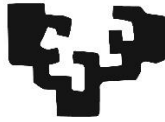


eman ta zabal zazu



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

Betekizunen trazabilitate inpaktu-analisi  
automatikoa eta dokumentazio formalaren  
sorkuntza automatikoa modeloetan oinarritutako  
ekosistemetan.

Gradu Amaierako Lana



Jon Legarda Gonzalez

Tutoreak: Juan Manuel Pikatza Atxa eta Mikel Larramendi

Ingeniaritza Informatikako Gradua

Euskal Herriko Unibertsitatea

Donostia, 2019ko iraila



## Esker onak

Astunegia izatearen beldur banaiz ere...

Eskerrik asko *CAF Signalling*eko lankideei, enpresara sartu bezain laster eskainitako nuen laguntzagatik. Bereziki eskerrak eman *ATO* Taldeko kideei: Algorriko *tupperen* artean izandako hizketaldi eta barre guztien errudun.

Gradu osoan zehar egindako lagun guztiei. Ezin izenik esan, norbait ahaztearen beldur egonda.

Lagunei. *Semper fidelis.*

Proiektuaren zuzendari izan diren Jon Goya eta Mikel Larramendiri, profesionaltasuna eta edozein momentuetan laguntzeko prestutasunarengatik.

Iñaki Berriotxoari, proiektuaren momentu guztietan emandako aholkuengatik eta adierazitako adiskidetasunagatik.

Juan Manuel Pikatza Atxari, proiektu honetan eguneko 24 orduetan laguntza, aholkuak eta dena delakoa behar izan dudanean izandako pazientziagatik. Ezin itzuli emandako eta irakatsitako guztia.

Alba, Kike, Arrate.

*“Hemen su horren bueltan dantzan gabiltza!”*

Jon Legarda Gonzalez



## Laburpena

Proiektu honetan software erduetan oinarritzen den ekosistema baten bi funtzionalitate gehitzeko soluzioa proposatu eta inplementatzen da: betekizunen trazabilitatea eta dokumentazio formalen sorrera automatikoa. Erduetan oinarritzen den ekosistema edo *framework* batek software generikotik espezifikora joatea ahalbidetzen du. Proiektu honen bitartez *CAF Signalling* enpresaren produktuetako bat definitzen duen ekosistematik betekizunen trazabilitatea zaindu eta dokumentazio automatikoa sortzen da. Lan honen bitartez aipatutako bi helburu horiek lortzeko bidea deskribatu egiten dira eta horietatik abiatuta sortutako bi produktuen inplementazioa ere azaltzen da. Proiektuaren garapenerako, *OpenUP* metodologia erabili da eta proiektuaren dokumentazioa eta memoria idazteko *CCII-N2016-02* estandarra erabili da.

**Hitz gakoak:** dokumentazio formala, betekizunak, trazabilitatea, automatizazioa.

## Resumen

Este proyecto propone e implementa una solución para añadir dos importantes funcionalidades a un ecosistema basado en modelos de software: trazabilidad de requisitos y creación de documentación formal automática. El *framework* o ecosistema que se basa en modelos de software ofrece la posibilidad de obtener un software específico a partir de un software más genérico. Mediante este proyecto se obtiene una solución para trazar requisitos y crear documentación formal para un importante producto de la empresa *CAF Signalling*. En este trabajo se describe el camino realizado y la implementación de la solución para lograr los dos citados objetivos. Para el desarrollo metodológico del proyecto, se ha usado la metodología *OpenUP* y el estándar *CCII-N2016-02* para la documentación y memoria del proyecto.

**Palabras clave:** documentación formal, requisitos, trazabilidad, automatización.

## Abstract

This project proposes and implements a solution that adds two important functionalities to a *framework* based on software models: automatic requirements traceability and automatic generation of formal documentation. The model-based *framework* enables the generation of specific software taking into account more generic software. Thanks to this project it is obtained a solution to trace requirements and create documentation for an important product of *CAF Signalling*. In this work it is described all the decisions taken to construct the solution as well as the implementation of it in order to achieve the two objectives. In order to use a well-known methodology, the project has been developed based on *OpenUP* and the documentation and memory has been written according to the standard *CCII-N2016-02*.

**Key words:** formal documentation, requirements, traceability, automatization.

## Aurkibide Orokorra

Esker onak.....	iii
Laburpena.....	v
Resumen .....	v
Abstract .....	vi
Aurkibide Orokorra .....	vii
Grafikoen aurkibidea .....	ix
Irudien aurkibidea.....	xi
Taula aurkibidea.....	xiii
1 Sarrera .....	1
2 Proiektuaren Xedea .....	7
3 Aurrekariak .....	9
4 Egungo egoera.....	13
4.1. Egungo Egoeraren Deskribapena.....	13
4.2 Identifikatutako Urritasunen Laburpena.....	14
5 Arauak eta Erreferentziak .....	15
5.1 Xedapen Legalak eta Aplikatutako Normak .....	15
5.2 Bibliografia.....	15
5.3 Metodoak, Tresnak, Ereduek eta Metrikak.....	18
5.3.1 Metodoak eta Tresnak.....	18
5.3.2 Eredua, Metrika eta Prototipoa.....	25
5.4 Proiektu Zeharreko Kalitate Plana.....	26
5.5 Beste Erreferentziak .....	27
6 Definizioak eta Laburdurak .....	31
7 Hasierako Betekizunak.....	39
8 Irismena.....	43
9 Hipotesiak eta Murriztapenak .....	45
10 Aukeren Egingarritasun Ikerketa.....	47

11	Proposatutako Sistemaren Deskribapena.....	55
12	Arrisku Analisia .....	75
13	Proiektuaren Antolaketa eta Kudeaketa.....	79
13.1	Antolaketa .....	79
13.2	Kudeaketa.....	79
14	Denbora Planifikazioa .....	87
15	Aurrekontua.....	95
16	Oinarrizko Dokumentuen Ordena.....	97
	ERANSKINAK .....	98
I.	Memoriaren Eranskinak.....	98
A1:	Sarrerako Dokumentazioa .....	98
A2:	Analisi eta Diseinua .....	98
A3:	Tamaina eta Esfortzu Estimazioa.....	98
A4:	Kudeaketa Plana .....	98
A5:	Segurtasun Plana.....	99
A6:	Beste Eranskinak .....	99
II.	Sistemaren Espezifikazioa .....	101
III.	Aurrekontua.....	101
IV.	Ikerlanak .....	101



## Grafikoen aurkibidea

1 grafikoa: Arriskuen probabilitatea eta inpaktuaren arteko lotura. ....	77
2 grafikoa: Proiektuaren LDE diagrama, beharrezko entregak erakusteko. ....	81
3 grafikoa: Atazen Gantt diagrama. ....	93



## Irudien aurkibidea

1 irudia: ATOren automatizazio-mailak.....	3
2 irudia: BETRADOK webgunearen hasierako interfazea. ....	4
3 irudia: BETRADOKeko webgunearen menu osoa. ....	5
4 irudia: CAMELen interfaze grafikoa.....	20
5 irudia: Draw.io-ren interfaze grafikoa. ....	21
6 irudia: GitLaben logoa.....	21
7 irudia: IBM Rational Rhapsodyko hasierako interfaze grafikoa.....	22
8 irudia: IBM Rational DOORSen interfaze grafikoa. ....	23
9 irudia: PlantUMLren lengoain idatzitako sekuentzia-diagrama, adibidea.....	24
10 irudia: V ereduaren errepresentazioa.....	37
11 irudia: DOORSen definitutako traza: B2.09.....	41
12 irudia: Irismen-diagrama.....	44
13 irudia: PlantUMLen diagramak sortzeko lengoaiaren adibidea. ....	51
14 irudia: PlantUMLren sekuentzia-diagramaren adibide bat. ....	51
15 irudia: "V" ereduaren errepresentazioa.....	55
16 irudia: CAMEL-Sistemak funtzionamendu orokorra.....	56
17 irudia: DocGenerator sistemaren arkitektura orokorra. ....	57
18 irudia: DocGeneratorren erabilpen kasuen eredia. ....	58
19 irudia: DocGeneratorreko aktibitate-diagrama (analisi-mailan). ....	58
20 irudia: DocGeneratorreko klase-diagrama (analisi-mailan).....	59
21 irudia: DocGeneratorreko analisi ereduko sekuentzia-diagrama (analisi-mailan). ....	60
22 irudia: DocGeneratorren pakete-diagrama (implementazio-mailan).....	60
23 irudia: RequirementTracerreko arkitekturaren diagrama.....	63
24 irudia: RequirementTracerren antolaketa hiru aldetan. ....	64
25 irudia: RequirementTracerren erabilpen kasuen eredia.....	65
26 irudia: RequirementTracerreko aktibitate-diagrama (analisi-mailan). ....	65
27 irudia: RequirementTracerren klase-diagrama (analisi-mailan).....	66

28 irudia: RequirementTracerreko sekuentzia-diagrama (analisi-mailan). .....	66
29 irudia: RequirementTracer barruko parametroen eskema. ....	67
30 irudia: RequirementTracerreko pluginaren pakete-diagrama. ....	68
31 irudia: RequirementTracerreko DXL aldeko antolaketa. ....	69
32 irudia: Continious Integrationekin exekuzioaren eskema. ....	72
33 irudia: Proiektuko giza baliabideen eskema orokorra. ....	83
34 irudia: Emangarrien araberako LDE diagrama. ....	88

## Taula aurkibidea

1 taula: Kalitatea bermatzeko rolak eta ardurak.....	27
2 taula: Laburduren taula. ....	38
3 taula: DocGeneratorentzako betekizun funtzionalak eta ez-funtzionalak. ....	40
4 taula: RequirementTracer-en helburuko betekizun funtzionala eta ez-funtzionalak...	41
5 taula: Parametroen hitzarmena. ....	59
6 taula: DocGenerator sistemaren proben taulak.....	62
7 taula: RequirementTracerren parametroen taula. ....	65
8 taula: RequirementTracerren proba kasuen taula. ....	71
9 taula: Identifikatutako arriskuen zerrenda.....	76
10 taula: Arriskuen probabilitatea eta inpaktua adierazteko taula. ....	76
11 taula: Arriskuak mitigatzeko estrategia garrantziaren arabera ordenatuta. ....	78
12 taula: Mugarri garrantzitsuen taula. ....	87
13 taula: Formazio eta lan-orduen banaketa (I).....	89
14 taula: Formazio eta lan-orduen banaketa (II).....	89
15 taula: Lan-Ataza eta orduen taula.....	91
16 taula: Iterazio banaketaren taula. ....	92
17 taula: Aurrekontuaren taula.....	96



## 1 Sarrera

Dokumentu hau Jon Legarda Gonzalez-en Ingeniaritza Informatikako Gradu Amaierako Lanaren memoria da, *CAF Signalling* enpresan egindakoa, Juan Manuel Pikatza tutore lanetan izan duena eta Euskal Herriko Unibertsitateko (UPV-EHU) Donostiako Informatika Fakultatean landutakoa.

Dokumentu honetan aurkituko da "BETRADOK"<sup>1</sup> proiektua gauzatzeko emandako pausu guztiak, zeinak zehaztuta egongo diren *OpenUP*<sup>2</sup> metodologia erabilita. Gainera, dokumentazio formala eta profesionala lortzeko asmoz, ezaguna den *CCII-N2016-02* arau estandarra erabili da dokumentazio honen forma eta atalak definitzeko. Arau hura *Informatika Ingeniaritzako Elkargo Profesionalen Kontseilu Nagusia (CCII) erakundeak*<sup>3</sup> ateratako estandarra da, UNEk (*Una Norma Española*) zehaztutako *UNE 157801:2007*<sup>4</sup> arau estandarra berrituz. Beraz, esan daiteke dokumentazio honek ongi betetzen dituela informatikako ingeniaritza proiektu batek beharrezko dituen betekizunak.

Aipatutako "BETRADOK" hitz-gakoa proiektuaren akronimo edo ezizena da. Titulu ofiziala honako hau da: **betekizunen trazabilitatea inpaktu-analisi automatikoa eta dokumentazio formalaren sorkuntza automatikoa modeloetan oinarritutako ekosistemetan.**

## Testuingurua

Software kalitatea oso garrantzitsua da software produktu seguru eta trinkoa lortzerako garaian. Software sistemaren kalitatea bermatzea funtsezkoa da produktua saltzeko. Batzuetan softwarearen kalitatea beste batzuetan baino garrantzitsuagoa da; izan ere, ez da berdina webgune baten software kalitatea edo tren bat automatikoki gidatzen duen software baten kalitatea.

---

<sup>1</sup> BETRADOK: proiektuaren izena.

<sup>2</sup> OpenUP: Siglak (ingelesez): *Open Unified Process*. RUP (*Rational Unified Process*) metodologiaren azpimultzo bat da. Proiektu informatiko batean kokatzeko eta informazioa antolatuta izateko balio du. Horretaz aparte, proiektuaren elaborazio fasean produktua zein izan daitekeen edo zein bidetik lortu daitekeen definitzeko balio du.

<sup>3</sup> CCII: Sigla (gaztelera): *Consejo de Colegios de Ingeniería Informática*. Estatu-mailan informatika ingeniari guztiak errepresentatu eta bateratzen dituen antolakundea da. Ikus, gainera: [CCII, webgunea](#).

<sup>4</sup> <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0039577>

## Dokumentazioa software kalitatea bermatzeko

Hori lortzeko proiektuak eta sistemak ondo dokumentatuak edukitzeak berebiziko garrantzia du, garatzaile eta analistek erroreak identifikatu eta soluzioak ahalik eta azkarren eta trinkoen sortzeko eta integratzeko. Dударik ez dago software ahaltsu bat garatzeko eta mantentzeko dokumentazio on eta eguneratua behar dela. Horren ildotik, oso garrantzitsua den hitz-gako bat automatizazioa da. Hainbat arloren testuinguruan azaldu daiteke automatizazioa, baina kasu honetan dokumentazioaren munduan esan dezakegu hurrengoa: software edo sistema handi baten dokumentazioa zenbat eta automatizatuen, orduan eta hobeto.

## Software baten betekizunak

Horretaz gain, oso garrantzitsua da software sistema baten betekizunak (*requirements*<sup>5</sup>) ondo betetzea. Betekizunak bezeroak zehazten dituen sistema baten propietateak dira. Askotan bezeroen gurak izan daitezke, eta beste batzuetan derrigorrezko murriztapenak beste faktore askorengatik. Halaber, betekizunek zehazten dute software sistema baten segurtasuna.

## Industria eta Trengintza

Industria kalitatezko software segurua lortzeko helburua duen arlo ezagun bat da. Industrian beharrezkoa da produktu seguruak eta kalitatezkoak lortzea. Horren adibide argi bat trengintza da.

Trengintzaren industria arlo zabala da: trenak ekoiztu, mantendu edo berria den teknologiarekin ekipatu dira horren adibide. Zentzu horretan, trenetan *safety*<sup>6</sup> kontzeptua derrigorrez aipatu behar den hitza da. Trenetan pertsonak bidaiatzen dute eta beraien bizitzekin ezin da jolastu. Automatizazioa hain presente dugun mundu honetan, trengintza ere ez da salbuespen bat. Geroz eta tren independenteagoak fabrikatu egiten dira eta, beraz, automatikoki gidatzeko gai diren trenak, gidaririk gabekoak.

Tren bat gidaririk gabe eta automatikoki gidatzeko software eta hardware sistema ahaltsua funtsezkoa da, *testing*<sup>7</sup> handia pasa duena eta kalitate handiko osagaiak dituena. Horren harira, proiektu honen helburua bikoitza da: alde batetik, software baten milaka betekizunak ondo betetzen direla eta horien trazabilitatea ziurtatzea eta, bestetik, software sistemaren osagai bakoitzaren dokumentazioa automatikoki lortzea.

---

<sup>5</sup> *Requirements*: Sistema edo aplikazio baten betekizunak.

<sup>6</sup> *Safety*: Euskaraz: software segurtasuna. Kasu honetan, pertsonen bizitzaz arduratzen den software sistema baten segurtasuna da.

<sup>7</sup> *Testing*: Software sistema baten funtzionalitatea probatzeko proba-multzoa.



## CAF Signalling eta ATO

Proiektu hau *CAF Signalling* enpresan garatu da. *CAF Signalling* CAF enpresa nagusiaren filiala da, Donostian (Euskadi) eta Alcobendasen (Madril) egoitzak dituena. Trenbide zirkulazioko seinaleztapen-sistemen garapenean lan egiten du: fabrikazioa, horniduran eta noski, mantentze-lanetan ere. Seinaleztapeneko irtenbide integralak eskaintzen ditu azpiegituratan eta material mugikorrean, Espainian eta nazioartean.

*CAF Signalling*-en estrategia hiru zutabe sendoetan oinarritzen da: batetik, berezko teknologiako produktu aurreratuak izatea, soluzio integralen baitan. Bigarrenez, esperientzia handia seinaleztapeneko ingeniartzan. Hirugarrenez, proiektu konplexuak kudeatzeko eta gauzatzeko gaitasun handia.

Enpresa honen ezagutza teknologikoa oso handia da. Haren proiektu garrantzitsuetako bat "ATO"<sup>8</sup> soluzio integrala da. *ATO* (ingelesez, *Automation Train Operation*) trenak automatikoki eta gidariaren kontrolik gabe gidatzen duen sistema da. Horretarako *embedded*<sup>9</sup> arloan software asko landu egiten da eta software horren garapenean laguntzeko tresnak ere bai. Ulergarria denez, tren bat automatikoki eta gidariaren laguntzarik gabe gidatzeko sistema oso ahaltsua behar da eta internazionalki ezagunak diren espezifikazioak bete behar ditu.

TABLE I. GRADES OF AUTOMATION					
Grade of Automation	Train Operation	Setting train in motion	Stopping Train	Closing Doors	Operation in the event of disruption
GoA1	ATP with Driver	Driver	Driver	Driver	Driver
GoA2	ATO with Driver	Automatic	Automatic	Driver	Driver
GoA3	Driverless	Automatic	Automatic	Train Attendant	Train Attendant
GoA4	Unattended	Automatic	Automatic	Automatic	Automatic

1 irudia: ATOren automatizazio-mailak.

<sup>8</sup> ATO: Sigla (ingelesez): *Automatic Train Operation*. *Auriga ATO* da izen osoa da eta tren bat automatikoki eta gidariaren kontrolik gabe gidatzea ahalbidetzen du hanbat automatizazio-mailatan. ERTMSaren interoperagarritasun espezifikoetan oinarritutako *GoA2* automatizazioa eskaintzen du. Sistema bi ataletan banatzen da: trenbideko *ATOa* eta ontziratutako *ATOa*. Proiektuan maiz agertzen den kontzeptua da; izan ere, *CAF Signalling* enpresan garatzen ari den sistema da eta proiektua bere testuinguruan azaltzen da.

<sup>9</sup> *Embedded*: Euskaraz: txertatuta. Software sistema bat txertatuta dela esaten da, funtzio espezifiko bat edo gehiago egiteko diseinatuta dagoenean, normalean denbora-errealako konputazio sistemetan.

CAF Signallingen ATO sistema garatzen hasi zenean funtsezkoa ikusi zuen software handi horren garapenerako tresna ahaltzu bat garatzea sistema horren espezifikazioan laguntzeko.

Hori dela eta, enpresak CAMEL<sup>10</sup> izeneko ekosistema bat garatzen hasi zen. CAMEL ATOaren funtzionalitatea definitzeko balio du, UML eta SysML modelatzeko erabiltzen diren tresnak baina gauza ahaltzuagoak egiten laguntzen duena. Enpresak produktu baten bizitza-zikloan definituta daukan bidean laguntzeko tresna da CAMEL. Proiektu hau CAMEL ekosisteman oinarrituta garatu da. Hurrengo ataletan ongi definituko da CAMEL eta proiektuaren lotura.

## Proiektuaren Webgunea

BETRADOK deitutako proiektu honen dokumentazioa biltzeko webgune bat erabili egin da, bezeroak eskura izan dadin proiektuaren garapenean zehar erabilitako eta sortutako dokumentu guzti-guztiak. Webgunea honakoa da: <http://betradok.000webhostapp.com/>

Ezkerreko menuari esker, proiektuan dauden hainbat motatako dokumentu lortu daitezke: memoria, eranskinak, barne kudeaketako dokumentuak etab. Gainera, webguneak bideo tutorial bat dauka webgunea nola erabili behar den jakiteko, baita egilearen informazio orokorra ere.

