekin jarraituko da. Aztertu beharreko teknologia gehienak aztertu ondoren, garapenari hasiera emango zaio. Memoria eta jarraipen eta kontrola proiektu osoan zehar burutuko dira. Azkenik, proiektuaren aurkezpena egiterakoan proiektua amaitutzat emango da.

Uztaila eta iraila bitartean 22 ordu kalkulatzen dira, aldiz, irailaren hirugarren astetik otsailaren lehenengo astera 15 ordu kalkulatzen dira astero lan egiteko. Ostiralero astean zehar egindako lanaren jarraipen eta kontrola egingo da, finkatutako ordu kopurua edo helbururen bat ez bada bete asteburua erabiliko da helburuak betetzeko.




3. irudia: Proiektuaren kronograma.

### 2.2.3 Emangarriak

2.2.2 atalean esan den moduan, memoria proiektu osoan zehar egingo da. Horretarako, emangarri batzuk finkatuko dira proiektuan zehar osatzeko, 2. taulan ikus daitekeenak.

| EMANGARRIAREN IZENA | ENTREGATZE DATA | DESKRIBAPENA |
|---|---|---|
| Plangintza | 14/09/10 | Proiektu osoaren plangintza zehaztuko du (LDE, atazak, arriskuak, kronograma … ). |
| Erremintak eta baliabideak | 14/10/24 | *GitHub, Gity* eta *FeatureIDE* baliabideen deskribapena (zer dira, zertarako erabiltzen dira … ). |
| *Balioztaketa* funtzionalitatea | 14/12/30 | Balioztaketa funtzionalitatearen inplementazioa zehaztuko du ( erabilitako funtzioak, funtzioen deskribapena, izandako arazoak bere konponbideekin … ). |
| *aukerakoEzaugarriaGehitu* funtzionalitatea | 15/01/03 | *aukerakoEzaugarriaGehitu* funtzionalitatearen inplementazioa zehaztuko du ( erabilitako funtzioak, funtzioen deskribapena, izandako arazoak bere konponbideekin … ). |
| *derrigorrezkoEzaugarriaGehitu* funtzionalitatea | 15/01/12 | *derrigorrezkoEzaugarriaGehit* funtzionalitatearen inplementazioa zehaztuko du ( erabilitako funtzioak, funtzioen deskribapena, izandako arazoak bere konponbideekin … ). |

2. taula: Emangarrien informazio taula.

## 2.3 Interesatuak

Proiektu honetan, 3. taulan ikusten den moduan, hainbat pertsonak hartuko dute parte. Proiektua aurrera eramango duena Eider Irigoyen ikaslea izango da. Arantza Irastorza irakaslea arituko da zuzendari lanetan, honen lana ikaslea aholkatzea izango da. Proiektu hau aurrera eraman ahal izateko ezinbestekoa izan da aurretik Leticia Montalvillo ikertzaileak egindako lana, eta bera izango da proiektu honen emaitza bezala inplementatukoaren bezeroa ere.

Interesatuekin komunikatzeko, posta elektronikoa erabiliko da. Gainera, beharren arabera, aurrez aurreko bilerak egingo da.

| ROLA | IZENA | POSTA HELBIDEA |
|---|---|---|
| Ikaslea | Eider Irigoyen | eirigoyen007@ikasle.ehu.es |
| Zuzendaria | Arantza Irastorza | arantza.irastorza@ehu.es |
| Tesiaren garatzailea | Leticia Montalvillo | leticia.montalvillo@ehu.es |

3. taula: Interesatuen informazio taula.

## 2.4 Kalitatearen Kudeaketa

Proiektu guztian zehar egiten diren ataza guztietan, ahalik eta kalitate mailarik handiena bilatuko da, bai proiektuaren kudeaketan bai produktuan.

### 2.4.1 Proiektuaren kalitatea

- Bilerak
  - Altua: interesatuekin egoeraren berri emateko maiztasunez bildu, bilera aurretik gai zerrenda interesatuei bidali, bilera akta egin.
  - Ertaina: interesatuekin egoeraren berri emateko maiztasunez bildu,bilera akta egin.
  - Baxua: interesatuekin oso gutxitan bildu, bilera aktarik gabe.

### 2.4.2 Produktuaren kalitatea

Produktuaren kalitatea lortzeko hiru atal izango dira kontuan: inplementazio lana, dokumentazioa eta aurkezpena.

- ✓ GitLine
  - Garatu beharreko hiru funtzionalitateak
    - Altua: ondo eta txukun inplementatuak daude
    - Ertaina: ondo inplementatuak daude, baina kodea ez dago txukun.
    - Baxua: badago funtzionalitateren bat ez dituela helburuak betetzen.
- ✓ Memoria
  - Edukia (izan beharreko kapituluak)
    - Altua: proiektuaren helburuen dokumentua, denboraren kudeaketa, eranskinak, glosategia, probak, ebazpenaren diseinua, ondorioak.
    - Ertaina: proiektuaren helburuen dokumentua, denboraren kudeaketa, eranskinak, probak, ebazpenaren diseinua, ondorioak.
    - Baxua: proiektuaren helburuen dokumentua, denboraren kudeaketa, ebazpenaren diseinua, ondorioak.

- Egitura
  - Altua: irudiek eta taulek oina daukate, algoritmo edo kode zatiak letra-mota desberdin batean daude, zuzentzaile ortografikoa erabili da, atal eta azpi-atalak erabili dira, goialdean proiektuaren eta egilearen izenak agertzen dira, estandarrak finkatutako iturri tamaina eta mota erabili da, kapituluak ondo bereizita daude.

  - Ertaina: algoritmo edo kode zatiak iturri desberdin batean daude, zuzentzaile ortografikoa erabili da, atal eta azpiatalak erabili dira, estandarrak finkatutako iturri tamaina eta mota erabili da, kapituluak ondo bereizita daude.
  - Baxua: algoritmo edo kode zatiak iturri desberdin batean daude, atal eta azpi atalak erabili dira, estandarrak finkatutako iturri tamaina eta mota erabili da.

✓ Aurkezpena
- Gardenkiak
  - Altua: lehenengo gardenkian izenburua eta UPV/EHU-ren logotipoa agertzen dira, aurkibidea, gardenkiak zenbakitu, irudiak ondo ikusten dira.
  - Ertaina: lehenengo gardenkian izenburua agertzen da, aurkibidea, irudiak ondo ikusten dira.
  - Baxua: lehenengo gardenkian izenburua agertzen da, irudiak ez dira oso ondo ikusten.

## 2.5 Arriskuen kontrola

Atal honetan, proiektuaren arrakasta arriskuan jar dezaketen elementuak identifikatuko dira. Arrisku bakoitza zertan datzan azaltzeaz gain, proiektuan izango lukeen eragina, gertatzeko probabilitatea eta ekiditeko estrategia edo behin gertatuta honen eragina gutxitzeko estrategia zehaztuko dira.

- Arriskua: proiektu honetan garatutako tresnak erabiltzen duen zerbitzuren batek (*GitHub, Gity* … ) erabiltzeari uztea.
  Deskribapena: Edozein arrazoia dela medio zerbitzua erori edo aldatu egiten bada, zerbitzu hori erabiltzen duten aplikazioen erabilgarritasuna murriztuta geratuko da.
  Eragina: altua

Gertatzeko probabilitatea: baxua. Orokorrean zerbitzuak eskaintzen dituzten konpainiak nahiko fidagarriak izaten dira, eta ez dute huts egiten.
Ekiditeko estrategia: konpainia fidagarrietako zerbitzuak soilik erabiliko dira.

- Arriskua: informazioa eta garatutako dokumentazioa galtzea.
  Deskribapena: galera hau, era desberdinetan gerta daiteke: eramangarriaren arazo teknikoak, USB-ren galera, erabiltzaileen hanka sartzeak …
  Eragina: oso altua. Zerbait galtzen bada, berriz ere egin beharko da eta denbora galera oso handia izan daiteke.
  Gertatzeko probabilitatea: txikia.
  Ekiditeko estrategia: arrisku hau ekiditeko informazioaren kopia periodikoak egingo dira eta leku desberdinetan biltegiratuko dira: eramangarrian, USB batean, Interneteko hodeian (drive).

- Arriskua: plangintzaren denbora-kalkuluak okerrak izatea
  Azalpena: planifikazioan egindako denbora kalkuluak okerrak izatea eta balioetsitakoaren eta benetan burututakoaren arteko aldea zabala izatea.
  Eragina:ertaina. Benetan dedikatutako ordu kopurua kalkulatutakoa baino askoz altuagoa bada, proiektuan atzerapenak sor daitezke eta kalitate murriztua ikus daiteke. Bestalde, benetan dedikatutako ordu kopurua kalkulatutakoa baino txikiagoa bada, kalitatea hobetzeko erabil daiteke.
  Gertatzeko probabilitatea: altua. Proiektu honen iraupena luzea denez, zaila da denbora kalkuluak egokiak egitea, eta ondorioz, huts egiteko aukera gehiago dago.
  Eragina gutxitzeko estrategia: jarraipen eta kontrolean desbideraketa handiak antzematen badira, birplanifikazio bat egin.

# 3. KAPITULUA

# Birplangintza

Proiektuaren birplangintzan, proiektuan zehar izandako desbiderapenak, proiektuaren LDE berria eta kronograma berria zehaztuko dira.

## 3.1 Proiektuaren birplangintza

Dedikazio faltarengatik eta proiektu kanpoko arrazoiengatik proiektuaren hasieran egindako planifikazioak aldaketa nabarmenak jaso dituenez, proiektuaren zuzendariarekin batera birplangintza bat egitea adostu da. Proiektua otsailean aurkeztea aurreikusita zegoenez eta helburua bete ez denez, birplangintza 2015eko otsailaren 2an egin da.

Ondoren, zehatzago azalduko dira 4. taulan agertzen diren desbiderapenen arrazoiak eta egin beharreko aldaketak plangintza zuzen bat lortzeko. Birplangintza egin beharreko momentuan, 210 lanordu burututa zeuden 335 lanordu ordez.

Argiago ikusteko, proiektuaren lan-paketeak banaka aztertuko dira gertatutako atzerapenak eta aldaketak (gertatu badira) azaltzeko.

Kudeaketa lan-paketean, ataza bat burututa dago (plangintza egin) eta beste biek oraindik irekiak jarraitzen dute (bilerak egin eta jarraipen eta kontrola egin). Plangintzarako, aurreikusitakoa baino 3 ordu gehiago behar izan dira. Beste bi atazetan ia estimatutako orduak inbertitu direnez eta oraindik lana gelditzen denez, aurreikusitako denbora 10 ordutan areagotu da bileren kasuan eta 8 jarraipen eta kontrolean.

Erabat bukatuta dagoen lan-pakete bakarra prestakuntza da. Pakete honetan ataza bat kendu (*Gity ezagutu*) eta beste bat gehitu (*Fama ikasi*) behar izan dira. Lehenengo kasuan, bakarrik OS X sistema eragilearentzako bezeroa denez eta ikasleak Windows erabiltzen duenez alde batera utzi da. Bigarren kasuan, *balioztaketa* funtzionalitatea egiteko FeatureIDE alde batera utzi eta

Fama erabili denez, funtzionalitate hau inplementatzeko erreminta berria ezagutzeko ataza gehitu da.

Garapenaren kasuan, inplementatu beharreko hiru funtzionalitateetatik bakarra inplementatu da, *balioztaketa*. Aurreikusitako denborarekin alderatuta desbiderapena nahiko handia izan da, 24 ordu gehiago behar izan dira ataza burutzeko. Desbiderapen hau bidean izandako arazoengatik (6.2 atalean azaltzen direnak) gertatu da. Beste bi funtzionalitateak inplementatzeko, ikasleak duen esperientziagatik, aurreikusitako denbora 5 orduetan gutxitu da bi kasuetan.

Azkenik, dokumentazio lan-paketean ataza guztiak irekiak daude. Prestakuntza atalean aurreikusitako denbora baino gutxiago erabili denez, ordu horietako batzuk memoria egiteko atazari gehitu zaizkio memoriaren kalitatea hobetu ahal izateko.

| Ataza | Plangintzan aurreikusitakoa | Egindakoa | Desbiderapena (Aurre-Egind) | | Birplangintzan aurreikusitakoa | Desbiderapena (Aurre-birplan) | | Burututa? |
|---|---|---|---|---|---|---|---|---|
| Kudeaketa | 25:00 | 24:35 | - | 0:25 | 46:30 | + | 21:30 | EZ |
| -Plangintza egin | 5:00 | 8:30 | + | 3:30 | 8:30 | + | 3:30 | BAI |
| -Bilerak egin | 10:00 | 7:20 | - | 2:40 | 20:00 | + | 10:00 | EZ |
| -Jarraipen eta kontrola egin | 10:00 | 8:45 | - | 1:15 | 18:00 | + | 8:00 | EZ |
| Prestakuntza | 150:00 | 104:00 | - | 46:00 | 104:00 | - | 46:00 | BAI |
| -FeatureIDE aztertu | 50:00 | 43:00 | - | 7:00 | 43:00 | - | 7:00 | BAI |
| -JavaScript ikasi | 20:00 | 6:00 | - | 14:00 | 6:00 | - | 14:00 | BAI |
| -GitHub ezagutu | 10:00 | 7:30 | - | 2:30 | 7:30 | - | 2:30 | BAI |
| ~~-Gity ezagutu~~ | 30:00 | 0:00 | - | 30:00 | 0:00 | - | 30:00 | ——— |
| -GitLine-rekin trebatu | 40:00 | 23:30 | - | 16:30 | 23:30 | - | 16:30 | BAI |
| -FaMa ikasi | 0:00 | 24:00 | + | 24:00 | 24:00 | + | 24:00 | BAI |
| Garapena | 100:00 | 74:30 | - | 25:30 | 104:30 | + | 4:30 | EZ |
| -Balioztaketa | 50:00 | 74:30 | + | 24:30 | 74:30 | + | 24:30 | BAI |
| -1.Inplementazioa | 30:00 | 0:00 | - | 30:00 | 15:00 | - | 5:00 | EZ |
| -2.Inplementazioa | 20:00 | 0:00 | - | 20:00 | 15:00 | - | 15:00 | EZ |
| Dokumentazioa | 60:00 | 7:45 | - | 52:15 | 90:00 | + | 30:00 | EZ |
| -Memoria idatzi | 50:00 | 7:45 | - | 42:15 | 80:00 | + | 30:00 | EZ |
| -Aurkezpena prestatu | 10:00 | 0:00 | - | 10:00 | 10:00 | - | 0:00 | EZ |
| GUZTIRA: | 335:00 | 210:50 | - | 124:10 | 345:00 | + | 10:00 | EZ |

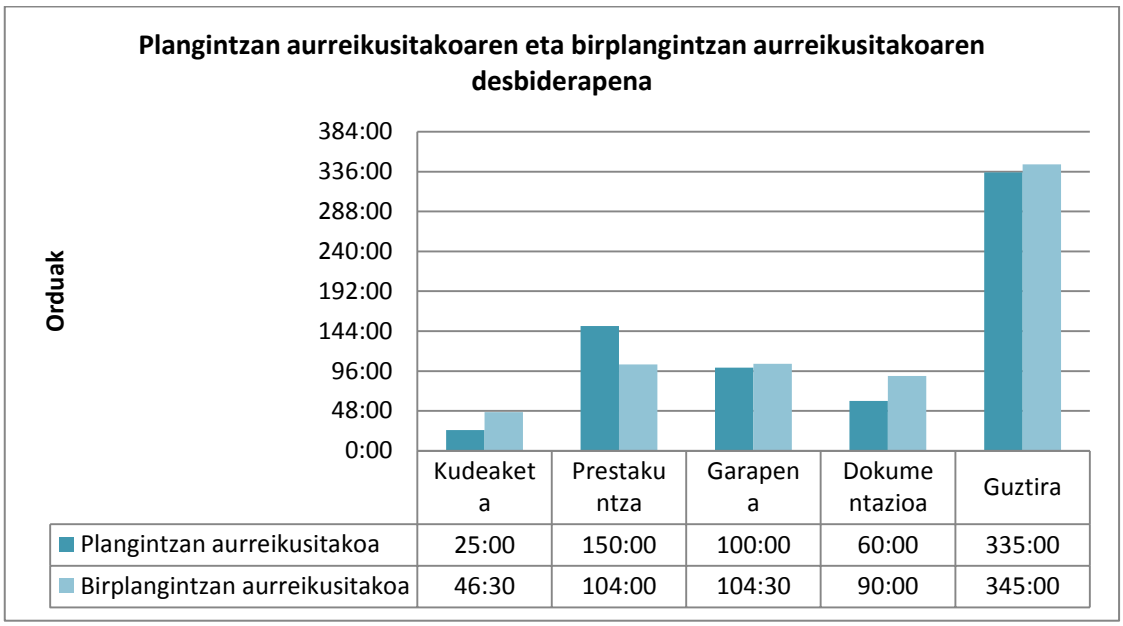
4. taula: Ordu taula.

Plangintzan aurreikusitako denbora eta birplangintzan aurreikusitako denborak alderatzen badira (1. grafikoa) proiektuari dedikatu beharreko lanordu totalean alde asko ez dagoela ikus daiteke. Lan-pakete guztietan, prestakuntzan izan ezik, plangintzan aurreikusitako lanorduak birplangintzan aurreikusitakoak baino gutxiago dira.

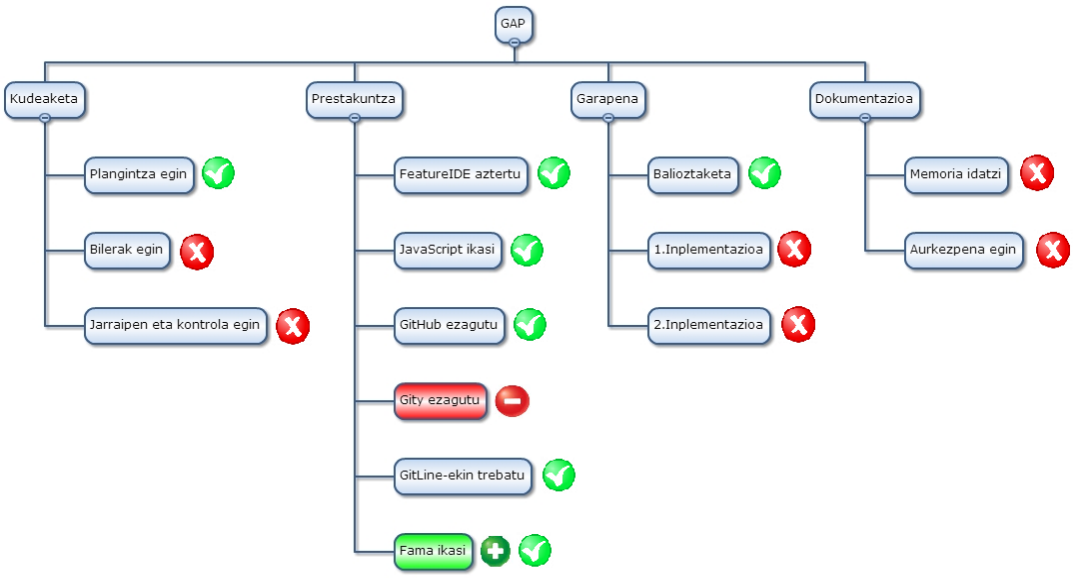
### 3.1.1 LDE berria

4. irudian ikus daitekeen moduan, aipatu diren aldaketak direla eta, proiektuaren LDE-a aldatu da. Gity ezagutu ezabatuta ( '-' bidez adierazita) gelditzen da eta *Fama ikasi* gehitzen da ('+'

bidez adierazita). Gainera ataza bakoitzaren egoera adierazten da, bukatutakoak 'tick' baten bidez eta bukatu gabeak 'x' baten bidez.




1. grafikoa: Desbiderapenen grafikoa



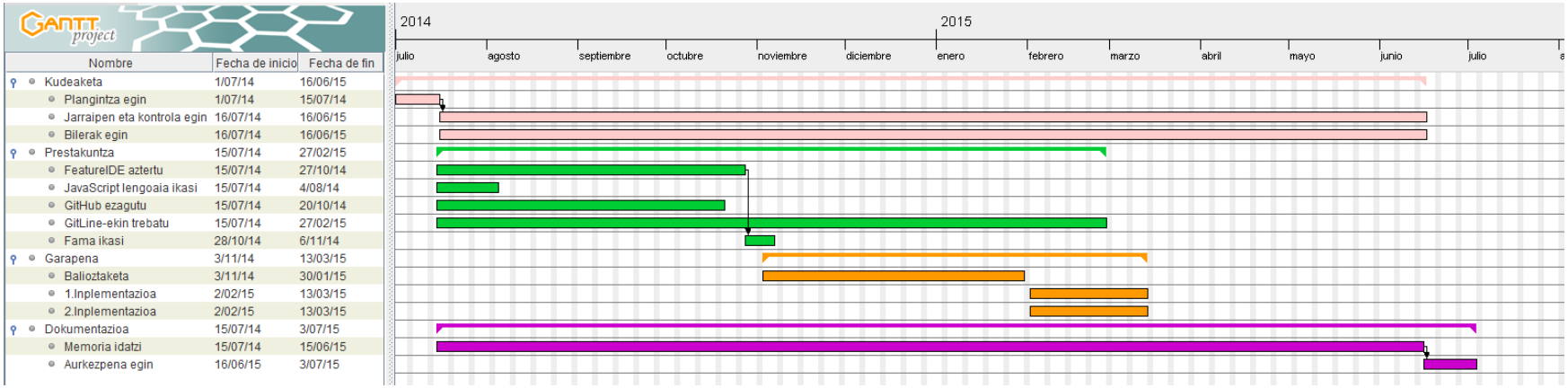

4. irudia: Proiektuaren LDE berria.

### 3.1.2 Kronograma

5. irudian ikus daiteke proiektuaren kronograma berria. Aldaketa nabarmenena proiektua amaitzeko data da. Dedikazio faltarengatik eta izandako arazoengatik, ikasleak ezin izan ditu astero finkatutako 15 orduak lan egin, beraz, proiektuaren defentsa otsailean egin beharrean uztailean egingo da.

Birplangintzan proiekturako 345 ordu kalkulatzen dira, 211 ordu jada erabili direnez 134 ordu gelditzen dira. Ordu horiek 22 asteetan zehar erabiliko direnez, astero batez beste 6 lanordu  kalkulatzen dira lan egiteko. Ikasleak lanean hasiko da eta astean zehar egin beharreko lan zama gutxitu du epeak betetzeko arazorik ez izateko.

Beste aldaketa nabarmen bat garapenean gertatu da. Plangintzan *balioztaketa* eta *1.inplementazioa* erdi-paraleloan egitea kalkulatu zen, baina azkenean *balioztaketa* bukatu arte ez da derrigorrezkoEzaugarriaGehitu (1.Inplementazioa) hasi, gainera derrigorrezkoEzaugarriaGehitu eta aukerakoEzaugarriaGehitu (2.inplementazioa) erabat paraleloan egingo direla kalkulatzen da.




| Nombre | Fecha de inicio | Fecha de fin |
|---|---|---|
| Kudeaketa | 1/07/14 | 16/06/15 |
| Plangintza egin | 1/07/14 | 15/07/14 |
| Jarraipen eta kontrola egin | 16/07/14 | 16/06/15 |
| Bilerak egin | 16/07/14 | 16/06/15 |
| Prestakuntza | 15/07/14 | 27/02/15 |
| FeatureIDE aztertu | 15/07/14 | 27/10/14 |
| JavaScript lengoaia ikasi | 15/07/14 | 4/08/14 |
| GitHub ezagutu | 15/07/14 | 20/10/14 |
| GitLine-ekin trebatu | 15/07/14 | 27/02/15 |
| Fama ikasi | 28/10/14 | 6/11/14 |
| Garapena | 3/11/14 | 13/03/15 |
| Balioztaketa | 3/11/14 | 30/01/15 |
| 1.Inplementazioa | 2/02/15 | 13/03/15 |
| 2.Inplementazioa | 2/02/15 | 13/03/15 |
| Dokumentazioa | 15/07/14 | 3/07/15 |
| Memoria idatzi | 15/07/14 | 15/06/15 |
| Aurkezpena egin | 16/06/15 | 3/07/15 |

5. irudia: Proiektuaren kronograma berria.

# 4. KAPITULUA

# Erreminta eta baliabideak

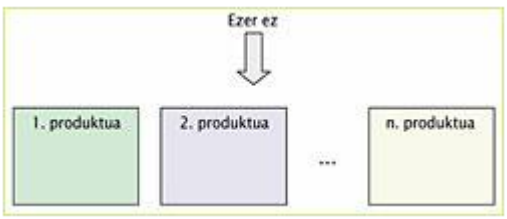
Kapitulu honetan, Software produktu lerroak, FeatureIDe, GitHub eta Fama erremintak azalduko dira.

## 4.1 Software Produktu Lerroak

Softwarea berrerabiltzea betidanik izan da Software Ingeniaritzaren helburuetariko bat. 1960ko hamarkadaren bukaeran azpi-errutinak erabiltzen hasi ziren; 70eko hamarkadan, berriz, moduluak, eta 80koan objektuei zuzendutako programazioa [**].

Gaur egun, ezaugarri komunak dituzten software-produktuak[1] garatzeko, berrerabiltze-maila ezberdinetako hainbat aukera dituzte erakundeek. Aukera horietako bakoitzak inbertsio handiagoa eskatzen du, baina berrerabiltze-maila ere handiagoa da.

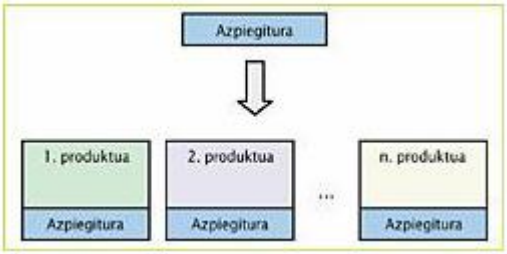
- **Produktuz produktu:** betiko ikuspegia da. 6. irudian ikusi daitekeen moduan, produktu bakoitza zerotik hasita garatzean datza.




6. irudia: Produktuz produktuko garapena.

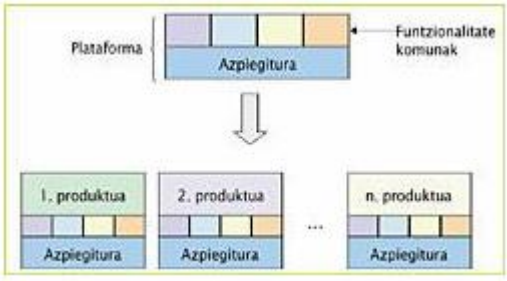
[**]Atal honetan azaldutakoa http://zientzia.net/site_media/pdf/A201_O38-41_1.pdf izena duen estekatik hartu da.

- **Azpiegitura estandarizatua:** produktuen azpiegitura estandarizatzean datza, 7. irudian ikusten den moduan.



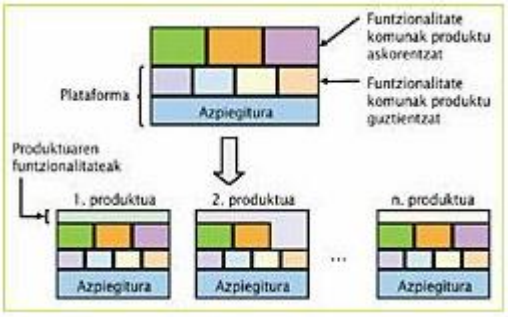

7. irudia: Azpiegitura estandarizatua.

- **Plataforma:** 8. irudian azalduta, produktuak plataforma bat oinarritzat hartuz garatzen dira; plataforma horrek azpiegitura estandarizatua izan ohi du, eta produktu guztien funtzio komunak ere biltzen ditu.



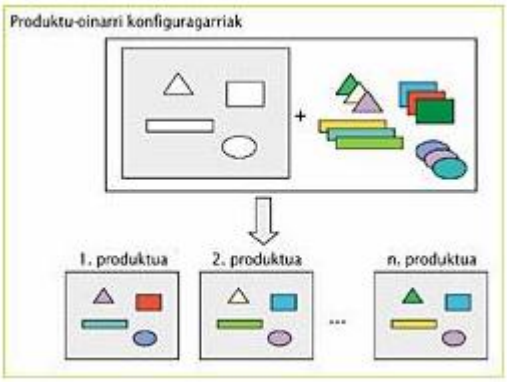

8. irudia: Plataforma.

- **Software-produktuen lerroa:** Plataformaz gainera, produktu gehientsuenentzat komunak diren funtzioak hartzen dira produktu oinarritzat (9. irudian azalduta); produktu baten edo gutxi batzuen funtzio zehatzak produktua oinarri horretatik eratortzen denean garatzen dira.




9. irudia: Software-produktuen lerroa.

- **Produktu-oinarri konfiguragarria:** 10. irudian ikusi daitekeen moduan, produktuak sortzeko oinarri bat garatzea da. Oinarri hori konfiguratu egin daiteke produktua sortzeko. Domeinu egonkorretan erabiltzen da batez ere.




10. irudia: Produktu-oinarri konfiguragarria.

Proiektu honetan, Software Produktu-Lerroekin (SPL) lan egin da. Produktu-lerroei buruzko lehendabiziko zantzuak 90eko hamarkadaren bukaeran agertu ziren, eta interesa izugarri handitu da 2000. urtetik aurrera. Aurreko filosofiek baino ikuspegi zabalagoa jorratzen dute, berrerabiltze planifikatu eta estrategikoa lantzen dutelako: ezaugarri teknikoak eta negozio-irizpideak kontuan hartzen dituzte, eta software-sektorearen industrializazioa dute helburu.

Software Produktu Lerroak ezaugarri komun batzuk dituzten software-sistema intentsiboak dira, eta merkatuaren alor jakin baten beharrak asetzen dituzte. Sistema horiek oinarri aktibo komunetatik abiatuta garatzen dira, aurrez erabakitako prozesuari jarraituz.

Aktibo bat erakundeak erabiltzen eta berrerabiltzen duen lan-produktu bat da. Berrerabiltzeko diseinatzen da, domeinuaren parte komuna hartzen du bere baitan eta aldakortasuna ahalbidetzen du, hau da, egoera desberdinetara egokitzen da. Beraz, oinarri aktiboek produktu-lerroaren ardatza osatzen dute. Arkitektura, software-osagarri berrerabilgarriak, domeinu-ereduak, dokumentuak eta test-kasuak oinarri aktiboen adibideak dira.

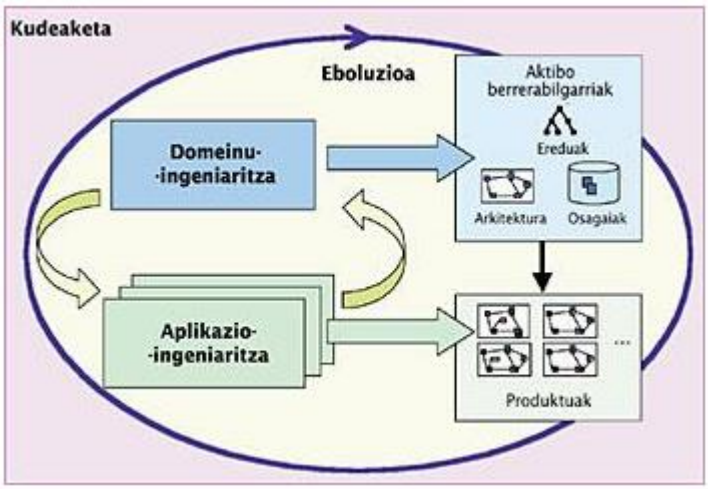
Produktu-lerro baten arrakastaren gakoa planifikatutako aldakuntzen kudeaketa era sistematikoan egitean datza. Beraz, funtsezkoa da produktu-lerroaren aldakortasuna (familiako kideak elkarrengandik bereizten dituen ezaugarria) identifikatzea, kontrolatzea eta kudeatzea.

Produktu-lerro batek domeinu (hots, terminologia komuna duen jakintza-arlo) batera zuzenduta egon behar du. Eta domeinu guztiak ez dira egokiak produktu-lerroak aplikatzeko, beharrezkoa da antzeko aplikazioen eskaera egotea. Domeinu horretako produktuen garapenean ezinbestekoa da esperientzia edukitzea.

Produktu-lerroek aldaketa dakarte produktuen bizitza-zikloan: proiektu ikuspegitik domeinu ikuspegira goaz. 11. irudian ikus daitekeen moduan, domeinu ikuspegian bi jarduera bereiz daitezke: *domeinu-ingeniaritza* eta *aplikazio-ingeniaritza* edo produktuen garapena.

Domeinu-ingeniaritzan, produktuen analisi sakona egin ondoren, aktibo berrerabilgarriak garatzen dira. Helburua produktuen parte komuna eta aldakorra identifikatu eta ahal den berrerabiltze-maila altuena lortzeko beharrezko azpiegitura edo oinarria sortzea da. Aktibo horiek enpresek produktuei buruz duten esperientzia eta ezagutza guztia biltzen dute. Aktibo nagusia produktuen arkitektura orokorra da, eta produktu-lerroak hori du oinarri. Produktu-lerro baten arkitekturak produktu guztiak hartzen ditu aintzat, modu orokor batean.