Hasiera batean aurkituko dugun orri nagusia 2. irudikoa da. Irudi horretan funtsezko bi atal agertzen dira: ezkerreko menua eta zati nagusia. 3. irudian zehatzago ikusi daiteke ezkerreko menua desplegatuta dauden aukera guztiak.



**BETRADOK: Betekizunen trazabilitate inpaktu-analisi automatikoa eta dokumentazio formalaren sorkuntza automatikoa modeloetan oinarritutako ekosistemetan.**

- BETRADOK proiektua
  - Portada
  - Laburpen Posterra
  - Sideo-tutoriala
  - Aurkibide orokorra
  - Memoria
  - Memoriaren eranskinak
  - Sistemaren Espezifikazioa
  - Aurrekontua
  - Itxerlanak
  - BETRADOK: Barne Kudeaketa
    - Proiektuaren Barne Kudeaketa

**BETRADOK: Betekizunen trazabilitate inpaktu-analisi automatikoa eta dokumentazio formalaren sorkuntza automatikoa modeloetan oinarritutako ekosistemetan.**

 <b>JON LEGARDA GONZALEZ</b> Tutorea (UPV-EHU): Juan Manuel Pikatza Tutorea (CAF Signalling): Mikel Lerramendi	 Universidad del País Vasco Euskal Herriko Unibertsitatea	
<b>HELBURUAK</b>	<b>2019ko Iraila</b>	<b>MOTIBAZIOA</b>
Implementazio eta diseinu edizio baten aurrean inpaktu-analisia egiteko betekizunen trazabilitate automatikoa burutzea.		Software garapenerako prozesuan, konponenten implementazioan egon daitezkeen aldatetak eta sistema-mailan dauden betekizunak paretatzea eta lotura horren kontrola egin.
Dokumentazio formalaren sorrera automatikoa modelo espezializatuaren informazioa erabiliz eta software garapenerako fase desberdinetarako.		"Model Based Engineering" araberako software produktu baten diseinu eta implementazioan diogen informazioaren irakurketa erraztu eta software analisi erraza egitea diagrama eta beharrezko diren kontzeptuen bisualizazioa erraztu.
<b>Enpresa ikuspenetik: "Software Kalitatea hartatu eta Software Segurua lortu"</b>		

2 irudia: BETRADOK webgunearen hasierako interfazea.

<sup>10</sup> CAMEL: Akronimoa (ingelesez): CAF Model Editor & Language. CAF Signalling-ek sortutako sistema da, ATOren funtzionalitatea definitzeko pentsatua, UML eta SysML modelatzeko erabiltzen diren tresnak baino gauza ahaltzuagoak egiten laguntzen duena. CAMEL proiektu honetan oso garrantzitsua da; izan ere, proiektua CAMEL sistemaren baitan egiten da.

**BETRADOK proiektua**

- Portada
- Laburpen Posterra
- Bideo-tutoriala
- Aurkibide orokorra
- Memoria
  - Sarrera
  - Proiektuaren Xedea
  - Aurrekaririk
  - Egungo egoera
  - Arauk eta erreferentziak
  - Definizioak eta laburdurak
  - Hasierako betekizunak
  - Insmena
  - Hipotesak eta Murriztapenak
  - Aukeren kerketa eta egingarritasuna
  - Proposatutako sistemaren deskribapena
  - Arrisku analisia
  - Proiektuaren antoiaketa eta kudeaketa
  - Denbora planifikazioa
  - Aurrekontuaren laburpena
  - Dokumentuaren Lehentasun Ordena
- Memoriaren eranskinak
  - A1 - Sarrerako Dokumentazioa
  - A2 - Anlisi eta Diseinua
    - Arkitectura
      - Arkitectura Koademoa I
      - Arkitectura Koademoa II
    - A3 - Tamaina eta Esfortzu Estimazioa
    - A4 - Kudeaketa Plana
      - Integrazioaren Kudeaketa
      - Irsmenaren Kudeaketa
      - Epeen Kudeaketa
      - Produktuaren Kostuen Kudeaketa
      - Kalitate Kudeaketa
      - Geza Babaldiaren Kudeaketa
      - Komunikazioen Kudeaketa
      - Arriskuaren Kudeaketa
      - Erosketen Kudeaketa
      - Interesatuen Kudeaketa
      - A5 - Segurtasun Plana
  - A6 - Beste Eranskinak
    - Hedapena
      - Produktuaren dokumentazioa
      - Erabilizale dokumentazioa - I
      - Erabilizale dokumentazioa - II
      - Backout Plana
      - Hedapen Plana
      - Azpiegitura
    - Garapena
      - Buld
      - Garatzaleen Probak
      - Diseinua - I
      - Diseinua - II
    - Ingurunea
      - Proiektuak Definitutako Prozesua
      - Tresnak
      - Garapen Kasua
    - Implementazioa
      - Implementazioa - I
      - Implementazioa - II
      - DocGenerator
      - RequirementTracer
    - Test
      - Proba Kasuak
      - Proba Log-a
      - Proba Script-a
- Sistemaren Espezifikazioa
  - Glosarioa
  - Bisioa
  - Betebeharren Espezifikazioa - I
  - Betebeharren Espezifikazioa - II
  - Erabilpen Kasuak
  - Erabilpen Kasuen Eredua
- Aurrekontua
  - Orokortasunak
  - Edukia
- Ikerlanak
  - Orokortasunak
  - Edukia

**BETRADOK: Barne Kudeaketa**

- Proiektuaren Barne Kudeaketa
  - Iterazio Plana
  - Proiektu Plana
  - Arrisku Zerranda
  - Lan-Aitzaren Zerranda
  - Barne Kudeaketa
  - Trebatzeko Materialak
  - Hizkuntza Hitzarmena
  - Bilera Aktak
    - 2019.02.06
    - 2019.02.13
    - 2019.02.20
    - 2019.03.06 - I
    - 2019.03.06 - II
    - 2019.03.20
    - 2019.04.02
    - 2019.04.03
    - 2019.04.11
    - 2019.04.23 - I
    - 2019.04.23 - II
    - 2019.05.14 - I
    - 2019.05.14 - II
    - 2019.05.22 - I
    - 2019.05.22 - II
    - 2019.06.12
    - 2019.06.17
    - 2019.07.04

**BETRADOK: Betekizunen trazabilitate inpaktu-analisi automatikoa eta dokumentazio formalaren sorkuntza automatikoa modeloetan oinarritutako ekosistemetan.**

<p><b>Bezeroa</b></p> <p>Izen soziala: CAF Signalling.</p> <p>I.F.Z.: -</p> <p>Helbidea: Juan F. Gillsagasti Kalea, 4, 20018 Donostia, Gipuzkoa.</p> <p>Telefonoa: 943 80 55 75</p>	<p><b>Hornitzailea</b></p> <p>Izen soziala: Jon Legarda Gonzalez</p> <p>I.F.Z.: -</p> <p>Helbidea: -</p> <p>Telefonoa: +34 646669502</p>
<p><b>Data eta sinadura:</b></p> <p>2019.02.04.</p> <p>Mikel Larramendi, Jon Legarda Gonzalez eta EHU (Euskal Herriko Unibertsitatea, Donostiko Informatika Fakultatea).</p> <p><b>Laburpena:</b></p> <p>Software sistema baten betekizunak trazatzeko sistema automatikoaren garapena eta dokumentazio formal automatikoki sortzeko aukera duen</p>	 <p><b>Egilea</b></p> <p>Izena eta abizenak: Jon Legarda Gonzalez</p> <p>Titulazioa: Ingeniari Informatikoa</p> <p>Profesionalen elkargoa eta elkargokide-zenbakia: Euskadiko Informatikako Ingeniarien Elkargo Ofiziala</p>

3 irudia: BETRADOKeko webgunearen menu osoa.

Ezkerreko menuari esker proiektuko dokumentu guztiak daude atzigarri. Gainera, webguneak ere *CCII N2016-02* estandarra bete egiten du (memoria, memoriaren eranskinak, sistemaren espezifikazioa...). Horri esker, irakurle adituak oso azkar identifikatuko du behar duen dokumentua. Behin menuko aukera batean klikatzen denean, zati nagusian PDF formatu gisa ireki egingo da eskatutako dokumentua.

**Oharra 1:** gerta daiteke menuko aukera batzuetan dokumentu garrantzitsurik ez egotea eta soilik deskribapen txiki bat egitea.

**Oharra 2:** webgunean dauden dokumentu asko badituzte, aldi berean, beste erreferentzia asko beste dokumentuetara. Halaber, dokumentu horiek bakarrik uler daitezke proiektuaren webgunearen testuinguruan, horregatik ez da gomendagarria dokumentuak deskargatu eta beste lekuetatik irakurtzea, erreferentziak galdu daitezkeelako.

## Glosategia

Dokumentu honetan zehar hainbat hitz berezi agertzen dira. Horietako gehienak orri-oinean agertzen dira deskribatuta. Hala eta guztiz ere, irakurleak beti jo dezake hurrengo estekara proiektuan agertu daitekeen hitz berezien esanahia ezagutzeko:

- [Glosategia](#). 2019. Jon Legarda Gonzalez.

## 2 Proiektuaren Xedea

Modu globalean esanda, proiektuaren helburua, azken finean, **CAMEL ekosistemaren funtzionalitatea hedatzea** da. Funtzionalitate horiek bi dira, beraz, bi dira proiektuaren helburu nagusiak:

1. Dokumentazioa formala automatikoki sortzeko sistema, software garapeneko bi fasetarako: softwarearen arkitektura eta software diseinu xehatua edo software diseinuko fasea.
2. *CAMEL* eta *IBM Rational DOORS*<sup>11</sup>en (betekizunak kudeatzeko aplikazioa) arteko lotura egitea, software betekizunen trazabilitatea automatikoki eguneratzeko sistema baten bidez.

Aipatu bezala, bi helburu horiek *CAMEL* ekosistemaren gainean lan eginez bete behar dira eta biak automatizazioa dute helburutzat.

Enpresa edo garatzaile ikuspegitik hartuta helburu garrantzitsu bi daude: alde batetik, software garapen prozesuaren automatizazioa; bestetik, software garapeneko hutsune garrantzitsu eta errorearen identifikazio azkarra. Hitz gutxitan esanda, denbora aurrezte eta kostu ekonomikoak gutxitzea.

Laburbilduz, proiektuaren bukaeran bi sistema izango ditugu. Baten helburua da *CAMEL* erabilia dokumentazioa sortzea (software sistema baten dokumentazio formala). Beste sistemaren helburua, software batek bete behar dituen betekizunak eta une jakin bateko inplementazioa lotzea, zein betekizun inplementatuta dauden jakiteko eta erraz identifikatzeko zein betekizun ez diren betetzen.

---

<sup>11</sup> *IBM Rational DOORS*: *IBM* enpresaren aplikazioa da, "*requirements*" edo betekizunen jarraipena eramateko tresna izanik. Hainbat motatako sistema edo baliabideen betekizunak kudeatu, analizatu eta trazatzeko erraztasunak ematen ditu.



### 3 Aurrekariak

Atal honen helburua proiektuaren helburu diren sistemen antzeko beste osagai edo sistemak deskribatzea da. Horrekin lotuta, aurrekariak ulertu baino lehenago ongi ulertu behar da zertarako erabili behar diren geroago azalduko diren tresnak proiektu ahaltzuetan.

Sarreran aipatu bezala, trengintzan oso garrantzitsua da *safety* kontzeptua. *Safety* kontzeptuaz ari garenean, ez gaude hitz egiten ari software segurua funtzionamendu ikuspegitik, baizik eta errore kantitateen ikuspegitik. Ikuspegi desberdina da; izan ere, trena gidatzeko software batek pertsonen bizitzak gidatzen ditu (nolabait esateko) eta pertsona horiek ziurtasunez heldu behar dira nahi duten metro estazioetara. Sistema horietan (*ATO* sisteman, esate baterako) errore kantitatea ia nulua izan behar da. Trengintzan *SIL* mailak (*safety integrity level*) definitzen du errore probabilitatea nolakoa izan behar den tren segurua izan dadin. Errore probabilitate txiki horiek lortzeko, sistema oso ahaltzuak behar dira eta horiek hobetu eta mantentzeko tresnak ere halakoak izan behar dira derrigorrez.

#### *CAF Signalling* enpresaren softwarearen garapena

*CAF Signalling* enpresan softwarearen garapenak bide luze bat izan du hasieratik, pixkanaka-pixkanaka bideratuz gaur egun erabiltzen diren metodologietara arte. Software garapenean, beste esparru askotan bezala, haren matrizea den *CAF* en esperientzia erabili izan dute.

Hasiera batean software garapenaren bidez ez ziren gaur egun bezain integralak diren soluzioak lortzen. Lehenik eta behin, *ISO 9001* ziurtagiriak lortzeko ahaleginak egin zituzten. Ziurtagiri horiek sistema baten kalitatea neurtzeko parametroak definitzen dute, beraz, ziurtagiria lortzerakoan software kalitatearen maila ziurtatu egiten zuten.

Ondoren, hiru helburu garrantzitsu finkatu ziren software garapenaren bidez kalitatezko softwareak lortzerako garaian:

1. Bezeroei eskainitako zerbitzua hobetu.
2. Produktuko erroreek eragindako intzidentziak gutxitu.
3. Proiektuen kostua optimizatu.

Enpresaren xede bihurtu ziren hiru helburu horiek lortzeko kalitatezko softwarea eta software seguruak lortzea ezinbesteko suertatu zen.

Hori dela eta, *CMMI*<sup>12</sup>ko zertifikatuak lortzea helburu bilakatu zen. Zertifikatu garrantzitsu horiek erakusten dute software bat kalitatezkoa dela eta prozesu zein metodologia sendoekin garatu egin dela. Bide horretan ezagunak diren metodologia anitz erabili izan ziren, *RUP (Rational Unified Process)*, esate baterako.

Oso garrantzitsua da ulertzea *CAF Signalling*ek seinaleztapen zerbitzuak eskaintzen dituela trengintzan, eta horrek esan nahi du haien sistema konplexuek ezin dutela akatsik izan; azken batean, gizakien bizitzak arriskuan egon daitezkeelako. Beraz, software sendoak eta kalitatezkoak lortzeko bidea argia da: hainbat eta hainbat arau eta ziurtagiri garrantzitsu definitzen duten mekanismo eta metodologiak ezarri.

Software sendo horien azken helburua *SIL* maila jakin bat ziurtatzea da. *SIL (safety integrity level)* ziurtagiriak arriskuen gutxitze maila eta segurtasun funtzionalaren maila zehazten du, horregatik derrigorrezko ziurtagiria bilakatu egin zen. Gainera, derrigorrez aipatu beharra dago segurtasun-mota hura ziurtatzeko beste arau garrantzitsu bat: *IEC 61508*. Estandar internazional honek zehaztu egiten ditu segurtasunarekin erlazionatuta dauden sistemak garatzeko metodo seguruak. Segurtasun funtzionala altua lortzeko metodoak biltzen ditu.

Ondorioz, *CAF Signalling*ek garatzen dituen soluzioen barnean arau eta estandar garrantzitsuak egon behar dira proiektuen fase guztietan zehar.

Software garatzeko enpresaren historia aipatu eta gero, proiektu honetan zehazki garrantzitsuak diren aurrekariak definitzea tokatzen da.

Proiektuaren aurrekari gisa, garrantzi handiko bi tresna-mota aurkezten dira: dokumentazio sortzailea eta betekizun kudeatzailea.

### *CAMELen aurrekaria: IBM Rational Rhapsody*

*CAMELen* aurrekari gisa, sistemak modelatzeko eta diseinatzeko prozesuan lagun dezaketen tresna ahaltsu eta profesionalak kontuan hartu ditugu.

Horien guztien artean, tresnarik garrantzitsuena *IBM Rational Rhapsody*<sup>13</sup> da. *Rhapsody UML* eta *SysML*en oinarritutako tresna komertziala da, software integratuaren garapenean eta

---

<sup>12</sup> *CMMI*: Sigla (ingelesez): *Capability Maturity Model Integration*. Software garapenean dauden praktika onak definitzen duen modelo edo eredua da. Software sistemak garatu, mantendu eta operatzeko eredua da, *CMMI Institutek* definitutakoa.

<sup>13</sup> *IBM Rational Rhapsody*: *IBM* enpresaren produktua da, *UML* eta *SysML*en oinarritutako software garapen integratuan laguntzen duena. Modeloa definitzen laguntzen du, baita dokumentazioa eta programazioko kodeketa sortzen ere.



*testingean* laguntzeko balio duena. Bere helburua software garapena automatizatzea da. Gainera, hainbat funtzionalitate gehiago integratuta ditu. Adibidez, dokumentazio sorkuntza.

*Rhapsody*-ri esker software handi bat espezifikatu daiteke maila gorenetik, maila baxuenera arte. Software osagai guztiak definitu daitezke eta pixkanaka-pixkanaka xehetasun handiagorekin definitu, kodeketa lortu arte. Oso tresna ahalsua da. *IBM* enpresak berak sei ezaugarri garrantzitsu azpimarratzen ditu:

- Kodearen sorkuntza automatikoa, sistemaren diseinua kontuan hartuta.
- Modeloetan oinarritutako probak eta simulazioak.
- Produktuaren bizitza-zikloari sostengua.
- Erroreen saihestea.
- Lan-inguru hobea edo erraza.
- Garatzailearen produktibitate areagotzea.

Tresna honi esker produktu baten bizitza-zikloan parte hartzen duten fase guztiak babestuta daude. Sistema baten arkitektura, sistema baten diseinua, osagaien espezifikazioa eta sistema bat definitzeko erabiltzen diren diagrama guztiak egiten laguntzen du. Horrek guztiak eragiten du *Rhapsody* oso tresna erabilia izatea sektorean.

Horretaz gain, badaude beste tresna interesgarri batzuk ere, esate baterako, *MagicDraw*. sistemak modelatzeko eta software ingeniarietan eman behar diren pausu guztiak ongi jarraitzeko balio duen softwarea da.

Hala ere, *CAF Signallingek* erabaki egin zuen software garapenerako *CAMEL* sortzea, gehienbat bi arrazoi nagusiengatik: lehenik eta behin, *Rational Rhapsodyn* identifikatutako urritasun eta murriztapenak direla eta; bigarrenik, trenen gidapenerako espezifikokoak diren ereduak modu zehatzago batean sortzeko.

## Dokumentazio Sorkuntza: Aurrekariak

Dokumentazioa automatikoki sortzeko aukera ematen diguten produktuen artean ere *IBM Rational Rhapsody* analizatuko dugu. *Rhapsodyk* integratzen dituen funtzionalitateei esker, dokumentazioa automatikoki sortu daiteke, konfigurazio txiki batekin kontrolatu daitekeena. *Rhapsodyko* proiektu batean hainbat diagrama sortzen dira (erabilpen kasuak, aktibitate-diagramak, analisi-ereduko diagramak etab.) eta horiek guztiak garrantzitsuak dira garatzaile batek analizatzeko zer falta daitekeen softwarean edo zergatik softwareak daukan konportamendu jakin bat edo beste.

Modu horretan, *Rhapsodyko* proiektu batean dauden diagramak dokumentu formaletara esportatu egiten dira testu-prozesadoreetan dauden marka berezien (*Worden* “markadoreak”, esate baterako) bidez, espezifikatutako dokumentu ataletan.

Proiektuan antzeko teknika bat erabili beharko da.

## Betekizunen Trazabilitate Automatikoa: Aurrekariak

Betekizunak trazatzeko aurrekari gisa, *IBM Rational Rhapsody* eta *IBM Rational DOORS*en arteko elkarrekintzaz hitz egin behar da. Betekizunak bi direkzioetan tratatzen ditu eta sendo mantendu egiten ditu. Diseinuko betekizunak kudeatzen laguntzen du.

Enpresa beraren bi produktu horiek elkarrekintza handia dute haien artean. Bi horiei esker orain arte *CAF Signalling*ek betekizunak ondo trazatu eta tratatu egin dituzte. *CAF Signalling*ek *IBM Rational DOORS* erabiltzen du haien produktu gehienek betekizunak trazatzeko eta definitzeko. *DOORS*en esker betekizunak estandarren arabera edo espezifikazio internazionalen arabera eguneratu daitezke. Oso tresna erabilgarria da.

Proiektu txiki batean betekizun kopuru gutxi kudeatzen dira normalean. Hala eta guztiz ere, enpresak kudeatzen dituen proiektuetan nazioartekoki errekonozituta dauden estandar eta espezifikazioak erabiltzen dira (derrigorrez). Espezifikazio horiek milaka betekizun definitzen dituzte. Betekizun gutxi kudeatzea ez da zaila, baina milaka betekizun bete behar direnean proiektu jakin baterako mekanismo jakin bat behar da horiek guztiak kontrolatzeko. Horretarako erabiltzen da *DOORS*.

Orain arte, *DOORS* erabili da, baina *CAMEL* garatzen eta erabiltzen hasi denetik, *ATO* taldeak beste tresna edo sistema bat behar du betekizunak kudeatzeko eta tratatzeko. Beste tresna bat esaten denean ez da derrigorrez sortu behar den software edo aplikazio bat. *DOORS* eta *CAMEL* arteko elkarrekintza egiten duen aplikazioa edo sistema ere izan daiteke.

Proiektuan antzeko elkarrekintza lortzea espero da.

## 4 Egungo egoera

Atal honetan azaldu egiten dira bi gauza: lehenik, proiektua hasi baino lehen zein den egoera eta, bigarrenik, proiektua bukatuta dagoela, zein den egungo egoera eta zein pausu eman daitezkeen etorkizunari begira, sistemak hobetzeko eta bizi-zikloa ahalik eta hobekien mantentzeko.

### 4.1. Egungo Egoeraren Deskribapena

#### Proiektua hasi baino lehen

Proiektuaren gauzatzea planteatzen den momentuan, *CAMEL*ek ez du sortzen inolako dokumentaziorik eta ez du betekizunen trazabilitatearen inguruko ezer egiten.

*CAMEL* gai da enpresak definituta duen software garapeneko “V” ereduan laguntzen. Horrek esan nahi du sistemako definitiotik inplementaziorako bidean laguntzen duela. Tresna horrek lagundu egiten du software garapen eredu honen fasetako eredu desberdinak sortzen automatikoki, eta, bide batez, garatzaileari definizio erroreak murriztea laguntzen du.

*CAMEL* ez dago oraindik %100ean egonkortuta. Momentuan egon daitezkeen erroreak hobetzeaz aparte, garapenean dagoen erreminta da. Proiektu honetan sartuta dauden ingeniariak uste dute tresna ahaltsuagoa egiteko hurrengo bi pausuak hauek direla: dokumentazioa automatikoki sortzeko gaitasuna eta moduren batean betekizunen trazabilitatea egitea.

#### Proiektua bukatu eta gero

Momentu honetan, 2019ko irailean, *CAMEL*ek bai *DocGenerator* bai *RequirementTracer* sistemak integratzen ditu. Etengabeko integrazioari esker, *CAMEL* komando bidez bi programak exekutatzeko gai da eta *ATO*ko kode nagusian (*ATO*a gordeta dagoen biltegiko adar nagusian) aldaketa bat dagoenean bi sistemak exekutatu egiten dira. Beraz, bi irteera daude: alde batetik, arkitektura eta diseinuko dokumentazio formalak; bestetik, betekizunen traza eguneratzen dira inpaktu-analisia egiteko prest egoteko. Bi sistema horien exekuzioen bidez, %100ean eguneratuta lortu egiten da dokumentazio formalak eta betekizunak trazatzeko sistema.

Gainera, oso garrantzitsua da aipatzea *CAMEL* garapen bidean dagoela eta modu paraleloan softwarearen probetarako beste tresna bat garatzen hasia dela *CAF Signalling*ek.

Beraz, ondo legoke dokumentazioa txosten gisa sortzea software proben txostenak ateratzeko. Horretaz aparte, agian software sistema (*Software System*) eta software definizioko (*Software Definition*) dokumentazioa sortzea ere positibo izan daiteke, maila guztietako dokumentazioa izateko asmoz.

Bestalde, betekizunen trazabilitateari buruz ondo legoke *IBM Rational DOORS*en egin daitezkeen aldaketak ondoren *CAMEL*en ikustea moduren batean edo bestean. Beraz, bigarren direkzioaren kobertura egongo litzateke: *DOORS* -> *CAMEL*. Horretarako, ikerlan bat egin beharko litzateke ikusteko zein behar dauden horretarako eta zein modutan egin daitezkeen hori.

## 4.2 Identifikatutako Urritasunen Laburpena

### Proiektua hasi baino lehen

Proiektua hasi baino lehen ez da urritasun tekniko berezirik antzeman. Beti ere kontuan hartu behar da *CAMEL*ek espezifikatzen dituen ereduak aldakorrek direla eta beraz, etengabe aldatzen joango dira modelo edo eredu gehien elementuak.

### Proiektua bukatu eta gero

Proiektua egina eta bukatuta dagoelarik, proiektuaren bukaeran identifikatu diren urritasunak zerrendatuko dira hurrengo lerroetan.

### Dokumentazio sorrera automatikoaren urritasunak

- *UML* diagramak sortzerakoan *PlantUML* ez ditu %100ean ongi erakusten diagramak. Zaila da osagai edo klase diagramak erakustea hamarnaka elementu erakutsi behar direnean, eta beraz, gertaera horiek kudeatzeko estrategia desberdinak erabili izan dira. Egia da *PlantUML* konponentea hobetzen joango dela eta horrek lagundu du elementu guztiak argi erakusteko.

### Betekizunen trazabilitateko urritasunak

Ez dira urritasun tekniko eta garrantzitsuak antzeman.

## 5 Arauak eta Erreferentziak

### 5.1 Xedapen Legalak eta Aplikatutako Normak

Proiektuaren helburu izan diren bi sistemak, *DocGenerator* eta *RequirementTracer*, diseinatu, eraiki eta inplementatzeko momentuan ez da bete behar izan arau estandarrik, ezta xedapen legalik ere. Bi sistema horiek ez daude *CAF Signalling*ek sortzen dituen produktuen arau estandarren menpe. Horrekin zera esan nahi da: bi sistema hauek ez dute trenaren kontrola izaten eta ez dute behar berezia den segurtasun-maila bat.

Azken batean, bi sistema hauen helburua da software garapenean laguntzea. *DocGenerator* eta *RequirementTracer*ri esker, proiektu jakin baten momentu jakin batean lortuko dugu dokumentazio automatikoa eskuratzea eta produktu horrek bete behar dituen betekizunak automatikoki trazatzea. Hau da, nahi den momentuan lortu daiteke informazioa jakiteko nola inplementatuta dagoen produktu hura eta zein betekizun betetzen dituen momentu jakin batean. Horrek esan nahi du bi sistema horien azken helburua dela kalitatearen inguruko informazioa erakustea; izan ere, enpresa honek kalitate handiko software soluzioak lortu behar ditu, aurreko ataletan esan bezala.

Ondorioz, ez da norma edo arau estandar garrantzitsurik bete behar izan bi soluzioak garatzeko. Hala ere, proiektua aurrera eramateko, hainbatetan aipatu bezala, *CCII N2016-02* estandarra erabili da, proiektua dokumentatzeko hain zuzen.

#### *CCII N2016-02* estandarra

Estandar honek ingeniari-tza informatikoko proiektuen dokumentazioa ondo antolatzeko balio du. Proiektuko memoria eta eranskinak honen arabera antolatu eta bete egin da. Dokumentazioarekin batera entregatu den webgunean ere memoria eta eranskinak estandarren arabera antolatuta dago.

### 5.2 Bibliografia

Proiektuan zehar hainbat liburu, gida eta webgune erabili izan dira informazioa bilatzeko edo datuak lortzeko. Hurrengo zerrendan, ordena lexikografikoan, aurkitu daitezke:

- *Agile Manifesto*. (2000). *Independent Signatories of the Manifesto for Agile Software Development*. Hemendik eskuratuta: <https://agilemanifesto.org/>.

- Apache POI Documentation. (2019). *The APACHE Software Foundation*. Hemendik eskuratuta: <https://poi.apache.org/apidocs/4.0/>.
- *Batch Script Tutorial*. (2019). *TutorialsPoint*. Hemendik eskuratuta: [https://www.tutorialspoint.com/batch\\_script/](https://www.tutorialspoint.com/batch_script/).
- *Customizing IBM Rational DOORS Using DXL*. (2010). *IBM Corporation*. Hemendik eskuratuta: [http://www.smartDXL.com/content/?page\\_id=842](http://www.smartDXL.com/content/?page_id=842)
- *DOORS: Curso*. (2012). *CAF Signalling*. Konfidentziala.
- *DOORS: Manual de Buenas Prácticas*. (2012). *CAF Signalling*. Konfidentziala.
- *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK), Quinta Edición*. (2013). *Project Management Institute, Inc*. Hemendik eskuratuta: [http://eimem.uniovi.es/c/document\\_library/get\\_file?uuid=2abb8e4b-78c3-4b02-879c-27541dcc1a7d&groupId=517164](http://eimem.uniovi.es/c/document_library/get_file?uuid=2abb8e4b-78c3-4b02-879c-27541dcc1a7d&groupId=517164)
- GrAL Eredua. (2019). Euskal Herriko Unibertsitatea. Hemendik eskuratuta: <https://www.ehu.eus/es/web/informatika-fakultatea/ikasketak/gradu-amaierako-proiektua>
- GrAL: Hitzarmena. (2019). UPV-EHU, CAF Signalling eta Jon Legarda.
- *Home – CAF Signalling*. (2019). *CAF Signalling, S.L*. Hemendik eskuratuta: <https://www.cafsignalling.com/>.
- *Irish Standard I.S. EN 50128:2011*. (2011). *NSAI Standards*. Konfidentziala.
- *Norma CCII-N2016-02: Norma Técnica para la realización de la Documentación de Proyectos en Ingeniería Informática*. (2016). *Consejo de Colegios de Ingenieros en Informática*. Hemendik eskuratuta: <http://cpiicm.es/wp-content/uploads/sites/3/2016/11/CCII-N2016-02-Norma-Tecnica-para-la-realizacion-de-la-Documentacion-de-Proyectos-en-Ingenieria-Informatica-V1.0-f.pdf>.

- *OpenUP methodology*. (2008). *Eclipse Foundation – OpenUP*. Hemendik eskuratuta: <http://www.utm.mx/~caff/doc/OpenUPWeb/index.htm> .
- Penadés Ribera, J. (2017). [\*Diseño de un sistema de gestión y sincronización de cuentas de Usuario para Informática MDM\*](#). *Universitat Politècnica de València*.
- *PlantUML: Getting Started*. (2019). *PlantUML Team*. Hemendik eskuratuta: <http://plantuml.com/es/starting>.
- *Presentación de la norma UNE 157801:2007 para los Proyectos de Ingeniería en Informática y Riesgos*. (2018-04-28). *Colegio Profesional de Ingenieros en Informática de la Comunidad de Madrid*. Hemendik eskuratuta: <https://pmi-mad.org/index.php/component/dropfiles/?task=frontfile.download&id=114&catid=253>
- *Rational Rhapsody API Reference Manual*. (2019). *IBM Corporation*. Hemendik eskuratuta: [https://www.ibm.com/developerworks/community/blogs/a8b06f94-c701-42e5-a15f-e86cf8a8f62e/resource/Documents/rhapsody\\_api\\_reference\\_manual.pdf?lang=en](https://www.ibm.com/developerworks/community/blogs/a8b06f94-c701-42e5-a15f-e86cf8a8f62e/resource/Documents/rhapsody_api_reference_manual.pdf?lang=en).
- *StackOverFlow Forum*. (2019). *Stack OverFlow*. Hemendik eskuratuta: <https://stackoverflow.com/>.
- *The DXL Reference Manual*. (2013). *IBM Corporation*. Hemendik eskuratuta: [https://www.ibm.com/support/knowledgecenter/SSYQBZ\\_9.5.0/com.ibm.doors.requirements.doc/topics/DXL\\_reference\\_manual.pdf](https://www.ibm.com/support/knowledgecenter/SSYQBZ_9.5.0/com.ibm.doors.requirements.doc/topics/DXL_reference_manual.pdf).
- *Unified Modelling Language*. (2019). *Wikipedia*. Hemendik eskuratuta: [https://eu.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://eu.wikipedia.org/wiki/Unified_Modeling_Language).

## 5.3 Metodoak, Tresnak, Ereduak eta Metrikak

Atal honetan proiektuan zehar erabilitako metodoak, tresnak, ereduak eta metrikak azaldu egiten dira. Proiektua gauzatzeko metodo desberdinak erabili dira proiektuaren eraikitze faserako eta ondoren, dokumentazioa zein memoria idazteko. Gainera, tresna asko erabili izan dira bai sistemak eraikitzeko bai dokumentazioa eta memoria sortzeko. Horretaz aparte, ereduak eta metrikak ere azaltzen dira.

### 5.3.1 Metodoak eta Tresnak

Lehenik eta behin, metodoak deskribatuko dira, bakoitza erabiltzeko zergatia eta ezaugarri bereziak azalduz.

#### Metodoak

##### *OpenUP*

*OpenUP* deritzon metodologia software garapenerako prozesu eta metodoa da. Enpresatalde batek garatu zuen eta 2007.urtean *Eclipse Foundation*eri utzita eman zioten. Prozesu minimo eta behar adinekoa da, beraz, soilik nahitaezkoa den edukia dago zehaztuta.

Metodologia hau erabili da proiektuan zehar dokumentazioa bildu eta proiektuaren kontrola eramateko modu pertsonalean. Horrek esan nahi du, enpresan ez dela metodologia honen berririk izan, soilik *BETRADOK* proiektuaren metodologia delako.

##### *Scrum* metodologia

*Scrum* metodologia, ordea, enpresako lan-taldean lan egiteko metodologia izan da. Nolabait *OpenUP* soilik proiektu espezifiko honetara bideratuta dagoen moduan, *Scrum* bidez, talde osorako metodologia izan da. Honen bidez, *daily meeting* deritzon bilera motzak eta iterazio bukaeran *sprint* bilera egin izan dira.

Metodologia horrek hiru ezaugarri nagusitan bereizten da: garapen inkrementaleko estrategiaren erabilera, soluzioaren emaitza kideen isilpeko ezagupenean oinarritzea, eta garapen fase desberdinak gainjartzea, bata bestearen atzetik, modu sekuentzialean egin ordez.



## Tresnak

Bigarrenik, proiektuan zehar erabilitako tresna garrantzitsuak zerrendatuko dira. Egia da tresna batzuk ez direla agertzen, zerrenda luzeegia izan daitekelako. Alfabetikoki daude ordenatuta:

### *Apache POI*

*Microsoft Office*ko hainbat dokumentu-mota manipulatzeko *API*a da. Proiektu honetan dokumentazio generazio automatikoan kokatzen da. Erabilitako bertsioa *Apache POI* 4.0.1 da. Software librea da eta, beraz, ez dauka kostu ekonomikorik. Webgunea: <https://poi.apache.org/>. Ez da *JAR* fitxategi bakarra duen *API*a, horrelako fitxategi gehiago behar ditu haien arteko dependentziak direla eta. Hauek dira aipatutako bertsiorako dependentziak:

- *commons-collections4-4.2.jar*
- *commons-compress-1.18.jar*
- *dom4j-2.1.1.jar*
- *log4j-1.2.17.jar*
- *poi-4.0.1.jar* (nagusia)
- *poi-ooxml-4.0.1.jar*
- *poi-ooxml-schemas-4.0.1.jar*
- *poi-scratchpad-4.0.1.jar*
- *xmlbeans-3.0.2.jar*

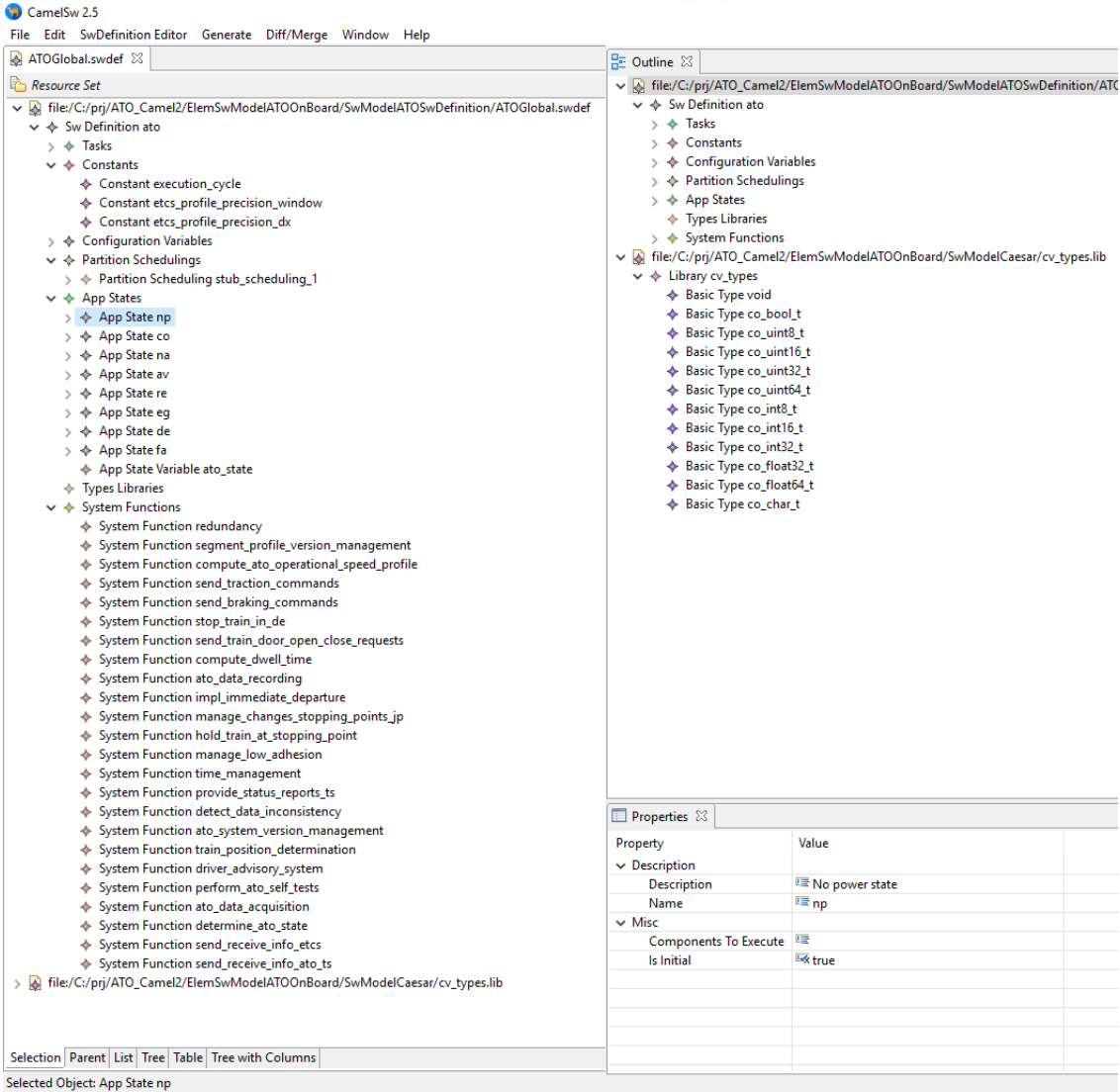
Ondorioz, kontuan hartu behar da noizbait bertsioa aldatzen bada, seguraski dependentzia guzti horien bertsioak ere aldatuko direla.

### *CAMEL (CAF Modeling & Editor Language)*

*CAMEL (CAF Modeling & Editor Language)* software sistema baten definizioa eta modelatua sortzeko balio duen software tresna da. *CAMEL CAF Signalling*ek garatutako erraminta da, *IBEk* eta *Ikerlan* enpresak garatutakoa, eta oraindik ere garapen eta hobekuntza fasean dago. Bestalde, *EMF framework*ari esker garatu da.