Aplikazio-ingeniaritzaren helburua, aldiz, aplikazioak sortzea da, baina ez zerotik hasita, baizik eta domeinu-ingeniaritzan sortutako oinarri aktiboak erabiliz.




11. irudia: Produktu lerroen jarduera.

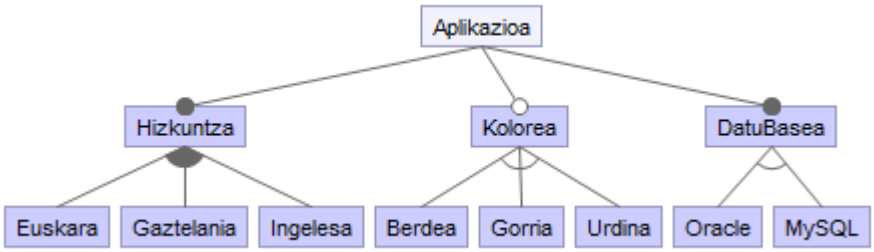
Bestalde, ezinbestekoa da beste jarduera bat kontuan hartzea, kudeaketa hain zuzen ere. Produktu-lerroa kudeatzeko, bai maila teknikoan bai enpresaren antolaketan, produktu-lerroaren planteamendua bide egokitik eta arrakastaz doala bermatu beharra dago. Horrekin batera, produktu-lerroa egoera berrietara ongi egokitzen dela bermatu beharra dago.

### 4.1.1 Zer dira ezaugarriak

Aplikazio oso bat, nolakotasun ezberdinak adierazten dituzten zatietan banatzen da (ingelesez *features* eta euskaraz *ezaugarriak*). Ideia hau, klaseen kontzeptuaren antzekoa dela badirudi ere, desberdintasun garrantzitsu bat dago. Testuinguru honetan, ezaugarri bat azken

erabiltzailearen aplikazioaren alderdi bat adierazten du. Ezaugarri bakoitzak, software tramankulu (klaseak, metodoak, grafikoak … ) ezberdinekin harremana izan dezake.

12. irudian ikus daitekeen adibidean, aplikazioak ezaugarri desberdinak ditu: hizkuntza (Euskara, Gaztelania eta Ingelesa), interfazearen kolorea (Berdea, Gorria eta Urdina) eta datu basea (Oracle eta MySQL). Beraz, azken erabiltzaileak ezaugarri horien artean hautatu dezake bere aplikazioa sortzeko. 4.1.3 atalean azalduko den moduan ezaugarriak ezin dira edonola hautatu.




12. irudia: Ezaugarri ezberdinak.

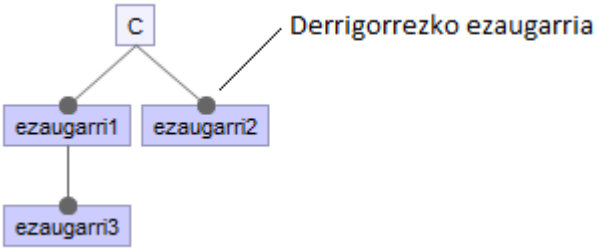
### 4.1.2 Zer da ezaugarri eredua

Ezaugarri ereduak kontzeptu instantzien ezaugarri komun eta ezaugarri aldakorrak irudikatzen ditu. Bi osagaiez osatuta dago: ezaugarri diagrama eta hainbat informazio osagarri (murriztapenak, mendekotasunak, ezaugarrien inguruko azalpen edo argudio... ).

Ezaugarri diagrama ziklorik gabeko zuhaitz bat da, bere erroan kontzeptu bat duena. Zuhaitzaren nodoak ezaugarriak dira (etiketadun laukien bitartez adierazita) eta arkuek ezaugarrien arteko erlazioak adierazten dituzte. Ezaugarrien konbinazio desberdinek konfigurazioak osatuko dituzte (ikus 4.1.3 atala). Mota ezberdineko ezaugarriak daude:

- Derrigorrezko ezaugarria (13. irudia): Kontzeptuaren instantzian (hots, azken erabiltzaileak sortutako aplikazioan) agertuko da, baldin eta bere gurasoa ere instantziaren deskribapenean agertzen bada. 5. taulan ikus daiteke baliozkoa den konfigurazio bakarra 13. irudian agertzen den ezaugarri ereduan oinarrituz. Konfigurazio posible bakarra dago ezaugarri guztiak derrigorrezkoak direlako.

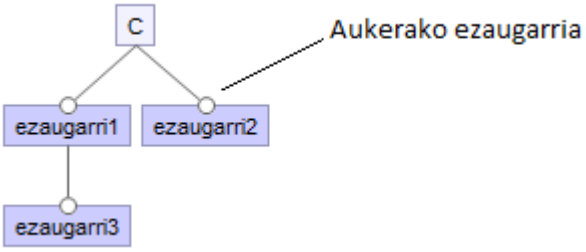
- Aukerako ezaugarria (14. irudia): Kontzeptuaren instantzian (azken erabiltzaileak sortutako aplikazioan) ager daiteke, baldin eta bere gurasoa ere instantziaren deskribapenean agertzen bada. 6. taulan ikus daitezke baliozkoak diren konfigurazioak 14. irudian agertzen den ezaugarri ereduan oinarrituz.




13. irudia: Derrigorrezko ezaugarria.

| 8.Irudia-ren ezaugarri diagramatik sortu daitezkeen instantziak: |
|---|
| C, ezaugarri1, ezaugarri2 eta ezaugarri3 |

5. taula: Baliozkoak diren instantziak.

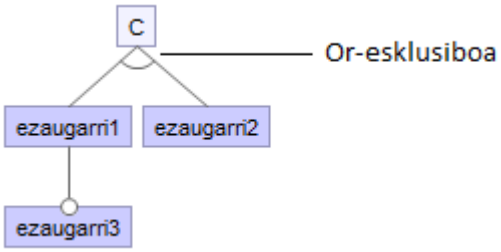



14. irudia: Aukerako ezaugarriak.

| 9.Irudia-ren ezaugarri diagramatik sortu daitezkeen instantziak: |
|---|
| C |
| C, ezaugarri1 |
| C, ezaugarri2 |
| C, ezaugarri1, ezaugarri3 |
| C, ezaugarri1, ezaugarri2 |
| C, ezaugarri1, ezaugarri2 eta ezaugarri3 |

6. taula: Baliozkoak diren instantziak.

- Ezaugarri or-esklusiboa (15. irudia): Ezaugarri or-esklusiboen multzoaren gurasoa kontzeptuaren instantzian (azken erabiltzaileak sortutako aplikazioan) agertzen bada, multzo horretako ezaugarri or-esklusibo bakarra agertuko da instantziaren deskribapenean (zehazki bat). 7. taulan ikus daitezke baliozkoak diren konfigurazioak 15. irudian agertzen den ezaugarri ereduan oinarrituz.

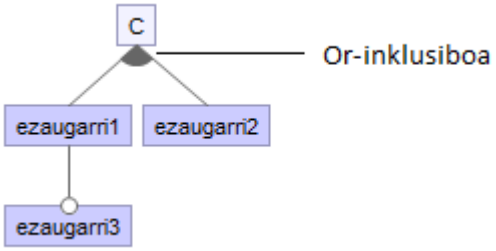



15. irudia: Or-esklusiboa.

| 10.Irudia-ren ezaugarri diagramatik sortu daitezkeen instantziak: |
|---|
| C, ezaugarri1 |
| C, ezaugarri2 |
| C, ezaugarri1, ezaugarri3 |

7. taula: Baliozkoak diren instantziak.

- Ezaugarri or-inklusiboa (16. irudia): Ezaugarri or-inklusiboen multzoaren gurasoa kontzeptuaren instantzian (azken erabiltzaileak sortutako aplikazioan) agertzen bada, multzo horren edozein azpimultzo ez-huts ere agertuko da instantziaren deskribapenean (gutxienez bat). 8. taulan ikus daitezke baliozkoak diren konfigurazioak 16. irudian agertzen den ezaugarri ereduan oinarrituz.




16. irudia: Or-inklusiboa.

| 11.Irudia-ren ezaugarri diagramatik sortu daitezkeen instantziak: |
|---|
| C, ezaugarri1 |
| C, ezaugarri2 |
| C, ezaugarri1, ezaugarri2 |
| C, ezaugarri1, ezaugarri3 |
| C, ezaugarri1, ezaugarri2 eta ezaugarri3 |

8. taula: Baliozkoak diren instantziak.

Informazio osagarri moduan, ezaugarri ereduetan, murriztapenak zehaztu daitezke. Murriztapenak osatzeko *not*, *and*, *or* eta *implies* erabili daitezke. 17. irudian murriztapen batzuk ikus daitezke:

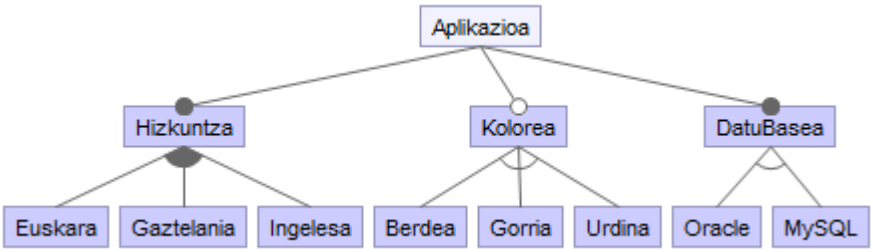
- Euskara ezaugarria aukeratzen bada, Berdea ezaugarria aukeratu behar da.
- Ingelesa ezaugarria aukeratzen bada, ezin da Berdea ezaugarria aukeratu.
- Ingelesa eta Gaztelania edo Ingelesa eta Euskara aukeratzen badira, MySQL aukeratu behar da.

$$\text{Euskara} \Rightarrow \text{Berdea}$$
$$\text{Ingelesa} \Rightarrow \neg\,\text{Berdea}$$
$$\text{Ingelesa} \wedge \text{Gaztelania} \vee \text{Ingelesa} \wedge \text{Euskara} \Rightarrow \text{MySQL}$$

17. irudia: Ezaugarri ereduaren murriztapenak.

### 4.1.3 Zer dira konfigurazioak

Konfigurazio batek instantzia zehatz batek izango dituen ezaugarriak definitzen ditu. Konfigurazio bakoitzaren ezaugarrien konbinaketak ezaugarri diagramak adierazitako baldintza eta murriztapenak bete behar ditu. 9. taulan ikus daitezke 18. irudian agertzen den ezaugarri ereduan oinarrituz, baliagarriak diren konfigurazioak, aldiz, 10. taulan baliagarriak ez diren konfigurazioak.




18. irudia: Ezaugarri diagrama.

| Baliagarriak diren konfigurazio batzuk: |
|---|
| Aplikazioa, Hizkuntza, Euskara, DatuBasea,Oracle |
| Aplikazioa, Hizkuntza, Euskara, Gaztelania, DatuBasea, MySQL |
| Aplikazioa, Hizkuntza, Ingelesa, Kolorea, Berdea, DatuBasea, Oracle |
| Aplikazioa, Hizkuntza, Euskera, Kolorea, Urdina, DatuBasea, Oracle |
| Aplikazioa, Hizkuntza, Ingelesa,Gaztelania, Kolorea, Berdea, DatuBasea, Oracle |

9. taula: Baliozkoak diren konfigurazio batzuk.

| Baliagarriak ez diren konfigurazio batzuk: |
|---|
| Aplikazioa |
| Aplikazioa, Kolorea, DatuBasea |
| Aplikazioa, Hizkuntza, Kolorea |
| Aplikazioa, Hizkuntza, Kolorea, DatuBasea |
| Aplikazioa, DatuBasea |

10. taula: Baliozkoak ez diren konfigurazio batzuk.

## 4.2 FeatureIDE

FeatureIDE Ezaugarriei Bideratutako Software Garapenerako (FOSD) Eclipse-n oinarritutako kode irekiko lan ingurunea da (Eclipse-ko plugin bat), Alemaniako Magdeburg Unibertsitatean garatuta [3]. FOSD-ek inplementazio teknika ezberdin asko eusten ditu: ezaugarriei bideratutako programazioa (FOP), aspektuei bideratutako programazioa (AOP), delta bideratutako programazioa (DOP)...

Beraz, FeatureIDE tresnak FOSD-ek proposaturiko faseak eusten ditu:

- Domeinuaren analisia: Ezaugarri eredua definitu.
- Domeinuaren inplementazioa: Kode aktiboak ezaugarrietara mapeatu.
- Betekizunen analisia: Aplikazio berriarentzako konfigurazio berria definitu.
- Softwarearen gauzatzea: Software aplikazioa automatikoki sortzen da.

FeatureIDE etengabe garatzen ari da, orain arte ezaugarri hauek dauzka inplementatuak:

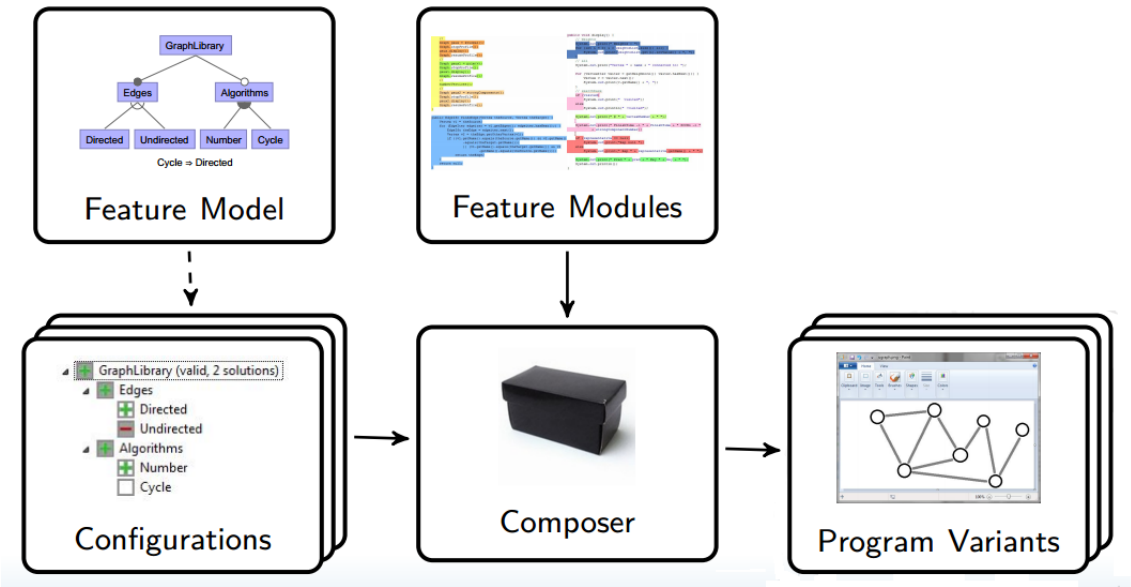
- Eclipse-rekin integrazio osoa.
- Ezaugarri ereduaren editorea, grafikoa edo testuan oinarritua.
- Murriztapenen editorea: eduki, sintaxi eta semantika egiaztatzailearen laguntzarekin.
- SPL iturburuaren abstrakzioa nabigazio erremintentzako.
- Konfigurazio editorea: konfigurazioak sortu eta aldatzeko.
- Software produktu lerroaren informazioa erakusten duen bista.
- Ezaugarrietara bideratutako programazioa:
    - AHEAD-ren integrazioa
    - FeatureC++-ren integrazioa
    - FeatureHouse-ren integrazioa
- Aspektuei bideratutako programazioa:
    - AspectJ-ren integrazioa
- Delta bideratutako programazioa:
    - DeltaJ-ren integrazioa
    - DeltaEcore-ren integrazioa
- Programaren aldaera guztien garapena.

### 4.2.1 Nola funtzionatzen du

Software produktu lerro bat sortzeko eta FeatureIDE-rekin lanean, lehenengo gauza ezaugarri eredua definitzea da (ikus 4.1.2 azpiatala). Lehen esan bezala, ezaugarri eredu batek ezaugarrien arteko konbinazioak zehazten ditu, grafikoki adierazitako ezaugarri diagrama batean. Domeinu bakoitzak bere ezaugarri eredua dauka eta honek Software Produktu Lerroa definitzen du.

Ondoren, ezaugarri bakoitzaren inplementazio kodea idatzi behar da. Kodea bata bestearen atzean konposatzen da, konposaketa horretarako *original()* hitz gakoa erabiltzen da eta aukeratutako aurreko ezaugarriari deia egiteko balio du. *original()* hitz gakoak *super()* hitz gakoaren antzekoa da baina superklase bati erreferentzia egin ordez, beste ezaugarri bati egiten dio erreferentzia.

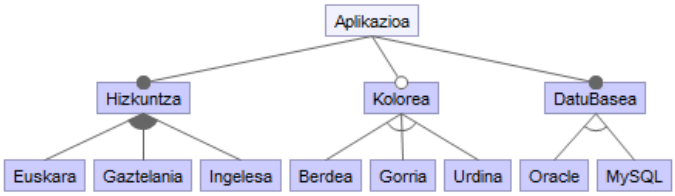
Hurrengo pausoa, konfigurazio berria definitzea da (ikus 4.1.3 atala). Konfigurazioa sortzerakoan ezaugarri ereduan zehaztutako murriztapen eta baldintzak bete behar dira, bestela konfigurazioa ez da baliagarri izango. Zuzena den konfigurazio bat sortu eta gordetzerakoan, dagokion kode inplementazioa automatikoki eraikitzen da, ezaugarrien inplementazioak (kodeak) konposatuz. 19. irudian, prozesu osoa agertzen da.




19. irudia: FeatureIDE-ren funtzionamendua.

### 4.2.2 Ezaugarri eredua garatu

FeatureIDE-n ezaugarri diagrama bai grafikoki (20. irudia) bai xml-a idatziz (21. irudia) garatu daiteke. Grafikoki egindako aldaketak automatikoki antzematen dira xml dokumentuan, eta alderantziz.



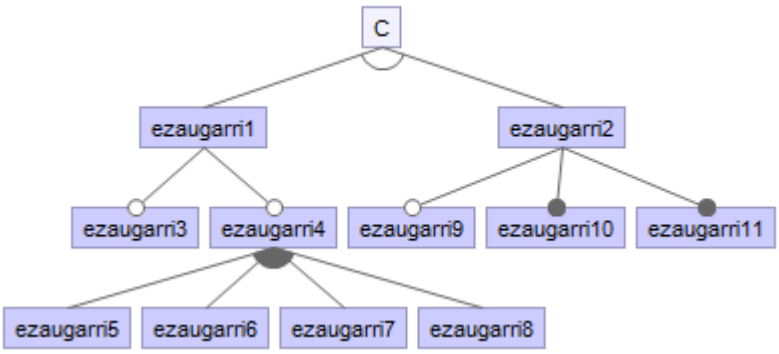

20. irudia: Ezaugarri diagrama grafikoki.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <featureModel chosenLayoutAlgorithm="1">
        <struct>
            <and abstract="true" mandatory="true" name="Aplikazioa">
                <or mandatory="true" name="Hizkuntza">
                    <feature mandatory="true" name="Euskara"/>
                    <feature mandatory="true" name="Gaztelania"/>
                    <feature mandatory="true" name="Ingelesa"/>
                </or>
                <alt name="Kolorea">
                    <feature mandatory="true" name="Berdea"/>
                    <feature mandatory="true" name="Gorria"/>
                    <feature mandatory="true" name="Urdina"/>
                </alt>
                <alt mandatory="true" name="DatuBasea">
                    <feature mandatory="true" name="Oracle"/>
                    <feature mandatory="true" name="MySQL"/>
                </alt>
            </and>
        </struct>
        <constraints>
            <rule>
                <imp>
                    <var>Euskara</var>
                    <var>Berdea</var>
                </imp>
            </rule>
            <rule>
                <imp>
                    <var>Ingelesa</var>
                    <not>
                        <var>Berdea</var>
                    </not>
                </imp>
            </rule>
            <rule>
                <imp>
                    <disj>
                        <conj>
                            <var>Ingelesa</var>
                            <var>Gaztelania</var>
                        </conj>
                        <conj>
                            <var>Ingelesa</var>
                            <var>Euskara</var>
                        </conj>
                    </disj>
                    <var>MySQL</var>
                </imp>
            </rule>
        </constraints>
        <calculations Auto="true" Constraints="true" Features="true" Redundant="true"/>
        <comments/>
        <featureOrder userDefined="false"/>
    </featureModel>
```

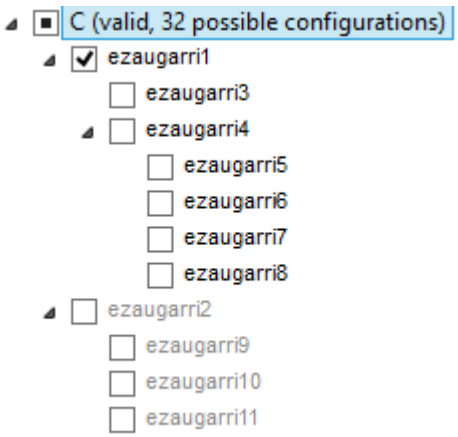
21. irudia: Ezaugarri diagrama xml egituran.

### 4.2.3 Konfigurazioak sortu

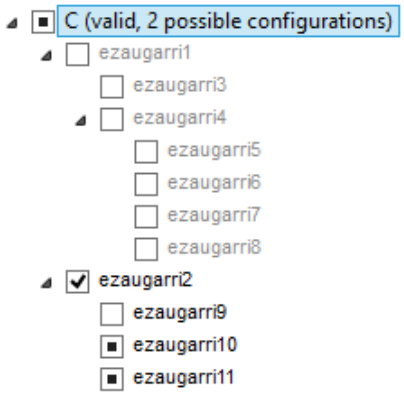
Laguntza moduan, FeatureIDE-k interfaze bat eskaintzen du zuzenak diren konfigurazioak soilik sortzeko. Adibidez, 22. irudian dagoen ezaugarri diagramaren konfigurazio bat sortzerakoan 23. irudian agertzen den interfazea agertzen da, *ezaugarri2* desgaituta *ezaugarri1* aukeratuta dagoelako eta beraien artean or-esklusibo erlazioa dutelako. Aldiz, *ezaugarri1* kentzerakoan eta *ezaugarri2* aukeratzerakoan, 24. irudian ikusten den moduan, *ezaugarri1* desgaituta eta *ezaugarri10* eta *ezaugarri11* zuzenean aukeratuta agertzen dira derrigorrezko ezaugarriak direlako.




22. irudia: Ezaugarri diagramaren adibidea.




23. irudia: Bigarren konfigurazioa




24. irudia: Lehenengo konfigurazioa.

## 4.3 GitHub

Git [4] bertsio kontrola erabiliz, GitHub-ek [5] softwareko lankidetza-proiektuak gordetzeko balio du. Kodea modu publikoan gordetzen da, modu pribatu batean gorde ahal izateko, gehienetan ordaindu beharreko kontu bat beharrezkoa da. UPV/EHUko posta helbidea erabiltzen bada kontua egiteko, aukera dago biltegi pribatu bat dohainik sortzeko.

### 4.3.1 Zertarako balio du

GitHub-ek garatzailearen kode biltegia gordetzen du eta proiektu baten barruan, taldean lan egiteko erreminta baliagarriak eskaintzen ditu. Gainera, garatzaile batek besteen softwarea hobetzen lagundu dezake. Helburu hauek bete ahal izateko, GitHub-ek *fork* eta *pull* funtzionalitateak inplementatzen ditu.

*Fork* egitea, garatzaile batek beste garatzaile baten biltegia klonatzea da, hau da, bere kontuan kopia bat egitea biltegi horrekin lan egin ahal izateko (akatsen bat konpondu, zerbait aldatu, etab). Behin eginbeharreko aldaketak egin dituela, *fork*-a egin duen garatzaileak, *pull* bat bidaliko dio proiektuaren jabeari. Honek erraz azter ditzake aldaketak, eta garatzailearen laguntza interesgarria iruditzen bazaio, jatorrizko biltegira gehitu.

### 4.3.2 Zer eskaintzen du

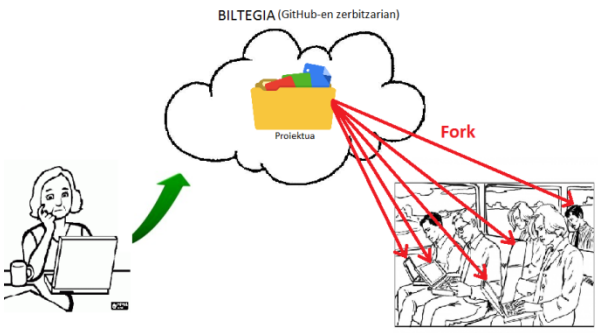
Gaur egun, GitHub-ek kodea gordetzeaz gain, beste erreminta batzuk eskaintzen ditu taldean lan egin ahal izateko:

- Proiektu barruko dokumentuen bertsio ezberdinen mantenuarentzako wiki bat.
- Arazoen jarraipen sistema bat. Sistema honi esker, kideek softwarearen inguruko arazoak edo iradokizunak zehaztasunez deskriba ditzakete.
- Kodea berrikusteko erreminta. Edozein fitxategiren edozein puntutan, oharrak txertatu daitezke eta *commit* zehatz batean egindako aldaketa batzuen buruzko eztabaidak sortu.
- Adar ikustailea. Biltegiko adar ezberdinetan egindako aurrerapenak alderatu daitezke.

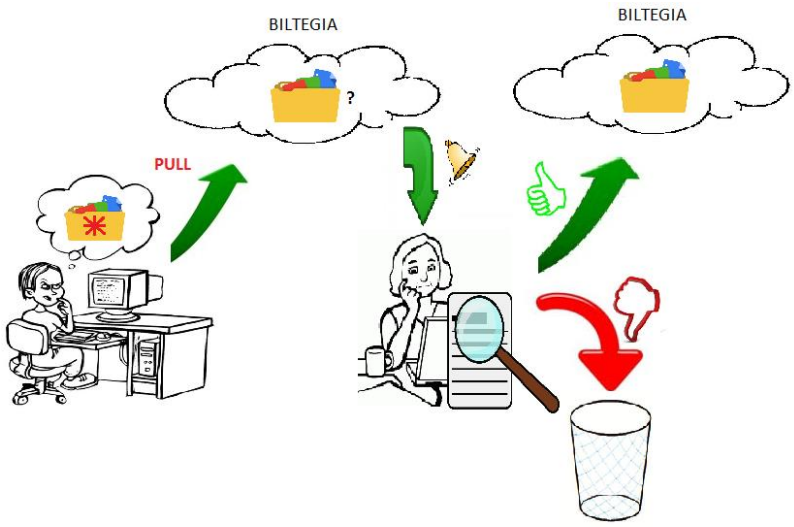
### 4.3.3 Nola erabiltzen da

Lanean hasteko, biltegi (*repository*) bat sortu behar da GitHub-en zerbitzarian. Ondoren, landu nahi den kodea bertara igo behar da. Behin kodea biltegian dagoela, taldeko beste kideek proiektuaren kopia bat egin dezakete nahi dituzten aldaketak egiteko, *fork* eginez , 25. irudian agertzen den moduan.

Kide horietako batek, bere kopian, aldatu beharreko zatiarekin bukatzerakoan, *pull* bat egin behar du. *Pull* hori egiterakoan, proiektuaren jabeari jakinarazpen bat iritsiko zaio. Jakinarazpen horretan, proiektuan egin nahi diren aldaketen zehaztapenak agertzen dira: nork bidali du, gehitutako / ezabatutako fitxategiak, gehitutako / ezabatutako kode lerroak. Proiektuaren jabeak aldaketak begiratzerakoan erabakitzen du *pull* hori onartu ala ezeztatu, 26. irudian ikus daitekeen moduan. *Pull*-a onartzen bada, aldaketak jatorrizko proiektuan txertatzen dira eta hortik aurrera *fork* egiterakoan, aldaketa horiek dituen proiektuaren kopia egingo da.




25. irudia: Biltegi baten *fork* egin.




26. irudia: *Pull*-aren prozesua.

Software proiektuetan, ohikoa da azpitaldetan banatzea eta azpitalde bakoitzak kode zati batekin lan egitea. Kasu hauetan, egokia da azpitalde bakoitzarentzako adar (*branch*) bat sortzea independenteki lan egiteko, 27. Irudian agertzen den moduan, eta eginbeharrekoa bukatzerakoan adarren artean *merge* egitea azpitalde ezberdinen lanak elkartzeko.

Demagun software garatzaile talde batek aplikazio berri bat garatzen ari dela. Orain arte garatutako aplikazioa adar nagusian (*master branch*) daukate, baina bi funtzionalitate berri garatu behar dituzte, beraien artean zerikusirik ez duten funtzionalitateak hain zuzen ere: sare sozialak txertatzea eta interfazea aldatzea. Beraz, bi azpitaldetan banatzea erabaki dute, talde bakoitzak funtzionalitate bat garatzeko.

Egoera honetan, bi adar sortzen dituzte sareadar eta interadar. Momentu honetan, master, sareadar eta interadar berdinak dira, 28. irudia ikus daitekeen moduan.

Ondoren bi taldeak lanean hasten dira independenteki, bakoitza bere adarrean (29. irudia).

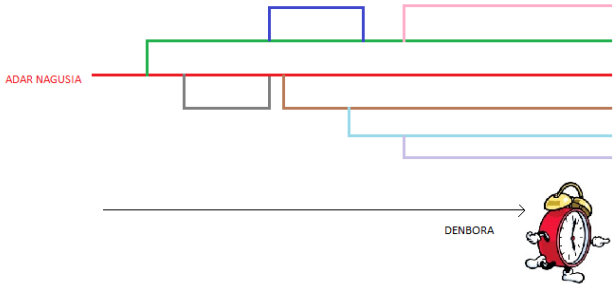
Sare sozialak txertatzea helburu bezala duen taldeak bere lanarekin bukatzen du, beraz *master* adarrarekin *merge* egiten du. Momentu honetan *master* adarrean sare sozialak txertatuak daude, 30. irudian ikus daitekeen moduan.

Interfazea bukatuta dagoenean, *interadar* eta *master*-en artean *merge* egingo da. Horrela, master adarrean, aplikazioa osoa (funtzionalitate guztiekin) izatea lortu da (31. irudia).

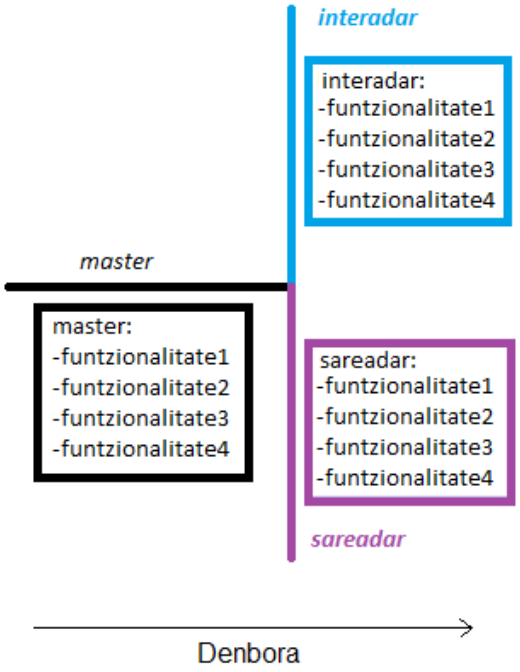
### 4.3.4 GitHub Windows-entzako

GitHub-ek Windows-entzako bezero bat sortu du [6]. Bezero honek interfaze grafikoa dauka eta hiru funtzionalitate inplementatzen ditu:

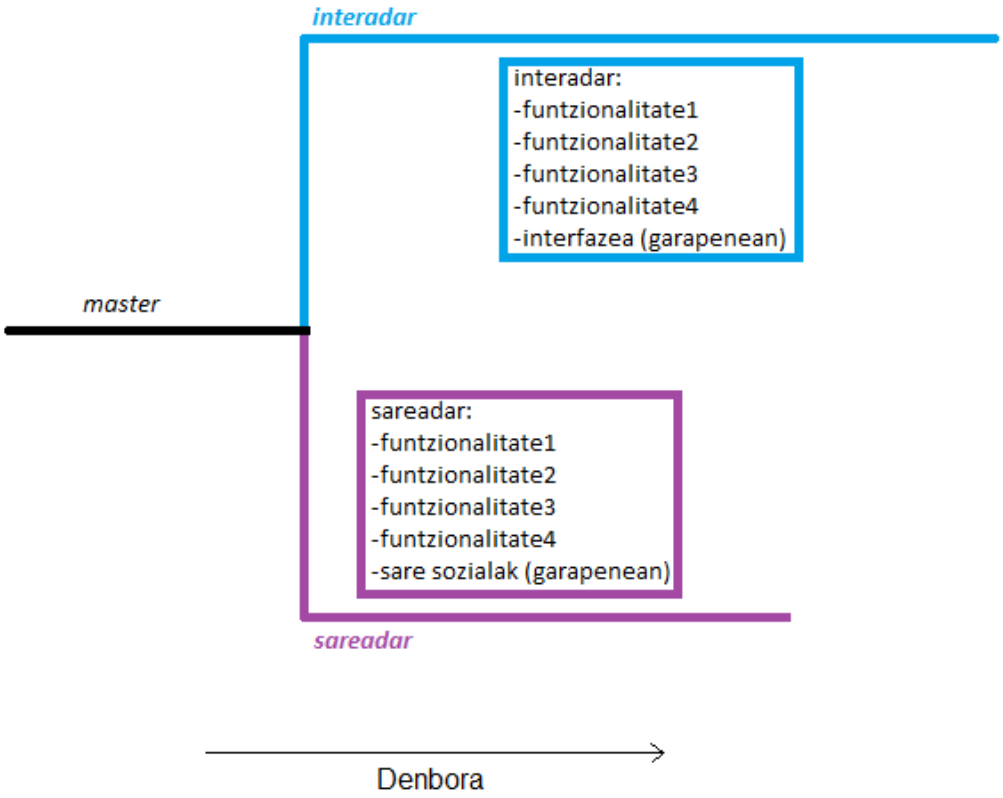
- Adar guztiak automatikoki sinkronizatzen ditu eta zerbait aldatzen denean jakinarazten dio bezeroaren erabiltzaileari.
- Biltegiak modu erraz batean klonatu (4.3 atalean azaldutako *fork* funtzionalitatea).
- Biltegi guztien eta beraien adar guztien kudeaketa: *commit* guztiak ikusi, aldaketak egin, adar berriak sortu …



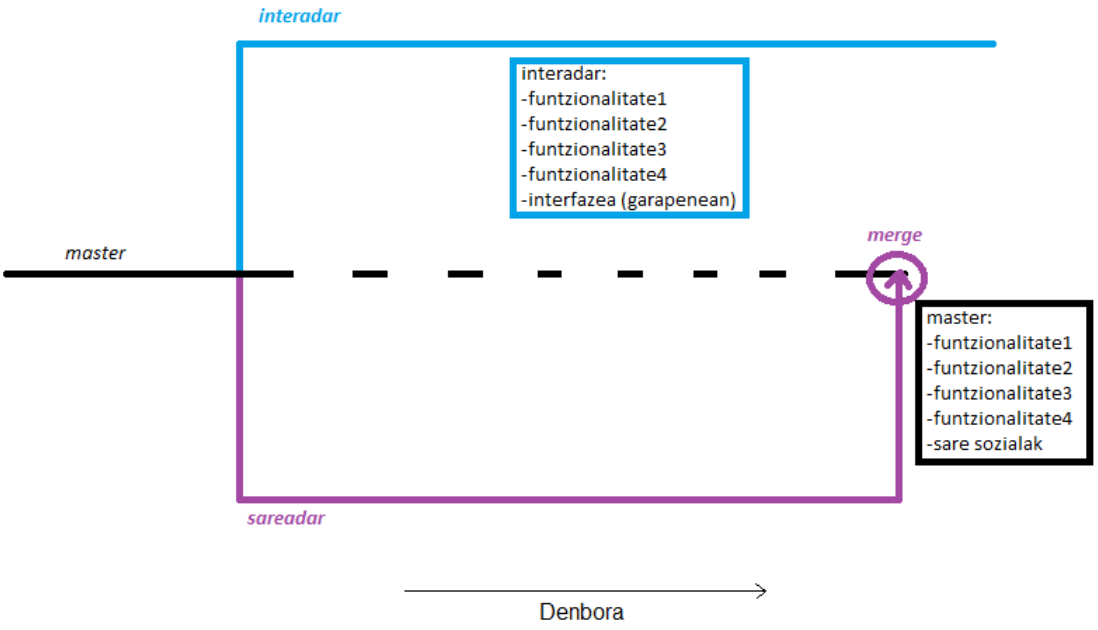

27. irudia: Biltegi baten adarrak.



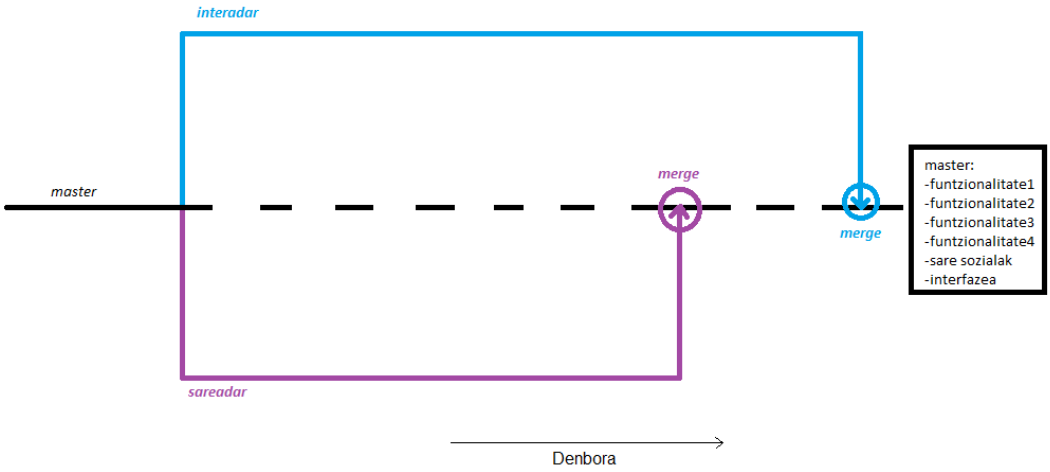

28. irudia: Hiru adarrek eduki bera dute.




29. irudia: Bi adarrak garapenean.




30. irudia: *sareadar* adarraren *merge*-a.




31. irudia: Produktu osoa *master* adarrean.

## 4.4 FaMa

FaMa (Feature Model Analyser) Sevillako unibertsitateko ISA taldeak garatutako ezaugarri ereduak aztertzeko erreminta da, ebatzaile (*solver*) ezberdinak (BDD[2], SAT[3] eta CSP[4]) erabiltzen dituen lehenengo lan ingurunea [2]. Ebatzaile bat software matematikoen atal bat adierazten duen hitz generikoa da, gehienetan arazo matematikoak ebazten dituen software liburutegi bat. FaMa lau modu ezberdinetan erabil daiteke.

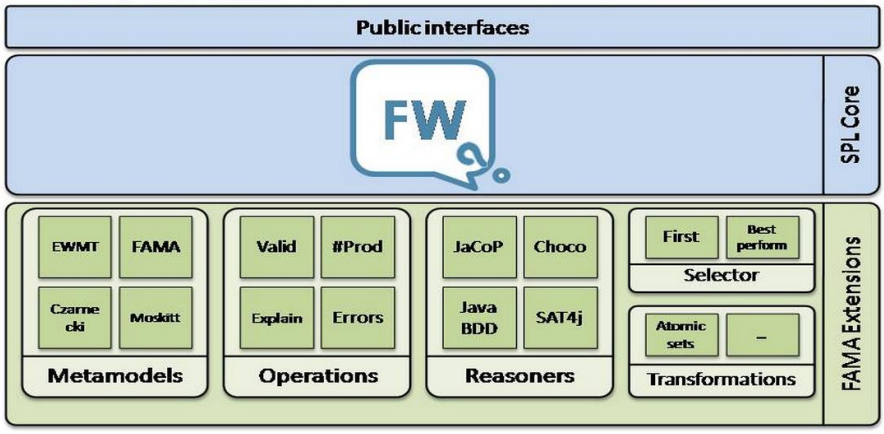
- FaMa shell: Komando lerroaren bidez ezaugarri ereduak[5] kargatu eta aztertze funtzioak exekuta daitezke.
- FaMa Web Service: FaMa-ren funtzio gehienek WSDL[6] ematen dute, bakoitzak bere aplikazioarentzat erabil ditzake.
- FaMa OSGI: FaMa ere OSGI paketeen multzo moduan erabil daiteke. OSGI, zerbitzuen integraziorako java-ren zehaztapen bat da.
- FaMa standalone: FaMa liburutegi hedagarri moduan erabilgarri dago ere.

Proiektu honen garapenean, FaMa standalone erabili da inplementaziorako.

[2] BDD: Erabaki binarioen diagramak lantzen ditu.
[3] SAT: Satisfakzio arazoak konpontzen ditu.
[4] CSP: Murriztapenen satisfakzio arazoak konpontzen ditu.
[5] FaMa-n ezaugarri diagramari ezaugarri eredua deitzen zaio.
[6] WSDL: Web zerbitzuak deskribatzeko erabiltzen den xml formatua.

32. irudian FaMa-ren arkitektura ikus daiteke.

- *QuestionTrader* edo *PublicInterfaces*: FaMa-rekin elkarrekintza ahalbidetzen duen interfazea da. Aldaketek azken erabiltzaileari ez eragiteko, *FaMa Core* ataletik bereizten da.
- *FaMaCore*: hedapen ezberdinen arteko komunikazioa ahalbidetzen du.
- *FaMaExtensions*: erabiltzailearen beharretara egokitzeko, modulu ezberdinen multzoa da.
    - *Metamodels*: Erabiltzailearen ezaugarri ereduak kargatu eta gordetzen ditu.
    - Reasoners: Beste garatzaile batzuk garatutako ebatzaileak gehitzen ditu. Ebatzaile lehentsiak: JavaBDD, Choco, JACOP GPL eta SAT4j.
    - *Operations*: arrazoitze funtzio berriak gehitzen ditu. Zenbat produktu sor daitezke, akatsen hautemate eta azalpena…
    - *ReasonerSelector*: exekuzio garaian, ebatzaile guztien arten, aukeratzen du egindako galderei erantzuteko. Horretarako, aurretik zehaztutako irizpide batean oinarritzen da (errendimendua, zehaztasuna…).
    - *Benchmarking*: eragiketa berdinerako, ebatzaile ezberdinen errendimendua konparatzen du.




32. irudia: FaMa-ren arkitektura.

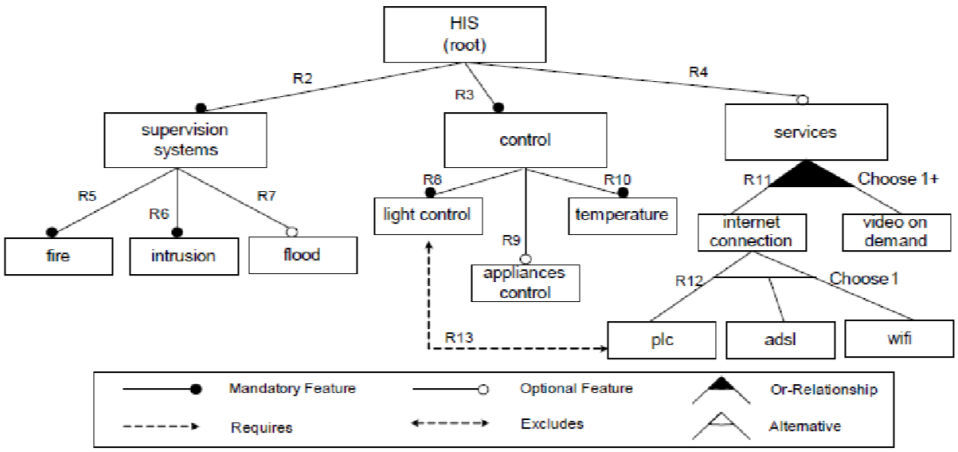
### 4.4.1 Nola funtzionatzen du

FaMa erabiltzeko aukera ezberdinen artean, proiektu honetan FaMa standalone erabili denez, aukera honen funtzionamendua azalduko da. Ezaugarri eredu baten informazioa lortzeko, ondorengo pausoak jarraitu behar dira:

- Ezaugarri eredua definitu (4.4.2 atalean azaltzen den moduan). Adibidez *featureModel.xml* fitxategian.
- QuestionTrader bat sortu, ezaugarri ereduaren inguruko galderak egin ahal izateko.
    - *QuestionTrader qt= new QuestionTrader();*
- Ezaugarri eredua kargatu.
    - *VariabilityModel fm= qt.openFile("featureModel.xml");*
- Ezaugarri eredua finkatu.
    - *qt.setVariabilityModel(fm);*
- Galdera sortu (adibidez, ezaugarri eredua baliozkoa den ala ez).
    - *ValidQuestion vq= (ValidQuestion) qt.createQuestion("valid");*
- Galdera exekutatu.
    - *qt.ask(vq);*
- Erantzuna lortu.
    - *boolean b = vq.isValid();*

### 4.4.2 Ezaugarri eredua garatu

FaMa-k lau fitxategi mota onartzen ditu ezaugarri ereduak adierazteko: XML, AFM, DOT (Graph Viz bidezko adierazpena) eta X3D (ezaugarri ereduen 3D adierazpena). Bi fitxategi mota erabilienak XML eta AFM dira. Adibide moduan, 33. irudian dagoen ezaugarri eredua, 34. irudian AFM moduan adierazita dago eta 35. irudian XML moduan.




33. irudia: HIS-en ezaugarri eredua.

```
%Relationships
HIS: SUPERVISION_SYSTEM CONTROL [SERVICES];
SUPERVISION_SYSTEM: FIRE INTRUSION [FLOOD];
CONTROL: LIGHT_CONTROL [APPLIANCES_CONTROL] TEMPERATURE;
SERVICES: [1,2]{VIDEO INTERNET};
INTERNET: [1,1]{POWER_LINE ADSL WIRELESS};

%Constraints
LIGHT_CONTROL EXCLUDES POWER_LINE;
```

34. irudia: HIS ezaugarri eredua AFM egituran adierazita.

```
<feature-model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.tdg-seville.info/benavides/featuremodelling/feature-model.xsd">
    <feature name="HIS">
        <binaryRelation name="R-1">
            <cardinality max="1" min="1" />
            <solitaryFeature name="SUPERVISION-SYSTEM">
                <binaryRelation name="R-4">
                    <cardinality max="1" min="1" />
                    <solitaryFeature name="FIRE" />
                </binaryRelation>
                <binaryRelation name="R-5">
                    <cardinality max="1" min="1" />
                    <solitaryFeature name="INTRUSION" />
                </binaryRelation>
                <binaryRelation name="R-6">
                    <cardinality max="1" min="0" />
                    <solitaryFeature name="FLOOD" />
                </binaryRelation>
            </solitaryFeature>
        </binaryRelation>
        <binaryRelation name="R-2">
            <cardinality max="1" min="1" />
            <solitaryFeature name="CONTROL">
                <binaryRelation name="R-7">
                    <cardinality max="1" min="1" />
                    <solitaryFeature name="LIGHT-CONTROL" />
                </binaryRelation>
                <binaryRelation name="R-8">
                    <cardinality max="1" min="0" />
                    <solitaryFeature name="APPLIANCES-CONTROL" />
                </binaryRelation>
                <binaryRelation name="R-9">
                    <cardinality max="1" min="1" />
                    <solitaryFeature name="TEMPERATURE" />
                </binaryRelation>
            </solitaryFeature>
        </binaryRelation>
        <binaryRelation name="R-3">
            <cardinality max="1" min="0" />
            <solitaryFeature name="SERVICES">
                <setRelation name="R-10">
                    <cardinality max="2" min="1" />
                    <groupedFeature name="VIDEO" />
                    <groupedFeature name="INTERNET">
                        <setRelation name="R-11">
                            <cardinality max="1" min="1" />
                            <groupedFeature name="POWER-LINE" />
                            <groupedFeature name="ADSL" />
                            <groupedFeature name="WIRELESS" />
                        </setRelation>
                    </groupedFeature>
                </setRelation>
            </solitaryFeature>
        </binaryRelation>
    </feature>
    <excludes name="ex-1" feature="LIGHT-CONTROL" excludes="POWER-LINE" />
</feature-model>
```

35. irudia: HIS ezaugarri eredua XML egituran adierazita.

### 4.4.3 Konfigurazioak sortu

Erreminta honetan konfigurazioak ezin dira grafikoki sortu. Konfigurazio bat sortzeko eta balioztatzeko ondorengo pausoak jarraitu behar dira, demagun sortu nahi den konfigurazioa *HIS*, *supervisionSystem*, *fire* eta *flood* ezaugarriak izan behar dituela:

- Ezaugarri eredua definitu (4.2.2 atalean azaltzen den moduan). Adibidez *featureModel.xml* fitxategian.
- QuestionTrader bat sortu, ezaugarri ereduaren inguruko galderak egin ahal izateko.
    - *QuestionTrader qt= new QuestionTrader();*
- Ezaugarri eredua kargatu.
    - *VariabilityModel fm= qt.openFile("featureModel.xml");*
- Ezaugarri eredua finkatu.
    - *qt.setVariabilityModel(fm);*
- Produktu berri bat sortu.
    - *Product nirep = new Product ();*
- Konfigurazioan egongo diren ezaugarriak lortu.
    - GenericFeature f1 = fm.searchFeatutreByName("HIS");
    - GenericFeature f2 = fm.searchFeatutreByName("supervision systems");
    - GenericFeature f3 = fm.searchFeatutreByName("fire");
    - GenericFeature f4 = fm.searchFeatutreByName("flood");
- Sortutako produktuari lortutako ezaugarriak gehitu.
    - *nirep.addFeature(f1);*
    - *nirep.addFeature(f2);*
    - *nirep.addFeature(f3);*
    - *nirep.addFeature(f4);*
- Galdera sortu (produktua baliozkoa den ala ez).
    - *ValidProductQuestion vpq = (ValidProductQuestion) qt.createQuestion("ValidProduct");*
- Galderari balioztatu nahi den produktua sortu.
    - *vpq.setProduct(nirep);*
- Galdera exekutatu.
    - *qt.ask(vpq);*
- Emaitza lortu.
    - *boolean baliozkoa= vpq.isValid();*

Kasu honetan, emaitza *false* izango litzateke derrigorrezkoak diren ezaugarri batzuk (*intrusion*, *control*, *light control* eta *temperature*) ez direlako produktuan agertzen.

# 5. KAPITULUA

# GitLine-ren zabaltzea

GitLine SPL biltegiekin lan egiteko GitHub-en hedapena da, ezaugarri eta produktu ezberdinen biltegiez osatuta dago. Ezaugarri biltegiak ezaugarrien inplementazioa dauka bere adarretan (branches), ezaugarri bakoitzeko adar bat. Produktu biltegiek, GitHub biltegi "normal" baten moduan, produktu bakoitzaren inplementazioa gordetzen dute.

Txosten honetan aurkezten den proiektuaren hasieran jasotako aplikazioak, ondorengo funtzionalitateak garatuak zituen:

- *Product Fork*:

  Funtzionalitate honek ezaugarri biltegi batetik, produktu biltegiak sortzen ditu. Aplikazioaren ingeniariak sortu nahi duen produktuaren konfigurazioa eskuz sartzen du eta GitLine-ek sartutako konfigurazioarekin, produktu biltegi bat sortzen du GitHub-en. Funtzionalitate honek aplikazio ingeniariak sartutako konfigurazioa zuzena dela suposatzen du.

- *Forward Propagation*:

  Funtzionalitate honek ezaugarri biltegiaren aldaketak produktu biltegietara hedatzen ditu. Hau ezaugarri baten kodea aldatzen denean eta aldaketa horiek ezaugarri hori duten produktu biltegietara hedatu behar denean gertatzen da. Produktu biltegiaren atal batean bere produktua osatzen duten ezaugarriak izan dituzten aldaketak ohartarazten dira. Aplikazio ingeniariak erabakitzen du aldaketa horiek onartu ala ez.

*Product fork* eta *Forward propagation* funtzionalitateak JavaScript-en [11] inplementatuak daude, Firefox-eko plugin [12] baten moduan. Js liburutegia erabiltzen da GitHub-eko API v3-a [7] atzitzeko. Kontsulta liburutegi bat da, bakarrik GET motako funtzioak inplementatzen ditu. POST eta DELETE funtzioak, GitHub web-ari HTTP eskaerak eginez burutzen dira.

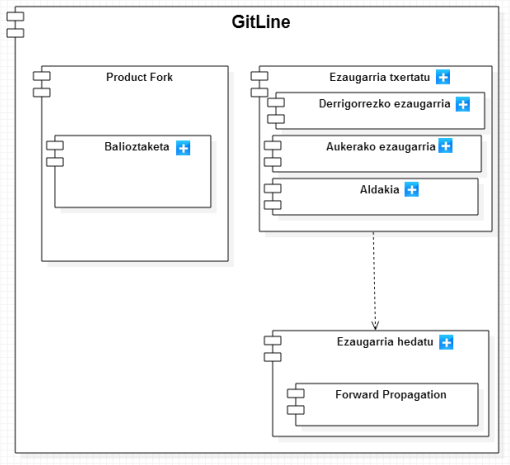
Egoera horretatik abiatuta, txosten honetan aurkezten den proiektuan ondorengo hobekuntzak gehitu zaizkio aplikazioari:

- *Balioztaketa*:
  Funtzionalitate honek, *Product fork* egiterakoan, aplikazio ingeniariak sartutako konfigurazioa zuzena dela ziurtatzen du. Modu honetan, sortutako produktu biltegien konfigurazioak ezaugarri biltegian dagoen ezaugarri ereduan oinarrituta zuzenak izango dira.

- *Ezaugarria txertatu*:
  Ezaugarri biltegien bizi-zikloa handitzeko asmoarekin, ezaugarri biltegietan ezaugarri berri bat gehitzeko aukera inplementatu da. Ezaugarri biltegian dagoen ezaugarri eredua eguneratzen da eta ezaugarri berriari dagokion adar berria sortzen da domeinuko ingeniariak ezaugarri berriari dagokion kodea gehitzeko.

- *Ezaugarria hedatu*:
  Ezaugarri biltegien eta produktu biltegien arteko sinkronizazioa bermatzeko, ezaugarri biltegi batean ezaugarri berri bat gehitu eta inplementatu ondoren, ezaugarri biltegi horretatik sortutako produktu biltegietara hedatzeko aukera inplementatu *da Forward propagation* funtzioaren laguntzarekin.

36. irudian ikus daitekeen moduan, GitLine-eko hiru moduluekin lan egin da: *Product Fork*, *Ezaugarria txertatu* eta *Ezaugarria hedatu*. "+" baten bidez adierazita daude ikasleak inplementatutako modulu berriak. *Product Fork* moduluan, *balioztaketa* gehitu da 6. kapituluan azalduta. *Ezaugarria txertatu* modulu osoa ikasleak garatu du, 7.1 atalean azalduta eta aurretik garatuta zegoen *ForwardPropagation* funtzionalitatea erabiliz, *ezaugarria hedatu* funtzionalitatea garatu da, 7.2 atalean azalduta.




36. irudia: GitLine-ren moduluak.

# 6. KAPITULUA

# Balioztaketa

Proiektu honetan garatutako *Balioztaketa* funtzionalitatea sakon aztertuko da, bere diseinua, inplementazioa eta interfazeak azalduz. Gainera, 6.2 atalean garapenaren inguruko erronkak laburbildu dira.

Funtzionalitate honek aurretik ikerlariak garatutako *productFork* funtzionalitatea hobetzea du helburu. Ikerlariak garatutako *productFork*-a funtzionalitateak: ezaugarri biltegi batetik, produktu biltegiak sortzen ditu. Aplikazioaren ingeniariak sortu nahi duen produktuaren konfigurazioa eskuz sartzen du eta GitLine-ek sartutako konfigurazioarekin, produktu biltegi bat sortzen du GitHub-en. Funtzionalitate honek aplikazio ingeniariak sartutako konfigurazioa zuzena dela suposatzen du.

Proiektu honetan garatutako hobekuntza, aplikazio ingeniariak sartutako konfigurazioa zuzena dela ziurtatzea da. Modu honetan, sortutako produktu biltegien konfigurazioak ezaugarri biltegian dagoen ezaugarri ereduan oinarrituta zuzenak izango dira. Horretarako, aplikazio ingeniariak sortu nahi duen konfigurazioa eskuz sartu ordez, konfiguradore batean aukeratu beharko du.

Balioztaketa funtzionalitatearen arkitektura 37. irudian ikus daiteke. Prozesua *gitHubDeltas* moduluan (modulu honek biltegiak eta interfazeak kudeatzen ditu) hasten da, bertan ezaugarri biltegian dagoen ezaugarri eredua eta aplikazio ingeniariak aukeratutako konfigurazioa lortzen dira. Ondoren, modulu horretan lortutako informazioa *script-compiler* modulura pasatzen da, modulu honek jasotako informazioarekin ebatzailea exekutatuko duen script-a exekutatzeaz arduratzen da (*famaValid.bat*). Script-ak ebatzailea exekutatzen du eta ebatzaileak lortutako informazioarekin fitxategi berriak sortzen ditu (*isValid.txt*, *featuresToDeselect.txt* eta *proposedProductFile.txt*). Bukatzeko, *gitHubDeltas* moduluak fitxategi horietan dagoen informazioa irakurri eta pantailaratzeaz arduratzen da.

Konfiguradore hau garatzerakoan bi zati nagusi nabarmendu dira: ebatzailea, 8. kapituluan azaldutakoa eta ebatzailearen eta GitLine-n arteko lotura.

## 6.1 GitLine eta ebatzailearen arteko lotura.

Atal honetan GitLine eta ebatzailea elkartzeko jarraitu den diseinua eta nola inplementatu den azalduko da. Hasteko, diseinua azalduko da, ondoren, inplementazioa eta bukatzeko inplementazioan erabilitako funtzio batzuen azalpenak.

### 6.1.1 Diseinua

38. irudian, balioztaketan parte hartzen duten interfazeen arteko trantsizioak ikus daitezke. ProductFork botoia sakatzerakoan, ezaugarri eredua osatzen duten ezaugarri guztiak listaratuak azaltzen dira, aplikazio ingeniariak konfigurazioa aukeratzeko. Aukeratutako konfigurazioa zuzena bada, aukeratutako konfigurazio hori agertuko da produktu biltegia sortzeko aukerarekin. Aukeratutako konfigurazioa zuzena ez bada, konfigurazioa zuzena izateko kendu beharreko ezaugarriak, proposatutako konfigurazio bat eta proposatutako konfigurazioarekin produktu biltegi bat sortzeko aukera agertuko da.

11. taulan eta 12. taulan , balioztatze funtzionalitatearen diagrama ikus daiteke. Taula bi zutabe nagusitan banatuta dago: bista eta logika. Bista zutabean, GitLine-en gitHubDeltas.js fitxategian gertatzen dena azaltzen da, eta logika zutabea beste bi zutabetan bereizten da: alde batetik, GitLine-en script-compiler.js fitxategian gertatzen dena azaltzen duena eta bestetik ebatzailea exekutatzeko beharrezko input-ak eta lortutako output-ak azaltzen dituena. Bigarren zutabean agertzen diren funtzioen azalpena 6.1.3-Funtzio gehigarriak atalean aurki daiteke.

### 6.1.2 Inplementazioa

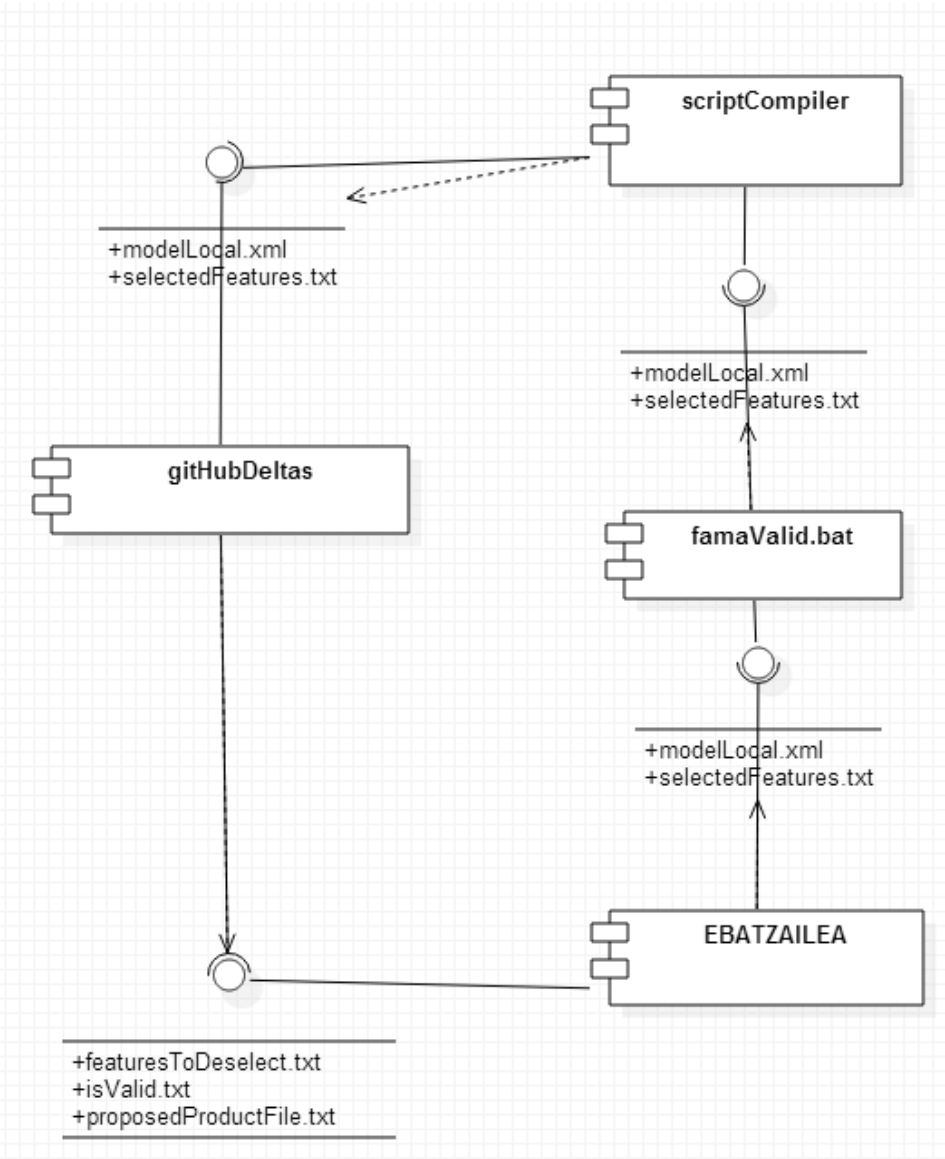
Funtzionalitate hau inplementatzeko JavaScript [11]*, mozilla developer* [12] eta GitHub plugina [7] erabili dira.

Interfazean agertzen den *ProductFork* botoia sakatzerakoan (1. taularen 1. pausoa), ondorengo pausoak jarraitzen dira:

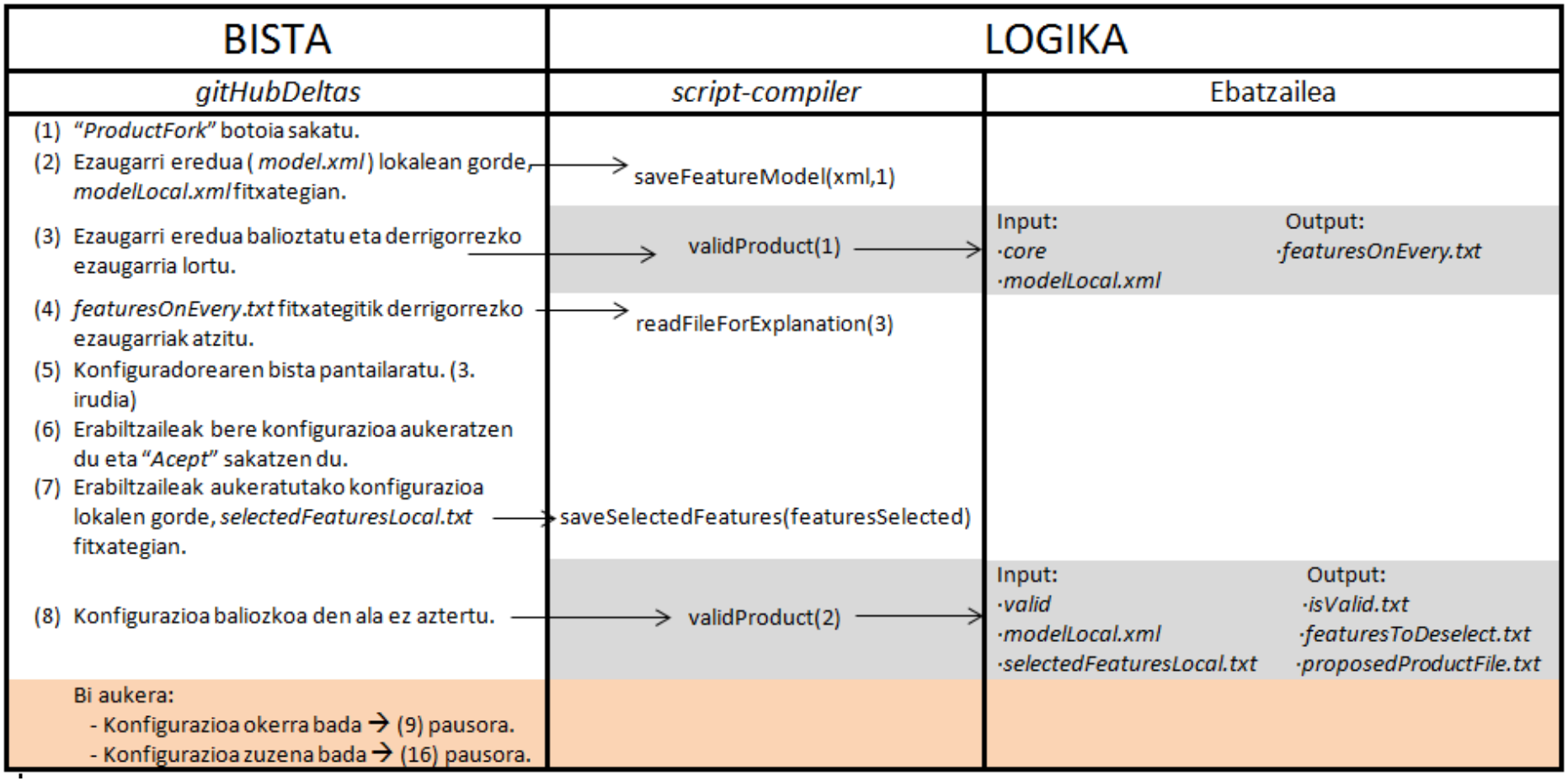
#### 1. Pausoa: Konfigurazioa sortzeko interfazea sortu.

- *InstallEController.prototype.execute=function(act){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:

- *window.prompt* baten bidez produktua eskuzko edo lagunduriko konfiguradore baten bidez egin nahi duen galdetzen zaio erabiltzaileari.
- *assisted* aukeratzen badu, lagunduriko konfiguradorea sortzen duen funtzioa deitzen da: *DeltaUtils.createConfigurator(0);*



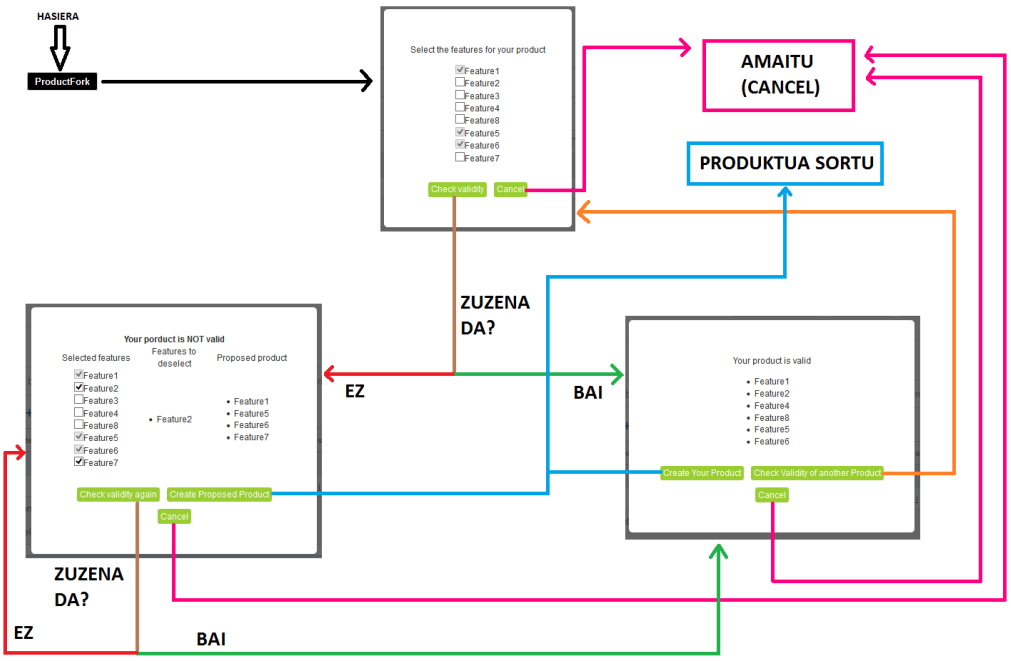

37. irudia: *Balioztaketaren* arkitektura.




11. taula: *Balioztatze* funtzionalitatearen diagrama.

| BISTA | LOGIKA | |
|---|---|---|
| *gitHubDeltas* | *script-compiler* | Ebatzailea |
| Konfigurazioa okerra denean: | | |
| (9) Erabiltzaileak bere konfigurazioan aukeratutako ezaugarriak atzitu. | → readFileExplanation(2) | |
| (10) Ezaugarri ereduaren derrigorrezko ezaugarriak atzitu. | → readFileExplanation(3) | |
| (11) Erabiltzaileak aukeratutako konfigurazioa egokia izateko, alde batera utzi beharreko ezaugarriak atzitu. | → readFileExplanation(0) | |
| (12) Proposatutako konfigurazioa atzitu. | → readFileExplanation(1) | |
| (13) Konfigurazioa okerra dela adierazten duen bista pantailaratu. (5. irudia)<br>Bi aukera:<br>- *"CheckValidity"* sakatu → (7) pausora.<br>- *"CreateProposedProduct"* sakatu → (14) pausora. | | |
| Biltegi berria sortu | | |
| (14) Azken konfigurazioaren ezaugarriak atzitu. | → readFileForExplanation(option) | |
| (15) *createPoduct* exekutatu eta biltegi berria sortu, AMAIERA. | | |
| Konfigurazioa zuzena denean: | | |
| (16) Aukeratutako ezaugarriak atzitu. | → readFileForExplanation(2) | |
| (17) Konfigurazioa zuzena dela adierazten duen bista pantailaratu. (4. irudia)<br>Bi aukera:<br>- *"CreateProduct"* sakatu → (14) pausora.<br>- *"CheckValidityOfAnotherProductAgain"* sakatu → (2) pausora. | | |

12. taula: Balioztatze funtzionalitatearen diagrama (jarraipena).

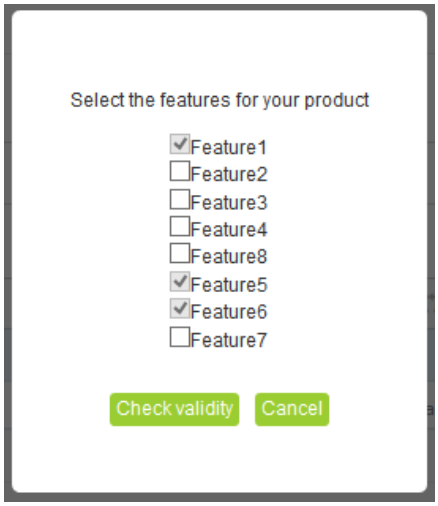



38. irudia: Interfazeen arteko trantsizioak

- *DeltaUtils.createConfigurator=function(option, kind)*{} funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *ghAuthorRepo.fetch* → biltegia atzitzen du.
    - *ghAuthorRepo.fetchBranches* → biltegiaren adar guztiak atzitzen ditu.
    - *ghAuthorRepo.getBranchByName("master")* → *master* izeneko adarra atzitzen du.
    - *master.fetchContents* → *master* adarrean dauden fitxategi eta direktorioak atzitzen ditu
    - *master.getFileByName("model.xml")* → *model.xml* fitxategia lortzen du.
    - *featureModelFile.fetchContent model.xml* → fitxategiaren edukia lortzen du.
    - *saveFeatureModel(xml,1)* → *model.xml* fitxategia lokalean gordetzen da (11. taularen, 2. pausoa).
    - Ondoren, derrigorrezkoak diren ezaugarriak lortuko dira (11. taularen, 3. eta 4. pausoak) *readFileForExplanation(3)* bidez eta konfiguradoreari dagokion html kodea sortuko zaio.
    - *UI.Dialog.show_product_configurator_dialog* funtzioari deitzen zaio, sortutako html kodeari botoiak gehitzeko eta pantailaratzeko.

- *UI.Dialog.show_product_configurator_dialog* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
    - Bukatzeko sortutako interfazea pantailaratzen da (11. taularen 5. pausoa).

Erabiltzaileak konfigurazio bat aukeratzen du 39. irudian agertzen den interfazean eta CheckValidity sakatzen du (11. taularen 6. pausoa). Momentu honetan konfigurazio hori egokia den ala ez aztertu behar da (2. pausoa).




39. irudia: Konfiguradorearen lehenengo interfazea.

2. Pausoa: Konfigurazioa baliozkoa den ala ez aztertu.

- DeltaUtils.selectedCheck(p) funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *inputElements = html_nodes.getElementsByClassName('features')* → parametro bezala jaso den html kodetik *class="features"* guztiak lortzen ditu.
    - *for* baten bidez aukeratutako ezaugarriak *String* batean gordetzen dira.
    - *saveSelectedFeatures(featuresSelected)* → aukeratutako ezaugarriak lokalean gordetzen ditu (11. taularen 7. pausoa).
    - Balioztaketa egingo duen funtzioari deitzen zaio (11. taularen 8. pausoa): *DeltaUtils.checkConfigurationValidity()*
- DeltaUtils.checkConfigurationValidity() funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Ebatzailea exekutatuko duen funtzioari deia *validProduct(2)* bidez.

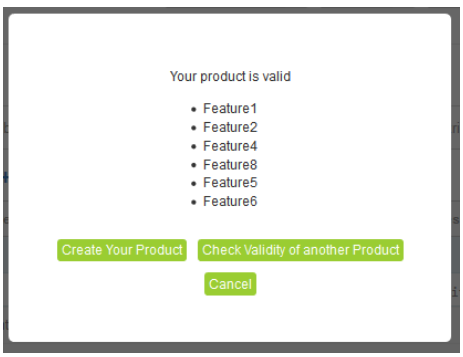
- *isValid=readFileIsValid()* ebatzaileak sortutako isValid.txt fitxategia irakurtzeko.
  - Baliozkoa bada *DeltaUtils.createConfigurator(1)* exekutatuko da.
  - Baliozkoa ez bada *DeltaUtils.createConfigurator(2)* exekutatuko da.

2. pausoa bukatzerakoan, bi aukera daude: konfigurazioa baliozkoa bada 3. pausoa exekutatuko da eta konfigurazioa baliozkoa ez bada 5. pausoa.

### 3. Pausoa: Konfigurazioa egokia dela adierazi.

- *DeltaUtils.createConfigurator(1)* funtzioa exekutatzen da eta ondorengoa gertatzen da:
  - *ghAuthorRepo.fetch* → biltegia atzitzen du.
  - *ghAuthorRepo.fetchBranches* → biltegiaren adar guztiak atzitzen ditu.
  - *ghAuthorRepo.getBranchByName("master")* → *master* izeneko adarra atzitzen du.
  - *master.fetchContents* → *master* adarrean dauden fitxategi eta direktorioak atzitzen ditu
  - *master.getFileByName("model.xml")* → *model.xml* fitxategia lortzen du.
  - *featureModelFile.fetchContent model.xml* → fitxategiaren edukia lortzen du.
  - *saveFeatureModel(xml,1)* → *model.xml* fitxategia lokalean gordetzen da.
  - Egoera honetan, konfigurazioa lokaletik irakurriko da (12. taularen 16. pausoa) eta dagokion html kodea sortuko da lista moduan agertzeko.
  - *UI.Dialog.show_product_configurator_dialog* funtzioari deitzen zaio sortutako html kodeari botoiak gehitzeko eta pantailaratzeko.
- *UI.Dialog.show_product_configurator_dialog* funtzioa exekutatzen da eta ondorengoa gertatzen da.
  - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
  - Bukatzeko sortutako interfazea pantailaratzen da (12. taularen 17. pausoa).

3. pausoa bukatzerakoan 40. irudian agertzen den interfazea agertuko da eta erabiltzaileak bi aukera izango ditu: baliozkoa den konfigurazioarekin produktua sortu (4. pausoa) edo beste konfigurazio bat baliozkoa den ala ez aztertu (1. pausoa).




40. irudia: Konfigurazio egokia dela adierazten duen interfazea.

### 4. Pausoa: Baliozkoa den konfigurazioarekin, produktua sortu.

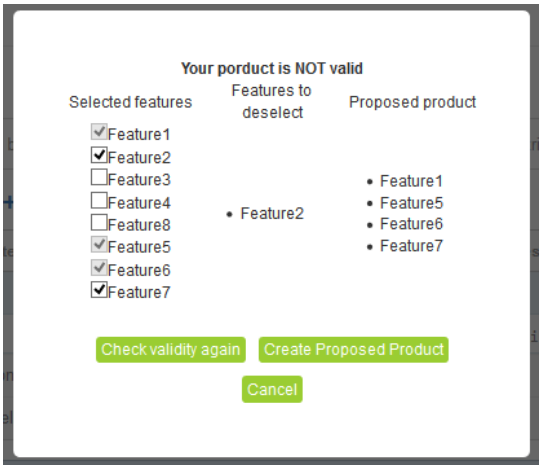
- *DeltaUtils.createProduct(2)* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *readFileForExplanation(2)* →selectedFeatures.txt fitxategitik produktua sortzeko ezaugarriak lortzen dira (12. taularen 14. pausoa).
    - Bukatzeko, produktua sortuko duen funtzioari deia egiten zaio *DeltaUtils.enactProductComposition()* (12. taularen 15. pausoa) (Hemendik aurrera, Leticia izan da garatzailea).

4. pausoa bukatzerakoan, aukeratutako ezaugarriekin biltegi berria sortuko da.

### 5. Pausoa: Konfigurazioa egokia EZ dela adierazi.

- *DeltaUtils.createConfigurator(2)* funtzioa exekutatu.
    - *ghAuthorRepo.fetch* → biltegia atzitzen du.
    - *ghAuthorRepo.fetchBranches* →biltegiaren adar guztiak atzitzen ditu.