*CAMEL ATO*a definitzeko balio duen tresna da. Hor *Software Definition* (software definizio maila) fasea modelatu egiten da eta tresnak hortik abiatuta *Software Architecture*, *Software Design* eta inplementazioa sortzen ditu automatikoki. Horren helburua hurrengoa da: soilik software definizio-mailan aldaketak egitea eta horrela hurrengoko pausuak automatikoki sortzea.

Azken batean, proiektuaren helburua *CAMELen* funtzionalitateak hedatzea eta tresna ahaltzuagoa sortzea da. Horregatik funtsezkoa da ondo ulertzea *CAMEL*ek nola funtzionatzen duen; izan ere, horren gainean lan egiten da. Hurrengo irudian agertzen da *CAMELen* interfaze grafiko bat.



The screenshot shows the CamelSw 2.5 software interface. The main window displays a project tree for 'ATOGlobal.swdef'. The tree is organized into several categories: Sw Definition ato, Tasks, Constants, Configuration Variables, Partition Scheduling, App States, Types Libraries, and System Functions. The 'App States' category is expanded, showing various states like 'App State np', 'App State co', 'App State na', etc. The 'App State np' state is selected, and its properties are displayed in the Properties panel on the right. The Properties panel shows the following details:

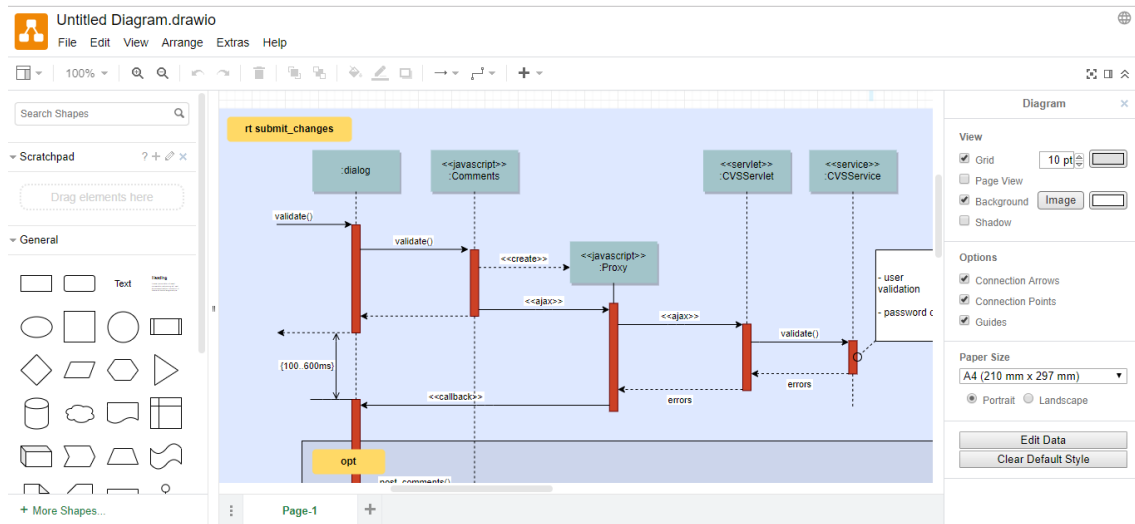
Property	Value
Description	
Description	No power state
Name	np
Misc	
Components To Execute	
Is Initial	true

At the bottom of the interface, the 'Selected Object' is identified as 'App State np'.

4 irudia: *CAMELen* interfaze grafikoa.

## Draw.io

Hainbat motatako diagramak sortzeko aukera ematen duen softwarea da. Eskuz sortzen dira diagramak eta bi aukera daude aplikazioa erabiltzeko: *online* edo *offline*. Proiektuan zehar aplikazio hau erabili izan da dokumentazioan agertzen diren diagrama gehienak egiteko, bezeroak dokumentazio ulergarria jaso dezan. Webgunea: <https://www.draw.io/>.



5 irudia: Draw.io-ren interfaze grafikoa.

## Eclipse Modeling Tool 2018-12

Eclipse *framework*eko bertsio espezifikoa da, 2018ko abendukoa. Tresna hau erabiltzen da *DocGenerator* eta *RequirementTracer* diseinatzeko. Webgunea: [Eclipse Modelling Tool 2018-12](https://www.eclipse.org/modeling-tool/).

## GitLab

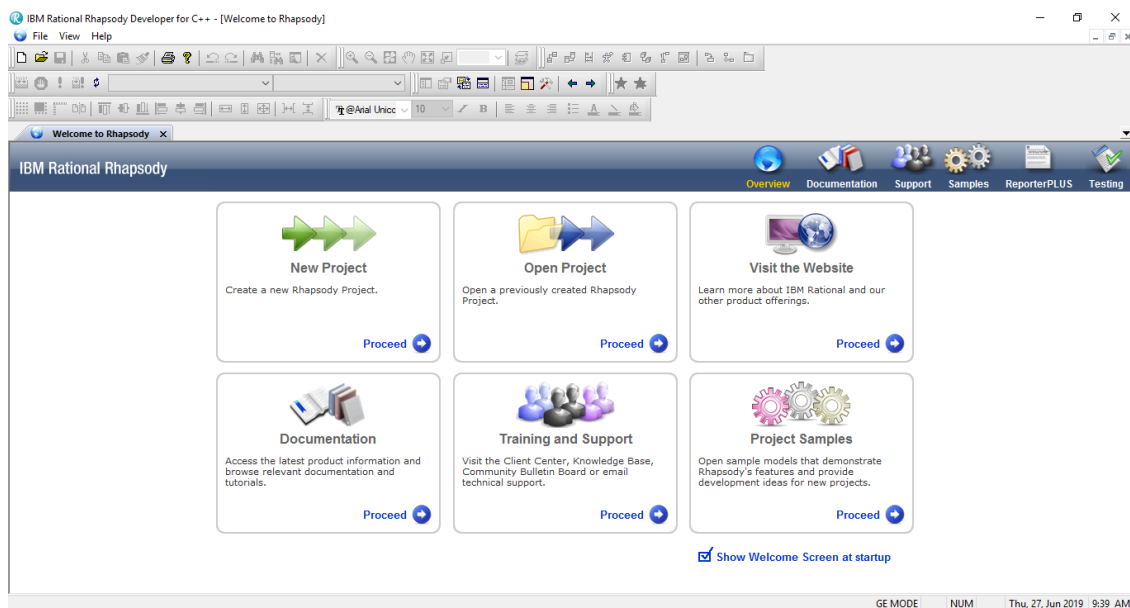
*Giten* oinarrituta dagoen web zerbitzua da, errepositorioen kudeatzailea izateaz aparte, *wiki*, erroreen jarraipena eta *Continuous Integration* ere baditu. Webgunea: <https://about.gitlab.com/>.



6 irudia: GitLaben logoa.

## IBM Rational Rhapsody

*Telelogic Rhapsody* deitua ere, *IBM*k garatutako tresna komertziala da, software integratuaren *testing* eta garapenerako balio duena, *UML* eta *SysML*an oinarritua. *Rhapsody*rekin modelatu daiteke eta dokumentazioa eta inplementazioa automatikoki sortu. Erabilitako bertsioa *IBM Rational Rhapsody Developer for C++ 7.5.3*. Proiektu honen hasierako helburua zen dokumentazioa sortzeko lehen pausua, modeloetatik *Rhapsody*rako pausua ematea.



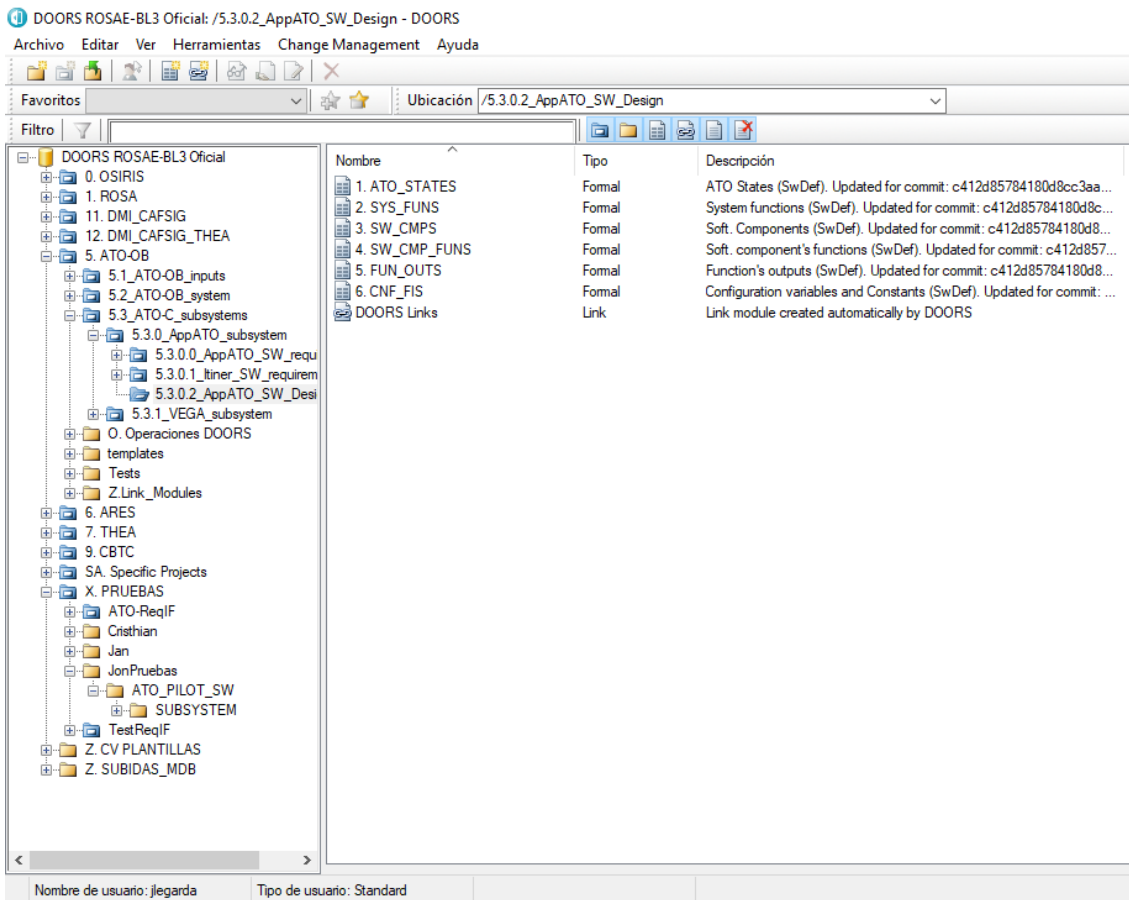
7 irudia: *IBM Rational Rhapsody*ko hasierako interfaze grafikoa.

## IBM Rational DOORS

*Telelogic* enpresak garatutako softwarea da, geroxeago *IBM*k erositakoa betekizunen kudeaketarako diseinatuta dagoena. *DOORS* datu-base jakin bat da non betekizunak gorde daitezkeen modu ordenatu eta estrukturatu batean, haien atributuekin batera. Erabiliko den bertsioa *IBM Rational DOORS 9.6*. Proiektuan *RequirementTracer* paketearen helburua *DOORS*en sortutako moduluak eguneratzea eta horrela automatikoki analisi-inpaktu bat egitea da.

*DOORS* enpresan oso erabilia da eta ez dago beste tresnarik betekizun edo *requirements*en trazabilitatea eta jarraipena egiteko. Hori dela eta, erabaki egin zen *CAMEL*ek estrategia zehatz batekin *DOORS*ekin elkarrekintza egin beharko zela.

*DOORS* “modulu” eta “esteka-modulu” kontzeptuak ditu. Moduluak dira betekizun zerrendak eta esteka-moduluak, aldiz, moduluen arteko traza egiteko objektu bereziak. Hurrengo irudian<sup>8</sup> ikus daiteke definitu egin den trazaren moduluak eta *DOORS*en interfaze grafikoaren adibide bat.



8 irudia: IBM Rational DOORSen interfaze grafikoa.

## Microsoft Word (Office)

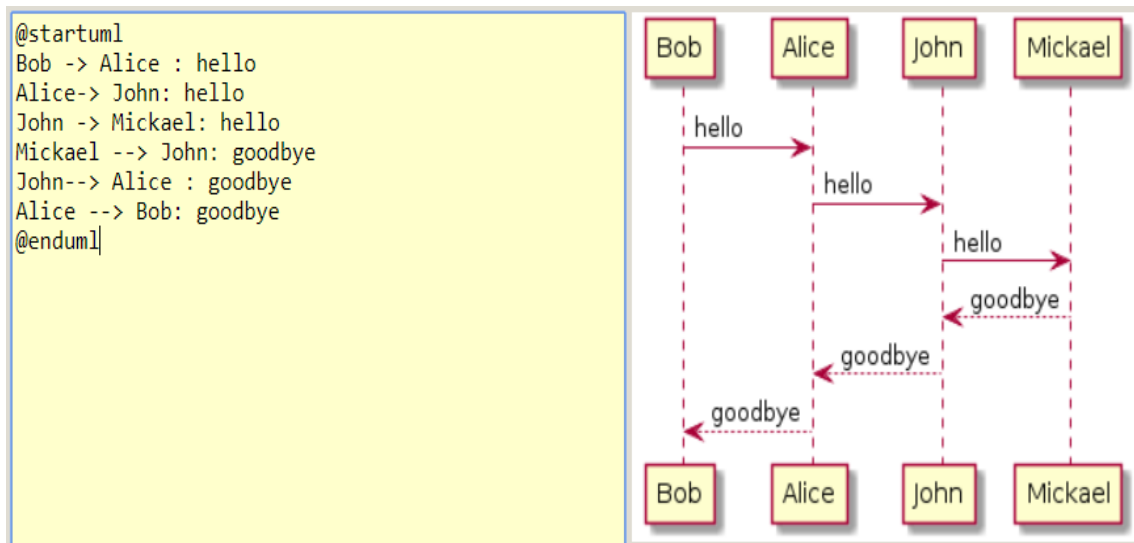
*Office* paketeko testu-prozesadore bat da, mundu osoan erabilia. *Microsoftek* garatu eta mantentzen duen softwarea da. Proiektuan bi gauzetarako erabiltzen da: proiektu eta produktuaren memoria eta dokumentazioa idazteko, eta *DocGeneratorrek* modelo jakin batzuetatik abiatuta dokumentazio formal automatikoa *DOCX* formatuan sortzeko.

## PlantUML

*Open Source* motako erraminta da. Testu planotik abiatuta eta etiketa bidezko lengoia definitu bat erabiliz, *UML* diagramak (eta *UML* ez diren beste diagrama batzuk ere) sortzeko balio du. Modu askotan integratu daiteke: aplikazioetan, testu-prozesadoretan konponente gisa, konponente desberdinetan, web-zerbitzuetan etab. Hainbat modu daude hura erabiltzeko. Webgunea: <http://plantuml.com/es/>.

Proiektu honetan zehar bi esparrutan erabiltzen da: alde batetik, *DocGeneratorek* dokumentazio formalean *UML* diagramak sortzeko, eta beste aldetik, proiektu eta produktuko dokumentazio eta memorian diagrama jakin batzuk sortzeko. Diagrama horiek hurrengo ataletan ageriko dira.

Hurrengo irudia<sup>9</sup> *PlantUML*ren webguneko aukeratik aterata dago. Adibide bezala, lengoia deskriptiboa erakusten da eta horretatik sortutako diagrama sinple baten erudia.



9 irudia: *PlantUML*ren lengoain idatzitako sekuentzia-diagrama, adibidea..

## Programazio-lengoaiak

Proiektuan zehar programazio-lengoia asko erabiltzen dira. Horien guztien deskribapena ez da egingo, baina zerrenda honako hau da:

- Java
- *DXL*
- CSS (dokumentazio webgunea)
- JavaScript (dokumentazio webgunea)
- HTML (dokumentazio webgunea)

## Besteak

Tresna hauetan aparte, badaude proiektuan zehar erabilitako beste software eta tresna batzuk ere. Horiek hurrengo dokumentuan agertzen dira:

- [Tresnak](#). 2019. Jon Legarda.

### 5.3.2 Eredua, Metrika eta Prototipoa

Atal honetan proiektuan zehar aplikatu eta erabilitako eredu, metrika eta prototipoak deskribatzen dira.

## Eredua

### *CCII N2016-02* arau estandarra

Hainbat alditan esan bezala, *CCII N2016-02* arauan oinarrituta idatzi egin da proiektuaren memoria eta dokumentazioa. Horrek eredu sendo bat erakusten du dokumentazioak jarraitu beharreko atalak ongi zehaztuz. Beraz, esan daiteke eredutzat erabili izan dela.

### *OpenUP* metodologiako ereduak

*OpenUP* metodologia jarraitzeko, metodologia berak eskainitako eredu eta txantiloak erabili egin dira. Modu horretan artefaktu (dokumentu) guztien atal eta azpiatal guztiak ongi beteta izatea lortu da.

### Dokumentazio formala sortzeko ereduak

Dokumentazio formala sortzeko sistema diseinatzerakoan eta bereziki inplementatzerakoan, aurrez erabilitako dokumentu ereduak erabili izan dira. Hori horrela izanik, duela urte askotako bi eredu lortu ziren eta modu horretan dokumentazioak zein atal izan behar dituen erraz definitu egin da.

## Metrika

### Proba kasuak onartzeko metrika

Proiektuan sortutako bi sistemak testeatzeko hainbat proba-kasu definitu egin dira. Proba kasu horiek sistemek pasatu behar dute %100ean onarpena izateko. Hots, bi motatako emaitza daude: onartuta edo ez onartuta. Proba-kasu bat pasatu dela jakiteko deskribapenean azaltzen den guztia ondo bete behar izan du sistemak.

### Lan-atazak eta orduak

Proiektu honetan lan-orduen estimazioak egiterako garaian, ez da erabili metrika berezirik orduen kontrola eramateko. Ez da ordu den beste edozer erabili, beraz, lan-orduen inguruan aipatzen denean, ordu bat eta 60 minutu berdin izango da.

## Prototipoak

Proiektuan zehar ez da prototipo sendorik sortu. Egia da proba pilotu txikiak egin direla proiektuan zehar; izan ere, bideragarritasun ikerlanak egiteko derrigorrezkoak bilakatu ziren. Hala eta guztiz ere, ez da prototipo finkorik erabili.

Erabilitako prototipoen artean, ordea, badaude aipamena merezi duten bi: alde batetik, dokumentazio formaleko prototipo bat. Prototipo hura *Wordeko* dokumentuak izan dira. Ez dira eranskinetan sartu konfidentzialtasuna mantentzeko asmoz. Bestalde, hasiera batean dokumentazioa sortzeko “erdiko pausu” bat erabili nahi zen: *Rhapsodyko* proiektua sortu. Horren bideragarritasuna ikertzeko prototipo txiki bat sortuta zegoen proiektua hasi baino lehen eta erabili egin zen hasiera batean. Hala ere, ondoren estrategia aldatu zenean, ez zen behar izan.

## 5.4 Proiektu Zeharreko Kalitate Plana

Atal honetan, proiektuaren garapenean zehar zein nolako kalitate plana erabili den zehaztuko da. Proiektuaren kalitatea kudeatzeko *PMBOK*aren gida jarraitu egin da proiektuan zehar.

### Kalitatearen Plangintza

Kalitatea zaintzeko plangintza proiektuaren egileak eta bere laguntzaileek egindako bileretan oinarritzen da, baita beste une askotan elkartutako momentuetan ere. Bilerak ez dira izan kalitate-kontrolerako bilera izan behar, soilik *feedbacka* emateko bilera sinpleak.

Proiektuko zuzendari eta laguntzaileekin batera, hiru motatako *feedback* edo hiru zutabe desberdinetan oinarritutako iruzkinak jaso egin dira:

- Inplementazioaren kalitatea: kode-mailako implementazioa egokia dela eta software arrunten patroiak jarraitzen direla bermatzeko.
- Funtzionalitate eta ezaugarrien kalitatea: produktuak bete behar dituen ezaugarri eta espezifikazioen kontrola.
- Memoria, dokumentazioa eta aurkezpenaren kalitatea: proiektuak atxikituta izan behar duen dokumentazio eta memoriaren kalitatea ere kontrolatu behar da. Adostutako estandarrak betetzen diren (*CCII* eta *OpenUP*) ziurtatu behar da; izan ere, estandar bat bakarrik betetzen da estandar jakin horrek eskatzen dituen artefaktu eta atal guztiak betetzen direnean. Horretaz aparte, memoria eta dokumentazioa erakusteko webgunearen kalitatea ere bermatu behar da.