    - *ghAuthorRepo.getBranchByName("master")* → *master* izeneko adarra atzitzen du.
    - *master.fetchContents* → *master* adarrean dauden fitxategi eta direktorioak atzitzen ditu
    - *master.getFileByName("model.xml")* → *model.xml* fitxategia lortzen du.
    - *featureModelFile.fetchContent model.xml* → fitxategiaren edukia lortzen du.
    - *saveFeatureModel(xml,1)* → *model.xml* fitxategia lokalean gordetzen da.
    - Egoera honetan interfazea hiru zatitan banatzen da:
        - *Selected Features* zutabea: zutabe honetan, erabiltzaileak aukeratutako okerreko konfigurazioa agertuko da. Horretarako, *readFileForExplanation(2)* funtzioari deituko zaio (12. taularen 9. pausoa). Gainera, derrigorrezko ezaugarriak desgaituak agertuko dira, horretareko *readFileForExplanation(3)* funtzioari deituko zaio (12. taularen 10. pausoa) derrigorrezko ezaugarriak lortzeko.
        - *Features To Deselect* zutabea: zutabe honetan, ebatzaileak konfigurazioa egokia izateko, alde batera utzi beharreko ezaugarriak agertuko dira. Horretarako *readFileForExplanation(0)* funtzioari deituko zaio (12. taularen 11. pausoa).
        - *Proposed Product* zutabea: zutabe honetan, ebatzaileak proposatutako konfigurazioa agertuko da. Horretarako *readFileForExplanation(1)* funtzioari deituko zaio (12. taularen 12. pausoa).
    - *UI.Dialog.show_product_configurator_dialog* funtzioari deitzen zaio sortutako html kodeari botoiak gehitzeko eta pantailaratzeko.
- *UI.Dialog.show_product_configurator_dialog* funtzioa exekutatzen da.
    - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
    - Bukatzeko sortutako interfazea pantailaratzen da (12. taularen 13. pausoa).

5. pausoa bukatzerakoan 41. irudian ikus daitekeen interfazea agertuko da eta bi aukera izango ditu erabiltzaileak: lehenengo zutabean konfigurazio berri bat aukeratu eta baliozkoa den ala ez aztertu (1. pausoa) edo ebatzaileak proposatutako konfigurazioarekin produktua sortu (4. pausoa).




41. irudia: Konfigurazioa egokia EZ dela adierazten duen interfazea.

### 6.1.3 Funtzio Gehigarriak

*6.1.1 Diseinua* atalean aurkeztutako tauletan, *script-compiler* zutabean agertzen diren funtzioak atal honetan sakonago azaltzen dira.

- *saveFeatureModel: function(featureModel, option){…}:*
  - Deskribapena: Funtzio honek lokalean gordetzen du ezaugarri eredu bat.
  - Sarrera:
    - *featureModel*: lokalean gorde nahi den ezaugarri eredua String batean.
    - *option*: parametro honek bi balio posible ditu.
      - 1: Ezaugarri eredua lokalean gordetzen da balioztaketa prozesuan atzitzeko.
      - 2: Ezaugarri eredua lokalean gordetzen da ondoren ezaugarri berri bat txertatzeko.
  - Irteera: *option* parametroaren arabera irteera ezberdina izango da.
    - 1: Fama izeneko direktorioan *modelLocal.xml* fitxategia sortzen du pasatako ezaugarri ereduarekin.
    - 2: Fama izeneko direktorioan *modelLocalInsert.xml* fitxategia sortzen du pasatako ezaugarri ereduarekin.

- *readFileForExplanation: function(option){..}:*
    - Deskribapena: Funtzio honek ebatzaileak emandako informazioa irakurtzeko balio du, baita erabiltzaileak aukeratu dituen ezaugarriak irakurtzeko ere.
    - Sarrera:
        - *option*: Parametro honek lau balio posible ditu.
            - 0: *featuresToDeselect.txt* irakurtzeko.
            - 1: *proposedProductFile.txt* irakurtzeko.
            - 2: *selectedFeaturesLocal.txt* irakurtzeko.
            - 3: *featuresOnEvery.txt* irakurtzeko.
    - Irteera: Aukeratutako fitxategian zeuden ezaugarri guztiak dituen *array* bat itzultzen du (*arrayOfFeatures*).
- *saveSelectedFeatures: function(checkedFeatures){..}:*
    - Deskribapena: Funtzio honek lokalean gordetzen ditu erabiltzaileak konfigurazioan aukeratutako ezaugarriak.
    - Sarrera:
        - *checkedFeatures*: String bat non erabiltzaileak aukeratutako ezaugarriak dauden.
    - Irteera: Fama direktorioan *selectedFeaturesLocal.txt* fitxategia sortzen du pasatako ezaugarri multzoarekin.
- *validProduct: function(option){..}:*
    - Deskribapena: Funtzio honek ebatzailea exekutatzen duten script-ak exekutatzen ditu.
    - Sarrera:
        - *option*: Parametro honek bi balio posible ditu.
            - 1: *famaCore.sh* edo *famaCore.bat* script-a exekutatzen du.
            - 2: *famaValid.sh* edo *famaValid.bat* script-a exekutatzen du.
    - Irteera: *option* parametroaren arabera irteera ezberdina izango da.
        - 1: Fama izeneko direktorioan *featuresOnEvery.txt* fitxategia sortzen du ebatzaileak emandako erantzunarekin.
        - 2: Fama izeneko direktorioan *isValid.txt, featuresToDeselect.txt eta proposedProductFile.txt* fitxategiak sortzen ditu ebatzaileak emandako erantzunarekin.

- *readFileIsValid: function(){..}:*
    - Deskribapena: Funtzio honek Fama izeneko direktorioan dagoen *isValid.txt* fitxategia irakurtzen du.
    - Sarrera: ez dauka sarrerako parametrorik.
    - Irteera: *isValid.txt* fitxategiaren edukia itzultzen du String batean.

### 6.1.4 Funtzio batzuen zehaztapenak

*6.1.2 Inplementazioa* atalean aurkeztutako funtzio batzuen funtzionamendua hobeto ulertzeko, atal honetan sakonago azaltzen dira.

- *DeltaUtils.createConfigurator=function(option, kind){…}*
  - Deskribapena: Funtzio honetan *balioztaketan* erabiltzen diren interfaze guztien html kodea sortzen da.
  - Sarrera:
    - *kind*: funtzionalitate honetan ez da erabiltzen.
    - *option*: parametro honek hiru balio posible ditu.
      - 0: Konfiguradorearen lehenengo interfazearen (ezaugarri guztiak listaratuak) html kodea sortzen du.
      - 1: Erabiltzaileak baliozkoa den konfigurazio bat balioztatzerakoan agertuko den interfazearen html kodea sortzen du.
      - 2: Erabiltzaileak baliozkoa ez den konfigurazio bat balioztatzerakoan agertuko den interfazearen html kodea sortzen du.
  - Irteera: Pantailaratu nahi den interfazearen html kodea itzultzen du.
- *DeltaUtils.createProduct=function(option){…}*
  - Deskribapena: Funtzio honek erabiltzaileak aukeratutako behin betiko konfigurazioarekin biltegi berria sortuko duen funtzioa exekutatzen du.
  - Sarrera:
    - Option: parametro honek bi balio posible ditu.
      - 1: Erabiltzaileak ebatzaileak proposatutako konfigurazioa aukeratzen duenean, ezaugarriak *proposedProductFile*.*txt* fitxategitik lortuko dira.
      - 2: Erabiltzaileak bere konfigurazioa aukeratzen duenean, ezaugarriak selectedFeaturesLocal.txt fitxategitik lortuko dira.
  - Irteera: -

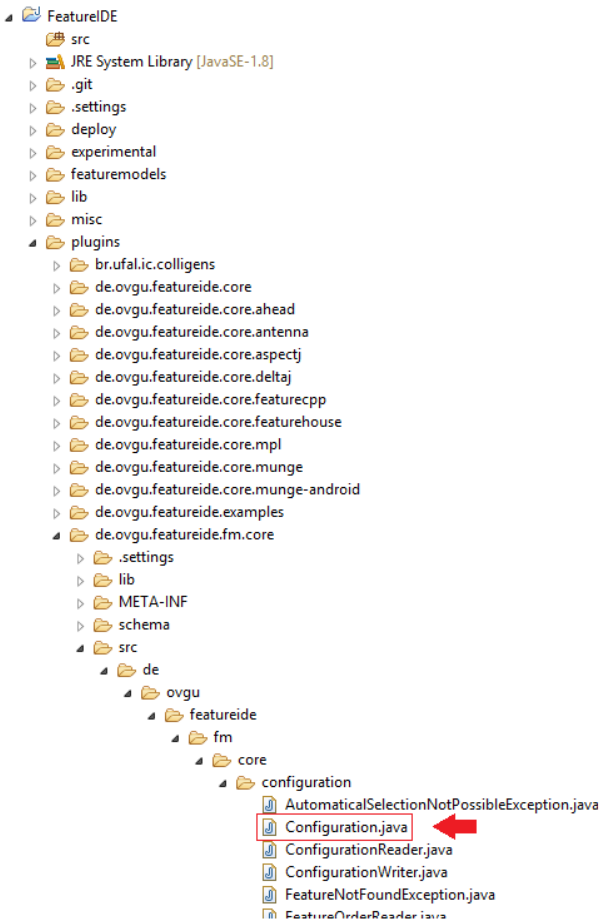
## 6.2 Garapenaren inguruko erronkak

Funtzionalitate hau inplementatzerakoan bi arazo nagusi egon dira: FeatureIDE plugin-ean oinarrituta, konfigurazioak balioztatzea, eta behin ebatzailea javan inplementatua zegoela, GitLine-ekin elkartzea.

### 6.2.1 Konfigurazioak balioztatu

Hasierako irismenean, konfigurazioak balioztatzea FeatureIDE plugin-ean oinarrituta egitea finkatu zen. Horretarako, ikasleak FeatureIDE plugin-a aztertu zuen. Azterketa horretan FeatureIDE-ren bi bertsio ezberdin aztertu zituen: FeatureIDE v2.7.1 (2014ko uztaila) eta FeatureIDE v.6.6 (2013ko iraila).

FeatureIDE-k dituen plugin ezberdinetatik konfigurazioen balioztaketaz arduratzen dena *de.ovgu.featureide.fm.core* da. Plugin horren direktorioaren barnean, /src/de/ovgu/featureide/fm/core/Configuration.java fitxategia aurkitzen da 42. irudian ikus daitekeen moduan. Fitxategi horretan, konfigurazio bat balioztatzeaz arduratzen den funtzioa aurkitzen da *isValid*() v2.7.2 bertsioan eta *valid*() v.6.6 bertsioan.




42. irudia: FeatureIDE-ren kodearen hierarkia.

Bi funtzioak era desberdinean inplementatuak daude, baina objektu berdinak erabiltzen dituzte:

- *features: ArrayList<SelectableFeatures>* → Array honetatik konfigurazioan egon daitezkeen ezaugarri guztiak lortzen dira.
- *children: LinkedList<Node>* → LinkedList honetan, konfigurazioan aukeratuta dauden ezaugarriak gordetzen dira *Literal* objektu moduan.
- *rootNode: Node* → ezaugarri ereduaren erroa da.

Objektuak behar moduan landu ondoren eta ebatzaileari pasa beharreko informazioa lortu ondoren, ebatzaileari deitzen zaio. Dei hori bi bertsioetan era desberdinean egiten da.

- v2.7.1 → *SatSolver(new And(allFeatures), TIMEOUT).isSatisfiable();*
- v.6.6 → *SatSolver( rootNode, TIMEOUT).isSatisfiable(children);*

Arazoa klase ezberdinen arteko erlazioak aztertzerakoan sortu zen. *Configuration* klasea erabili ahal izateko *SelectableFeature*, *Node*, *FeatureModel* , *And*, *Literal* eta *SatSolver* erabili behar ziren, baina horietako klase bakoitza erabiltzeko, beste hainbat klase erabili behar ziren 13. taulan ikus daitekeen moduan, hiru maila bakarrik sakonduta 27 klase ezberdinek parte hartzen dute. Beraz, balioztaketan parte hartzen duten klase guztiak kudeatzea denbora kostu handia suposatzen zuen.

| | | | | | |
|---|---|---|---|---|---|
| **Configuration** | SelectableFeatures | Selection | - | | |
| | | Feature | ColorList | | |
| | | | Constraint | FeatureModel | ... |
| | | | | Node | ... |
| | | | | FMPoint | ... |
| | | | | Feature | ... |
| | | | | ConstraintAtribute | .. |
| | | | FeatureStatus | ... | ... |
| | | | FMPoint | ... | .. |
| | | Configuration | ... | ... | ... |
| | Node | - | | | |
| | FeatureModel | Feature | ... | ... | ... |
| | | Constraint | ... | ... | .. |
| | | FeatureModelAnalyzer | Feature | ... | ... |
| | | | FeatureModel | ... | .. |
| | | | IProgressMonitor | ... | .. |
| | | | FeatureDependencies | ... | ... |
| | | RemainingManager | Renaming | ... | .. |
| | | | Ifolder | ... | ... |
| | | FeatureModelLayout | FMPoint | ... | .. |
| | | ColorschemaTable | FeatureModel | ... | ... |
| | | FMComposerName | Iproject | ... | .. |
| | | | QualifiedName | ... | ... |
| | SatSolver | Isolver | ... | ... | .. |
| | And | - | | | |
| | Literal | Object | ... | ... | ... |

13. taula: Klaseen arteko erlazioak.

FeatureIDE plugin-a aztertzeko plangintzan kalkulatu zen denbora 50 ordukoa zen eta bakarrik kodea aztertzeko eta 13. taulan agertzen den taula lortzeko 43 ordu inbertitu ziren. Agian, kodea aztertzeko erremintaren bat erabili izan bazen eskuz egindako azterketa horren denbora kostua eta klase guzti horiek kudeatzeko denbora kostua ezberdina izango litzatekeen. Horietako erreminta bat *JDepend* [10] izan daiteke, erreminta honek Java aplikazio batean parte hartzen duten paketeen azterketa egiten du, beraien arteko dependentziak eta berrerabilera zehaztuz. Baina, ikasleak nahiz eta software espezialitatean jardun, ez du mota honetako erreminten berri izan ikasketetan zehar eta ez zuen aukera horretan pentsatu ere.

Egoera horretan, FeatureIDE-rekin aurrera jarraitzeko denbora, beste erreminta bat bilatu eta horrekin aurrera jarraitzeko denbora baino altuagoa zela kalkulatu zenez, beste baliabide bat bilatzeko aukera hautatu zen.

FeatureIDE alde batera utzita, konfigurazioak balioztatzeko beste baliabide batzuk aurkitu zituen. Alde batetik *guidsl* [14] erreminta, erreminta honek *.cnf* edo *.m* luzapenak dituzten ezaugarri ereduetan oinarrituta konfigurazioak balioztatzen ditu. Beste aldetik FaMa [2] erreminta, *.xml* edo *.fm* luzapenak dituzten ezaugarri ereduetan oinarrituta konfigurazioak balioztatzen ditu.

Bi erreminten artean, FaMa erreminta aukeratu zuen balioztaketa aurrera eramateko. Erabaki hau bi puntutan oinarritzen da:

- Nahiz eta bi erremintak FeatureIDE-n ezaugarri ereduak adierazteko era desberdinak izan, FaMa-ren *.xml* fitxategiak hurbilago daude eta ikasketa errazagoa da.
- FaMa erremintarekin konfigurazioa baliozkoa den ala ez jakiteaz gain, informazio gehiago eskura daiteke (kendu beharreko ezaugarriak, proposatutako ezaugarriak, derrigorrezko ezaugarriak…).

### 6.2.2 Ebatzailea GitLine-ekin elkartu

FaMa berrerabiltzen duen ebatzailea inplementaturik, ebatzailea GitLine-rekin lotzeko hiru aukera aztertu dira:

- Lehenengo aukera:
    - Deskribapena: Aukera honetan, interfazea eta ebatzailea jar-aren barnean garatuta daude eta applet-aren bidez *“java –jar Balioztapena.jar… “* exekutatzen da.
    - Abantaila: konfiguradorearen atala ondo dabil.
    - Desabantaila: *Balioztapena.jar* exekutatzerakoan, GitLine-ek ez daki erabiltzaileak noiz bukatzen duen bere konfigurazioa aukeratzeaz, erabat independente exekutatzen den aplikazio bat delako. Beste modu batera esanda, GitLine-etik *Balioztapena.jar*-erako *zubia* ondo dabil (GitLine-etik

*Balioztapena.jar* exekutatzen da), baina *Balioztapena.jar*-etik GitLine-era ez dago zubirik, GitLine-ek ezin du *Balioztapena.jar* sortzen duen informazioa irakurri.

- Bigarren aukera:
  - Deskribapena: Aukera honetan, interfazea eta ebatzailea applet-ean garatuta daude.
  - Abantaila: konfiguradorearen atala ondo dabil.
  - Desabantaila: Applet-a exekutatzerakoan, erabat independentea den firefox leihoa irekitzen da. GitLine-ek ez daki erabiltzaileak noiz bukatzen duen bere konfigurazioa aukeratzeaz eta gainera GitLine-etik "*kanpo*" zaudela dirudi.
- Hirugarren aukera:
  - Deskribapena: Aukera honetan interfazea html lengoaian garatuta dago (beste bi kasuetan, java lengoaian) eta ebatzailea applet-ean.
  - Abantaila: aukerarik integratuena.
  - Desabantaila: ebatzaileari deitzerakoan baimen arazoak daude. Firefox nabigatzailetik jar-ean dauden FaMa liburutegiak atzitzeko baimenik ez.

Ikerlariak hirugarren aukera hautatu zuen aukera integratuena zelako, baina ebatzaileari deitzeko baimen arazoak egun batzuetan konpontzen ez baziren alde batera uztea adostu zen.

Azkenean, hirugarren aukera baztertu eta beste aukera berri bat hautatu zen, interfazea GitLine-en garatu zen. Ebatzailearen kasuan, aurretik garatutako java aplikazioa mantendu zen, baina interfazea alde batera utzita, hiru mailako arkitektura jarraituz inplementatuta dagoenez, interfazea kentzea eta *main* berri bat sortzeak ez zuen debora kostu handia suposatu.

# 7. KAPITULUA

# Ezaugarria txertatu eta hedatu

Proiektu honetan garatutako *ezaugarria txertatu* eta *ezaugarria hedatu* funtzionalitateak sakon aztertuko dira, bere diseinua, inplementazioa eta interfazeak azalduz. Gainera 7.3 atalean garapenaren inguruko erronkak laburbildu dira.

Bi funtzionalitate hauek erlazio handia dute, ezin da ezaugarri bat hedatu ezaugarri berri bat txertatu gabe. Beraz, lehenengo *ezaugarria txertatu* azalduko da, ondoren *ezaugarria hedatu* eta bukatzeko garapenean zehar izandako gorabeherak.

*Ezaugarria txertatu* eta *ezaugarria hedatu* bata bestearen ondoren egin behar direnez baina ez modu jarrai batean (ezaugarria hedatu baino lehen domeinuko ingeniariak ezaugarri berriari dagokion kodea inplementatu behar du), ezaugarri biltegian jakinarazpen (issue) bat sortzen da domeinuaren ingeniariari hedatu gabeko ezaugarri bat duela gogorarazteko. Jakinarazpenak GitHub biltegien atal bat da, bertan biltegiaren inguruko oharrak pilatzen dira. Txertatzen den ezaugarri bakoitzeko, jakinarazpen berri bat sortzen da ezaugarri biltegian. Jakinarazpen hauek domeinuaren ingeniariari prozesua bukatu gabe dagoela gogoratzeaz gain, hedatu behar den ezaugarriaren informazioa gordetzeko balio du. Ezaugarri bat txertatzerakoan sortzen den jakinarazpenaren izenburua eredu bera jarraitzen du *New_(ezaugarri mota)_feature_of_(gurasoa)_(izena)*, txertatu den ezaugarriaren informazioa hedaketa egiten den momentura arte gordetzeko.

## 7.1 Ezaugarria txertatu

*Ezaugarria txertatu* funtzionalitateari dagokion informazio guztia ondorengo ataletan azaltzen da.

Ezaugarri biltegien bizi-zikloa handitzeko asmoarekin, ezaugarri biltegietan ezaugarri berri bat gehitzeko aukera inplementatu da. Ezaugarri biltegian dagoen ezaugarri eredua eguneratzen da eta ezaugarri berriari dagokion adar berria sortzen da domeinuko ingeniariak ezaugarri berriari dagokion kodea gehitzeko.

Funtzionalitate hau inplementatzeko 7. kapituluan azaldutako ebatzailea erabili da, jatorrizko ezaugarri ereduari ezaugarri berria txertatu ondoren sortutako ezaugarri eredua baliozko dela ziurtatzeko.

43. irudian ezaugarria txertatu funtzionalitatearen arkitektura ikus daiteke. Prozesua *gitHubDeltas* moduluan hasten da, bertan ezaugarri ereduan dagoen ezaugarri eredua eta ezaugarri berriaren informazioa (mota, gurasoa eta izena) lortzen dira. Ondoren, modulu horretan lortutako informazioa *script-compiler* modulura pasatzen da, modulu honek jasotako informazioarekin ebatzailea exekutatuko duen script-a exekutatzeaz arduratzen da (*famaInsert.bat*). Script-ak ebatzailea exekutatzen du. Ebatzaileak ezaugarri berria jatorrizko ezaugarri ereduan txertatzen du, sortutako ezaugarri eredu berria baliozkoa dela ziurtatzen du eta sortutako ezaugarri eredu berria *modelLocalInsertChange.xml* fitxategian gordetzen du. Bukatzeko, *gitHubDeltas* moduluak ezaugarri biltegiaren ezaugarri eredua eguneratzeaz arduratzen da.

Hasteko, diseinua azalduko da, ondoren, inplementazioa eta bukatzeko inplementazioan erabilitako funtzio batzuen azalpenak.

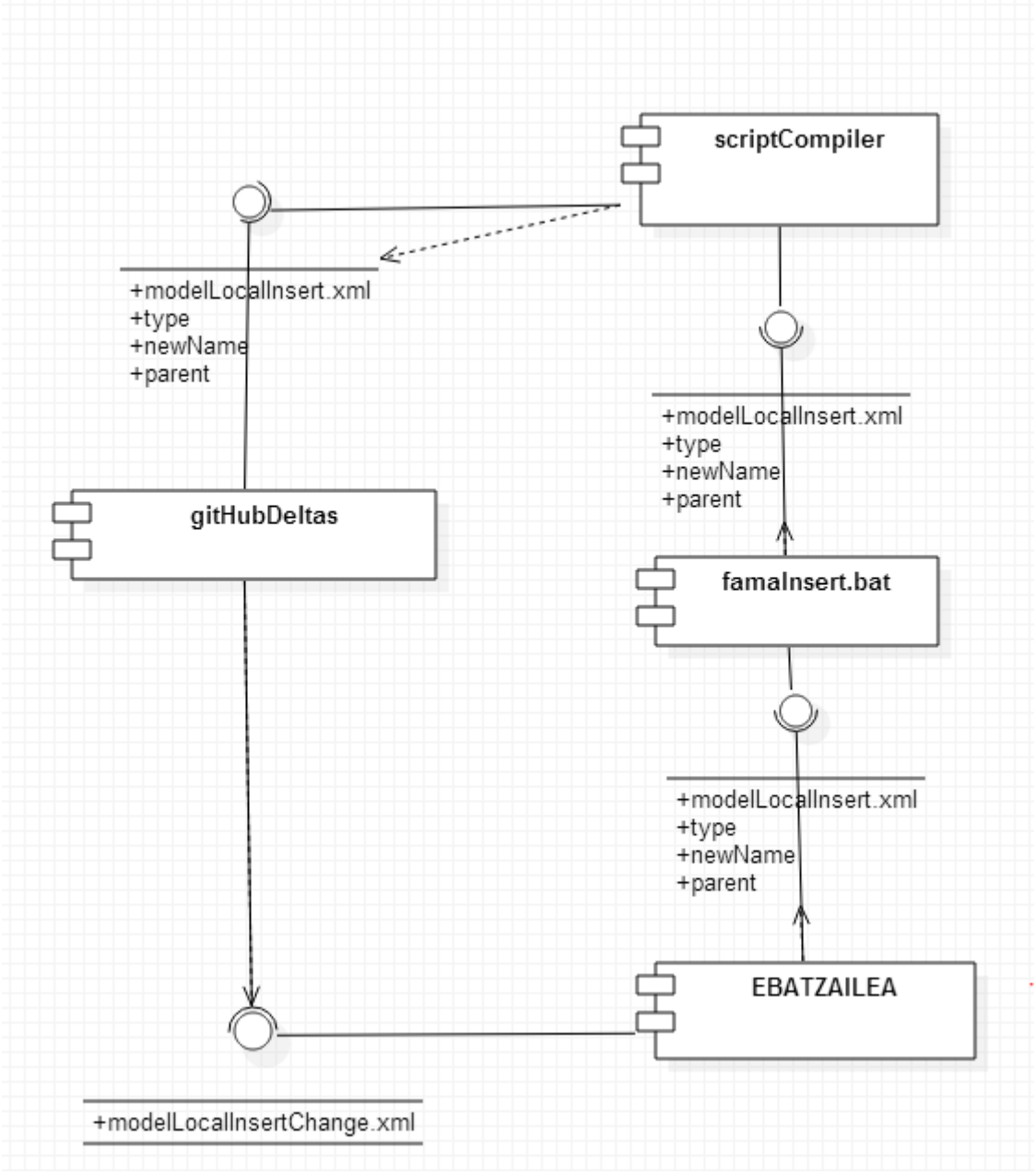
### 7.1.1 Diseinua

15. taulan *ezaugarri txertatu* funtzionalitatearen diagrama ikus daiteke. Taula bi zutabe nagusietan banatuta dago: bista eta logika. Bista zutabean, GitLine-en *gitHubDeltas* fitxategian gertatzen dena azaltzen da, eta logika zutabea beste bi zutabeetan bereizten da: alde batetik, GitLine-en *script-compiler* fitxategian gertatzen dena azaltzen duena eta bestetik ebatzailea exekutatzeko beharreko *input*-ak eta lortutako *output*-ak azaltzen dituena. Bigarren zutabean agertzen diren funtzioen azalpena 7.*1.3-Funtzio gehigarriak* atalean aurki daiteke.

### 7.1.2 Inplementazioa

Funtzionalitate hau inplementatzeko *JavaScript* [11], *mozilla developer* [12] eta GitHub plugina [7] erabili dira.

Interfazean agertzen den *InsertFeature* botoia sakatzerakoan (14. taularen 1. pausoa), ondorengo pausuak jarraitzen dira:




43. irudia: *Ezaugarria txertatu*-ren arkitektura.

| BISTA | LOGIKA | |
|---|---|---|
| *gitHubDeltas* | *script-compiler* | Ebatzailea |
| (1) *"InsertFeature "* botoia sakatu.<br>(2) Ezaugarri mota aukeratzeko bista pantailaratu (2. irudia).<br>(3) Erabiltzaileak txertatu nahi duen ezaugarri berriaren mota aukeratu eta *"Accept"* sakatzen du.<br>(4) Bi aukera (3) pausoan aukeratutakoaren arabera:<br>- *mandatory* edo *optional* aukeratu bada → ezaugarri ereduko ezaugarri guztiak lortu.<br>- *alternative* aukeratu bada → *or-inclusive* egitura baten gurasoak diren ezaugarriak lortu.<br><br>(5) Ezaugarri eredua (*model.xml*) lokalean gorde, *modelLocalInsert.xml* fitxategian.<br>(6) Ezaugarri berriaren gurasoa aukeratzeko bista pantailaratu (3. irudia).<br>(7) Erabiltzaileak txertatu nahi duen ezaugarri berriaren guraso aukeratu eta *"Accept"* sakatzen du.<br>(8) Ezaugarri berriaren izena eskatzen duen leihoa pantailaratzen da (4. Irudia). | | |

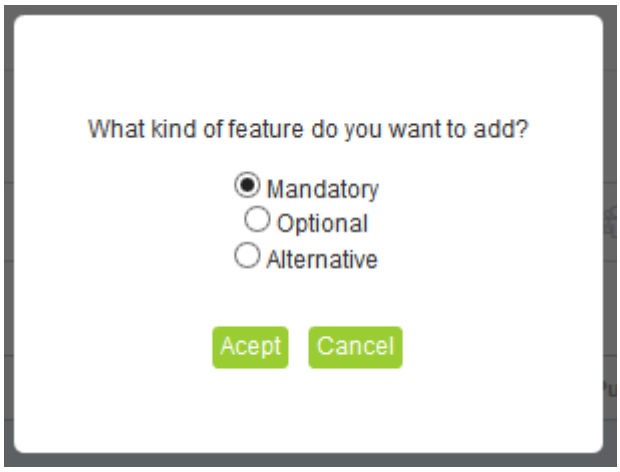
14. taula: Ezaugarri bat txertatzeko diagrama.

| BISTA | LOGIKA | |
|---|---|---|
| *gitHubDeltas* | *script-compiler* | BalioztapenaVCMOA2 |