Hurrengo taulan<sup>1</sup> adierazten da kalitatea kontrolatzeko rol desberdinak eta horien bakoitzaren ardurak:



Rolak	Ardura
Proiektuaren egilea: JLE	Etengabeko kalitate-kontrola esparru guztietan.
Proiektuaren zuzendaria: MLA	Funtzionalitateen kalitate-kontrola.
Proiektuaren zuzendariordea: JOG	Funtzionalitateen kalitate-kontrola.
Laguntzailea : IBE	Funtzionalitateen kalitate-kontrola. Implementazioaren kalitate-kontrola (sistemen arkitektura eta diseinuaren kalitatea eta zuzentasuna).
Unibertsitateko tutorea: JMP	Dokumentazioaren kalitate-kontrola. Memoriaren kalitate-kontrola. Memoria aurkezteko webgunearen kalitate-kontrola. Aurkezpenaren kalitate-kontrola.

1 taula: Kalitatea bermatzeko rolak eta ardurak.

Horretaz gain, kalitatea bermatzeko proba-kasuen testuingurua azaltzea komeni da. Sistemen diseinua eta inplementazioa probatzeko, sistema bakoitzarentzat hainbat proba-kasu definitu dira. Proba-kasu guzti horien helburua da sistemak bete beharreko funtzioak ongi probatzea.

Proba-kasu horiek frogatzeko ez da metodo automatikorik erabili. Hots, ez da erabili *continuous integration* bezalako erremintarik; izan ere, oso zaila litzateke bi sistemetan ongi definitzea automatikoki probatzeko *script* bat (seguraski beste proiektu oso bat izan zitekeen). *Script* horrek denbora luzea hartuko luke eta ez guke aprobetxatuko denbora *DocGenerator* eta *RequirementTracer* hobetzen.

Hori dela eta, proba-kasurak exekutatu egin dira era manulean. Adibidez, probetako batek eskatzen zuen CSV fitxategian lerro pare bat ausaz borratzea. Hori eskuz borratu egin da eta eskuz exekutatu egin da ondoren.

Honek ez du esan nahi ez denik ongi probatu. Izan ere, denbora luzea eman da probetan, bi sistemen kalitatea ahalik eta handiena izateko.

## 5.5 Beste Erreferentziak

Beste erreferentzietan eranskinei erreferentzia egingo zaie esteka hiperesteka bidez. Eranskinetan hainbat motatako dokumentuak daude eta guzti-guztiak beharrezkoak izan dira proiektuaren memoria idazterako garaian.

- [Memoriaren eranskinak](#)
  - [A1: Sarrerako Dokumentazioa](#)
  - [A2: Analisi eta Diseinua](#)
    - [Arkitektura Koaderno I](#)
    - [Arkitektura Koaderno II](#)
  - [A3: Tamaina eta Esfortzu estimazioa](#)
  - [A4: Kudeaketa Plana](#)
    - [Integrazioaren Kudeaketa](#)
    - [Irismenaren Kudeaketa](#)
    - [Epeen Kudeaketa](#)
    - [Produktuaren Kostuen Kudeaketa](#)
    - [Kalitate Kudeaketa](#)
    - [Giza Baliabideen Kudeaketa](#)
    - [Komunikazioen Kudeaketa](#)
    - [Arriskuen Kudeaketa](#)
    - [Erosketen Kudeaketa](#)
    - [Interesatuen Kudeaketa](#)
  - [A5: Segurtasun Plana](#)
  - [A6: Beste Eranskinak](#)
    - [Hedapena](#)
      - [Produktuaren Dokumentazioa](#)
      - [Erabiltzaile Dokumentazioa I](#)
      - [Erabiltzaile Dokumentazioa II](#)
      - [Backout Plana](#)
      - [Hedapen Plana](#)
      - [Azpiegitura](#)
    - [Garapena](#)
      - [Build](#)
      - [Garatzaileen Probak](#)
      - [Diseinua - I](#)
      - [Diseinua - II](#)
    - [Ingurunea](#)
      - [Proiektuak definitutako prozesua](#)
      - [Tresnak](#)
      - [Garapen Kasua](#)
    - [Inplementazioa](#)
      - [Inplementazioa – I](#)
      - [Inplementazioa – II](#)

- [Test](#)
  - [Proba kasuak](#)
  - [Proba Log-a](#)
  - [Proba Script-a](#)
  
- [Sistemaren Espezifikazioa](#)
  - [Glosarioa](#)
  - [Bisioa](#)
  - [Betebeharren Espezifikazioa I](#)
  - [Betebeharren Espezifikazioa II](#)
  - [Erabilpen Kasuen Eredua](#)
  - [Erabilpen Kasuak](#)
- [Aurrekontua](#)
  - [Orokortasunak](#)
  - [Edukia](#)
- [Ikerlanak](#)
  - [Orokortasunak](#)
  - [Edukia](#)
  
- [Proiektuaren Barne Kudeaketa](#)
  - [Iterazio Plana](#)
  - [Proiektu Plana](#)
  - [Arrisku Zerrenda](#)
  - [Lan-Atazen Zerrenda](#)
  - [Barne Kudeaketa](#)
  - [Trebartzeko Materialak](#)
  - [Hizkuntza Hitzarmena](#)
  - [Bilera aktak](#)



## 6 Definizioak eta Laburdurak

Atal honetan definitu egiten dira proiektuan zehar eta proiektuaren dokumentazio zein memorian zehar ateratako kontzeptu, hitz gako, akronimo, sigla edo laburdurak, haien esanahia argitzeko asmoz.

### Definizioak

Bilduma honetan garrantzitsuak diren kontzeptuak azaldu egiten dira, baina hauek aparte, beste hainbat kontzeptu daude eranskinetako glosarioan bildu egiten direnak.

- **ALI:** Siglak (gaztelaraz): *Asociación de Titulados en Ingeniería Informática edo Asociación de Titulados Universitarios Oficiales en Informática*. 1980an sortutako elkarte da, informatikako doktoreak, lizentziatuak, diplomatuak eta gradu desberdinetako Ingeniariak biltzen dituena. Ikus, gainera: [ALI, webgunea](#).
- **Apache POI:** Java programazio-lengoaiarako APIa da, zeinek *Microsoft Office* formatuko dokumentuak manipulatzeko, editatzeko eta sortzeko ahalbidetzen duen. *Apache Software Foundation*-ek zuzendutako proiektua da. Proiektu honetan erabilitako bertsioa *Apache POI 4.1.0* da. Ikus, gainera: [Apache POI, webgunea](#).
- **API:** Sigla (ingelesez): *Application Programming Interface*. Funtzio, operazio eta prozedura bilduma da, beste software batek berrerabil dezakeena. API horiek libreak edo lizentziadunak izan daitezke.
- **ATO:** Sigla (ingelesez): *Automatic Train Operation*. *Auriga ATO* da izen osoa da eta tren bat automatikoki eta gidariaren kontrolik gabe gidatzea ahalbidetzen du hainbat automatizazio-mailatan. *ERTMS*aren interoperagarritasun espezifikokoetan oinarritutako *GoA2* automatizazioa eskaintzen du. Sistema bi ataletan banatzen da: trenbideko ATOa eta ontziratutako (*embedded*) ATOa. Proiektuan maiz agertzen den kontzeptua da; izan ere, *CAF Signalling*en garatzen ari den sistema da eta proiektua bere testuinguruan azaltzen da.
- **BAT:** *Batch* motako *script* fitxategien formatuaren izena eta atzizkia da. Sorta bidezko prozesamendua exekutatzen ahalbidetzen du. Izatez, formatu gabeko testu fitxategiak dira, ".bat" estentsioa dutena, *MS-DOS* motako aginduak dituena. Horren adibidea: "adibidea.bat".
- **BATCH:** *DOS*, *OS/2* eta *Microsoft Windows*en sorta bidezko prozesamendua exekutzeko fitxategia da. Formatu gabeko testu fitxategiak dira, *.BAT* formatua dutenak.

- **BETRADOK:** Akronimoa (euskaraz): Betekizunen Trazabilitatea eta Dokumentazio sorkuntza automatikoa. Proiektuaren izena da. Proiektuaren titulua luzeegia da maiz erabiltzeko, beraz, adostua izan zen "BETRADOK" deitzea proiektuko tutorearekin batera. Proiektuko dokumentazio eta memorian zehar maiz agertu daitekeen izena edo kontzeptua da.
- **CAF:** Sigla (gaztelaraz): *Construcciones y Auxiliar de Ferrocarriles*. Beasainen (Euskadi) egoitza soziala duen trengintzako enpresa da, 1917an sortua izan zena. Tranbia, metro edo trenbideko sareak mantendu, trenak eta burdinbideko materialak sortzen eta mantentzen ditu. Eskumendeko 45 enpresa ditu mundu osoan. Ikus, gainera: [CAF, webgunea](#).
- **CAF Signalling:** CAF Taldearen eskumendeko enpresa bat da, Donostiko Zuatzu enpresa-parkean eta Alcobendasen egoitza duena. Trenbideetarako kontrola eta seinaleztapen soluzio integralak sortzen dituen enpresa da. Ikus, gainera: [CAF Signalling, webgunea](#).
- **CAMEL:** Akronimoa (ingelesez): *CAF Model Editor & Language*. CAF Signalling-ek sortutako sistema da, ATOn funtzionalitatea definitzeko pentsatua, UML eta SysML modelatzeko erabiltzen diren tresnak baino gauza ahaltsuagoak egiten laguntzen duena. CAMEL proiektu honetan oso garrantzitsua da; izan ere, proiektua CAMEL sistemaren baitan egiten da.
- **CCII:** Sigla (gaztelaraz): *Consejo de Colegios de Ingeniería Informática*. Estatu-mailan informatika ingeniari guztiak errepresentatu eta bateratzen dituen antolakundea da. Ikus, gainera: [CCII, webgunea](#).
- **CCII N2016-02:** CCII erakundeak sortutako arauen-bilduma da, zeinek ingeniaritza informatikoko proiektuetarako dokumentazioaren estruktura eta beharrezkoak diren dokumentu eta sekzioak definitzen dituen. Ikus, gainera: [CCII N2016-02: Norma Técnica para la realización de la Documentación de Proyectos en Ingeniería Informática](#).
- **CI:** Sigla (ingelesez): *Continuous Integration*. Erreferentzia: [Continuous Integration](#).
- **CMMI:** Sigla (ingelesez): *Capability Maturity Model Integration*. Software garapenean dauden praktika onak definitzen duen modelo edo eredua da. Software sistemak garatu, mantendu eta operatzeko eredua da, CMMI Institutek definitutakoa. Ikus, gainera: [CMMI Institute](#).
- **Continuous Integration:** Euskaraz, etengabeko integrazioa edo integrazio etengabea da. Martin Fowler-ek proposatu zueneko eredu informatikoa da, non software garapen fasean akatsak identifikatzeko eta zuzentzeko aukera ematen duen. Etengabeko integrazioaren bi adar garrantzitsuenak konpilazioa eta probak dira.

- **DOC:** Laburdura (ingelesez): *document*. *DOC* edo *doc* fitxategi estentsio bat da *Microsoft Word* dokumentuak prozesatzeko.
- **DocGenerator:** Akronimoa (ingelesez): *Documentation Generator*. Proiektu honetan sortutako pakete bat da, irteera *DOCX* formatuan dokumentazioa daukana.
- **DOCX:** Laburdura (ingelesez): *document XML*. *DOCX* edo *docx* *Microsoft Open XML* bidez konprimatutako fitxategiaren formatua da. Nolabait *DOC* formatuaren bertsio berrizatzen daiteke.
- **DXL:** Sigla (ingelesez): *DOORS Extended Language*. Script bidezko programazio-lengoaia da, *IBM* enpresaren *IBM Rational DOORS* aplikazioa zabaltzeko balio duena. Programazio-lengoaia C eta C++-ren antzekoa da eta espezifiko da *IBM Rational DOORS*entzat.
- **Eclipse:** Software plataforma bat da, programazio-tresna multzo askok osatutakoa kode irekia. Hasiere batean *IBM* enpresak garatutako tresna izan zen, baina orain *Eclipse Foundation*ek garatzen du. Gaur egun, *Eclipse Public License* deitutako lizentziaren menpe dago. Ikus, gainera: [Eclipse Foundation](#).
- **Eclipse Modelling 2018-12:** proiektu honetan erabilitako *Eclipse* programako bertsioa da, 2018. urteko abenduan atera zena. Ikus, gainera: [Eclipse Modelling 2018-12, webgunea](#).
- **Embedded:** euskaraz, txertatuta. Software sistema bat txertatua dela esaten da, funtzio espezifiko bat edo gehiago egiteko diseinatuta dagoenean, normalean denbora-errealeko konputazio sistemetan.
- **EMF:** Sigla (ingelesez): *Eclipse Modelling Framework*. Sistemak eta aplikazioak modelatzeko eta kodea sortzeko tresna da, egituratutako ereduen arabera egindakoa. *CAMEL* garatzeko erabilitako *framework*-a da. Ikus, gainera: [EMF, webgunea](#).
- **ERTMS:** Sigla (ingelesez): *European Rail Traffic Management System*. Europar Batasunaren inizatibaz sortutako estandarra da, tren-sareen arteko interoperabilitatea ziurtatzeko. Mundu-mailan bakarra da funtzio hori daukana.
- **Eredu:** Hainbat definizio eduki ditzakeen kontzeptua da, baina proiektuaren baitan kokatuko dugu haren definizioa. Eredu bat sistema baten errepresentazio kontzeptuala da, hainbat kontzepturekin sortutakoa eta erabiltzaileak ulertzeko egina. Azken batean, kontzeptu orokor baten errepresentazioa da.
- **GitLab:** Bertsio kontrolerako eta garapenerako software kolaboratiboa da, *Git* erremintan oinarrituta. *Git*-ek eskaintzen dituen funtzionalitateaz aparte, *GitLab* errore jarraipen eta beste hainbat gauza integratuta ditu, hala nola, *CI* edota *wikia*. Ikus, gainera: [GitLab, webgunea](#).

- **GoA:** Sigla (ingelesez): *Grade of Automation*. UITP-k (*Unión Internacional del Transporte Público*) definitutako eta estandarizatutako kontzeptua trenak automatikoki gidatzeko mailak definitzeko. Hainbat maila desberdin daude, automatizazioaren araberakoak: *GoA0*, *GoA1*, *GoA2*, *GoA3*, *GoA4*. Ikus, gainera: [UITP](#), [webgunea](#).
- **GrAL:** Akronimoa (euskaraz): Gradu Amaierako Lana. Unibertsitateko Gradu baten bukaeran egin beharreko nahitaezko proiektua.
- **IBM:** Sigla (ingelesez): *International Business Machines Corporation*. New York-en egoitza duen teknologia eta aholkularitza enpresa famatua da, 378.000 langile inguru dituena. Ikus, gainera: [IBM](#), [webgunea](#).
- **IBM Rational DOORS:** IBM enpresaren aplikazioa da, "requirements" edo betekizunen jarraipena eramateko tresna izanik. Hainbat motatako sistema edo baliabideen betekizunak kudeatu, analizatu eta trazatzeko erraztasunak ematen ditu. Ikus, gainera: [IBM Engineering Requirements Management DOORS Family](#).
- **IBM Rational Rhapsody:** IBM enpresaren produktua da, UML eta SysMLan oinarritutako software garapen integratua laguntzen duena. Modeloa definitzen laguntzen du, baita dokumentazioa eta programazioko kodeketa sortzen ere. Ikus, gainera: [IBM Engineering Systems Design Rhapsody - Developer](#).
- **Iterazioa:** Software garatzeko modu arineko testuinguruan, denbora-unitate bat da. Denbora gehiago edo gutxiago irau dezakeen espazio-unitate da. Normalean, bi aste edo hiru asteko luzera izaten du.
- **Java:** *Sun Microsystems* enpresak komertzializatutako programazio-lengoaia da, baita plataforma informatikoa ere. Ikus, gainera: [Java](#), [webgunea](#).
- **LaTeX:** Tipografikoki kalitate handiko edukia sortzeko sistema da, testuen konposizioaz sortuta dagoena. Artikulu zientifiko eta liburu zientifikoetan asko erabiltzen den formatua eta sistema da. Ikus, gainera: [LaTeX](#), [webgunea](#).
- **Meta-eredu:** Software Ingeniaritzan, eredu bat definitzeko erregela, murriztapen, teoria eta eskemen bilduma gisa definitzeko lengoaia mota da. Metaeredu batek eredu bat definitzen du, eredu horrek beharrezkoak dituen ezaugarri guztiak azalduz. Espezifikoak izaten da edozein sistema, aplikazio edo lengoaia adierazteko. Lau maila abstraktuetan banatu ohi dira: M0, M1, M2, M3 mailak.
- **Meta-modelo:** Ikusi [meta-eredu](#).
- **Microsoft Word:** Testu prozesamendura bideratutako programa informatikoa da, *Microsoft Office* deitutako paketean integratuta dagoena eta *Microsoft* enpresak garatutakoa.

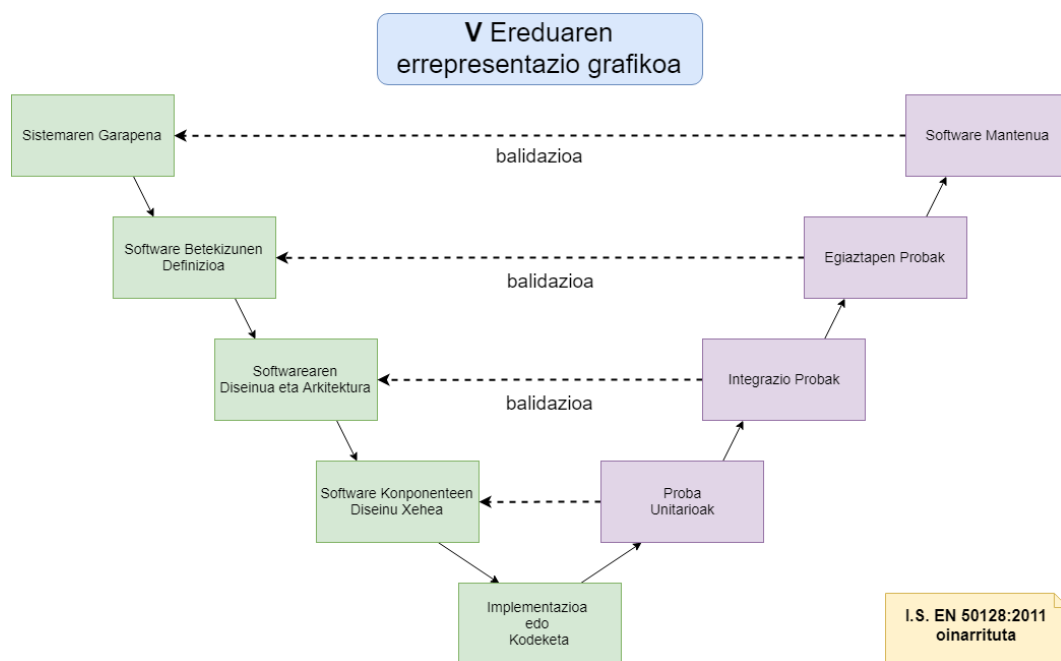


- **Modelo:** Modelo hitza azaltzen denean, *MDA*ren (*Model Driven Architecture*) testuinguruan azalduko da eta hala ulertu behar da ere. Software diseinurako gerturapen bat bezala ulertzen da. Domeinu baten barnean sistema bat definitzeko balio duena da.
- **ODT:** Akronimoa (ingelesez): *OpenDocument*. *OASIS*en dokumentu irekiko formatua da.
- **OpenUP:** Siglak (ingelesez): *Open Unified Process*. *RUP* (*Rational Unified Process*) metodologiaren azpimultzo bat da. Proiektu informatiko batean kokatzeko eta informazioa antolatuta izateko balio du. Halaber, proiektuaren elaborazio fasean produktua zein izan daitekeen edo zein bidetik lortu daitekeen definitzeko balio du. Ikusi, gainera: [OpenUP](#).
- **PlantUML:** Hainbat motatako diagramak lortzeko osagaia da. *Open Source* motakoa da eta hainbat motatako aplikazio eta tresnetan integratuta dago, baita integrazeko erraztasunak eman ere.
- **RedMine:** Proiektuen kudeaketarako erreminta da, erabiltzaileak proiektuak antolatzen eta jarraipena egiten ahalbidetzen duena. Horretaz gain, bere funtzionamendua hobetu daiteke funtzionalitate gehiago eranstean, *Open Source* motakoa delako.
- **RequirementTracer:** Proiektu honetan sortutako paketetako bat da, *java*, *DXL* eta *batch* programazio-lengoaieran idatzita dagoena eta betekizunen trazabilitatea errazten laguntzen duena.
- **RTF:** Siglak (ingelesez): *Rich Text Format*. *Microsoft* enpresak 1987. urtean garatutako fitxategi informatiko jakin baten formatua da, testu-prozesadore gehienek irakur eta uler dezaketena.
- **Safety:** Euskaraz: software segurtasuna. Kasu honetan, pertsonen bizitzaz arduratzen den software sistema baten segurtasuna da.
- **Scrum:** “Software Garapen Arineko” lan-metodologia da. Bere helburua proiektuaren eraikuntza iterazioetan (edo *Sprint* delakoetan) banatzea da. Horren helburua hurrengo da: proiektu handiak kudeatzen eta kontrolatzen lagundu eta aldaketa berriei modu eraginkorren aurre egin.
- **SIL:** Sigla (ingelesez): *Safety Integrity Level*. Arrisku gutxitze alderako segurtasun funtzioak eskaintzen duen estandarra da. Europar mailan 4 SIL desberdin definitzen dira, 1 maila baxuena izanik eta 4 maila altuena.
- **Software Architecture:** Euskaraz: software arkitektura. Sistema, aplikazio edo osagai baten ikuspegia orokorra da, zeinek sistema baten estruktura maila altuena osatzen duen. Software arkitektura bat aukeratu eta diseinatu egiten da helburu eta betekizunen arabera; modu abstraktu batean definitzen ditu osagaien arteko interfaze eta

komunikazioak; eta, azken batean, patroiz eta abstrakzio multzo jakin bat da. Proiektuan askotan aipatzen da "*Software Architecture*" edo "software arkitektura" kontzeptuak. Proiektuan software arkitektura definituta daukan eredu batekin asko lan egiten da, hortik dokumentazio automatikoa (*DocGenerator*) sortuz.

- **Software Definition:** Euskaraz: software definizioa. Software sistema baten osagai, murriztapen eta erlazioak definitzen dituen ikuspegia da. Software arkitektura baino maila altuago batean da, sistemako definiziotik oso hurbil eta betekizunak kontuan hartuz. Proiektu honetan asko aipatuko da "*Software Definition*" edo software definizioa kontzeptuak; izan ere, betekizunen trazabilitatea egiteko analisia software definiziotik hartutako eredu batetik abiatuta egiten da (*RequirementTracer*).
- **Software Design:** Euskaraz: software diseinua edo software diseinu xehea. Software sistema bateko konponente bakoitzaren deskribapenari dagokion kontzeptua da, non konponente horrek izango dituen klaseak eta haien arteko erlazio estatiko eta dinamikoak zehazten diren. Software garapen maila hau software arkitekturako hurrengo pausua da, eta inplementaziotik baino lehen kokatzen da. Proiektu honetan asko aipatzen da "*Software Design*" edo software diseinua; izan ere, software diseinuak definitzen dituen eredu batetik abiatuta lan egiten da dokumentazio automatikoa sortzeko (*DocGenerator*).
- **Software Agile Development:** Ikusi: [Software Garapen Arina](#).
- **Software Garapen Arina:** Software garatzeko modu jakin bat da garapen iteratibo eta inkrementalean oinarrituta, non sistemako betekizunak eta soluzioak eboluzionatu egiten diren denboran zehar eta proiektuaren beharren menpe. Softwarea garatzeko modu honetan talde antolatuetan lan egiten da eta epe motzerako erabakiak hartzen dira. Normalean plangintzak iterazio baten hasieran egiten dira iterazio horietan zer egingo den definituz. Ikusi, gainera: [Manifesto for Agile Software Development](#). Proiektu honetan software garapen arineko bi metodo erabiltzen dira: *OpenUP* eta *Scrum*.
- **StackOverflow:** Informatika garatzaileen komunitate handi bat biltzen dituen webgune bat da, non erabiltzaileek programazioko hainbat lengoaietako galderako egin ditzaketen. Ikusi, gainera: [StackOverflow, webgunea](#).
- **SysML:** Akronimoa (ingelesez): *Systems Modeling Language*. Sistemak espezifikatzeko lengoia estandarra da, *UML*aren azpimultzo hedatua dena.
- **SVN:** Laburdura (ingelesez): *Apache Subversion*. Bertsioen kontrolerako sistema da, *Open Source* motakoa da eta *Apache* garatutako produktua da.
- **SwArchitecture:** Laburdura (ingelesez). Ikusi: [Software Architecture](#).
- **SwDefinition:** Laburdura (ingelesez). Ikusi: [Software Definition](#).

- **SwDesign:** Laburdura (ingelesez). Ikusi: [Software Design](#).
- **Testing:** Software sistema baten funtzionalitatea probatzeko proba-multzoa.
- **UITP:** Siglak (gaztelera edo ingelesez): *Unión Internacional del Transporte Público* edo *International Association of Public Transport*. Transporte publikoaren baitan sortutako elkarte da.
- **UML:** Siglak (ingelesez): *Unified Modelling Language*. Modelaketarako lengoia bateratua da. Sistemak zehaztu, diseinatu eta eraikitzeo lengoia da, printzipioz objektuei orientatutako programaziorako prestatuta dagoena.
- **UNE:** Siglak (gaztelera): *Una Norma Española*. *AENOR*ren sortutako arau-esperimental, txosten eta arau-multzoen bilduma da. Normalizazio eta estandarizazio arauen bilduma dira.
- **"V" Zikloa, "V" Metodoa, "V" Modeloa:** Sistemak garatzerako garaian definituta dagoen prozedura edo lan egiteko modu bat da, zeinetan saiatzen den arriskuak minimizatzea eta proiektu zein produktuaren kalitatea hobetzea. Metodo honen helburua da sistema baten garapenerako dauden fase desberdinak definitzea, haren garapena balioztatzeo. Hau da, metodo honen bidez softwareak bete behar dituen betekizunak eta garapenerako prozedurak balidatzen ditu. Azken batean, modelo mota honen bidez grafikoki adierazten da sistema baten garapenerako bititza-zikloa. Sistemaren analisia eta diseinua software probekin bateratzen ditu eta haien arteko erlazioak nolakoak diren erakusten du. Hurrengo irudian horren errepresentazio grafikoa<sup>10</sup> agertzen da:



10 irudia: V ereduren errepresentazioa.

- VPN: Siglak (ingelesez): *Virtual Private Network*. Sare birtuala.

## Laburdurak

Hurrengo taula<sup>2</sup> honetan aipatzen dira memoria eta dokumentazioan zehar agertzen diren laburduren esanahia:

Laburdura	Esanahia
GrAL	Gradu Amaierako Lana
IBE	Iñaki Berriotxo
JLE	Jon Legarda Gonzalez
JOG	Jon Goya
MLA	Mikel Larramendi

*2 taula: Laburduren taula.*

## 7 Hasierako Betekizunak

Proiektua hasi bezain pronto, betekizunak definitu behar izan dira. Horiek bi zutabe nagusitan banatzen dira: *CAF-Signalling* enpresak garatzen dituen betekizunen trazabilitatea eta dokumentazio sorkuntza automatikoa egitea. Ondoren, sistema bakoitzeko betekizun espezifikoak zerrendatuko dira. Labur esanda, hauek dira proiektu honen betekizun nagusiak:

- *CAMEL* ekosistematik abiatuta proiektu generikoentzako dokumentazio formalaren sorkuntza automatikoa.
- *CAMEL* ekosistematik abiatuta proiektu generikoentzako inpaktu-analisia ahalbidetuko duen betekizunen trazabilitate automatikoa.

Bi helburu nagusi horiek hainbat betekizun espezifiko dituzte. Hurrengo azpialetan azalduko dira.

### Dokumentazio Sorkuntza Automatikoa: Betekizunak

Hurrengo taulan definitu eta azaldu egiten dira dokumentazio formalaren sorkuntza automatikoa lortzeko helburuaren betekizunak. Betekizun funtzionala eta ez-funtzionalak desberdin egiten dira.

Betekizunaren IDa	Betekizuna	Mota
B1.01	Lortutako dokumentazioa <i>Software Architecture</i> eta <i>Software Design</i> mailetatik lortu behar dira.	Funtzionala
B1.02	<i>Software Architecture</i> mailatik dokumentu bat lortu behar da.	Funtzionala
B1.03	<i>Software Design</i> mailatik osagai bakoitzeko ( <i>Component</i> ) dokumentu bat lortu behar da.	Funtzionala
B1.04	Garatzen den sistema <i>CAMEL</i> ekosistemaren parte izan behar da.	Funtzionala
B1.05	Garatzen den sistema proiektuekiko independentea izan behar da.	Funtzionala
B1.06	Garatzen den sistemako datu-iturria <i>CAMEL</i> ek sortutako ereduak izan behar dira.	Funtzionala
B1.07	Garatzen den sistema <i>Eclipse Framework</i> an garatutako plugina izan behar da.	Funtzionala
B1.08	Garatzen den sistema JAVA programazio-lengoiaren bidez idatzita egon behar da.	Funtzionala

Betekizunaren IDa	Betekizuna	Mota
B1.09	Garatzen den sistema <i>Continuous Integration</i> <sup>14</sup> bidez exekutatu behar da.	Funtzionala
B1.10	<i>CAMEL 2.0</i> bertsiorako prest egon behar du eta ahal den heinean, <i>CAMEL 3.0</i> bertsiorako ere prest egon behar du.	Ez funtzionala
B1.11	Sortutako dokumentazio formalean agertzen diren diagramak <i>Open Source</i> motako sistemarekin lortu behar dira.	Ez funtzionala
B1.12	Sortutako dokumentazio formala ahalik eta erabilgarrien den formatu batean izan behar da.	Ez funtzionala
B1.13	Garatzen den sistema dokumentuak manipulatzeko sistema edo <i>APIa Open Source</i> motakoa izan behar da.	Ez funtzionala
B1.14	Garatzen den sistema memoria fisikoa ez betetzea eta zaintzea ahalbidetu behar du.	Ez funtzionala

3 taula: DocGeneratorentzako betekizun funtzionalak eta ez-funtzionalak.

## Betekizunen Trazabilitate Automatikoa: Betekizunak

Hurrengo taulan definitu eta azaldu egiten dira betekizunen trazabilitate automatikoa lortzeko helburuaren betekizun funtzionalak eta ez-funtzionalak.

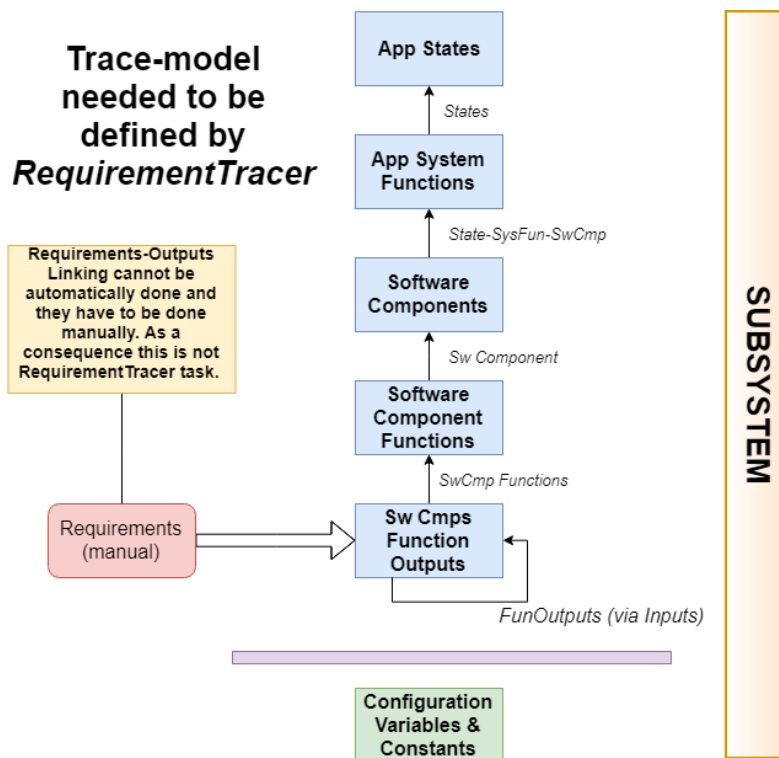
Betekizunaren IDa	Betekizuna	Mota
B2.01	Betekizunen trazabilitatea lortzeko <i>Software Definition</i> mailatik abiatutako traza lortu behar da.	Funtzionala
B2.02	Garatzen den sistema <i>CAMEL</i> ekosistemaren parte izan behar da.	Funtzionala
B2.03	Garatzen den sistema proiektuekiko independentea izan behar da.	Funtzionala
B2.04	Garatzen den sistemako datu-iturria <i>CAMEL</i> ek sortutako software definizioko ereduak izan behar da.	Funtzionala
B2.05	Betekizunak eta traza <i>IBM Rational DOORS</i> en bidez egin behar da.	Funtzionala
B2.06	Garatzen den sistemak hiru "alde" izan behar ditu: <ol style="list-style-type: none"> <li><i>Eclipse Framework</i>an garatutako plugina JAVA programazio-lengoan idatzita.</li> <li>Plugina eta <i>IBM Rational DOORS</i> bateratzen dituen <i>BATCH</i> bidezko exekuzioa.</li> <li><i>IBM Rational DOORS</i>en traza eguneratuko duen sistema <i>DXL</i> programazio-lengoan idatzitakoa.</li> </ol>	Funtzionala
B2.07	Pluginetik esportatu behar dira CSV formatuko fitxategi bat trazako elementu bakoitzeko datuak gordetzeko.	Funtzionala

<sup>14</sup> *Continuous Integration*: Euskaraz, etengabeko integrazioa edo integrazio etengabea da. Martin Fowler-ek proposatu zueneko eredu informatikoa da, non software garapen fasean akatsak identifikatzeko eta zuzentzeko aukera ematen duen. Etengabeko integrazioaren bi adar garrantzitsuenak konpilazioa eta probak dira.

Betekizunaren IDa	Betekizuna	Mota
B2.08	IBM Rational DOORSen trazako datuak inportatzeko CSV formatua erabili behar da.	Funtzionala.
B2.09	Traza honakoa izan behar da: <ul style="list-style-type: none"> <li>AppStates -&gt; App System Functions -&gt; Software Components -&gt; Software Component Functions -&gt; Software Component Function Outputs -&gt; Output (via Inputs).</li> <li>Configuration Variables &amp; Constants.</li> </ul>	Funtzionala.
B2.10	Garatzen den sistema <i>Continuous Integration</i> bidez exekutatu behar da.	Funtzionala
B2.11	CAMEL 2.0 bertsiorako prest egon behar du eta ahal den heinean, CAMEL 3.0 bertsiorako ere prest egon behar du.	Ez funtzionala
B2.12	Garatzen den sistema memoria fisikoa ez betetzea eta zaintzea ahalbidetu behar du.	Ez funtzionala
B2.13	Exekuzio denbora ahalik eta txikien lortzeko ahalegina egin behar da.	Ez funtzionala

4 taula: RequirementTracer-en helburuko betekizun funtzionala eta ez-funtzionalak.

Zehaztutako traza proiektuan sartutako enpresako zuzendariak diseinatutako traza da, hori baita beharrezkoa duten informazioa. B2.09 betekizunean zehaztutako trazaren errepresentazio bisuala honako irudi honetan ageri da:



11 irudia: DOORSen definitutako traza: B2.09.

Hala eta guztiz ere, hurrengo fitxategietan eta proiektuaren webgunean, eskuragarri daude betekizunen inguruko argibide gehiago:

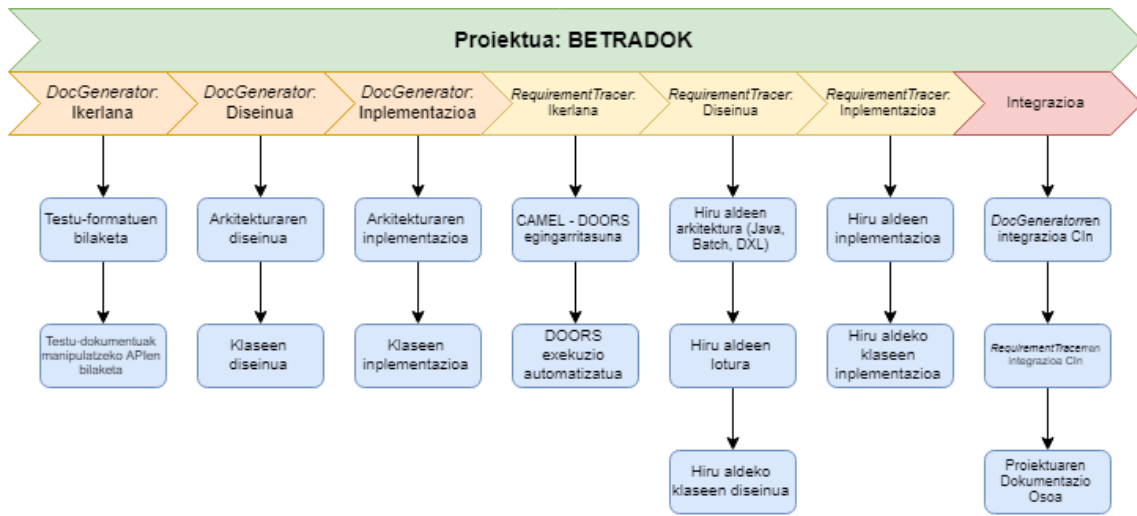
- [Project Vision \(Bisioa\)](#). 2019. Jon Legarda.
- [Betekizunen espezifikazioak – I.](#) 2019. Jon Legarda.
- [Betekizunen espezifikazioak – II.](#) 2019. Jon Legarda.
- [Erabilpen kasuak](#). 2019. Jon Legarda.
- [Erabilpen kasuen eredia](#). 2019. Jon Legarda.



## 8 Irismena

Proiektu honen irismenari dagokionez, hainbat puntu azpimarratu behar dira BETRADOKen *scopea* definitzeko. Horretarako ulertzeko garrantzitsua da jakitea **softwarearen bizi-zikloko hiru fase beteko direla: hasiera, elaborazioa eta eraikuntza** (inplementazioa eta probekin). Trantsizioa, ordea, proiektutik kanpo geratu egingo da, denbora mugatutako proiektua izango delako eta bezeroarekin hitzartutako baldintza delako. Hurrengo zerrendan deskribatu egiten dira irismena definituko duten ezaugarriak:

1. **OpenUP metodologiak** eskatzen dituen dokumentuak metodologia hura betetzeko. Dokumentu horiek metodologiak planteatzen dituen txantiloiekin egon behar dira osatuak. Aldi berean, **dokumentu horiek integratu behar dituzte enpresako lan-metodologia nagusia: SCRUM metodologia.**
2. **CCII N2016-02 arauak eskatzen duen dokumentazio egitura.** Ingeniaritza informatikako proiektu profesional baten dokumentazioa ere profesionala izan dadin, arau estandar bat erabiltzea oso garrantzitsua da. Kasu honetan, *CCII* elkargoarena.
3. **Proiektuaren webgunea** ([www.betradok.000webhostapp.com](http://www.betradok.000webhostapp.com)). Proiektuaren webgune honi esker *CCII N2016-02* araua jarraitzen duen webgune bat eskura izango du bezeroak. Webgune horrek hasiera, elaborazioa eta eraikuntza (inplementazioa eta probekin) faseen dokumentazio osoa izango du.
4. **Proiektuaren webgunea azaltzeko bideo-tutoriala.** Bideo horri esker bezeroak gida motz bat izango du BETRADOK proiektuko webgunea erabiltzeko eta ulertzeko.
5. **BETRADOK proiektuaren defentsako aurkezpena.** Proiektuaren defentsa Euskal Herriko Unibertsitateko Informatika Fakultatean aurkeztu behar da.
6. **DocGenerator eta RequirementTracer sistemaren egikaritzea.** Azken batean, proiektuaren helburua da bi sistema horien bizi-ziklo osoaren betetzea (mantenua izan ezik). Proiektuaren alde teknikoko irismena hobeto ulertzeko asmoz, *“Table Top Drawing”* deritzon irismen-diagrama diseinatu da. Diagrama horretan<sup>12</sup> irudia: Irismen-diagrama. hiru gauza garrantzitsu erakusten dira:
  - Errepresentazio eskematikoa da, aurrerapen lineala erakusten duena.
  - Proiektuaren egile bakoitza biltzen du, kasu honetan, soilik JLE.
  - Maila bakoitza kolore desberdin batekin markatuta dago, atazak eta azpiatazak desberdintzeko asmoarekin. Halaber, bigarren mailan kolore antzeko baina desberdinak erabiltzen dira sistema bakoitza desberdintzeko asmoz.