| (9) Ezaugarri berriaren informazio guztia (mota, izena eta gurasoa) jasotzen duen konfirmazio leihoa pantailaratu (5. irudia).<br>(10) Erabiltzaileak *"Accept"* sakatuko du. | | |
| (11)Lokalean gordetako ezaugarri ereduari (*modelLocalInsert.xml*) ezaugarri berria txertatu eta eredu berria (*modelLocalInsertChange.xml*) balioztatu. | insertFeature(parent, name, kind) | Input:<br>·*mandatory/optional/alternative*<br>·*modelLocalInsert.xml*<br>·*parent*<br>·*name*<br>Output:<br>·*modelLocalInsertChange.xml* |
| (12)Ezaugarri berriari dagokion *adar* berria sortu.<br>(13) Ezaugarri biltegian dagoen *model.xml* eguneratu.<br>(14) Ezaugarri berriari dagokion jakinarazpena (*issue*) sortu (6. irudia) . | | |

15. taula: Ezaugarri bat txertatzeko diagrama (jarraipena.

1. Pausoa: Ezaugarri mota aukeratzeko interfazea sortu.

- *DeltaUtils.interfaceOfInsertFeature=function(insertoption){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Sortu beharreko interfazearen html kodea *String* batean gordetzen da.
    - *UI.Dialog.show_insertFeatureInterface* funtzioari deitzen zaio sortutako html kodeari botoiak gehitzeko eta pantailaratzeko.
- *UI.Dialog.show_insertFeatureInterface* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
    - Bukatzeko, 44. irudian ikus daitekeen interfazea pantailaratzen da (14. taularen 2. pausoa).



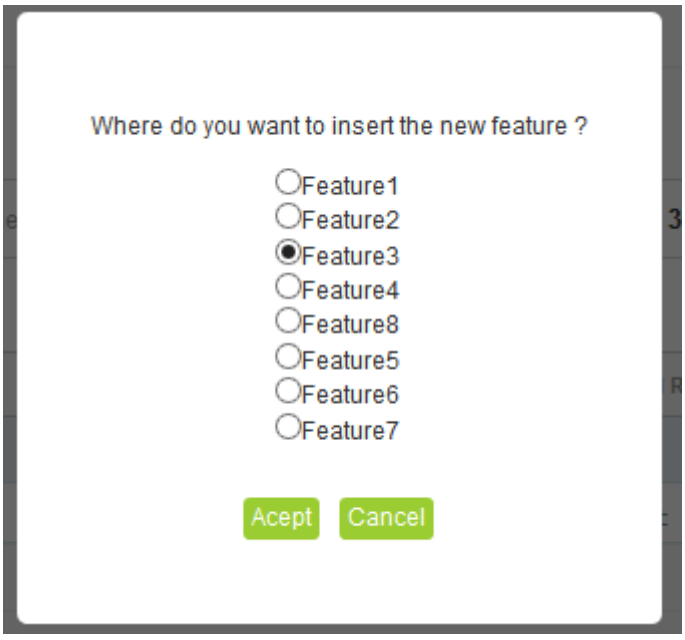

44. irudia: Ezaugarri mota aukeratzeko interfazea.

2. Pausoa: Gurasoa aukeratzeko interfazea sortu.

- *DeltaUtils.selectedInsert=function(docu,phase,allFeatures, kind){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Aurreko pausoan derrigorrezko edo aukerakoa aukeratu bada, ezaugarri guztiak erakutsiko dituen interfazeari deitzen zaio (14. taularen 4. pausoa) → DeltaUtils.*createConfigurator(3,checkedOption)*
    - Aurreko pausoan aldakia aukeratu bada, egitura hori duten ezaugarriak bakarrik erakutsiko dituen interfazeari deitzen zaio (14. taularen 4. pausoa)→ *DeltaUtils.createConfigurator(4,checkedOption)*
- *DeltaUtils.createConfigurator=function(option, kind){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *ghAuthorRepo.fetch* → biltegia atzitzen du.

- *ghAuthorRepo.fetchBranches* → biltegiaren adar guztiak atzitzen ditu.
- *ghAuthorRepo.getBranchByName("master")* → *master* izeneko adarra atzitzen du.
- *master.fetchContents* → *master* adarrean dauden fitxategi eta direktorioak atzitzen ditu
- *master.getFileByName("model.xml")* → *model.xml* fitxategia lortzen du.
- *featureModelFile.fetchContent model.xml* → fitxategiaren edukia lortzen du.
- *saveFeatureModel(xml,2)* → *model.xml* fitxategia lokalean gordetzen da (14. taularen 5. pausoa).
- Ondoren, egoeraren arabera pantailaratu beharreko ezaugarriekin html kodea sortuko da.
- *UI.Dialog.show_insertFeatureInterface* funtzioari deitzen zaio sortutako html kodeari botoiak gehitzeko eta pantailaratzeko.

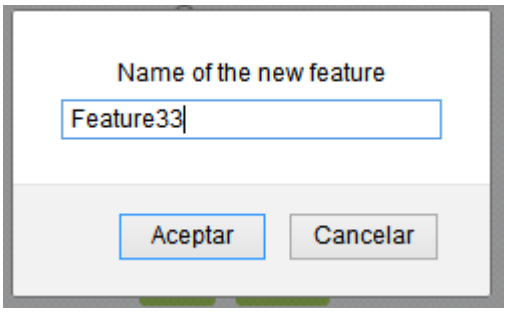
- *UI.Dialog.show_insertFeatureInterfaze* funtzioa exekutatzen da eta ondorengoa gertatzen da:
  - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
  - Bukatzeko 45. irudian ikus daitekeen interfazea pantailaratzen da (14. taularen 6. pausoa).




45. irudia: Gurasoa aukeratzeko interfazea.

### 3. Pausoa: Izena zehaztu eta baliozkoa dela ziurtatu.

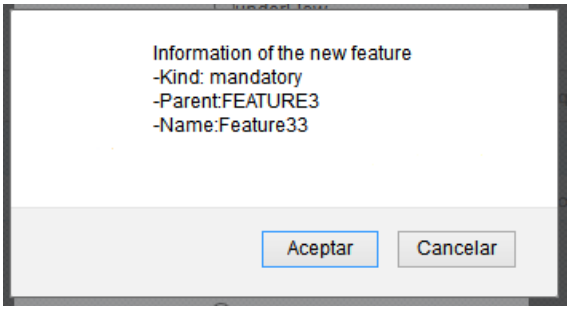
- *DeltaUtils.validNameOfNewFeature=function(allFeatures,checkedOption,kind){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *window.prompt* baten bidez ezaugarri berriaren izena zehazteko eskatzen zaio erabiltzaileari (46. irudia) (14. taularen 8. pausoa).
    - Jasotako izenari zuriuneak kentzen zaizkio.
    - Izena ezaugarri ereduan jada ez dela existitzen ziurtatzen da. Ez bada existitzen, ezaugarria sortzeko prozesua hasten da.




46. irudia: Izen berria finkatzeko leihoa.

### 4.Pausoa: Ezaugarri ereduan ezaugarria txertatu eta balioztatu.

- *DeltaUtils.createFeature=function(checkedOption,kind,newName){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - 47. irudian ikus daitekeen konfirmazio leihoa agertzen da (15. taularen 9. pausoa) eta *Acept* sakatzerakoan (15. taularen 10. pausoa), ondorengo pausoekin jarraitzen da.
    - Ebatzailea exekutatuko duen funtzioari deia (15. taularen 11. pausoa) → *insertFeature(checkedOption, newName,kindOption)*
    - Txertaketa ondo burutzen bada *adar* berria sortzeko funtzioa exekutatzen da (15. taularen 12. pausoa).




47. irudia: Konfirmazio leihoa.

5.Pausoa: Adar berria sortu.

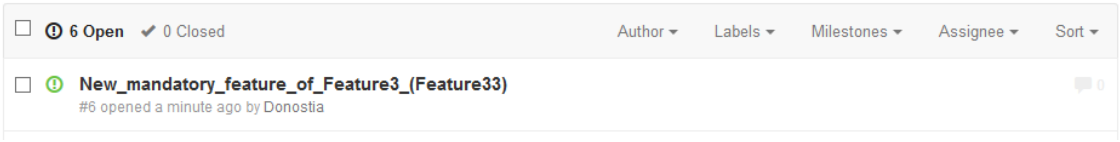
- *DeltaUtils.createBranch=function(parent, newBranchName,user,repo,token,f){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - GitHub-en *adarrak* sortzen dituzten GET eta POST deiak simulatzen dira.

6.Pausoa: Ezaugarri biltegian dagoen ezaugarri eredua eguneratu.

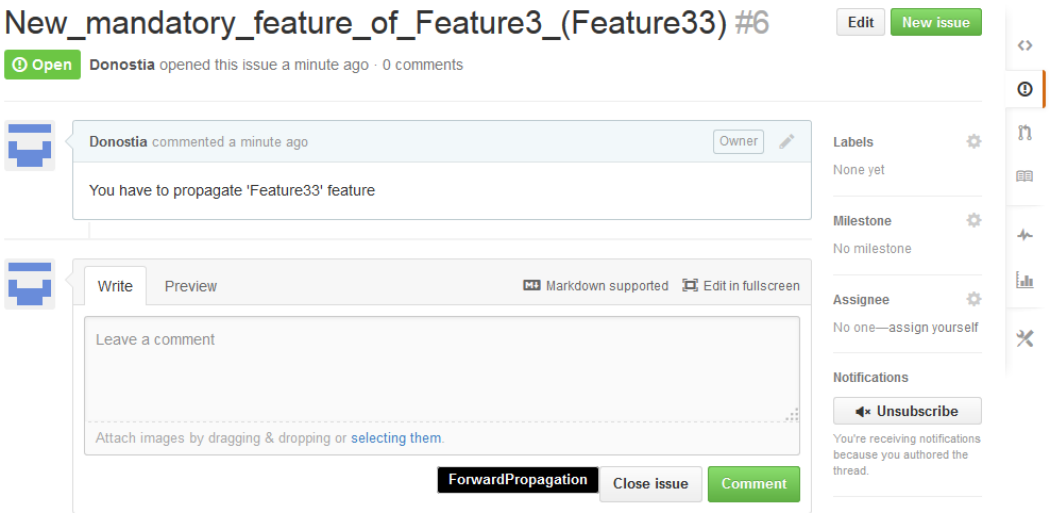
- DeltaUtils.editModelFile=function(kind,insertValid,newName,checkedOption){} funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *ghAuthorRepo.fetch* → biltegia atzitzen du.
    - *ghAuthorRepo.fetchBranches* → biltegiaren adar guztiak atzitzen ditu.
    - *ghAuthorRepo.getBranchByName("master")* → *master* izeneko adarra atzitzen du.
    - *master.fetchCommits* → *master* adarrean *commit* guztiak atzitzen ditu.
    - master.getLastCommit() → master adarraren azkeneko *commit*-a atzitzen du.
    - *DeltaUtils.editFile(user, repo, "master","model.xml",commit, token, insertValid, "new model.xml",null);* → *model.xml* fitxategia *insertValid* (ezaugarri eredu berria) aldagaian dagoen balioarekin eguneratzen da (15. taularen 13. pausoa).

7.Pausoa: Jakinarazpena sortu.

- *DeltaUtils.createIssue=function(newName,body,checkedOption,kind){}* funtzioa exekutatzen da eta ondorengoa gertatzen da.
    - Zehaztutako eredua jarraitzen duen izenburua zehazten zaio jakinarazpenari: *New_(ezaugarri mota)_feature_of_(gurasoa)_(izena)*.
    - GitHub-en *jakinarazpenak* sortzen dituzten GET eta POST deiak simulatzen dira.
    - Biltegian, 48. irudian eta 49. irudian agertzen den moduan, jakinarazpen berria sortzen da (15. taularen 14. pausoa).




48. irudia: Jakinarazpen lista.




49. irudia: Jakinarazpen bat aukeratuta.

### 7.1.3 Funtzio gehigarriak

*7.1.1 Diseinua* atalean aurkeztutako tauletan, *script-compiler* zutabean agertzen diren funtzioak atal honetan sakonago azaltzen dira.

- *saveFeatureModel: function(featureModel, option){…}:*
  - Deskribapena: Funtzio honek lokalean gordetzen du ezaugarri eredu bat.
  - Sarrera:
    - *featureModel*: lokalean gorde nahi den ezaugarri eredua, String batean.
    - *option*: parametro honek bi balio posible ditu.
      - 1: Ezaugarri eredua lokalean gordetzen da, balioztaketa prozesuan atzitzeko.
      - 2: Ezaugarri eredua lokalean gordetzen da, ondoren ezaugarri berri bat txertatzeko.
  - Irteera: *option* parametroaren arabera irteera ezberdina izango da.
    - 1: Fama direktorioan *modelLocal.xml* fitxategia sortzen du pasatako ezaugarri ereduarekin.
    - 2: Fama direktorioan *modelLocalInsert.xml* fitxategia sortzen du pasatako ezaugarri ereduarekin.
- *insertFeature: function(parent, newFeature, option){..}:*
  - Deskribapena: Funtzio honek, ezaugarri bat txertatzeko, ebatzailea exekutatzen duten script-ak exekutatzen ditu.
  - Sarrera:
    - *parent*: Ezaugarri berriaren gurasoa izango den ezaugarriaren izena.
    - *newFeature*: Ezaugarri berriaren izena.
    - *option*: Parametro honek hiru balio posible ditu.

- 0: *famaMandatory.sh* edo *famaMandatory.bat* script-a exekutatzen du.
- 1: *famaOptional.sh* edo *famaMandatory.bat* script-a exekutatzen du.
- 2: *famaAlternative.sh* edo *famaAlternative.bat* script-a exekutatzen du.

  - Irteera: Balioztatua dagoen ezaugarri eredu berria String batean itzultzen du.

### 7.1.4 Funtzio batzuen zehaztapenak

*7.1.2 Inplementazioa* atalean aurkeztutako funtzio batzuen funtzionamendua hobeto ulertzeko, atal honetan sakonago azaltzen dira.

- *DeltaUtils.createConfigurator=function(option, kind){…}*
  - Deskribapena: Funtzio honek ezaugarri berriaren gurasoa aukeratzeko interfazearen html kodea sortzen du.
  - Sarrera:
    - *kind*: Txertatu nahi den ezaugarri berriaren mota.
    - *option*: parametro honek hiru balio posible ditu.
      - 3:Ezaugarri ereduko ezaugarri guztiak listaratzen ditu (ezaugarri berria mandatory edo optional delako).
      - 4:Bakarrik or-inklusiboa egitura duten ezaugarriak listaratzen ditu (ezaugarri berria alternative delako).
  - Irteera: Pantailaratu nahi den interfazearen html kodea itzultzen du.

## 7.2 Ezaugarri berria hedatu

Atal honetan, *ezaugarria hedatu* funtzionalitateari dagokion inplementazioa azalduko da. Ezaugarri bat hedatu ahal izateko, beharrezkoa da aurretik ezaugarri berri bat txertatu izana.

Ezaugarri biltegien eta produktu biltegien arteko sinkronizazioa bermatzeko, ezaugarri biltegi batean ezaugarri berri bat gehitu eta inplementatu ondoren, ezaugarri biltegi horretatik sortutako produktu biltegietara hedatzeko aukera inplementatu da *ForwardPropagation* funtzioaren laguntzarekin.

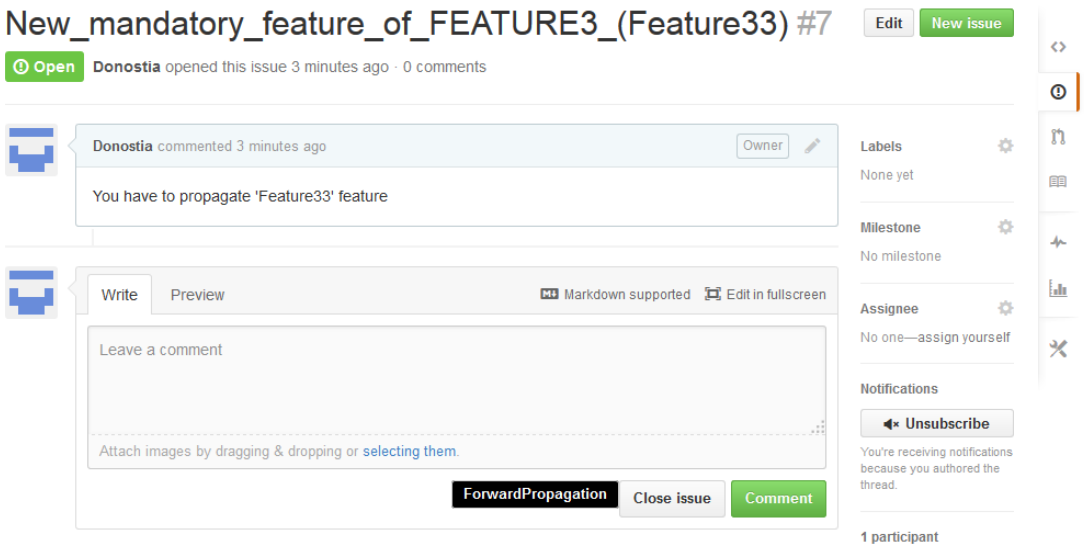
Ezaugarri bat hedatzeko, ezaugarri biltegi horietatik sortutako produktu biltegi guztiak (fork-ak) aztertu behar dira. Ezaugarri berriaren gurasoa inplementatua duten produktu biltegietara

bakarrik hedatzeko aukera eman behar zaio domeinuko ingeniariari. Horretarako, produktu biltegi bakoitzaren *product.config* fitxategia aztertu behar da.

### 7.2.1 Inplementazioa

Funtzionalitate hau inplementatzeko *JavaScript* [11], *mozilla developer* [12] eta GitHub plugina [7] erabili dira.

Hedatu nahi den ezaugarriaren jakinarazpena ireki eta 50. irudian ikus daitekeen *ForwardPropagation* botoia sakatzerakoan, ondorengo pausoak jarraitzen dira:




50. irudia: Jakinarazpen bat aukeratu ondoren.

**1. Pausoa: Jakinarazpenaren izenburutik beharreko informazioa lortu.**

- *DeltaUtils.selectedCheckIssue=function(docu,title){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
  - *ghUserRepo.fetch* → ezaugarri biltegia atzitzen du.
  - *ghUserRepo.fetchIssues* → bitegiaren jakinarazpen guztiak atzitu.
  - *var issues= ghUserRepo.getIssueByNumber(number);* → interfazetik lortutako zenbakiaren bidez (number) irekita dagoen jakinazpena lortu.
  - Izenburua osatzen duen String-a kudeatu beharreko informazioa jasotzeko: ezaugarri mota, ezaugarri berriaren gurasoa eta ezaugarri berriaren izena.

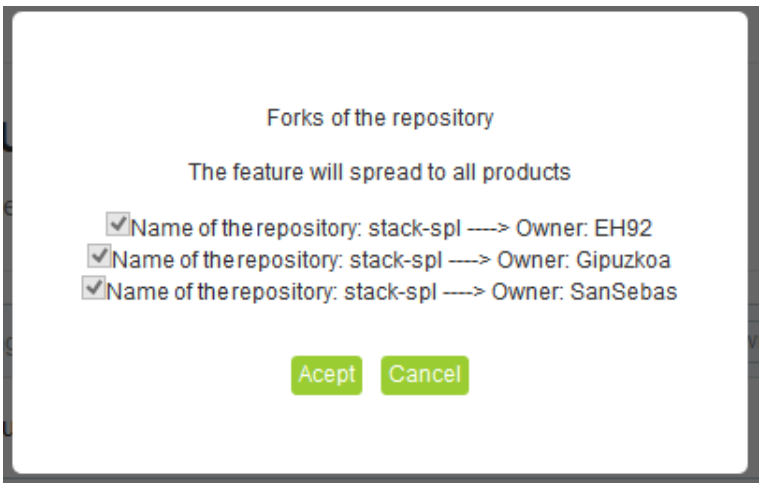
2. Pausoa: Ezaugarri biltegiaren produktu biltegietan (fork-etan) ezaugarri berriaren gurasoa inplementatuta dagoen ala ez begiratu, informazio hori produktu biltegien master adarrean dagoen *product.config* fitxategia aztertzen lortzen da. Inplementatua dagoen kasuetan, produktu biltegia pantailaratu.

- *DeltaUtils.createConfiguratorForPropagation=function(kind,parent,newFeature,forksWithParent,issue){}* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - *ghUserRepo.fetch* →ezaugarri biltegia atzitzen du.
    - *ghUserRepo.fetchFork* → ezaugarri biltegiaren produktu biltegiak (fork-ak) atzitzen ditu.
    - *var Forks= ghUserRepo.getForks();* → Forks aldagaian, fork guztiak gordetzen ditu.
    - Forks-en neurria 0 bada → ezaugarri biltegiak fork-erik ez dituela adierazten du.
    - Forks-en neurria 0 baino handiago bada → fork bakoitzaren *product.config* fitxategia aztertzen da.
- *DeltaUtils.readProductConfig=function(Forks,parent,configString,kind,kont,newFeature, forksWithParent,issue){}* funtzioa errekurtsiboa da, Forks aldagaian dauden fork guztiak aztertzeko, funtzio hau exekutatzerakoan ondorengoa gertatzen da:
    - *fork.fetch* → uneko produktu biltegia atzitzen du.
    - *fork.fetchBranches* → uneko produktu biltegiaren adarrak atzitzen ditu.
    - *var master=fork.getBranchByName("master");* → master adarra atzitu.
    - *master.fetchContents*→ master adarraren edukia atzitu.
    - *var productConfig = master.getFileByName("product.config");* → *product.config* fitxategia atzitu.
    - *productConfig.fetchContent* → *produc.config* fitxategiaren edukia lortu.
    - *product.config* fitxategia aztertzen da, ea ezaugarri berriaren gurasoaren izena agertzen den. Izena agertzen bada, produktu biltegia interfazera gehitzen da. Izena ez bada agertzen, beste jakinarazpen-en izenburua aztertzen dira gurasoa hedatu gabe dagoen jakiteko.

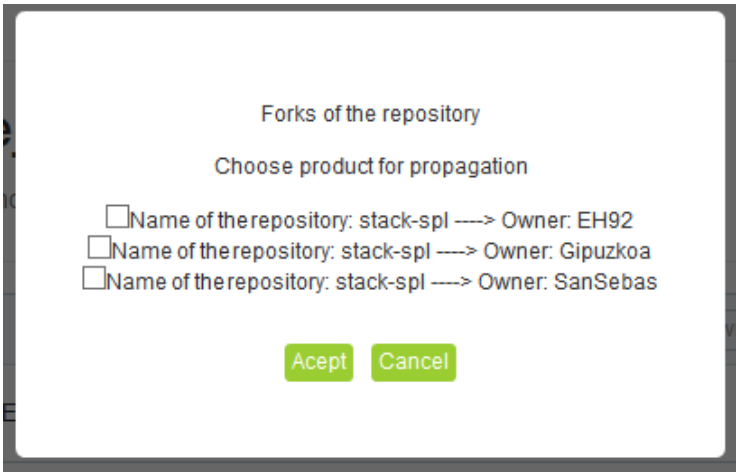
- *UI.Dialog. show_ForksOfRepository* funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Funtzio honetan, interfazearen botoiak sortzen dira eta ekintza bat zehazten zaie.
    - Bukatzeko sortutako interfazea pantailaratzen da.

Erabiltzaileak interfazean agertuko diren produktu biltegietatik hedaketarako biltegiak aukeratzen ditu eta *Acept* sakatzen du. Hedatu nahi den ezaugarria derrigorrezkoa bada, hedaketa biltegi guztietara burutzea derrigortuta egongo da erabiltzailea, 51. irudian ikusten

den moduan. Aukerakoa edo aldaki bat bada, erabiltzaileak erabaki ahal izango du, 52. irudian ikusten den moduan.




51. irudia: Biltegi guztietara hedatzeko derrigortuta.



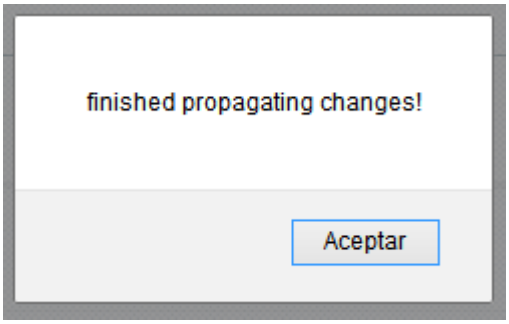

52. irudia: Hedaketarako biltegiak aukeratzeko aukera.

### 3. Pausoa: Aukeratutako biltegietara hedaketa egin.

- DeltaUtils.selectedCheckForks=function(docu, Forks, newFeature, issue){} funtzioa exekutatzen da eta ondorengoa gertatzen da:
  - Erabiltzaileak aukeratutako biltegiak lortu.
  - *DeltaUtils.editIssue(titleIssue);* → jakinarazpena itxi.
  - window.setInterval baten bidez → *DeltaUtils.forwardRecursive(forkSelected, newFeature, DeltaUtils.recursivekont, issue);* exekutatu 15 segunduro, aukeratutako produktu biltegi bakoitzeko behin.

- DeltaUtils.forwardRecursive=function(Forks, newFeature, multiPropagationCount, issue) {} funtzioa exekutatzen da eta ondorengoa gertatzen da:
    - Hedaketa fork guztietara burutu bada, hedaketa bukatu dela adierazten du.
    - Bestela, *DeltaUtils.enactForwardPropagation(ghUser,Forks[DeltaUtils.recursivekont],newFeature,true);* exekutatzen da hedaketa gauzatzeko. Funtzio hau ikertzaileak inplementatu du.

Erabiltzaileari hedaketa bukatu dela adieraziko zaio (53. irudia).




53. irudia: Hedaketa burutu dela adierazten duen leihoa.

## 7.3 Izandako arazoak

Proiektuaren zati honetan, nagusiki hiru arazo egon dira: txertatutako ezaugarri berriaren informazioa nola gorde hedaketa burutzeko momenturako, or-esklusibo egitura baten kudeaketa eta hedaketa ezberdinen ordena mantentzea.

### 7.3.1 Ezaugarri berriaren informazio gorde

Ezaugarri bat txertatu eta hedatu ezin da era jarrai batean egin. Lehenengo, garatzaileak ezaugarria txertatu behar du, ondoren ezaugarri horri dagokion kodea inplementatu eta azkenik ezaugarri hori produktu biltegietara hedatu. Arazoa hedaketa egiteko momentuan sortzen da, nondik lortzen da ezaugarri berriaren informazioa (ezaugarri mota, gurasoa eta izena)?

Aukera bat lokalean fitxategi bat sortzea zen, txertatutako ezaugarrien informazioa gordetzen zuena, baina FaMa-rekin lokalean fitxategi asko sortzen direnez, aukera hau alde batera utzi nahi zuen ikertzaileak.

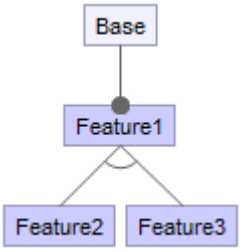
Azkenean, jakinarazpenaren izenburutik lortzen da informazioa hedaketa egiteko momentuan. Horretarako, jakinarazpenen izenburu guztiek eredu bera jarraitzen dute: *New_(ezaugarri mota)_feature_of_(gurasoa)_(izena)*. Adibidez, *Base* ezaugarriari *Feature11* izeneko derrigorrezko bat txertatu bazaio, jakinarazpenaren izenburua ondorengoa izango da: *New_mandatory_feature_of_Base_(Feature11)*.

### 7.3.2 Or-esklusibo egitura baten kudeaketa

Hasierako irismenean, bakarrik txertatutako derrigorrezko ezaugarriak bakarrik hedatzea finkatu zen, baina funtzionalitate hau garatzerakoan aukerako bat edo aldaju bat ere hedatzeak denbora kostu txikia suposatzen zuela ikusita, bi kasu hauek hedaketara gehitzea finkatu zen.

Baina or-esklusibo batean txertatutako ezaugarri bat hedatzea egoera berezia da. Kasu honetan, ez du balio ezaugarri berria produktu biltegietan gehitzearekin, aurretik or-esklusibo horren ezaugarria ezabatu behar da. Argiago ikusteko, adibide batekin azalduko da.

Demagun 54. irudian ikus daitekeen ezaugarri eredua duen ezaugarri biltegi bat dugula eta 16. taulan ikus daitekeen konfigurazioa duen produktu biltegi bat ere.

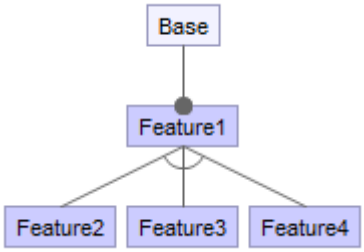



54. irudia: Or-esklusiboa duen ezaugarri eredua.

| Produktu biltegiaren konfigurazioa |
|---|
| Base |
| Feature1 |
| Feature3 |

16. taula: Produktu biltegiaren hasierako konfigurazioa.

Egoera honetan, garatzaileak Feature1-i Feature4 izeneko aldaki bat gehitzen dio (55. irudia) eta produktu biltegira hedatzen du, kasu honetan ez du konfigurazioan Feature4 gehitzearekin balio (17. taula), Feature3 ezabatu beharko litzateke ere (18. taula).




55. irudia: Or-esklusiboa duen ezaugarri eredu berria.

| Produktu biltegiaren konfigurazio OKERRA |
|---|
| Base |
| Feature1 |
| Feature3 |
| Feature4 |

17. taula: Produktu biltegiaren konfigurazio okerra.

| Produktu biltegiaren konfigurazio ZUZENA |
|---|
| Base |
| Feature1 |
| Feature4 |

18. taula: Produktu biltegiaren konfigurazio zuzena.

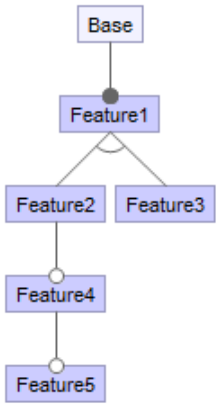
Beraz, hasierako irismenean finkatuta ez zegoenez eta denbora kostua handiago zenez, kasu hau alde batera utzi da. Ezaugarri bat txertatzerako momentuan, aldaki bat txertatu nahi bada bakarrik or-inklusibo egitura duten ezaugarriak agertuko dira.

### 7.3.3 Hedaketa ezberdinen ordena mantendu.

Demagun 54. irudian agertzen den ezaugarri eredua duen ezaugarri biltegi bat dugula eta *Feature2*-ri *Feature4* gehitzen zaiola. *Feature4* hedatu gabe, *Feature4*-ri *Feature5* gehitzen zaio (56. irudia). Hedaketa *Feature5*-ekin hasten bada, sistemak *Feature4* inplementatua duten produktu biltegirik ez dagoela adieraziko du eta hori modu batean, okerra da. Arazoa *Feature4* (*Feature5*-en gurasoa) hedatu gabe dagoela da.

Arazo hori konpontzeko, bi kasu ezberdin bereiztu dira hedaketa egiteko momentuan: ezaugarri berriaren gurasoa inplementatua duten biltegirik ez egotea edo ezaugarri berriaren gurasoa hedatu gabe egotea. Modu honetan, *Feature5* hedatzeko garaian, lehenago *Feature4* hedatu behar dela adieraziko du sistemak, hedaketa ezberdinen ordena mantenduz.

Laburbilduz, hedaketa bat egiteko garaian, ezaugarri berriaren gurasoa hedatu den ala ez begiratzen da. Gurasoa hedatua ez dagoen kasuetan, aukeratutako ezaugarria hedatu baino lehen hedatu behar duen ezaugarria adierazten zaio erabiltzaileari.




56. irudia: Lortutako ezaugarri eredua.

# 8. KAPITULUA

# Ebatzailea

Ebatzailea software matematikoen atal bat adierazten duen hitz generikoa da, gehienetan arazo matematikoak ebazten dituen software liburutegi bat. Proiektu honetan, ebatzailea ezaugarri ereduekin lan egiteko erabiltzen da. Ebatzaile horrek ondorengo funtzionalitateak inplementatuak ditu: ezaugarri eredu baten derrigorrezko ezaugarriak lortu, konfigurazio bat baliozkoa den ala ez zehaztu eta ezaugarri ereduan ezaugarri bat txertatu ondoren, ezaugarri eredu berria baliozkoa den ala ez zehaztu.

- Derrigorrezko ezaugarriak lortu: Parametro bezala jasotzen duen ezaugarri ereduaren derrigorrezko ezaugarriak lortzen ditu, beste modu batera esanda, parametro bezala jasotako ezaugarri eredu horretatik sortutako produktu guztietan egongo diren ezaugarriak zehazten ditu.
- Konfigurazio bat balioztatu: Parametro bezala jasotzen duen ezaugarri ereduan oinarrituta, parametro bezala jasotako konfigurazioa (ezaugarri multzoa), baliozkoa den ala ez adierazten du.
- Ezaugarri eredu batean ezaugarri berri bat txertatu eta ezaugarri eredu berria balioztatu: Parametro bezala jasotako ezaugarri ereduan, parametro bezala jasotako ezaugarri berria txertatzen du. Gainera, sortutako ezaugarri eredu berri hori balioztatzen du.

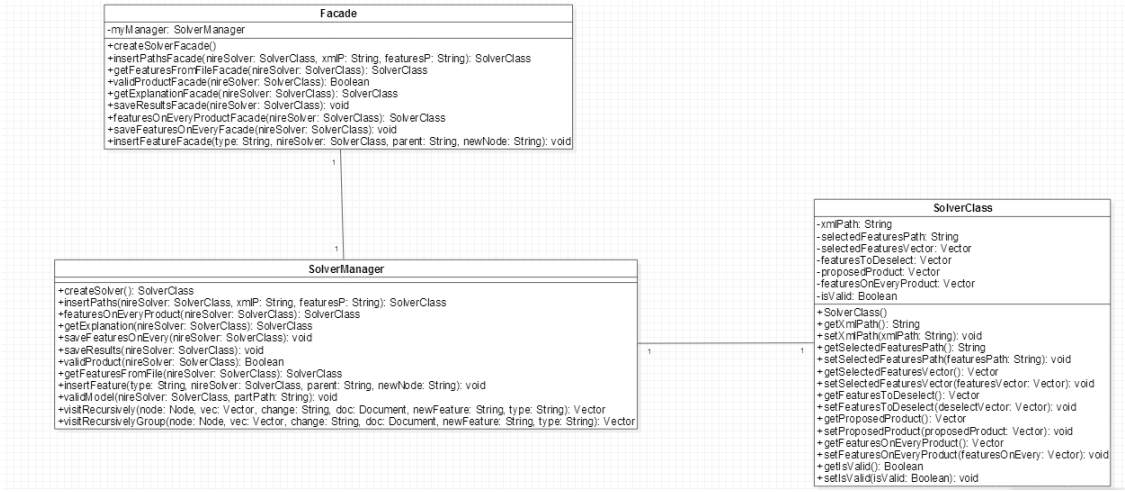
Ebatzailearen inplementazioan FaMa [2] liburutegia erabili da.

## 8.1 Analisi eta diseinua

Atal honetan, ebatzailearen UML –ko klase diagrama eta sekuentzia diagramak deskribatuko dira.

### 8.1.1 Klase diagrama

57. irudian, UML-n inplementatutako domeinu7 eredua zehazten da. Ebatzailea hiru mailako arkitektura jarraitzeko diseinatu da. Facade klaseak main-a eta logika bereizten ditu, Facade motako objektu batek SolverManager objektu batekin bakarrik edukiko du erlazioa, eta alderantziz. SolverManager klaseak SolverClass motako objektuak kudeatzen ditu, SolverManager objektu bat SolverClass objektu batekin bakarrik egongo da erlazionatuta, eta alderantziz. SolverClass klaseak get eta set erako metodoak ditu bakarrik.




57. irudia: Klase diagrama.

### 8.1.2 Sekuentzia diagramak

Sekuentzia diagramen bidez, ebatzailearen funtzionalitate bakoitzean parte hartzen duten klaseak eta gertakarien ordena azaltzen da.

#### 8.1.2.1 Balioztaketa

Atal honetan *Balioztaketa* funtzionalitatearen sekuentzia diagrama (59. irudia eta 60. irudia) deskribatzen da. *Balioztaketa* burutzeko, ebatzaileak ezaugarri eredua eta balioztatu beharreko konfigurazioa jaso behar ditu parametro bezala.

[7] Domeinu eredua *StarUML* softwarearekin garatu da.

Lehenengo pausoan prozesu osoan zehar erabiliko den *SolverClass* motako objektua sortzen da, *nireSolver* izenarekin. Ondoren, *nireSolver* objektuan exekuzio-deian jasotako parametroak gordetzen dira (4. pausoa): ezaugarri ereduaren bide-izena, eta balioztatu nahi den konfigurazioa osatzen duten ezaugarriak dauzkan fitxategiaren bide-izena. Balioztaketarekin hasi baino lehen, ezaugarriak dituen fitxategia aztertzen da eta informazio hori *nireSolver* objektuan gordetzen da (8. pausoa).

*nireSolver* objektuan dagoen informazioarekin balioztaketa egiten da (11. pasuoa), balioztaketa egiteko FaMa-k inplementatutako klaseak erabiltzen dira (*QuestionTrader*, *GenericFeatureModel*, *Product*, *ValidProductQuestion* eta *ExplainInvalidProductQuestion*), hau da, balioztaketa bera FaMa liburutegiak egiten du. FaMa-k bere prozesuarekin amaitzerakoan, *nireSolver* objektuan emaitza gordetzen da.

Ondoren, konfigurazioa okerra baldin bada berriz ere FaMa liburutegia erabiltzen da zergatiaren informazioa jasotzeko (24. Pausoa). Bukatzeko, FaMa-tik jasotako informazioa Fama izeneko direktorioan fitxategi ezberdinetan (*featuresToDeselec*t.*txt* eta *proposedPoductFile.txt*) gordetzen da (39. pausoa).

#### 8.1.2.2 Derrigorrezko ezaugarriak lortu

Atal honetan derrigorrezko ezaugarriak lortzeko funtzionalitatearen sekuentzia diagrama deskribatzen da (61. irudia). Ebatzaileak ezaugarri eredu bat jaso behar du parametro bezala.

Lehenengo pausoan prozesu osoan zehar erabiliko den *SolverClass* motako objektua sortzen da, *nireSolver* izenarekin. Ondoren, *nireSolver* objektuan exekuzio-deian jasotako parametroa gordetzen da (4. pausoa): ezaugarri ereduaren bide-izena.

*nirSolver* objektuan dagoen informazioarekin eta FaMa liburutegiko klaseak erabilita (*QuestionTrader* eta *CoreFeatureQuestion*), derrigorrezko ezaugarriak lortzen dira eta *nireSolver* objektuan gordetzen dira (7. pausoa).

Bukatzeko, *nireSolver* objektuan gordetako informazio hori Fama izeneko direktorioan, fitxategi berri baten gordetzen da (*featuresOnEvery.txt*) (17. pausoa).

#### 8.1.2.3 Ezaugarri bat txertatu.

Atal honetan ezaugarri bat txertatu funtzionalitatearen sekuentzia diagrama deskribatzen da (62. irudia). Ebatzaileak ezaugarri bat txertatu ahal izateko, ezaugarri eredu bat, ezaugarri

berriaren mota, ezaugarri berriaren gurasoa eta ezaugarri berriaren izena jaso behar ditu parametro bezala.

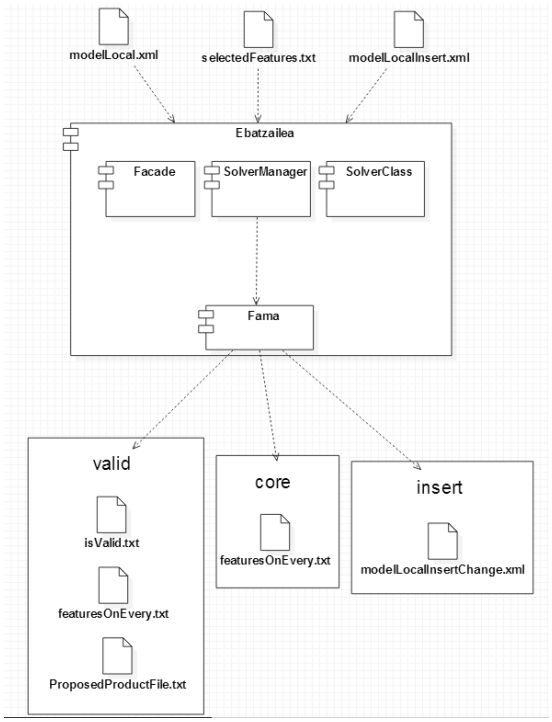
Lehenengo pausoan prozesu osoan zehar erabiliko den *SolverClass* motako objektua sortzen da, *nireSolver* izenarekin. Ondoren, *nireSolver* objektuan exekuzio-deian jasotako ezaugarri ereduaren bide-izena gordetzen da (4. pausoa).

Ezaugarri berria txertatzeko, ezaugarri eredua errekurtsiboki aztertzen da ezaugarri berriaren gurasoa aurkitu arte (7. pausoa). Ezaugarri berriaren gurasoa aurkitzerakoan, ezaugarri berria txertatzen da.

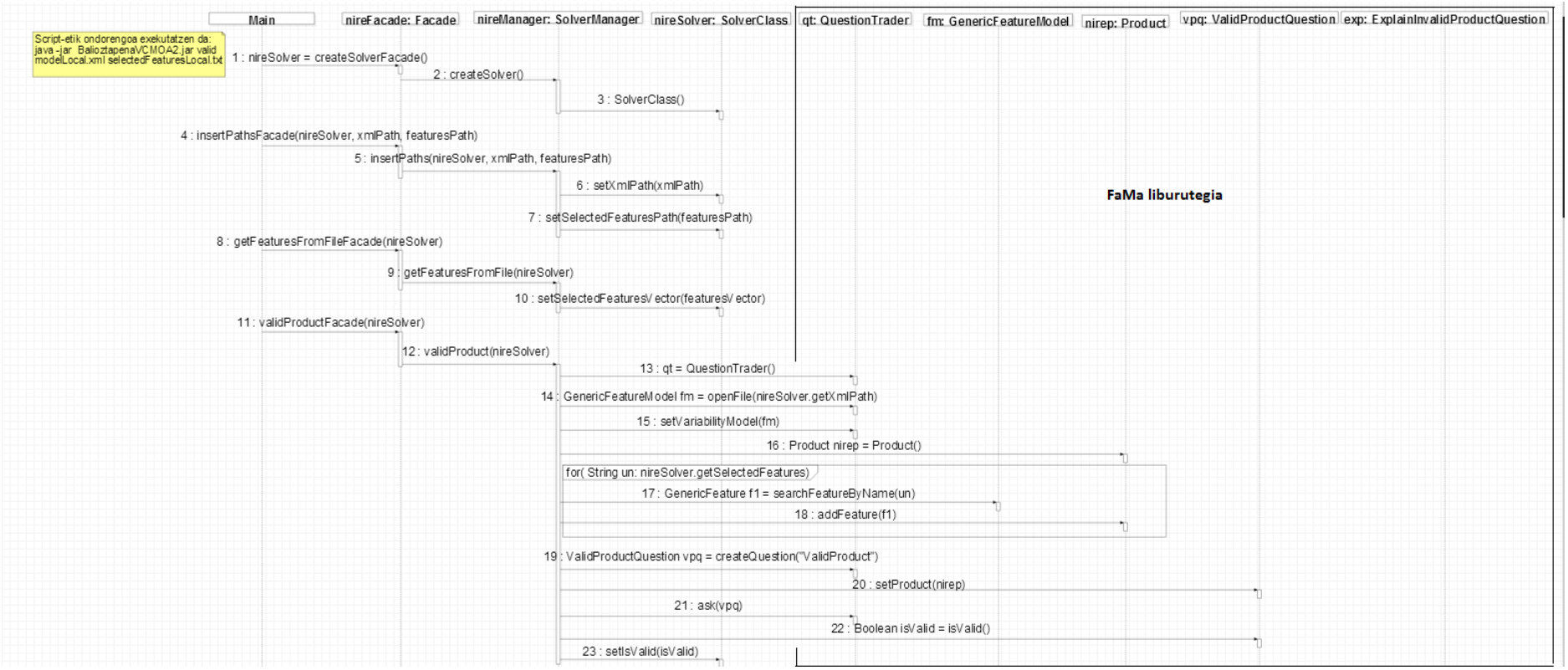
Ezaugarri eredu berria lortu ondoren, FaMa liburutegiko klaseak (*QuestionTrader* eta *ValidQuestion*) erabiliz baliozkoa den ala ez aztertzen da (12. Pausoa). Baliozkoa ez bada, ezaugarri eredu berria duen fitxategia ezabatzen da.

## 8.2 Inplementazioa

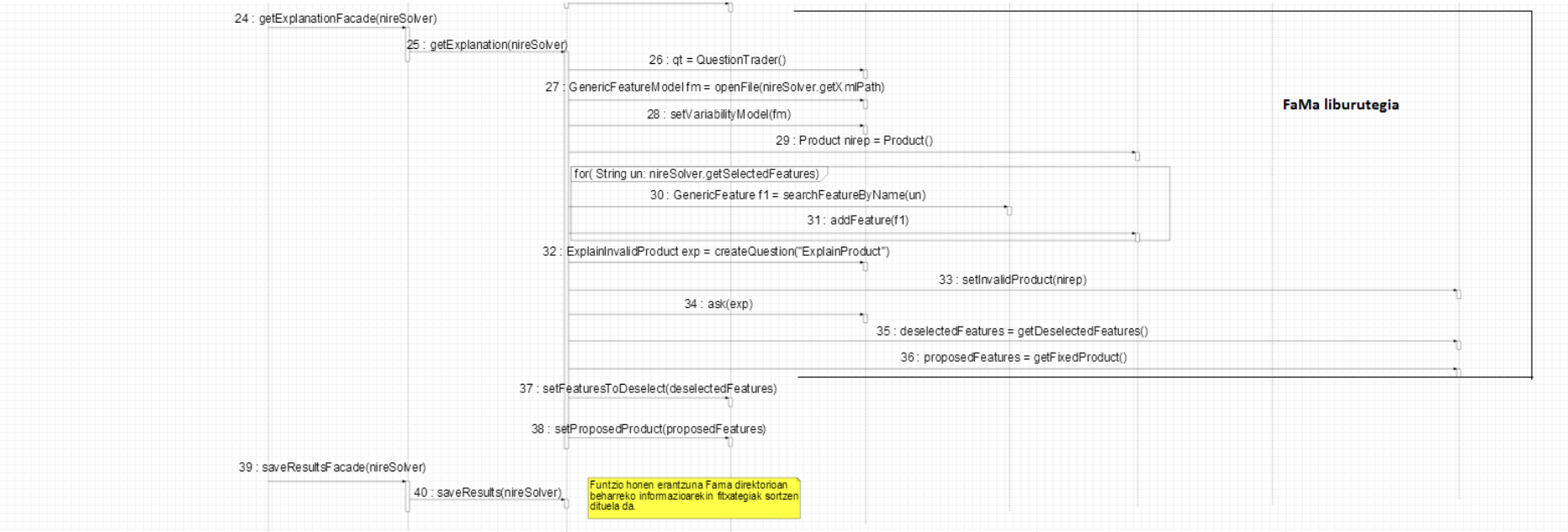
Atal honetan, ebatzaileak erabiltzen dituen fitxategiak eta ebatzailea exekutatzeko erabiltzen diren script-ak deskribatuko dira. 58. irudian ikusten den moduan ebatzaileak input moduan fitxategi ezberdinak jasotzen ditu, eta output moduan beste fitxategi batzuk sortzen ditu.



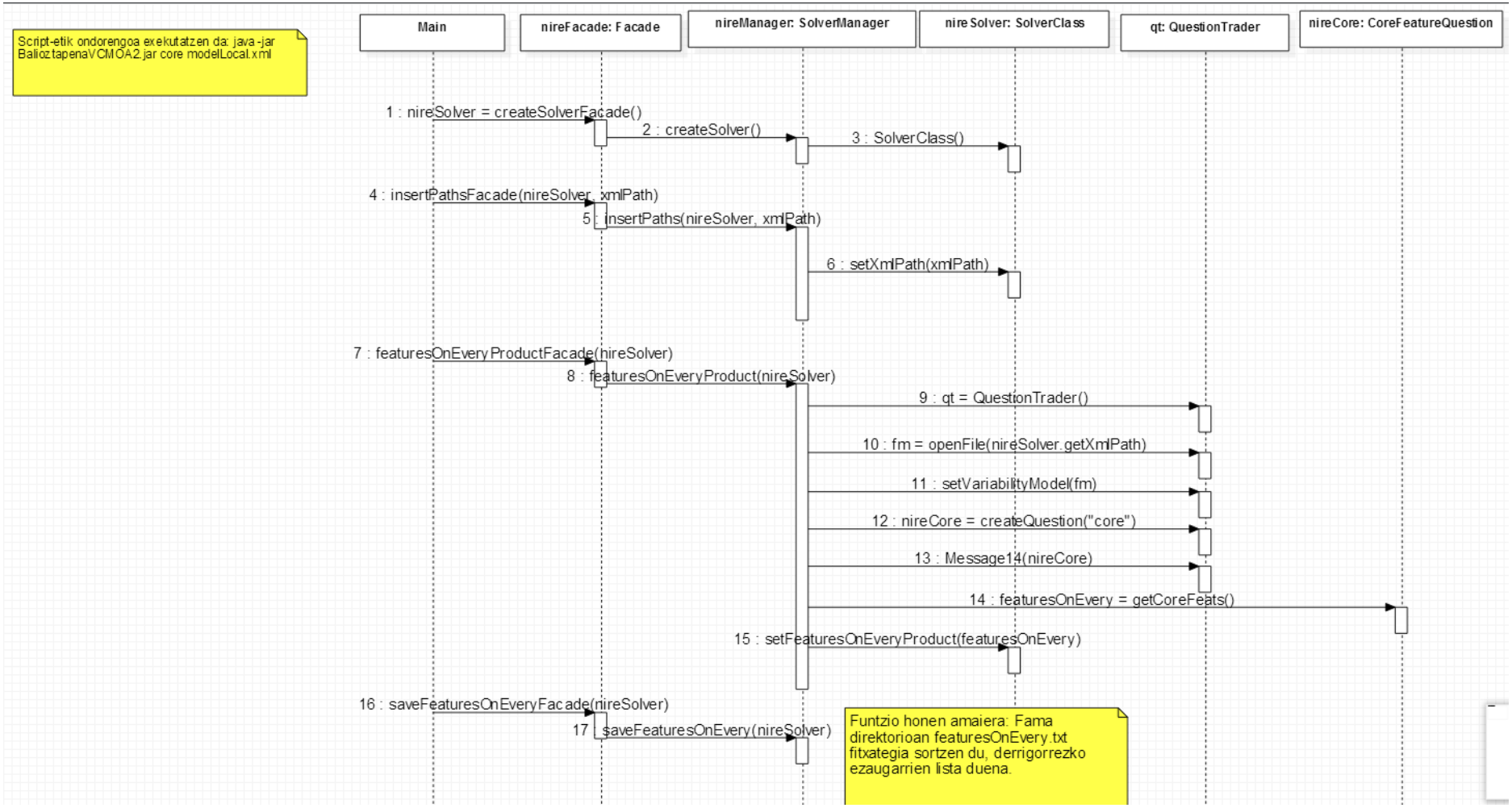

58. irudia: Ebatzailearen arkitektura.



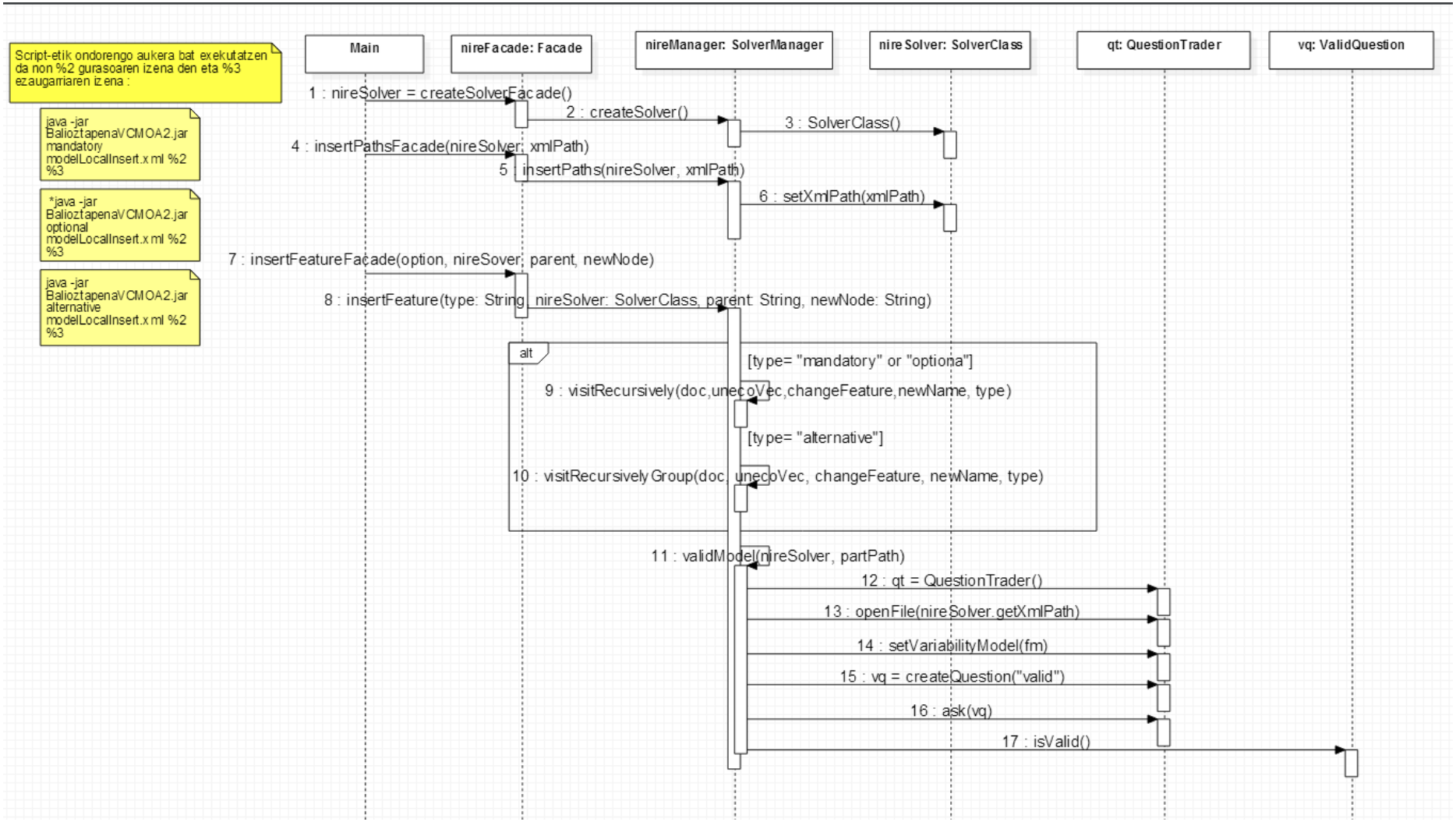

59. irudia: Sekuentzia diagrama: *Balioztaketa*.




60. irudia: Sekuentzia diagrama: *Balioztaketa* (jarraipena).




61. irudia: Sekuentzia diagrama: Derrigorrezko ezaugarriak lortu.




62. irudia: Sekuentzia diagrama: Ezaugarri bat txertatu.

### 8.2.1 Erabiltzen dituen fitxategiak

Ebatzaileak fitxategi ezberdinekin lan egiten du, batzuk sarrera moduan eta beste batzuk irteera moduan. Atal honetan, fitxategi hauek deskribatuko dira.

- Sarrera moduan erabiltzen dituen fitxategiak:
    - *modelLocal.xml:* Konfigurazio bat balioztatzeko, biltegian dagoen ezaugarri eredua gordetzen du. Formatua ez dago finkatuta, ezaugarri bakoitza lerro batean, ezaugarri eredu osoa lerro batean edo edonola egon daiteke, baldintza bakarra ezaugarri eredua ondo egituratuta egotea da (beharrezko etiketak erabili eta etiketak behar moduan itxiak egotea).
    - *modelLocalInsert.xml*: Ezaugarri berri bat txertatzeko, biltegian dagoen ezaugarri eredua gordetzen du. Formatua ez dago finkatuta, ezaugarri bakoitza lerro batean, ezaugarri eredu osoa lerro batean edo edonola egon daiteke, baldintza bakarra ezaugarri eredua ondo egituratuta egotea da (beharrezko etiketak erabili eta etiketak behar moduan itxiak egotea).
    - *selectedFeatures*.txt: Erabiltzaileak balioztatu nahi duen konfigurazioa osatzen duten ezaugarriak gordetzen ditu. Ezaugarri guztiak lerro batean eta zuriunearen bitartez bereiztuta egon behar dira.
- Irteera moduan erabiltzen dituen fitxategiak:
    - *isValid.txt*: Konfigurazio bat baliozkoa den ala ez itzultzen du. 1 baliozkoa bada, 0 baliozkoa ez bada.
    - *featuresOnEvery*.txt: Ezaugarri eredu baten derrigorrezko ezaugarriak, hau da, konfigurazio guztietan egon behar diren ezaugarriak gordetzen ditu. Ezaugarri guztiak lerro batean eta zuriunearen bitartez bereiztuta.
    - *featuresToDeselect.txt*: Konfigurazio bat zuzena ez denean, alde batera utzi behar diren ezaugarriak gordetzen ditu. Jatorrizko konfigurazioa zuzena bada, fitxategia hutsik egongo da.
    - *proposedProductFile.txt*: Konfigurazio bat okerra denean, ebatzaileak konfigurazio horretan oinarrituta, zuzena den konfigurazio bat proposatzen du. Proposamen hori fitxategi honetan gordetzen da. Konfigurazioa zuzena denean, fitxategia hutsik dago.
    - *modelLocalInsertChange.txt:* Ezaugarri eredu batean ezaugarri bat txertatzerakoan, fitxategi honetan ezaugarri eredu berria gordetzen da. Formatua ez dago finkatuta, ezaugarri bakoitza lerro batean, ezaugarri eredu osoa lerro batean edo edonola egon daiteke, baldintza bakarra ezaugarri eredua ondo egituratuta egotea da.

### 8.2.2 Script-ak

GitLine aplikazioan dagoen Fama izeneko direktorioan, GitLine aplikazioaren eta ebatzailearen arteko informazio trukaketa ahalbidetzen duten script-ak daude. Script bakoitzak ebatzailearen funtzionalitate bat exekutatzen du. Script hauek GitLine-eko kodetik exekutatzen dira, ebatzailea exekutatu ahal izateko.

- *famaValid.bat edo famaValid.sh*: Produktu bat balioztatzen du.
- *famaCore.bat edo famaCore.sh*: Ezaugarri eredu baten derrigorrezko ezaugarriak lortzen ditu.
- *famaMandatory.bat edo famaMandatory.sh*: Derrigorrezko ezaugarri bat txertatzen du ezaugarri ereduan.
- *famaOptional.bat edo famaOptional.sh*: Aukerako ezaugarri bat txertatzen du ezaugarri ereduan.
- *famaAlternative.bat edo famaAlternative.sh*: Aldaki bat gehitzen du ezaugarri ereduan.

## 8.3 Erabilpena

Ebatzailearen inplementazioa BalioztapenaVCAMOA2.jar fitxategian paketatuta dago eta diseinuan aipatu den bezala berarekin hiru funtzio burutu daitezke.

- Produktu baten balioztaketa:
  - Deia: *java –jar BalioztapenaVCAMOA2.jar valid model.xml selectedFeatures.txt*
  - Parametroak:
    - *valid*: Produktu bat balioztatu nahi dela adierazten duen parametroa.
    - *model.xml*: Balioztaketarako erabiliko den ezaugarri eredua.
    - *selectedFeatures.txt*: Balioztatu nahi den produktua osatzen duten ezaugarriak. Ezaugarriak zuriunearen bitartez bereiztuta egon behar dira.
  - Irteera: Konfigurazio bat ezaugarri eredu batentzako baliozkoa den ala ez itzultzen du *isValid.txt* fitxategian (1 baliozkoa da, 0 ez da baliozkoa). Gainera, konfigurazioa okerra bada, *featuresToDeselect.txt* eta *proposedProductFile.txt* fitxategietan konfigurazioa baliozkoa izateko aholkuak itzultzen ditu.
- Ezaugarri eredu baten derrigorrezko ezaugarri (produktu guztietan egon behar diren ezaugarriak) guztiak lortu.
  - Deia: *java –jar BalioztapenaVCAMOA2.jar core model.xml*
  - Parametroak:
    - core: Ezaugarri eredu baten derrigorrezko ezaugarriak lortu nahi direla adierazten duen parametroa.

        - model.xml: aztertu nahi den ezaugarri eredua.
    - Irteera: *featuresOnEvery.txt* fitxategian parametro bezala sartutako ezaugarri ereduaren derrigorrezko ezaugarriak itzultzen ditu, zuriunez bereizita.

- Ezaugarri berri bat txertatu ezaugarri eredu batean eta ezaugarri eredu berria balioztatu.
    - Deia: *java –jar BalioztapenaVCAMOA2.jar type model.xml parent new*
    - Parametroak:
        - type: Ezaugarri eredu batean ezaugarri mota ezberdin daudenez, parametro honetan zein motako ezaugarria txertatu nahi den adierazi behar da: *mandatory*/*optional*/*alternative*.
        - model.xml: Jatorrizko ezaugarria non ezaugarri berria txertatuko den.
        - parent: Txertatu nahi den ezaugarriaren gurasoa.
        - new: Ezaugarri berriaren izena.
    - Irteera: Ezaugarri eredu berria baliozkoa bada, modelLocalInsertChange.xml fitxategian ezaugarri eredu berria itzultzen du.

# 9. KAPITULUA

# Test-kasuak

Kapitulu honetan, erabiltzaileak sor ditzakeen egoera ezberdinen aurrean sistemak erantzuteko modua azalduko da.

Test kasuak hiru ataletan bereiztu dira, inplementatutako funtzionalitate bakoitzeko bana: *balioztaketa*, *ezaugarria txertatu* eta *ezaugarria hedatu*.

## 9.1 Balioztaketa

*Balioztaketak* ezaugarri biltegian dagoen ezaugarri ereduan oinarrituta, baliozkoak diren produktu biltegiak sortzen direla ziurtatzen du *6-Balioztaketa* kapituluan azalduta.

*Balioztaketa* egiterako garaian ondorengo egoerak aurki daitezke.

- EGOERA: Ezaugarri eredua hutsik dago.
  - Erantzuna: Ezaugarri eredua hutsik dagoela adierazten duen leihoa agertzen da.
- EGOERA: Ezaugarri eredua hasieratuta dago *<feature-model>* etiketarekin baina ez dago errorik.
  - Erantzuna: Ezaugarri eredua hutsik dagoela adierazten duen leihoa agertzen da.
- EGOERA: Ezaugarri eredua gaizki egituratua dago.
  - Erantzuna: Ezaugarri eredua gaizki dagoela adierazten duen leihoa agertzen da.
- EGOERA: Ezaugarririk gabeko konfigurazioa aukeratzen da.
  - Erantzuna: Ezin da egoera hau eman, beti gutxienez zuhaitzaren erroa aukeratuta eta desgaituta egongo da.
- EGOERA: *ProductFork* sakatu ondoren, produktu biltegiaren sorrera alde batera utzi nahi da.
  - Erantzuna: Interfaze guztietan *Cancel* botoia aurkitu daiteke prozesua alde batera uzteko.

- EGOERA: Egokia den konfigurazio bat lortzerakoan, beste konfigurazio batekin probatu nahi da.
    - Erantzuna: *CheckValidityOfAnotherProduct* sakatuz, berriz ere beste konfigurazio bat balioztatu daiteke.

## 9.2 Ezaugarria txertatu

*Ezaugarria txertatu* funtzionalitateak, ezaugarri biltegian dagoen ezaugarri ereduan ezaugarri berri bat txertatzen du 7.1 *Ezaugarria txertatu* atalean azalduta.

Ezaugarri eredu batean ezaugarri berri bat txertatzerako garaian ondorengo egoerak aurki daitezke.

- EGOERA: Ezaugarri eredua hutsik dago.
    - Erantzuna: Gurasoa aukeratzeko garaian, ezaugarri eredua hutsik dagoela adierazten duen interfazea agertzen da.
- EGOERA: Ezaugarri eredua hasieratuta dago *<feature-model>* etiketarekin baina ez dago errorik.
    - Erantzuna: Gurasoa aukeratzeko garaian, ezaugarri eredua hutsik dagoela adierazten duen interfazea agertzen da.
- EGOERA: Ezaugarri eredua gaizki egituratua dago.
    - Erantzuna: Gurasoa aukeratzeko garaian, ezaugarri eredua gaizki egituratua dagoela adierazten duen interfazea agertzen da.
- EGOERA: Ez dago ezaugarri motarik aukeratuta *Acept* sakatzerakoan.
    - Erantzuna: Ezaugarri mota aukeratu behar dela adierazten duen interfazea agertzen da, eta ezaugarri mota aukeratzeko interfazera bueltatzen da.
- EGOERA: Ezaugarri berriaren gurasoa aukeratzeko ezaugarri ereduaren ezaugarriak listaratzen dituen interfazean, ez da gurasorik aukeratu.
    - Erantzuna: Gurasoa aukeratu behar dela adierazten duen interfazea agertzen da, eta gurasoa aukeratzeko interfazera bueltatzen da.
- EGOERA: Ezaugarri berriaren izena existitzen da jada ezaugarri ereduan.
    - Erantzuna: Izena jada existitzen dela adierazten duen interfazea agertzen da, eta izena zehazteko interfazera bueltatzen da.
- EGOERA: Ezaugarri berriaren izena txuri utzi da.
    - Erantzuna: Izen bat zehaztu behar dela adierazten duen interfazea agertzen da, eta izena finkatzeko interfazera bueltatzen da.
- EGOERA: Izena zehazteko momentuan, gurasoa gaizki aukeratu dela konturatzen da.
    - Erantzuna: *Cancel* sakatu eta gurasoa aukeratzeko interfazera bueltatuko da.
- EGOERA: *InsertFeature* sakatu ondoren, ezaugarri berri bat txertatzea alde batera utzi nahi da.

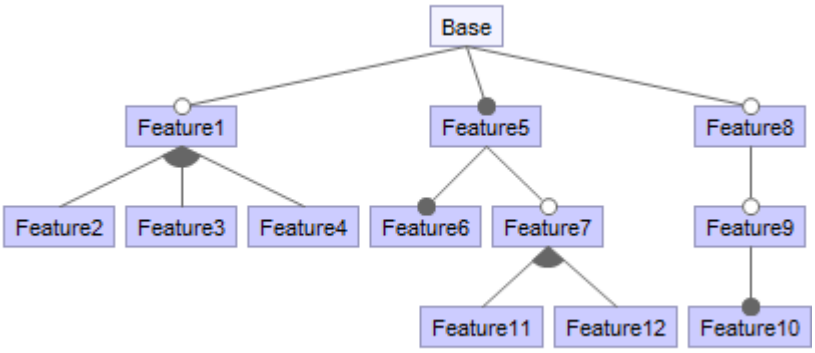
- Erantzuna: Interfaze guztietan *Cancel* botoia aurkitu daiteke prozesua alde batera uzteko.

- EGOERA: Ezaugarri berriaren inguruko informazioa ondo zehaztu den ala ez zalantzak sortzen dira.
  - Erantzuna: Ezaugarri berriaren informazio guztia zehaztu ondoren, informazio guztia azaltzen duen konfirmazioa leiho agertzen da.

## 9.3 Ezaugarria hedatu.

Ezaugarri berri bat txertatu eta garatu ondoren, hedaketa egiteko garaian egoera ezberdinak aurki daitezke, baina egoera hauek hobeto ulertzeko abiapuntu bat finkatuko da.

❖ ABIAPUNTUA: *Donostia* izeneko erabiltzaile batek 63. irudian agertzen den ezaugarri ereduarekin ezaugarri biltegi bat sortu du. Ondoren *SanSebas*, *Gipuzkoa* eta *EH92* erabiltzaileek biltegi horretatik *ProductFork* egin dute konfigurazio ezberdinekin. Produktu biltegi bakoitzak aukeratutako konfigurazioa 19. taulan ikus daiteke.



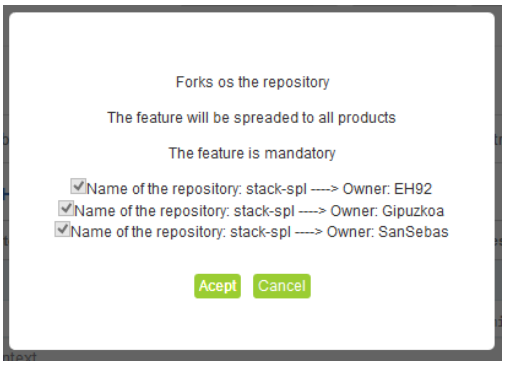

63. irudia: Ezaugarri biltegiaren ezaugarri eredua.

| PRODUKTU BILTEGIAK | | |
|---|---|---|
| **SanSebas** | **Gipuzkoa** | **EH92** |
| Base | Base | Base |
| Feature5 | Feature5 | Feature5 |
| Feature6 | Feature6 | Feature6 |
| Feature7 | Feature1 | Feature8 |
| Feature11 | Feature2 | Feature7 |
| - | Feature3 | Feature12 |

19. taula: Produktu biltegi ezberdinen konfigurazioak.

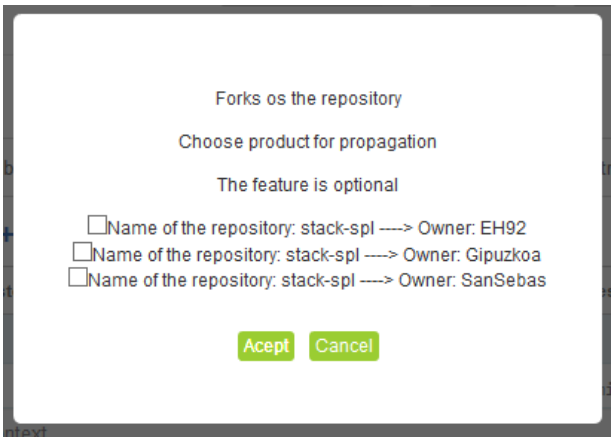
Behin abiapuntua finkatuta, aurki daitezkeen egoera ezberdinak aztertuko dira.

- 1-EGOERA: Base-ri *derrigorrezko* ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Base ezaugarria hiru produktu biltegietan dagoenez eta txertatutako ezaugarria derrigorrezkoa denez, hedaketarako hiru biltegiak aukeratuta eta desgaituak agertuko dira (64. irudia). *Acept* sakatu ondoren, hedaketa hiru biltegietara egingo da.



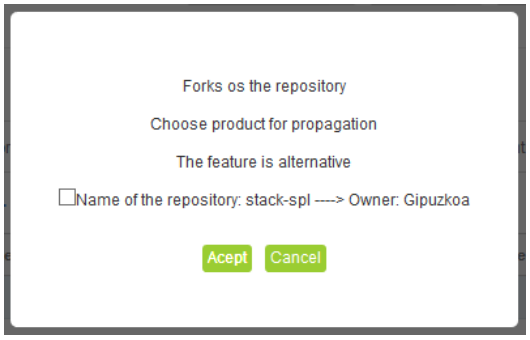

64. irudia: 1-Egoeran jasotako erantzuna.

- 2-EGOERA: Base-ri *aukerako* ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Base ezaugarria hiru produktu biltegietan dagoenez, eta txertatutako ezaugarria aukerakoa denez, hedaketarako hiru biltegiak agertuko dira (65. irudia). Erabiltzaileak zein biltegietara egin nahi duen hedaketa aukeratu ondoren *Acept* sakatuko du. Hedaketa erabiltzaileak aukeratutako biltegietara egingo da soilik.



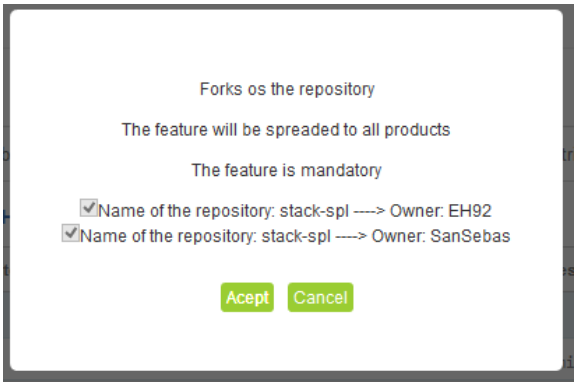

65. irudia: 2-Egoeran jasotako erantzuna.

- 3-EGOERA: Feature1 ezaugarriari *aldaki* bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Feature1 *Gipuzkoa* biltegian soilik dagoenez eta txertatutako ezaugarria aldaki bat denez, Gipuzkoa biltegia soilik agertuko da (66. irudia). Erabiltzaileak *Gipuzkoa* biltegia aukeratu eta *Acept* sakatu ondoren, hedaketa bakarrik *Gipuzkoa* biltegira egingo da.




66. irudia: 3-Egoeraran eta 10-Egoeran jasotako erantzuna.

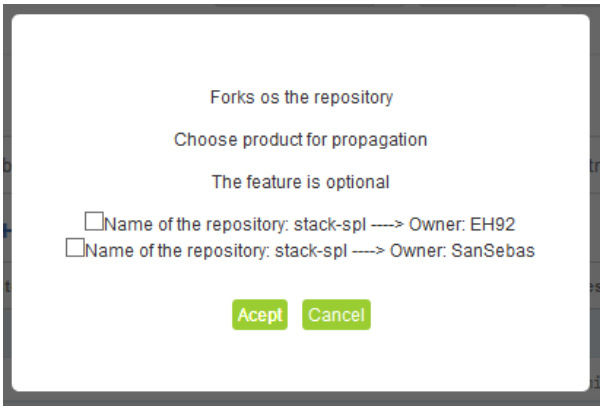
- 4-EGOERA: Feature7 ezaugarriari *derrigorrezko* ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Feature7 ezaugarria SanSebas eta EH92 biltegietan dagoenez eta txertatutako ezaugarri berria derrigorrezkoa denez, hedaketarako bi biltegiak aukeratuta eta desgaituak agertuko dira (67. irudia). Acept sakatu ondoren, hedaketa bi biltegietara egingo da.




67. irudia: 4-Egoeran jasotako erantzuna.

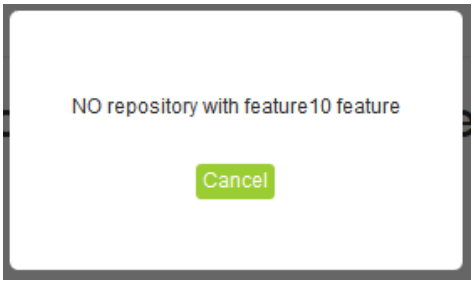
- 5-EGOERA: Feature7 ezaugarriari *aukerako* ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.

- Erantzuna: Feature7 ezaugarria *SanSebas* eta *EH92* produktu biltegietan dagoenez, eta txertatutako ezaugarria aukerakoa denez, hedaketarako bi biltegiak agertuko dira (68. irudia). Erabiltzaileak zein biltegietara egin nahi duen hedaketa aukeratu ondoren Acept sakatuko du. Hedaketa erabiltzaileak aukeratutako biltegietara egingo da soilik.



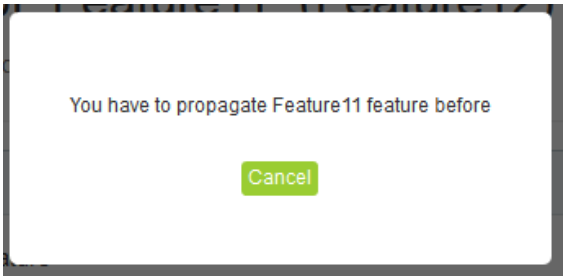

68. irudia: 5-Egoeran jasotako erantzuna.

- 6-EGOERA: Feature10 ezaugarriari *derrigorrezko* edo aukerako ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
  - Erantzuna: Feature10 duten biltegirik ez dagoela adieraziko da (69. irudia).



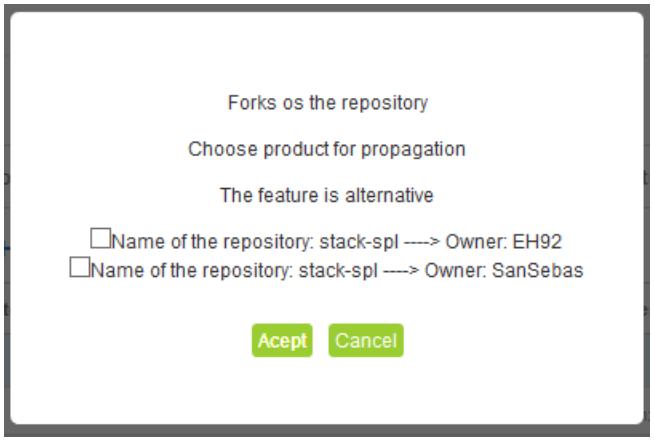

69. irudia: 6-Egoeran jasotako erantzuna.