12 irudia: Irismen-diagrama.

## 9 Hipotesiak eta Murriztapenak

Proiektuaren hasierako planteamenduaren helburua automatizatu daitekeen edozer automatizatzea zen. Azken finean, zenbat eta automatizatuen, orduan eta langile gutxiago “denbora galtzen” ataza automatizagarri horietan. Proiektu handietan oso zaila da proiektuko bizi-zikloetan dokumentazio formal sendoa mantentzea, horregatik dokumentazio hura automatikoki sortzea oso garrantzitsua da. Gainera, oso interesgarria da betekizunak egoki eta automatikoki tratatzea, momentu oro jakiteko bezeroaren espezifikazioak betetzen diren edo ez jakiteko.

Lehenik eta behin, esan beharra dago kostu ekonomikoa ez dela bat ere aldatu hasierako planteamendutik. Izan ere, proiektuan zehar ez da kostu gehigarri sortu. Epeen inguruan, ordea, egia da luzatu egin dela pixka bat proiektuaren garapen denbora. Proiektuaren “denbora planifikazioa” atalean aipatzen den modura.

Horretaz aparte, *DocGenerator* sisteman formatu ireki bat (*RTF*, adibidez) erabiltzea pentsatu zen, baina azkenean *DOCX* formatua erabiltzea erabaki egin zen, *API* egokia aurkitu zelako. Halaber, aipatu egingo da bai “denbora planifikazioa” atalean bai “proposatutako soluzioaren deskribapena” atalean, baina hasiera batean *CAMEL*etik *Rhapsody*-ko proiektua sortzea nahi zen, ondoren *Rhapsody*-ko proiektutik abiatuta dokumentazioa sortzeko. Bide hori deuseztatu zen mantentze eta denbora-kostuak zirela eta. Aipatutako ataletan hobeto azaltzen da.

Gainera, *RequirementTracer* sistema diseinatzerako garaian, proiektua baino lehenagoko hipotesiak (*DOORS*en erabilera eta *CSV* formatuaren erabilera gehienbat) bete egin ziren eta ez zen aldaketa berezirik egin izan. Horrek, noski, erraztu egin zuen diseinu eta inplementazio faseak.

Aipatutako gauza guzti horietaz aparte, ez da murriztapen berezirik egon *BETRADOK*en garapenean zehar.



## 10 Aukeren Egingarritasun Ikerketa

BETRADOKeko helburuak lortzeko aukera desberdinen egingarritasun ikerketa bat burutu egin da hasiera batean. Egingarritasun ikerketa horren xedea proposatzen den soluzioa egiteko aukerak balioztatzea izan da.

Egingarritasun ikerketan hainbat gauza desberdinak begiratu dira.

### *IBM Rational Rhapsody* ardatz gisa hartuta: dokumentazio automatikoa sortzea

Proiektuaren helburuetako bat dokumentazio formala automatikoki lortzea da. Horretarako, hasierako planteamendu moduan bi pausu eman behar dira: lehenik eta behin, *IBM Rational Rhapsody*ko proiektu bat sortu *Rhapsody*ko APIa erabilia eta *CAMEL*ek definitutako modeloetatik abiatuta; bigarrenik, *Rhapsody*ko proiektua edukita dokumentazioa sortu definitzke dagoen formatu bat erabilia (Word, ODT, RTF...). Beraz, ikerketa moduan lehenengo aztertu behar dena zera da: *CAMEL*ek definitzen dituen modeloetatik abiatuta *Rhapsody*ko proiektu bat (.rpy formatua daukana) nola sortzen den eta elementu guztiak sortzeko ahalmena.

Hori dela eta, 2017an hasitako meta-modelotik *Rhapsody*-ko dokumentaziora transformatzeko egingarritasuna aztertu behar izan zuen enpresak. Horregatik, meta-modeloa diseinatu zuten pertsonak erabaki zuten “proba pilotu” bat egitea eta ikustea une hartan zuten meta-modelotik *Rhapsody*-rako transformazioa nolakoa izan zitekeen.

Proba “pilotu” horretan ikusi zen hasierako planteamendu hau egingarria zela. Hots, *CAMEL*ek definitzen dituen klase eta elementu guztiak, modu batean edo bestean, *Rhapsody*ko proiektu batera esportatu daitezkeela. Egia da *CAMEL*ek definitutako modeloko elementu jakin batzuk *Rhapsody*ko elementuekin parekatzea “zaila” dela. Hala eta guztiz ere, beti aurkitzen da bide egoki bat elementu guztiak ongi definitzeko. Ikerketa hori egin eta gero, taldeak utzi egin zuen bigarren pausua etorkizunerako: *Rhapsody*ko proiektu batetik dokumentazio automatikoa sortzea.

### *DocGenerator*: Dokumentazioa sortzeko formatua

Behin *IBM Rational Rhapsody*n software garapeneko fase bateko modeloa ongi sortuta edukita, dokumentazioa fisikoki sortzea geratzen da. Horretarako, oso garrantzitsua den gauza bat definitu behar zen ongi: dokumentazio horren formatua.

Zentzu horretan hainbat formaturen bideragarritasuna aztertu zen. Horien artean, garrantzitsuenak lau izan ziren:

- *Microsoft Word: DOC/DOCX.*
- *Open Office Writer: ODT.*
- *RTF (Rich Text Format).*
- *LaTeX.*

Horien azalpena eta hartutako erabakien zergatiak hurrengo lerroetan daude

### *LaTeX* deuseztatu

*LaTeX* oso formatu egokia da estetikoki itxura onarekin dokumentazioa sortzeko. Formatu hau testu liburu zientifiko askotan erabiltzen da, formatua behin zehaztuta oso itxura profesionalarekin sortzen dituelako.

Hala eta guztiz ere, ez da dokumentua aldatzeko eta maneiatzeko errazena den formatua. Hots, behin dokumentua sortuta, testu bat gehitzeko edo irudiak aldatzeko orduan ez du beste askok bezain malgutasun handia ematen; izan ere, aditua izan behar da formatua kontrolatzen eta *LaTeX* bidez idazten. Horregatik, deuseztatu zen lehen aukera izan zen.

Gainera, ez dago *API* erabilgarririk formatua ondo maneiatzeko. Beraz, sortzeko garaian, inplementazio oso sakona eta luzea aurreikusten zen eta ez zuen aurrera-pausu garrantzitsurik suposatuko.

### RTF deuseztatuta: *API* falta

RTF (*Rich Text Format*) formatu librea da, eta malgutasun handia du beste formatu batzuetara konbertsioa egiteko (Word, ODT...). Hori oso puntu positiboa da. Horregatik hau izan zen *API* bat bilatzeko lehen aukera.

Hala eta guztiz ere, arraroa bada ere, ez dago *API* osatua formatu hau erabiltzeko *Open Source* motakoa, hots, software librea dena. Aurkitutako bakarra garatzaile partikular batek sortutako liburutegia izan da:

- *RTF Document Constructor Library*<sup>15</sup> (Dima Popov).

---

<sup>15</sup> *RTF Document Constructor Library*: <https://www.codeproject.com/Articles/98062/RTF-Document-Constructor-Library>

Horrek ez du ematen fidagarritasun handirik; izan ere, gerta daiteke mantenerik ez edukitzea eta, beraz, momenturen batean euskarririk gabe geratzea eta gure sistemaren trinkotasuna galtzea.

## ODT vs DOC/DOCX

Azkenean bi izan dira geratutako formatuak: ODT eta DOC/DOCX, edo beste batera esanda, *OpenOffice Writer* eta *Microsoft Word*. Biren arteko diferentzia da bata librea dela eta bestea ez. Horregatik, erabakigarria izan diren ezaugarriak izan dira bata eta bestearentzako dauden *API*ak.

*Microsoft Word*erako prestatuago daude *API*ak une honetan. Horregatik, egindako ikerketa eta gero .DOC edota .DOCX formatuak erabiliko dira dokumentazioa *DocGenerator* sisteman automatikoki sortzeko.

## *DocGenerator*: *Microsoft Word* dokumentuak manipulatzeko *API*aren bilaketa

Behin erabakita formaturik egokiena *Microsoft Word* dela, *API*a erabakitzeko unea heldu da. Horien balorazioa egiteko hauek izan dira kontuan hartutako bi parametroak:

- *Open Source* motako *API*a izatea, lizentzia pribatuengan dependentziarik ez edukitzeko.
- Mantenugarritasun erraza duen *API*a izatea.
- *API*a Java programazio-lengoaian idatzia izatea.
- Komunitate "handia" liburutegia erabiltzen.

Zentzu horretan bi izan dira betekizunak hobekien bete dituzten liburutegiak. Bi parametro horiek betetzen dituzten *API*ak *Microsoft Word* dokumentuak hauek dira:

- *Apache POI: the Java API for Microsoft Documents*.
- *OpenOffice API*.

Bien arteko konparaketa guztiz praktikoa izan da. Dokumentu honetan ez da atxikitu egindako proben inplementazioa, non ikusi den zein *API*k ematen dituen erraztasun gehien inplementazioa aurrera eramateko eta zeinek eskaintzen dituen ahalik eta propietate gehien dokumentua manipulatzeko.

Azken erabakia honako hau izan da: *Apache POI* erabiltzea *Microsoft Word* dokumentuak manipulatzeko.

## DOC vs DOCX

Erabiliko dugun *API*a zein izango den badakigunez, erabakitzeko azken gauza da zein formatutan aterako den irteera. Garrantzitsua da hori ere erabakitzea eta oso argi edukitzea; izan ere, guztiz desberdinak diren objektuak (Objektuei Orientatutako Programazioa) erabili beharko dira.

Zentzu honetan ez da egon eztabaida handirik eta proiektuaren zuzendariarekin (MLE) erabaki da DOCX formatua erabiltzea, hori delako bietatik berriena.

## DocGenerator: Modeloetatik zuzenean dokumentazioa sortzea

Proiektuan zehar azaldu zen beste gai garrantzitsu bat: beharrezkoa al da benetan **Modelo -> Rhapsody -> Dokumentazioa** bidea egitea? Merezki du bi konbertsio egitea? Konbertsio bakar batean egin daiteke?

Egia esan ez dago pisuzko arrazoirik bi konbertsio egiteko, hasiera batean *IBM Rational Rhapsody*tik dokumentazioa sortzea errazago izango zela pentsatzea baino. Hala ere, horrek mantengarritasun arazo handia ekartzen du; izan ere, bi inplementazio (edo bi azpisistema) mantendu behar dira. Hau da, hasierako ideiaaren arabera, lehenik eta behin, *CAME*leko modeloetatik *IBM Rational Rhapsody*rako proiektua sortu behar da; eta bigarrenik, *IBM Rational Rhapsody*ko proiektutik dokumentazio formala sortu. Azken finean, horrek derrigorrez bi konbertsio egia ekartzen du eta ondorioz, bi inplementazio. Horrek ekartzen du lan bikoitza egin behar izatea. Egia da ere onura badaukala; izan ere, diagramak sortzeko erraztasun handiagoak ematen ditu *Rhapsody*ko proiektuak (eta *API*ak). Hala eta guztiz ere, taldean egindako bilerak eta gero ondorioztatu egin zen bide horretatik ez jotzea, eta bi tarteko pausu hori (*Rhapsody*ko proiektua sortzea) deuseztatzea.

Aipatutako guztia kontuan hartuta beharrezkoa suertatu zen diagramak sortzeko *API* edo liburutegi baten bilaketa egitea, modu horretan erdiko "pausu" hori ezabatu eta konbertsio bakarra egin daitekeelako. Zentzu honetan aipatu beharra da azkar aurkitu zela liburutegi hori: *PlantUML*.

## PlantUML diagramak sortzeko

*PlantUML* liburutegi libre bat da, eta erraztasunak ematen ditu hainbat motatako *UML* diagramak sortzeko, baita beste motatako diagramak ere. *GraphViz* erabiltzen du dependentsia gisa eta hainbat plataformetan erabili daiteke: *Eclipse*, *Word*, *GitLab*, *WordPress*... Horrek erakusten du oso erabilia den tresna dela eta merezi duela. Liburutegi honek sor ditzakeen diagramak hauek dira:

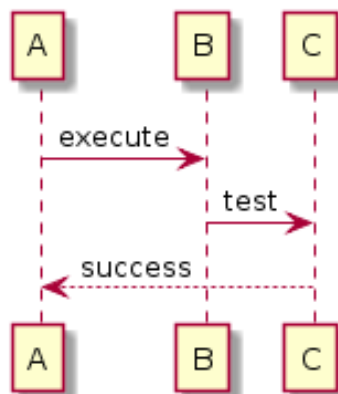


- Sekuentzia diagramak
- Klase diagramak
- Aktibitate diagramak
- Osagai diagramak
- *Egoera makina (State-Machine)* diagramak
- Objektu diagramak
- Hedapen diagramak
- *Timing* diagramak
- *SDL* diagramak
- *Gantt* diagramak
- *etab.*

Lengoaia espezifikoa erabiltzen da hura sortzeko, horrelako itxura daukana (adibide gisa, sekuentzia diagrama sortzen du):

```
@startuml
A -> B : execute
B -> C : test
C --> A : success
@enduml
```

13 irudia: PlantUMLen diagramak sortzeko lengoiaren adibidea.



14 irudia: PlantUMLren sekuentzia-diagramaren adibide bat.

Laburbilduz, ez da lengoaia zaila eta aukera ugari ematen dituen osagaia da.

## Dokumentazioan irudia txertatzeko estrategia

Ikertzeko beste gai garrantzitsua da zein modutan txertatuko diren irudiak behin kodea sortu dela dokumentuan. Bi modu daude horretarako, hurrengo lerroetan deskribatzen direnak.

*Java* bidez irudia sortu eta “irudi gisa” txertatu dokumentuan; azken batean, irudi finko bat edukita. Horrek ez du uzten irudia editatzen.

Bigarren bidea da meta-kodea *Word* dokumentuan txertatu eta *PlantUML*ren *Word*eko osagaiarekin exekutatu. Modu horretan, dokumentuan bertan (*.DOCX*) aurrez ateratako kodea edukiko genuke eta botoi baten bidez irudia sortuko litzateke. Horren alde garrantzitsua da irudia editatu daitekeela, baina horrek badu alde txarra: irudia modelotik inferitu egiten da, momentu jakin baten modeloan dagoen informazioarekin; beraz, ez du zentzurik editatzeko aukerarik egotea, ez litzatekeelako trinkoa eta segurua. Horretaz aparte, probak egin eta gero ikusi da errore eta hutsune batzuk daudela *carriage return*ekin.

Ondorioz, lehenengo bidea jarraituko da: irudia zuzenean txertatu (automatikoki).

## *RequirementTracer*: *IBM Rational DOORS* komando bidez exekutatzeko aukera

*RequirementTracer* sisteman oso garrantzitsua da dena ahalik eta era automatizatuan exekutatzeko lortzea, hasierako planteamendua kontuan hartuta. Dena nahiko argi zegoen hasieran, baina automatizaziorako bide horretan hutsune argi bat zegoen: posiblea al da *IBM Rational DOORS* komando bidez exekutatzeko eta parametro batzuk bidaliz nahi den funtzioa exekutatzeko?

Hori egiaztatzen, enpresako beste kide batzuei galdetuta eta ikerlan txiki batekin lortu da erantzuna: bai, posiblea da *DOORS* exekutatzeko komando bidez parametroak pasata eta nahi den *DXL* funtzio espezifiko exekutatzeko. Horretarako, beharrezko gauza bakarra *batch* bidezko *scripta* da, eta komando espezifiko bat. Horrek ere erakusten du *Continuous Integration*en sartzeko aukera bideragarria dela, *batch* motako scriptak erabili daitezkeelako *Clan*. Ikusi hurrengo webgunetako informazio eta parametroen aukera guztiak:

- *IBM, DXL* exekutatu komando bidez<sup>16</sup>.

---

<sup>16</sup> *IBM, DXL* exekutatu komando bidez:

<https://www.ibm.com/developerworks/community/forums/html/topic?id=77777777-0000-0000-0000-000014794906>

- *IBM*, komando aukera guztiak<sup>17</sup>.

Hori bideragarria dela ikusita, egiaztatu da betekizunen trazabilitatea %100ean automatiza daitekeela.

---

<sup>17</sup> IBM, komando aukera guztiak:

[https://www.ibm.com/support/knowledgecenter/en/SSYQBZ\\_9.6.1/com.ibm.doors.configuring.doc/topic/s/c\\_clientcommandswitches.html](https://www.ibm.com/support/knowledgecenter/en/SSYQBZ_9.6.1/com.ibm.doors.configuring.doc/topic/s/c_clientcommandswitches.html)



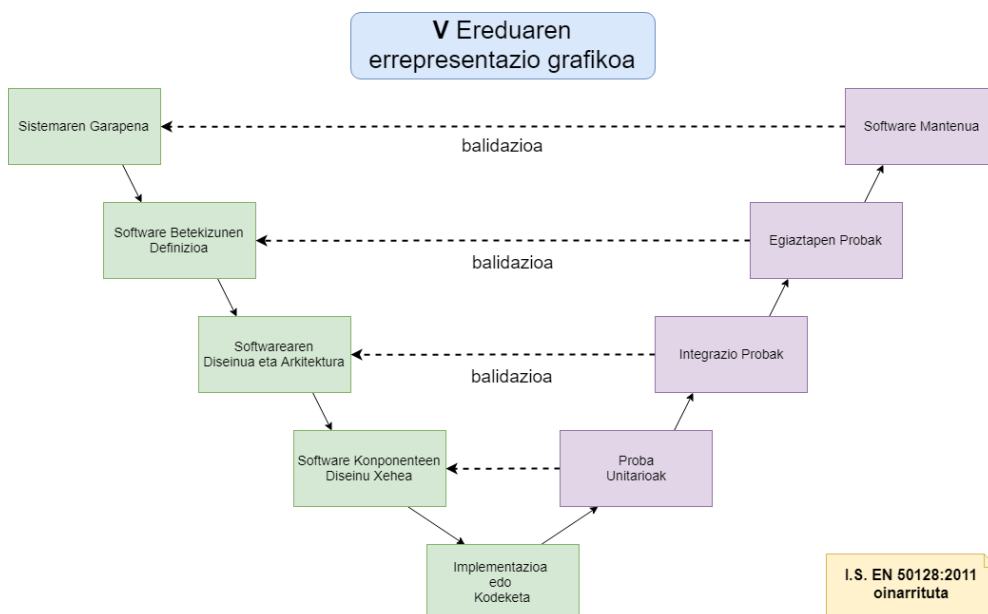
## 11 Proposatutako Sistemaren Deskribapena

Proposatutako soluzioak BETRADOK proiektuan bi izan dira: batetik, dokumentazio sorkuntza automatikoa ahalbidetzen duen sistema; bestetik, betekizunen trazabilitate automatikoa ahalbidetzen duen sistema. Hori dela eta, bi deskribapen egin dira hurrengo lerroetan. Hasiera batean, *CAMEL* eta bi sistemen arteko elkarrekintza nola egiten den deskribatuko da eta, ondoren, sistema bakoitzaren deskribapena egingo da.