- 7-EGOERA: Feature6-ri Feature11 txertatu eta Feature11 hedatu gabe, Feature11-ri Feature12 txertatu eta garatu ondoren, Feature12 hedatu nahi da.
  - Erantzuna: Feature12 hedatu baino lehen Feature11 hedatu behar dela adieraziko da (70. irudia).



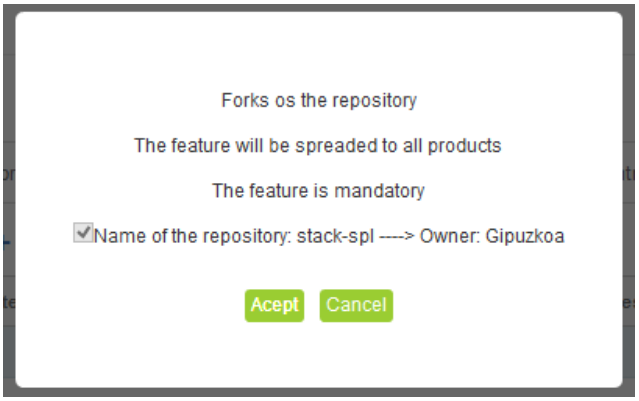

70. irudia: 7-Egoeran jasotako erantzuna.

- 8-EGOERA: Feature7 ezaugarriari *aldaki* bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Feature7 ezaugarria *SanSebas* eta *EH92* biltegietan dagoenez eta txertatutako ezaugarria aldaki bat denez, hedaketarako bi biltegiak agertuko da (71. irudia). Hedaketa erabiltzaileak aukeratutako biltegietara egingo da soilik.




71. irudia: 8-Egoeran jasotako erantzuna.

- 9-EGOERA: Feature2 ezaugarriari *derrigorrezko* ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Feature2 bakarrik *Gipuzkoa* biltegian dagoenez eta txertatutako ezaugarria derrigorrezko denez, Gipuzkoa biltegia soilik agertuko da aukeratuta eta desgaituta (72. irudia). Erabiltzaileak *Acept* sakatu ondoren, hedaketa bakarrik *Gipuzkoa* biltegira egingo da.




72. irudia: 9-Egoeran jasotako erantzuna.

- 10-EGOERA: Feature2-ri aukerako ezaugarri bat txertatu eta garatu ondoren, ezaugarri berri hori hedatu nahi da.
    - Erantzuna: Feature2 ezaugarria *Gipuzkoa* biltegian dagoenez eta txertatutako ezaugarri berria aukerakoa denez, bakarrik *Gipuzkoa* biltegia agertuko da (72. irudia). Acept sakatu ondoren, hedaketa Gipuzkoa biltegira egingo da. Erabiltzaileak Gipuzkoa biltegia aukeratu eta Acept sakatu ondoren, hedaketa bakarrik *Gipuzkoa* biltegira egingo da.

# 10.KAPITULUA

# Jarraipen eta kontrola

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

## 10.1 Proiektuaren jarraipena eta desbiderapenak

Proiektua burutzeko 352 ordu inbertitu dira. Proiektu hasieran egindako plangintzan kalkulatutakoarekin alderatuta, %4.9-ko desbiderapena egon da eta proiektu erdian egindako birplangintzan kalkulaturakoarekin alderatuta bakarrik %0,9-eko desbiderapena egon da.

Zehazki atal bakoitzean egondako desbiderapena 20. taulan ikus daiteke. Birplangintzan kalkulatutako denborak eta burutzeko behar izan den denbora alderaturik desbiderapen handiena kudeaketa atalean gertatu dela ikus daiteke. Dokumentazioaren atalean, nahiz eta birplangintzan kalkulatutako denbora handitu, ez da nahikoa izan eta 8 ordu gehiago inbertitu behar izan dira.

Plangintzan, *1. inplementazioa* eta *2. inplementazioa* moduan identifikatutako atazak, irismenean egondako aldaketak direla eta (aldakia txertatu funtzionalitatea eta txertatu daitezkeen ezaugarri mota guztiak hedatu funtzionalitatea gehitu dira), *ezaugarria txertatu* eta *ezaugarria hedatu* moduan identifikatu dira jarraipen eta kontrolean:

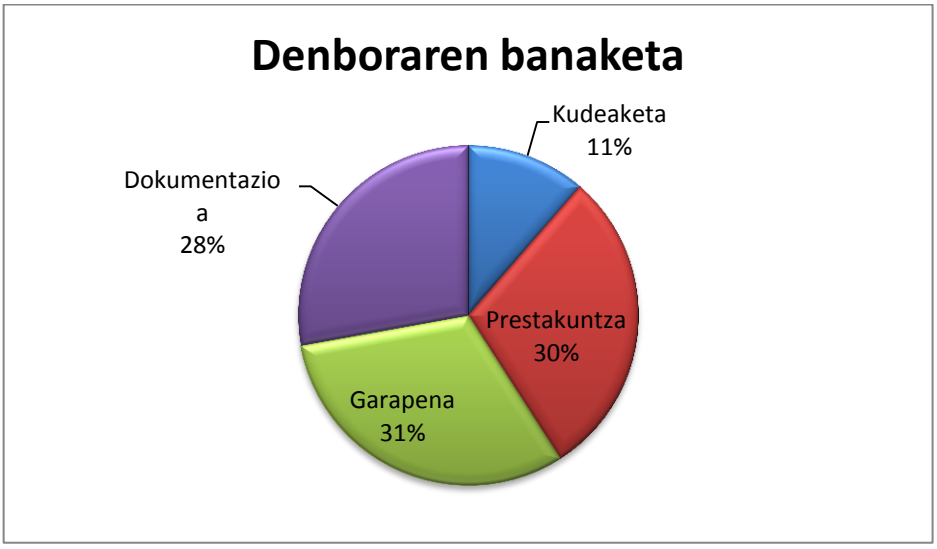
- Ezaugarria txertatu: Derrigorrezko ezaugarria, aukerako ezaugarria edo aldaki bat txertatu ezaugarri ereduan.
- Ezaugarria hedatu: Txertatutako ezaugarriaren gurasoa inplementatua duten produktu biltegi guztietara hedatzeko aukera.

Proiektuan zehar dokumentazioari, prestakuntzari eta garapenari dedikatutako denbora oso antzekoa izan da, bakarrik %3-ko aldearekin, 2. grafikoan ikus daitekeen moduan. Kudeaketa atalean ordu kopuru txikiena inbertitu da, proiektu osoaren %11-a.

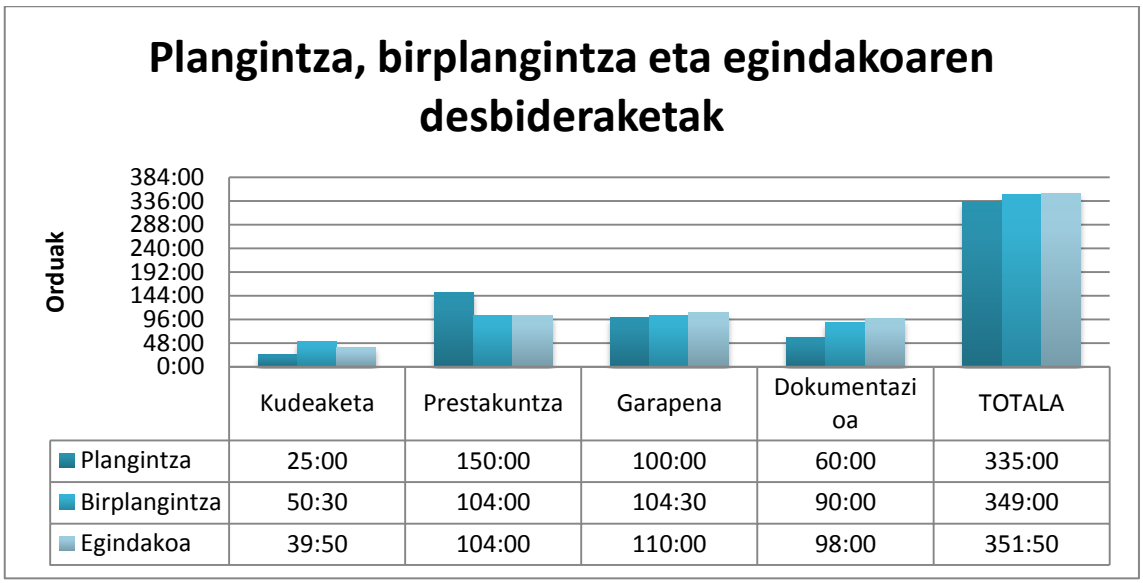
Plangintzan eta birplangintza kalkulatutako denborak eta azkenean inbertitutako denborak hobeto aztertzeko, 3. grafikoan ikus daitekeen grafikoa sortu da. Kudeaketa atalean, plangintzan denbora gutxiegi kalkulatu zen, ikertzaile baten kodearekin lan egingo zela kontuan hartuta. Birplangintzan debora hori asko handitu zen ikertzailearekin beharrezko bilerak egiteko, baina azkenean denbora gutxiago behar izan da. Nahiz eta orokorrean hasieran finkatutako denbora kalkuluak nahiko okerrak izan, ondoren egindako birplangintzan denbora kalkulatzeko gaitasuna hobetu da.

| Ataza | Plangintzan aurreikusitakoa | Birplangintzan aurreikusitakoa | Egindakoa | Desbiderapena (Plang-Egind) | | Desbiderapena (Birplang-Egind) | | Burututa? |
|---|---|---|---|---|---|---|---|---|
| Kudeaketa | 25:00 | 50:30 | 39:50 | + | 14:50 | - | 10:40 | BAI |
| -Plangintza egin | 5:00 | 8:30 | 8:30 | + | 3:30 | - | 0:00 | BAI |
| -Birplangintza egin | 0:00 | 4:00 | 5:30 | + | 5:30 | + | 1:30 | BAI |
| -Bilerak egin | 10:00 | 20:00 | 14:20 | + | 4:20 | - | 5:40 | BAI |
| -Jarraipen eta kontrola egin | 10:00 | 18:00 | 11:30 | + | 1:30 | - | 6:30 | BAI |
| Prestakuntza | 150:00 | 104:00 | 104:00 | - | 46:00 | - | 0:00 | BAI |
| -FeatureIDE aztertu | 50:00 | 43:00 | 43:00 | - | 7:00 | - | 0:00 | BAI |
| -JavaScript ikasi | 20:00 | 6:00 | 6:00 | - | 14:00 | - | 0:00 | BAI |
| -GitHub ezagutu | 10:00 | 7:30 | 7:30 | - | 2:30 | - | 0:00 | BAI |
| ~~-Gity ezagutu~~ | 30:00 | 0:00 | 0:00 | - | 30:00 | - | 0:00 | ~~———~~ |
| -GitLine-rekin trebatu | 40:00 | 23:30 | 23:30 | - | 16:30 | - | 0:00 | BAI |
| -FaMa ikasi | 0:00 | 24:00 | 24:00 | + | 24:00 | - | 0:00 | BAI |
| Garapena | 100:00 | 104:30 | 110:00 | + | 10:00 | + | 5:30 | BAI |
| -Balioztaketa | 50:00 | 74:30 | 74:30 | + | 24:30 | - | 0:00 | BAI |
| -Ezaugarria txertatu | 30:00 | 15:00 | 12:30 | - | 17:30 | - | 2:30 | BAI |
| -Ezaugarria hedatu | 20:00 | 15:00 | 23:00 | + | 3:00 | + | 8:00 | BAI |
| Dokumentazioa | 60:00 | 90:00 | 98:00 | + | 38:00 | + | 8:00 | BAI |
| -Memoria idatzi | 50:00 | 80:00 | 83:30 | + | 33:30 | + | 3:30 | BAI |
| -Aurkezpena prestatu | 10:00 | 10:00 | 14:30 | + | 4:30 | + | 4:30 | BAI |
| GUZTIRA: | 335:00 | 349:00 | 351:50 | + | 16:50 | + | 2:50 | BAI |

20. taula: Denbora taula.




2. grafikoa: Inbertitutako denbora ehunekotan.




| | Kudeaketa | Prestakuntza | Garapena | Dokumentazioa | TOTALA |
|---|---|---|---|---|---|
| Plangintza | 25:00 | 150:00 | 100:00 | 60:00 | 335:00 |
| Birplangintza | 50:30 | 104:00 | 104:30 | 90:00 | 349:00 |
| Egindakoa | 39:50 | 104:00 | 110:00 | 98:00 | 351:50 |

3. grafikoa: Plangintza, birplangintza eta egindakoaren desbideraketak.

Proiektuaren plangintzan, proiektuaren garapena urtarrilaren 6-ean amaituko zela finkatu zen, baina dedikazio faltarengatik, balioztaketa egiteko erreminta aldatu behar izanak eta sortutako ebatzailea GitLine-rekin elkartzeko izandako arazoak direla eta ( 6.2 atalean sakonago azaldutako arazoak), azkenean proiektuaren garapena martxoaren 3an amaitutzat eman da 73. irudian agertzen den Gantt diagraman ikus daitekeen moduan . Otsailean egindako birplangintzari esker, produktuaren kalitatea hobetzeko denbora zegoela konturatu zen ikaslea. Beraz, zuzendariak eta ikertzaileak proposatutako hobekuntzak garatu zituen. Alde batetik, aukerako eta derrigorrezko ezaugarriak txertatzeaz gain, or-esklusibo egitura batean aldaki bat txertatzea inplementatu da ere. Beste aldetik, plangintzan bakarrik txertatutako derrigorrezko ezaugarriak hedatzea finkatu zen, baina azkenean aukerako ezaugarriak eta aldakiak hedatzeko aukera inplementatu da ere.

Plangintzan, *balioztaketa* eta *ezaugarria txertatu* ia modu paralelo batean garatuko zirela kalkulatu zen, baina *balioztaketarekin* izandako arazoak direla eta, *balioztaketarekin* amaitu arte ez da *ezaugarria txertatu* garatu. Aldiz, *ezaugarria txertatu* eta *ezaugarria hedatu* modu paraleloan garatu dira oso lotura estua dutelako.

## 10.2 Komunikazioa

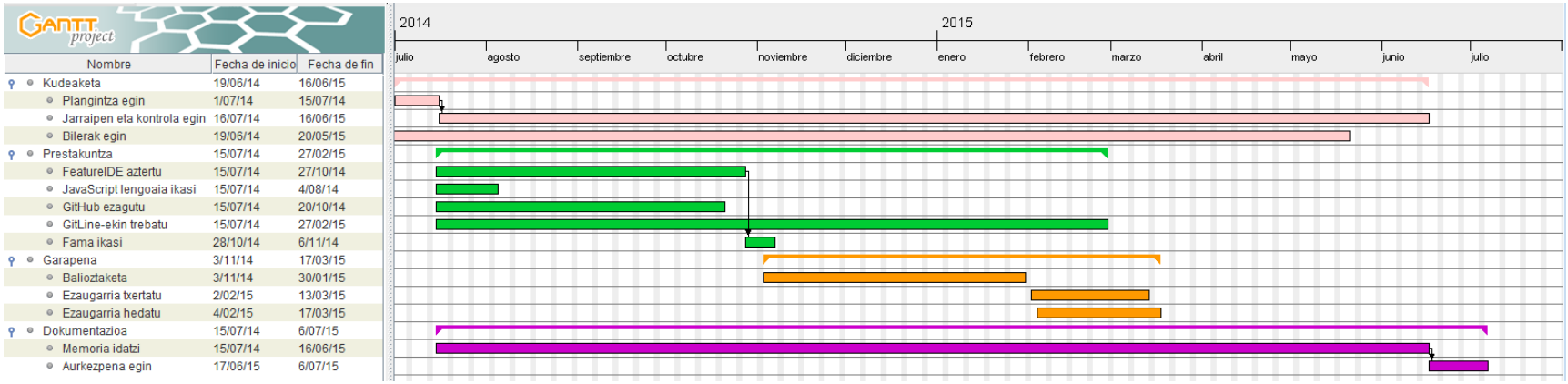
Proiektuan zehar komunikazioak aurrera eramateko metodoa posta elektronikoa eta bilerak izan dira. Posta elektronikoaren bidez gauzatutako komunikazioa ezin hobea izan da; garrantzitsuak ziren gaietarako erabili da, eta formaltasuna eta egokitasuna mantendu da.

Bilerak ere egin dira, hauetan garrantzizko erabakiak hartu eta lanaren jarraipena egin da. *10.2.1 Bilera aktak* atalean zehatzago azaltzen da bilera bakoitzean parte hartutako interesatuak eta aipatutako puntuak. Bilera batzuetan zuzendaria bakarrik, beste batzuetan zuzendaria eta ikerlaria, besteetan ikerlariak bakarrik.

Ikasleak eta ikertzaileak GitHub erabili dute kodea konpartitzeko.

### 10.2.1 Bilera aktak

Izandako bilera bakoitzeko, taula bat bete da ondorengo datuekin: eguna, ordua, iraupena, partaideak eta aipatutako puntuak. Bilera aktak C eranskinean ikus daitezke.




| Nombre | Fecha de inicio | Fecha de fin |
|---|---|---|
| Kudeaketa | 19/06/14 | 16/06/15 |
| Plangintza egin | 1/07/14 | 15/07/14 |
| Jarraipen eta kontrola egin | 16/07/14 | 16/06/15 |
| Bilerak egin | 19/06/14 | 20/05/15 |
| Prestakuntza | 15/07/14 | 27/02/15 |
| FeatureIDE aztertu | 15/07/14 | 27/10/14 |
| JavaScript lengoaia ikasi | 15/07/14 | 4/08/14 |
| GitHub ezagutu | 15/07/14 | 20/10/14 |
| GitLine-ekin trebatu | 15/07/14 | 27/02/15 |
| Fama ikasi | 28/10/14 | 6/11/14 |
| Garapena | 3/11/14 | 17/03/15 |
| Balioztaketa | 3/11/14 | 30/01/15 |
| Ezaugarria txertatu | 2/02/15 | 13/03/15 |
| Ezaugarria hedatu | 4/02/15 | 17/03/15 |
| Dokumentazioa | 15/07/14 | 6/07/15 |
| Memoria idatzi | 15/07/14 | 16/06/15 |
| Aurkezpena egin | 17/06/15 | 6/07/15 |

73. irudia: Egindakoaren Gantt diagrama.

## 10.3 Kalitatea

Atal honetan proiektuaren kalitatearen jarraipena azalduko da, bai proiektuarena eta bai produktuarena ere. Proiektuaren kalitatea neurtzeko irizpideak honako hauek izan dira:

- Proiektuaren kudeaketa:

  Proiektuaren kudeaketaren kalitate-maila neurtzeko elementu bakarra identifikatu zen: egindako bilerak. Planean ezarritakoaren arabera, kudeaketaren kalitatea altua izateko, interesatuekin egoeraren berri emateko maiztasunez bildu, bilera aurretik gai zerrenda interesatuei bidali eta bilera aktak egin beharra zegoen. Proiektuan zehar hiru baldintza horiek bete direnez, proiektuaren kudeaketaren kalitate-maila altua lortu da.

- Garatutako produktua:

  Garatutako produktuaren kalitate-maila altua lortzeko, garatu beharreko hiru funtzionalitateak ondo (ondo funtzionatzen dute) eta modu txukun batean (ondo egituratuta eta komentarioekin) inplementatuak egon behar dira. Irismenean zehaztutako hiru funtzionalitateak ondo eta modu txukun batean inplementatzeaz gain, funtzionalitate gehiago inplementatu dira, aldakia txertatu eta txerta daitezkeen ezaugarri mota guztiak hedatu, beraz garatutako produktuan kalitate-maila altua lortu da.

- Proiektuaren memoria:

  Proiektuaren memoriaren kasuan, bi ziren identifikatutako kalitate-elementuak: dokumentuaren edukia eta egitura.

  Edukiari dagokionez, kalitate-maila altua lortzeko, hauek ziren memoriak izan beharreko kapituluak: proiektuaren helburuen dokumentua, denboraren kudeaketa, eranskinak, glosategia, probak, ebazpenaren diseinua, ondorioak. Memoriak atal guztiak dituenez, kalitate-maila altua lortu da.

  Egiturari dagokionez, hauek ziren betebeharreko baldintzak kalitate-maila altua lortzeko: irudiek eta taulek oina daukate, algoritmo edo kode zatiak letra mota desberdin batean daude, zuzentzaile ortografikoa erabili da, atal eta azpi-atalak erabili dira, goialdean proiektuaren eta egilearen izenak agertzen dira, GAP-en araudiak finkatutako iturri tamaina eta mota erabili da, kapituluak ondo bereizita daude. Memoriak baldintza guztiak betetzen dituenez, kalitate-maila altua lortu da.

- Proiektuaren aurkezpena:

  Proiektuaren defentsarako gardenkien kalitate-maila altua izateko, ondorengo baldintzak bete behar dira: lehenengo gardenkian izenburua eta UPV/EHU-ren logotipoa agertzen dira, aurkibidea, gardenkiak zenbakitu, irudiak ondo ikusten dira. Proiektuaren aurkezpenak baldintza guztiak betetzen dituenez, kalitate- maila altua lortu da.

## 10.4 Arriskuak

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

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

Proiektuaren defentsaren atzerapenaren kasuan eta atazen estimazio okerraren kasuan, arriskua arazoa bihurtu da eta plangintzan zehaztu zen moduan, birplangintza bat egin behar izan da.

Proiektu honetan garatutako tresnak erabiltzen dituen zerbitzuekin (GitHub, Firefox… ) ez da arazorik egon.

# 11.KAPITULUA

# Ondorioak

Kapitulu honetan, proiektu osoaren ondorioak biltzen dira, baita etorkizuneko lanak ere.

## 11.1 Ondorioak

Proiektuaren helburua, GitLine-en [8] funtzionalitate batzuk inplementatzea zen. GitLine, GitHub-en funtzionalitate gehigarriak eskaintzen dituen Firefox-eko (37.0 bertsioan) plugin bat da, plugin honek SPL produktu eraikitzaileari laguntza eskaintzen dio ezaugarri biltegiak eta produktu biltegiak sortu eta beraien arteko sinkronizazioa mantentzeko. SPL edo software produktu-lerroak, antzekoak diren software sistemen bilduma bat eraikitzea dute helburu gisa, eta software artefaktuen multzo partekatu batetik abiatzen dira, produkziorako bide komunak erabiliz.

GitLine aplikazioari plangintzan adierazitako funtzionalitate guztiak inplementatu zaizkio: *balioztaketa*, *ezaugarria txertatu* eta *ezaugarria hedatu*. Funtzionalitate hauek garatzeko *java* [19], *javaScript* [11], *mozilla developer* [12]eta *GitHub Api* [7] erabili dira.

Lehenengo funtzionalitateari*, balioztaketa*-ri, dagokionez, hasieran sortu ziren zailtasunen aurrean soluzioak topatzeko gai izan da ikaslea. Irismenean finkatu zen erremintarekin zituen arazoak ikusirik (FeatureIDE erabiltzeko, klase dependentzia asko) beste baliabide bat aurkitu zuen helburuak bete ahal izateko, FaMa [2] erreminta. Behin FaMa aurkituta, konfigurazioak balioztatzen dituen ebatzailea sortu zuen. Ebatzailea eta GitLine elkartzeko modu ezberdinak aztertu ondoren, azkenik, GitLine-en dinamikoki sortzen diren interfazeak eta ebatzailearekin loturak inplementatu zituen.

Beste bi funtzionalitateen helburuak zabaldu egin dira proiektuan zehar. Azkenean, ondorengoa inplementatu da: derrigorrezko ezaugarri bat, aukerako ezaugarri bat edo aldaki bat (bakarrik or-inklusibo egitura batean) ezaugarri ereduan gehitzeko aukera eta ezaugarri hori produktu biltegietara hedatzeko aukera. Ezaugarri berri bat txertatzerakoan, FaMa bidez ezaugarri eredu

berria balioztatzen da. Ezaugarri berri bat txertatzerakoan, garatzaileak ezaugarri berria inplementatu behar duenez, ezin da txertaketa eta hedaketa era jarrai batean burutu. Egoera honen aurrean, txertaketa burutzerakoan jakinarazpen bat sortzen da eta garatzaileak ezaugarria inplementatu ondoren, jakinarazpena ireki eta bertan agertzen den *ForwardPropagation* botoiaren bidez hedaketa burutu dezake. Hedaketan, bakarrik ezaugarri berriaren gurasoa inplementatuta duten produktu biltegietara egiten dela ziurtatzen da.

Funtzionalitateak garatu baino lehen, beste pertsona batzuek garatutako kodeak begiratu behar izan ditu ikasleak: GitLine eta FeatureIDE. GitLine-ren kasuan, ikertzaileak asko lagundu dio ikasleari kodea menperatzeko, aldiz, FeatureIDE-ren kasuan, agian hasieratik kodea era automatiko batean aztertzen duen software bat erabili behar zuen ikasleak denbora aurrezteko. Mota horretako softwarerik ez erabiltzeagatik, FeatureIDE aztertzen inbertitutako denbora gehiegizkoa izan da eta beste erraminta bat, FaMa, erabili behar izan da funtzionalitateak garatzeko.

Kudeaketari dagokionez, hasieran egindako plangintzan proiektua otsailean amaituko zela finkatu zen, baina ikaslearen dedikazio faltarengatik eta izandako arazoengatik ekaina arte atzeratu egin da. Atzerapen honi esker, denbora gehiago egon da produktuaren kalitatea hobetzeko, funtzionalitate gehiago inplementatuz.

Laburbilduz, ikasleak ondorengo konpetentziak landu ditu: git eta GitHub-ekin lanean ibili ikertzailearekin kodea konpartitzeko; spl-etan eta bereziki ezaugarri ereduen balioztaketa ezagutu eta jorratu; *java script*-ekin, Firefox pluginekin eta GitHub pluginakin ibiltzea eta arazoen aurrean soluzioak topatzeko gai izatea. Gainera, bere denbora kudeatzen trebetasun handia lortu du, ia urtebetez aritu baita lanean eta denbora horretan era konstantean egin behar izan du lan.

## 11.2 Etorkizuneko lana

Etorkizunari begira, GitLine hobetzeko lerroa jarraitu beharko litzateke. Hauek dira hobekuntza posible batzuk.

*Balioztaketa* funtzionalitateari dagokionez, konfigurazioa aukeratzeko garaian ezaugarriak era hierarkiko batean agertzea, ezaugarrien arteko erlazioak (guraso-umeak) argiago ikusteko.

Ezaugarri biltegien bizi-zikloa handitzeari dagokionez, or-inklusibo egitura batean aldaki bat gehitzeko aukera gehitzea ondo legoke. Beste aukera bat, or egiturarik ez duen ezaugarri bati or egitura bat gehitzea da.

Ezaugarri ereduan ezaugarriak gehitzeaz gain, murriztapenak gehitzeko aukera ere gehitu daiteke GitLine aplikazioan.

# 12.KAPITULUA

# Bibliografia


- [1] AccessToken
  - https://help.github.com/articles/creating-an-access-token-for-command-line-use/
- [2] FaMa
  - http://www.isa.us.es/fama/?FaMa_Framework
- [3] FeatureIDE
  - http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/
- [4] Git
  - https://git-scm.com/
- [5] GitHub
  - https://github.com/
- [6] GitHub for Windows
  - https://windows.github.com/
- [7] GitHub API v3
  - https://developer.github.com/v3/
- [8] GitLine
  - http://letimome.github.io/GitLine/
- [9] Java
  - http://elvex.ugr.es/decsai/java/
- [10] JDepend
  - http://clarkware.com/software/JDepend.html
- [11] JavaScript
  - http://www.w3schools.com/js/
- [12] Mozilla Developer
  - https://developer.mozilla.org/es/
- [13]Onekin Talde ( UPV/EHU Euskal Herriko Unibertsitatea)
  - http://onekin.org/

- [14] The guidsl Tool
  - http://www.cs.utexas.edu/~schwartz/ATS/fopdocs/guidsl.html
- [15] Sat4j
  - http://www.sat4j.org/

# ERANSKINAK

# A. ERANSKINA

# Instalatzeko eskuliburua

Eranskin honetan, ikertzaileak eta ikasleak garatutako GitLine aplikazioa nola instalatu azalduko da.

GitLine, GitHub-en funtzionalitate gehigarriak eskaintzen dituen Firefox-eko plugin bat da, plugin honek SPL produktu eraikitzaileari laguntza eskaintzen dio, ezaugarri biltegiak eta produktu biltegiak sortu eta beraien arteko sinkronizazioa mantentzeko.

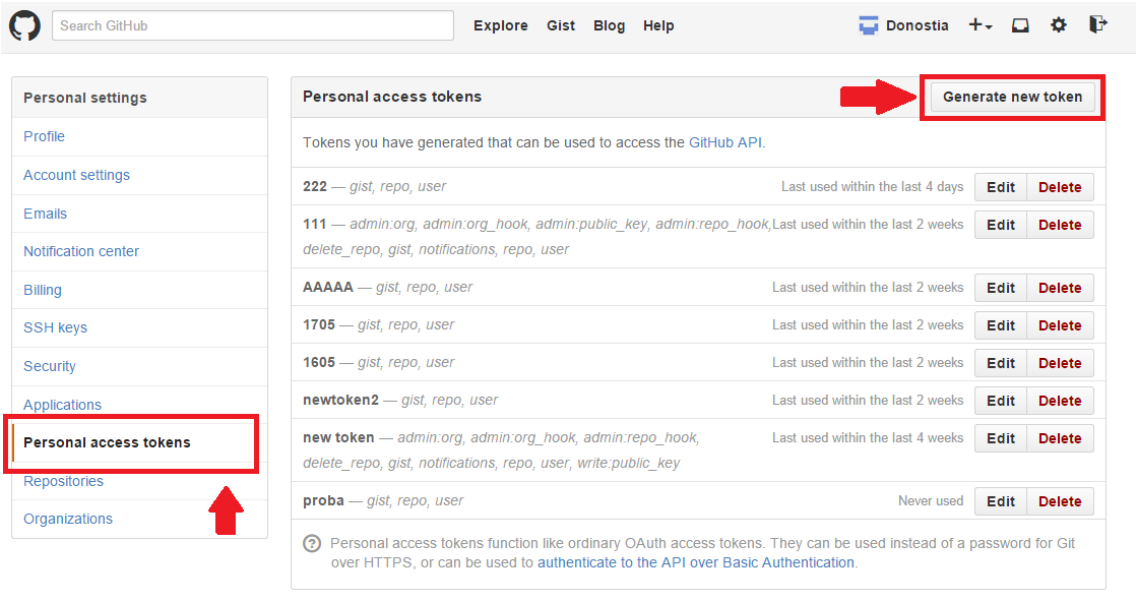
Hauek dira GitLine erabili ahal izateko aurrebaldintzak:

- Firefox-eko 37.0 bertsioa izatea.
- GitHub-en kontu bat izatea.

GitLine instalatzeko, GitHub kontuan, *access token* [1] (bezero eta zerbitzari gako parea da http bidez GitHub-en kautotzeko) bat sortu behar da. Ondoren, aplikazioaren direktorioa Firefox-eko direktorio zehatz batean kokatu behar da eta bukatzeko, Firefox arakatzailearen konfigurazioan bi aldaketa egin. Pauso hauek ondorengo puntuetan zehazten dira.

GitLine instalatzeko ondorego pausoak jarraitu behar dira:

1) GitHub kontuan*, acces token* bat sortu eta ondo gorde, behin bakarrik ikusteko aukera dago eta. Token-a sortzeko GitHub kontuan sartu eta *editProfile* sakatu. Bertan 74. irudian agertzen den interfazea ikusten da. Ezkerreko zutabean *Personal access tokens* sakatu eta ondoren *Generate new token*.



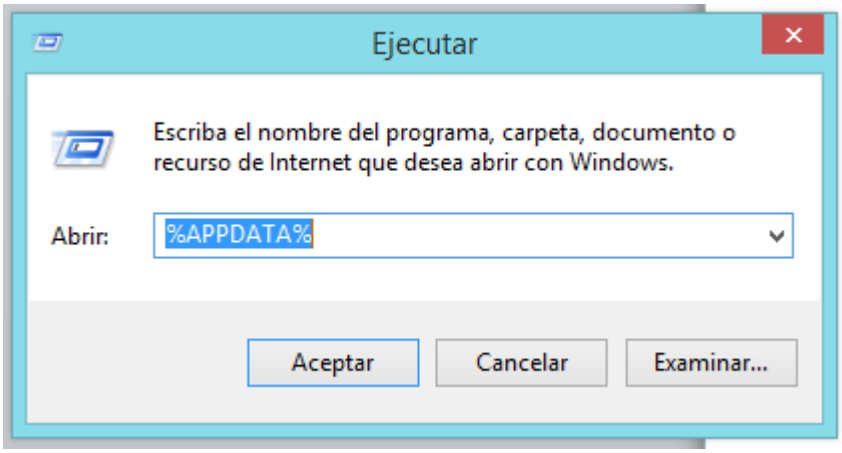

74. irudia: Token berria sortu.

2) GitLine proiektuko */content/githubdeltas.js* fitxategian, sortutako *acces token*-a pegatu. Zehazki, 75. Irudian agertzen den *DeltaUltils.getUserAccesToken* funtzioan, *your token here* jartzen duen tokian .

```
DeltaUtils.getUserAccessToken=function(){
    return "cda3c4c6d92c32677bb3cd0eb2c0a1f88fab4259"; // <--your token here
}
```

75. irudia: *getUserAccess* funtzioa.

3) Aplikazioa, Firefox plugin bat da. Beraz, proiektuaren direktorioa Firefox-eko *extensions* direktorioan kokatu behar da. Direktorio hori aurkitzeko, *Ejecutar* aplikazioa ireki eta *%APPDATA%* direktorioa ireki (76. irudia). Behin direktorio horretan kokatuta, */Roaming/Mozilla/Firefox/Profiles/xo8uhidn.default/extensions-era* joan eta aplikazioa bertan kokatu *scxmlGitDelta@onekin.org* izenarekin.

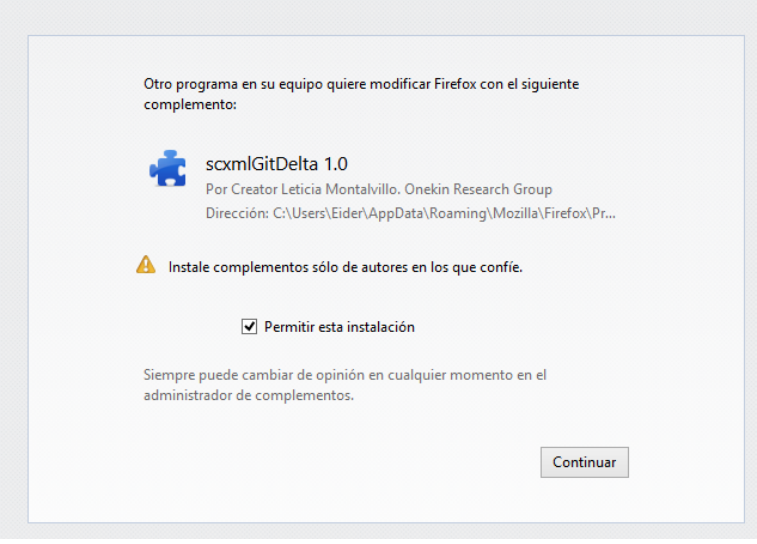



76. irudia: *Ejecutar* aplikazioa.

4) Firefox-eko bilatzaile lerroan, *about:config* bilatu eta ondorengo parametroak *false* moduan jarri:
   a. *security.csp.enable=false*
   b. *security.mixed-content.block_active_content=false*

5) Firefox exekutatu. Gauzak ondo egin baldin badira Firefox-ek plugin berria detektatuko du eta instalatzeko baimena eskatuko du (77. irudia).

6) Plugin-a instalatzerakoan instalazioa amaitzen da.




77. irudia: Plugin-a instalatzeko baimena.

# B.ERANSKINA

# Glosategia

## *A*

Aplikazio ingeniariak
*Application engineer* inglesez.

## *B*

Biltegia
*Repository* inglesez.

## *D*

Domeinuko ingeniariak
*Domain engineer* inglesez.

## *E*

Ebatzailea
*Solver* inglesez.
Ezaugarri biltegia
*Feature repository* inglesez.
Ezaugarria
*Feature* inglesez.

## *P*

Produktu biltegia
*Product repository* inglesez.

# C.ERANSKINA

## Bilera aktak

| | |
|---|---|
| Eguna: | 2014/06/25 |
| Ordua: | 11:30 |
| Iraupena: | 1o |
| **Partaideak:** | Arantza Irastorza, eta Eider Irigoyen |
| **Aipatutako puntuak:** | • Proiektuaren zuzendariak proiektuaren aurkezpena egin dio ikasleari.<br>• Zuzendariak plangintza egiteko adierazi dio ikasleari. |

| | |
|---|---|
| Eguna: | 2014/09/06 |
| Ordua: | 11:30 |
| Iraupena: | 45min |
| Partaideak: | Arantza Irastorza eta Eider Irigoyen |
| Aipatutako puntuak: | • Proiektuaren zuzendariak plangintzan egin beharreko zuzenketak adierazi dizkio ikasleari.<br>• Proiektuaren zuzendariak plangintza zuzentzeko eta berriz ere bidaltzeko adierazi dio ikasleari. |

| | |
|---|---|
| Eguna: | 2014/10/15 |
| Ordua: | 11:30 |
| Iraupena: | 25min |
| Partaideak: | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Proiektuaren zuzendariak plangintzan egin beharreko azken zuzenketak adierazi dizkio ikasleari.<br>• Ikasleak GitLine kodearen azterketarekin atzeratuta dabilela adierazi du, arazoak izan ditu exekutatzeko eta.<br>• Ikasleak astebete barru entregatu beharreko emangarriaren ("Erreminta eta baliabideak") egoeraren berri eman du. Dena ondo dabil, epean entregatzeko arazorik ez. |

| | |
|---|---|
| Eguna: | 2014/10/24 |
| Ordua: | 12:15 |
| Iraupena: | 45min |
| Partaideak: | Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Ikasleak arazoak ditu FeatureIDE-rekin, aurkitu du "solver"-ari egiten zaion deia, baina ez daki nola adierazi behar zaion ezaugarri eredua "solver" horri.<br>• Elkarrekin aztertu dute FeatureIDE eta bertsio ezberdinekin ari zirela konturatu dira, ikertzailearen kodea hobeto ulertzen da eta bertsio hori bidali dio ikasleari. |

| | |
|---|---|
| Eguna: | 2014/10/15 |
| Ordua: | 12:00 |
| Iraupena: | 1o eta 30min |
| Partaideak: | Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Ikertzailea ikaslearekin bildu da GitLine exekutatzeko arazoa konpontzeko, lortu du konpontzea.<br>• Ikertzaileak GitLine aplikazioaren egungo kodea azaldu dio ikasleari. Aldatu behar dituen kode zatiak eta alde batera utzi beharreko kode zatiak adieraziz, hau da, bere kodea non txertatu behar duen adierazi dio zehazki. |

| | |
|---|---|
| Eguna: | 2014/11/05 |
| Ordua: | 11:30 |
| Iraupena: | 45min |
| Partaideak: | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Ikasleak FeatureIDE-rekin izandako arazo guztiak adierazi dizkie.<br>• FeatureIDE alde batera utzita aztertu dituen beste baliabideak azaldu ditu ikasleak: .cnf fitxategiak, .m fitxategiak... Azkenean, aurkitutako konponbidea (Fama deituriko aplikazioa) azaldu du demo txiki baten bidez. Zuzendariari eta ikertzaileari konponbide egokia iruditu zaie.<br>• Aurkitutako konponbidea java-n idatzita dagoenez eta GitLine javaScript-en bi zatiak elkartzeko bi proposamen egin ditu ikasleak: .jar eta applet... Ikertzaileak applet-a baztertu dut. |

| | |
|---|---|
| Eguna: | 2014/12/29 |
| Ordua: | 12:00 |
| Iraupena: | 25min |
| Partaideak: | Arantza Irastorza eta Eider Irigoyen |
| Aipatutako puntuak: | • Proiektuaren zuzendariak "Erreminta eta baliabideak" emangarrian egin beharreko zuzenketak adierazi dizkio ikasleari.<br>• Ikasleak java-n ondo dabilen konfiguratzaile bat duela adierazi dio zuzendariari (baina ikerlariaren onarpenaren zain dago).<br>• Ikasleak zuzendariari konfiguratzailearen lehen prototipoa bidaliko dio. |

| | |
|---|---|
| Eguna: | 2015/01/08 |
| Ordua: | 11:00 |
| Iraupena: | 1 ordu |
| Partaideak: | Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Garatu duen konfiguratzailea ikerlariaren kodean integratzeko, ikasleak hiru proposamen aurkeztu ditu.<br>• Ikerlariak aukerarik "integratuena" aukeratu du: interfazea javaScript bidez eta FaMa deiak applet baten bidez. Proposamen honetan, FaMa deitzerakoan arazoa gertatzen da (baimen arazoa dela suposatzen da). Ikerlariak arazoa konpontzen lagunduko diola adierazi dio ikasleari. |

| | |
|---|---|
| Eguna: | 2015/01/08 |
| Ordua: | 12:00 |
| Iraupena: | 30min |
| Partaideak: | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Zuzendariari aurreko bileran hartutako erabakiak adierazi zaizkio.<br>• Prototipoa epe labur batean konpontzen ez bada, FaMa script batetik exekutatzea proposatu da.<br>• Seguruenik otsailean proiektua aurkezteko denborarik ez da egongo. Batez ere, integrazio arazoak luzatzen badira. |

| | |
|---|---|
| Eguna: | 2015/01/14 |
| Ordua: | 10:00 |
| Iraupena: | 1o eta 15min |
| Partaideak: | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Fama deiak applet baten bidez egikaritzearen aukera baztertu, sortzen den arazoa (baliteke baimen arazoa izatea) konpontzeko denbora Fama exekutatzeko script-a egiteko baino denbora gehiago eramango duela aurreikusten da eta.<br>• GitLine-tik script-a exekutatzeko jarraitu beharko dituen pausuak adierazi dizkio ikerlariak ikasleari.<br>• Ikerlariak "Mozilla developer guide" begiratzeko gomendatu dio ikasleari.<br>• Zuzendariak birplangintza egiteko adierazi dio ikasleari. |

| | |
|---|---|
| Eguna: | 2015/02/04 |
| Ordua: | 15:00 |
| Iraupena: | 1o |
| Partaideak: | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Ikasleak inplementatutako lehenengo funtzionalitatea (balioztaketa) erakutsi die ikerlariari eta zuzendariari. Ados daude egindakoarekin.<br>• Inplementatu beharreko beste bi funtzionalitateak zehaztu dira. Aldaketa txiki bat egon da hasieran finkatu zenarekin, *aldakia gehitu* izan beharrean, *aukerako ezaugarria gehitu* izango da momentu honetan, baina denbora izan ezkero hirurak inplementatuko dira:<br>❖ Derrigorrezko ezaugarria gehitu: derrigorrezko ezaugarria gehitu ("•" bidez adierazitakoak) eta aurretik sortutako produktu guztietara zabaldu aldaketa.<br>❖ Aukerako ezaugarria gehitu: aukerako ezaugarria gehitu ("o" bidez adierazitakoak) eta aurretik sortutako produktuetatik, erabiltzailean aukeratzen dituenei, aldaketa zabaldu.<br>❖ Aldakia gehitu: aldaki bat gehitu "or" egitura bati eta aurretik sortutako produktuetatik, erabiltzaileak aukeratzen dituenei, aldaketa zabaldu. |

| | |
|---|---|
| Eguna: | 2015/02/04 |
| Ordua: | 16:00 |
| Iraupena: | 1o 15min |
| Partaideak: | Leticia Montalvillo eta Eider Irigoyen |
| Aipatutako puntuak: | • Inplementazio berriarentzako beharreko funtzioak eta baliabideak ("GitHub Developer Guide") adierazi dizkio ikerlariak ikasleari. |

| | |
|---|---|
| Eguna: | 2015/02/18 |
| Ordua: | 13:00 |
| Iraupena: | 1o 15min |
| **Partaideak:** | Leticia Montalvillo eta Eider Irigoyen |
| **Aipatutako puntuak:** | • Ikasleak inplementatu duena adierazi dio ikerlariari<br>❖ model.xml aldatzen da.<br>❖ model.xml berria Fama bidez balioztatzen da.<br>❖ branch berria sortzen da.<br>❖ interfazeak.<br><br>• Ezaugarri eredu bat aldatzerakoan, hedaketa nola egin zehaztu:<br>1) Zein produktuetara hedatuko den aukeratu:<br>❖ Derrigorrezko ezaugarria gehitu bada: ezaugarri berriaren aita dituzten produktu guztietara hedatu .<br>❖ Aukerako ezaugarria gehitu bada: ezaugarri berriaren aita dituzten produktu guztien artean erabiltzaileak aukeratuko du zein produktuetara hedatu .<br>❖ Aldakia gehitu bada: ezaugarri berriaren aita dituzten produktu guztien artean erabiltzaileak aukeratuko du zein produktuetara hedatu . Momentuz bakarrik aldaki bat baino gehiago onartzen dituen egituran inplementatuko da.<br>❖ *Restriction*-ak alde batera utziko dira.<br>2) Hedaketa egin beharraren oharra (*issue*) itxi. |

| | |
|---|---|
| Eguna: | 2015/03/03 |
| Ordua: | 12:30 |
| Iraupena: | 1o 15min |
| **Partaideak:** | Leticia Montalvillo eta Eider Irigoyen |
| **Aipatutako puntuak:** | • Interfazean botoiak txertatzeko arazoak dituela adierazi dio ikasleak ikerlariari eta arazoa bilatzen egon dira. Ez dute konpontzea lortu.<br>• *ProductFork* botoia sakatzerakoan, batzuetan sortzen diren arazoak aztertzen egon dira. Biltegiak nola egon behar diren fork-ak egiteko adierazi dio ikerlariak ikasleari. Ikasleak berriz ere probatzearen pendiente gelditu da arazoa.<br>• Ikerlariak inplementatutako *forwardPropagation* funtzioari nola deitu aztertzen egon dira. |

| | |
|---|---|
| Eguna: | 2015/03/17 |
| Ordua: | 15:00 |
| Iraupena: | 1o 15min |
| **Partaideak:** | Arantza Irastorza, Leticia Montalvillo eta Eider Irigoyen |
| **Aipatutako puntuak:** | • Zuzendariari proiektua erakutsi<br>• Hedapena egiterakoan, ezaugarrien arteko erlazioak kontuan izan → alde batera utzi.<br>• Inplementazioa bukatutzat eman da. |

| | |
|---|---|
| Eguna: | 2015/05/20 |
| Ordua: | 10:30 |
| Iraupena: | 1o |
| **Partaideak:** | Arantza Irastorza, eta Eider Irigoyen |
| **Aipatutako puntuak:** | • Ikasleak memoriaren inguruan zituen duda batzuk galdetu dizkio zuzendariari.<br>• Ikasleak bere egoeraren berri eman dio zuzendariari. |