### Proposatutako Sistemak eta *CAMEL*: Testuingurua

Lehenik eta behin, ulertu behar da proposatutako bi sistemak *CAMEL*en menpe egiten dutela lan. Horrek esan nahi du *CAMEL*ek daukan software garatzeko faseen espezifikazioak kontuan hartu behar direla bi sistemen inplementazioa eta diseinua gauzatzeko.

*CAMEL*ek bide osoa "V" ereduko software garapenean egiten du. Software garatzeko eredu honek sistema baten espezifikazio orokorretik espezifikorako bidea egiten du, ondoren fase bakoitzerako *softwarearen probak* ere lotuz. Hurrengo "V" ereduaren errepresentazio grafikoa<sup>15</sup> agertzen da (aurreko atal batean ere ageri da).

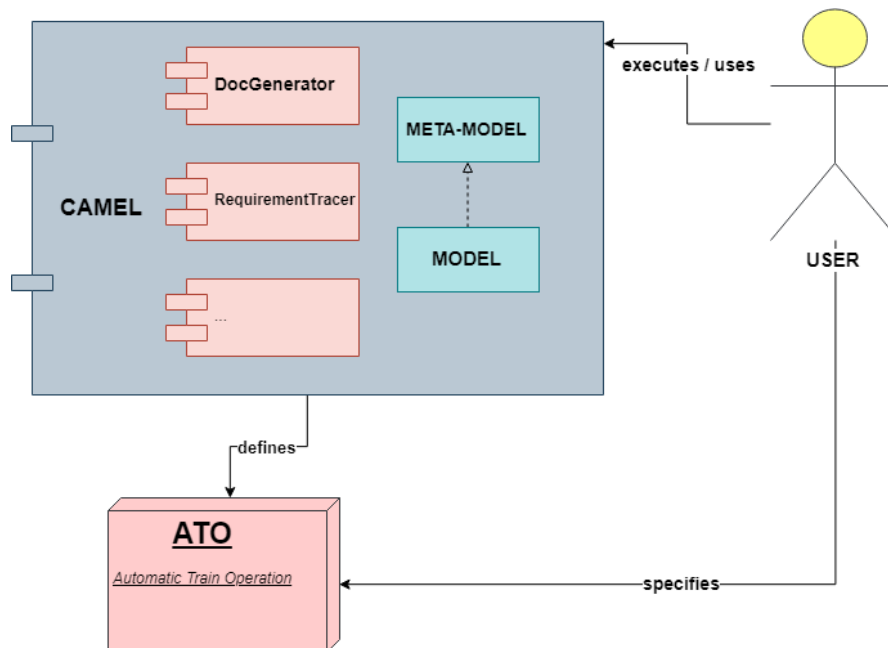


15 irudia: "V" ereduaren errepresentazioa.

Eredu hura jarraitzeko *CAMEL* tresna sortu zen. Beraz, BETRADOKeko bi sistemak software definizioa, software arkitektura eta software diseinua espezifikatzen duten metaereduak eredu hori erabiltzen dute. Metaeredu (meta-modelo) bat erabilita, eredu asko

ateratzen dira. Gure helburua da proiektua dena delakoa izanik ere, sistemak funtzionatzea. Horregatik metaereduen espezifikazioa kontuan hartzen da sistemak diseinatzeko eta inplementatzeko garaian.

Hori horrela izanik, lehenik ulertu behar da *CAMEL* eta proiektuko bi sistemen funtzionamendu orokorra. Hurrengo irudiak<sup>16</sup> adierazten du elkarrekintza horren nondik norakoa.



16 irudia: *CAMEL*-Sistemak funtzionamendu orokorra.

Irudian<sup>16</sup> ikusi daitekeen bezala, *CAMEL* ATO sistema definitzeko balio du eta erabiltzaileek (software garatzaileek, gure kasuan) *CAMEL* erabiltzen dute horretarako. Proiektuko bi sistemak, *DocGenerator* eta *RequirementTracer* *CAMEL*en barruan dauden bi osagai modura ulertu daitezke.

## Osagaien Deskribapena

Hurrengo bi atal nagusiak *BETRADOK* proiektuan diseinatu eta garatu diren bi sistemen azalpenak emango dira. Horretarako hainbat diagrama eta atal jorratu nahi izan dira:

- Arkitektura orokorraren diagrama.
- Erabilpen kasuen eredia eta aktibitate-diagrama.
- Anlisi ereduko sekuentzia-diagrama eta klase-diagramak.
- Inplementazioko pakete-diagrama orokorra eta xehetasun orokorrak.
- Froga edo proben kobertura.

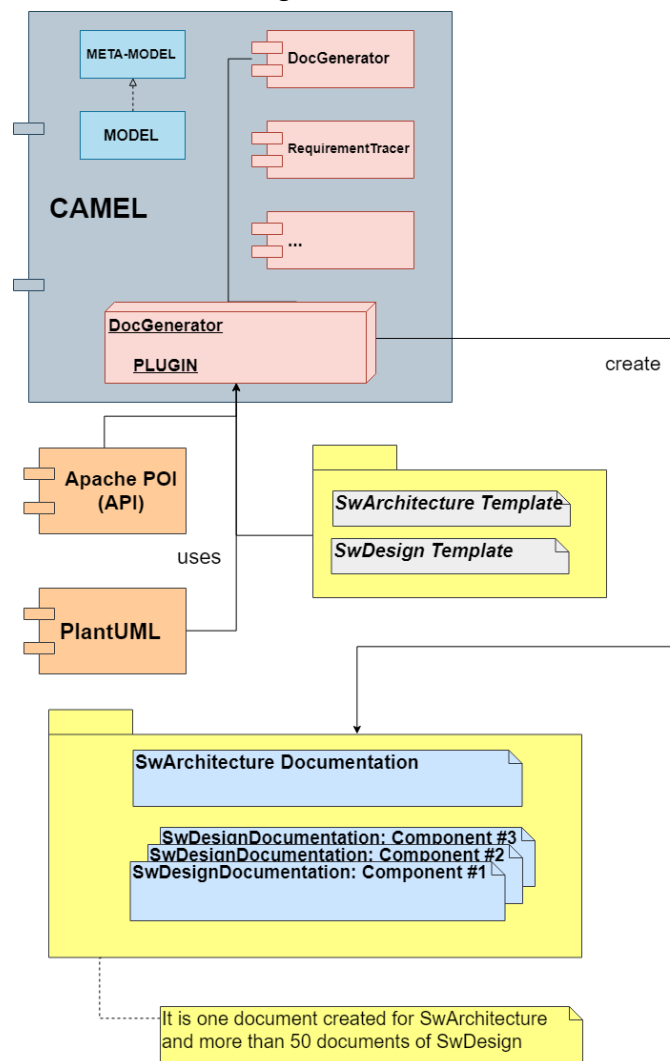
Hori guztiari esker, espero da dokumentu honen irakurleak argi izatea diseinuaren nondik norakoak. Horren ildotik, ez da kodea azalduko; izan ere, pentsatzen da ez dela beharrezkoa proposatutako sistemen deskribapenerako.

## DocGenerator sistema

Proiektuko lehenengo helburua dokumentazio sorkuntza automatikoa izan da. Horren barne, bi dokumentazio formal jakinak eskatu dira: software arkitekturako dokumentazioa eta software diseinuko dokumentazioa.

### DocGenerator: Arkitektura

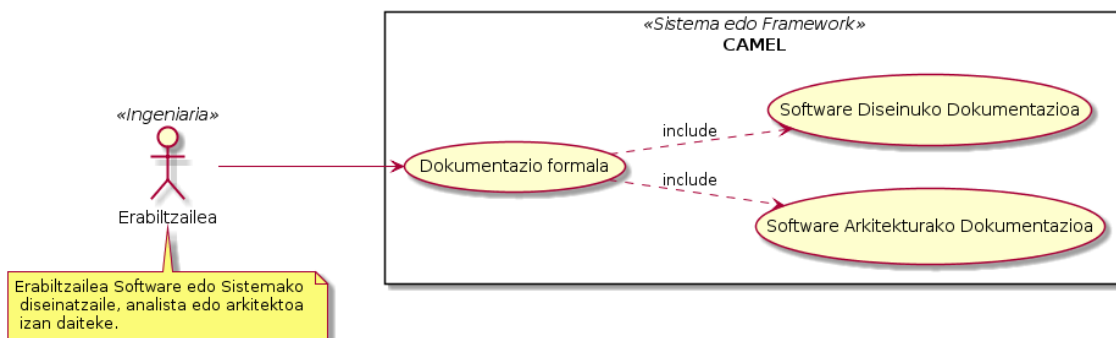
DocGenerator sistema edo osagaiarentzako diseinatutako arkitektura honakoa da:



17 irudia: DocGenerator sistemaren arkitektura orokorra.

## Erabilpen kasuen eredua

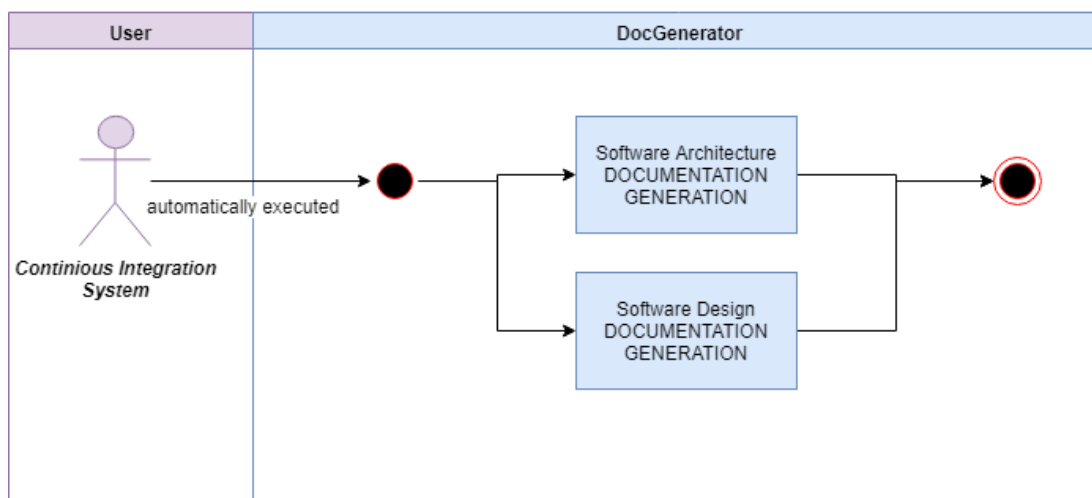
Erabilpen kasuak hurrengo irudian deskribatzen dira.



18 irudia: DocGeneratorren erabilpen kasuen eredua.

Honi lotuta aktibitate-diagrama simple bat ere diseinatu da:

## DocGenerator



19 irudia: DocGeneratorreko aktibitate-diagrama (analisi-mailan).

## DocGeneratorren erabilera: Parametroen hitzarmena

Aipatu bezala, *DocGenerator CAMEL* bidez exekutatu da, haren gainean diseinatu eta inplementatutako osagaia delako. Hori kontuan hartuta parametro bidez informazioa pasatzea hitzartu egin zen, proiektuekiko independentea den osagaia bihurtzeko asmoz. Modu horretan, *ATO* ez den beste proiektu batentzako prest egon daiteke. Horregatik bi argumentu eskatzen zaizkie, lortu nahi den dokumentazio-motaren arabera:



Komandoa	Aukera	1. parametroa	2. parametroa
<b>docGenerator</b>	-a (Software arkitekturako dokumentazioa lortzeko).	sw_arkitekturako_ereduko_patha. (.swarch).	dokumentazioa gordetzeko <i>patha</i>
	-d (Software diseinuko dokumentazioa lortzeko).	sw_diseinuko_ereduko_patha. (.swdsg).	

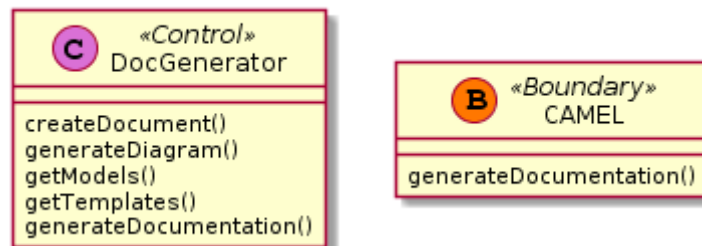
5 taula: Parametroen hitzarmena.

## Analisi Eredua

### Klase-Diagramak

Horretaz gain, ulertzeko hobeto zein funtzio daukan entitate bakoitzak hurrengo klase-diagramak sekuentzia-diagramako entitate eta objektu bakoitzak analisi-mailan dauzkaten funtzio nagusiak adierazten dira:

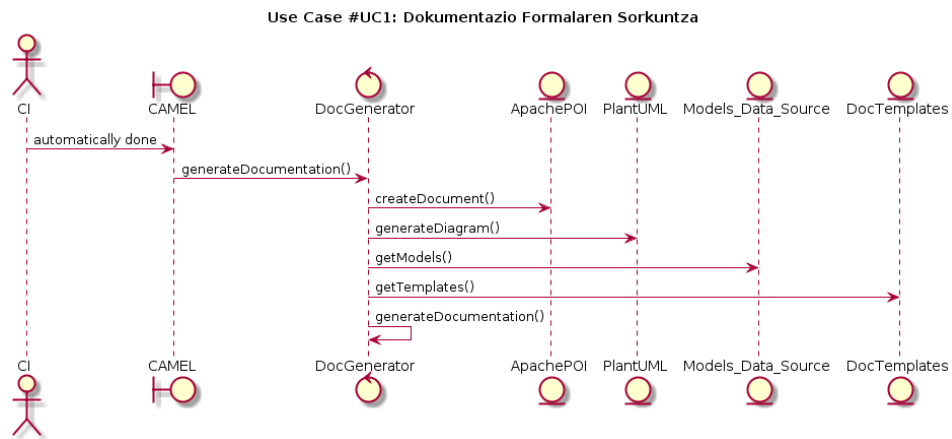
### DocGenerator Class Diagrams



20 irudia: DocGeneratorreko klase-diagrama (analisi-mailan).

### Sekuentzia-Diagrama

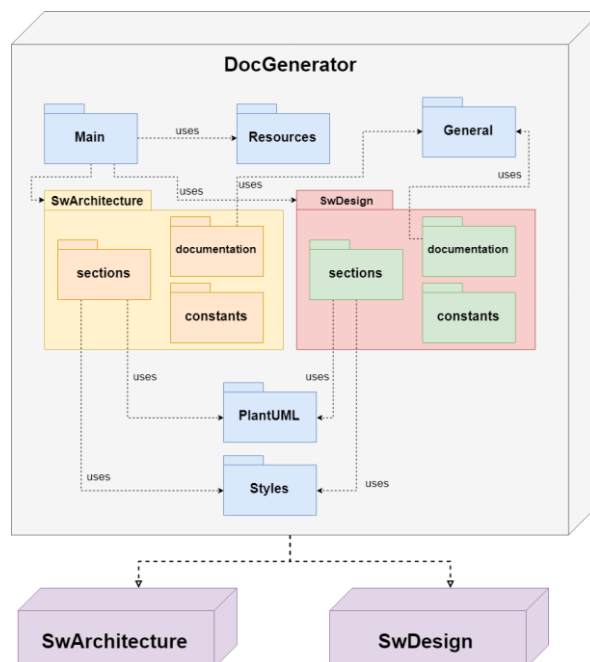
Aurreko ataletan ageri dira sistema honek erabiltzen dituen bi osagai nagusi: *Apache POI* eta *PlantUML*. Bi osagai horiekin konektatzeko dagokion sekuentzia-diagrama ulertzeko hobeto haren funtzionamendu honakoa da: *continious integration* bidez *CAMEL*eko komando bat exekutatu *DocGenerator* sistema exekutatzen da. Horrek diagramak sortzen ditu *PlantUML* bidez eta dokumentuak manipulatzeko *Apache POI* bidez eta txantiloak dokumentuetan idazten du modeloetan jasotako informazioa.



21 irudia: *DocGenerator*reko analisi ereduiko sekuentzia-diagrama (analisi-mailan).

## Implementazioa

Implementazioaren inguruan, jakin behar da *DocGenerator* azken finean *CAMEL* barnean dagoen *plugin* bat dela eta *Eclipse Modelling Tool 2018-12* tresnarekin garatu dena eta *java* programazio- lengoian idatzita dagoena. Hurrengo irudian<sup>22</sup> azaldu nahi da oso modu orokorrean eta xehetasun txikietan sartu gabe zeintzuk dira estruktura horren pakete garrantzitsuenak eta haien arteko dependentziak zeintzuk diren. Aurretik azaldu moduan, ez da kodearen azalpenik emango.



22 irudia: *DocGenerator*ren pakete-diagrama (implementazio-mailan).

Bukatzeko, deskribatu nahi da erabilitako patroik nagusiak inplementazioa gauzatzeko garaian. Patroi horiek bi izan dira (nagusienak):

- Klase estatikoen erabilera. Memoria asko ez kargatzeko eta ahal den heinean ez memoria ez erabiltzeko, klase estatikoak erabili izan dira gehienetan.
- Klase abstraktuen erabilera. Klase bat baino gehiagok metodo berdinak erabili behar badituzte, klase abstraktu eta interfazeak diseinatu dira. Erabaki horri esker, kodea mantengarria eta hedagarria da.

### Xehetasun teknikoak

Ikus daitekeen modura, sistema alde edo zati bakarra dauka, hots, ez da beste azpisistemetan banatzen inondik inora. Horrek mantentzea eta hobetzea erraztu egiten du.

Datu gisa, 2019ko uztailaren 10eko bertsioak (adibide gisa soilik) 10 minutu behar ditu *DocGenerator* exekutatzeko. Irteera gisa 51 dokumentu berri ateratzen dira (software arkitekturako bat eta software diseinuko berrogeita hamar). Batz besteko kalkulua eginez, 51 dokumentu horietan gutxi gora behera 8000 orri daude guztira. Proiektuaren tamaina hobeto ulertzeko datu aproposak dira horiek.

Hurrengo estekan aurkitu daitezke ateratako dokumentu automatikoen ereduak (ez daude sortzen diren guztiak, konfidentziasuna dela eta):

- [Dokumentazio automatikoaren ereduak](#).

Hurrengo estekan aurkitu daiteke *DocGenerator* pakete osoa:

- [DocGenerator](#). 2019. Jon Legarda.

**Oharra:** ez saiatu exekutatzeko *DocGenerator*; izan ere, ez du funtzionatzeko *CAMEL*en eredurik gabe.

### *DocGenerator*: Probak

Hurrengo taulak<sup>6</sup> ongi laburbiltzen du sistemaren gainean egindako proba guztien informazioa. Oso taula orokorra da, ez dira probak egikaritzeko komandoak zehaztu. Hala ere, ongi definitzen dute zein kasu dauden frogatuta.

Proba Titulua	Deskribapena	Emaitza
<i>DocGeneratorek</i> ez ditu txantiloak ( <i>template</i> ) aurkitzen.	Dokumentazioa sortzeko garaian, txantiloirik ez pasatzea.	Ongi

Proba Titulua	Deskribapena	Eraitza
<i>DocGeneratorek</i> behar dituen eta txantiloien markadoreak ez doaz bat.	Dokumentazioa sortzeko garaian, txantiloak hitzarmen batean dauden markadoreak izan behar ditu kontuan. Gerta liteke hitzarmeneko markadore horiek ez egotea.	Ongi
<i>DocGeneratorri</i> parametro desegokiak pasatzen zaizkio I.	<i>DocGenerator</i> exekutatzeko parametro hiru pasa behar zaizkio: 1) Modeloaren karpeta <i>patha</i> . 2) Dokumentazioa sortzeko karpeta. 3) Modelo bertsioa. Probatu behar da parametro desegokiak pasatzen. Kasu honetan, kantitate gutxiago.	Ongi
<i>DocGeneratorri</i> parametro desegokiak pasatzen zaizkio II.	<i>DocGenerator</i> exekutatzeko parametro hiru pasa behar zaizkio: 1) Modeloaren karpeta <i>patha</i> . 2) Dokumentazioa sortzeko karpeta. 3) Modelo bertsioa. Probatu behar da parametro desegokiak pasatzen. Kasu honetan, kantitate gehiago.	Ongi
<i>DocGeneratorri</i> parametro desegokiak pasatzen zaizkio III.	<i>DocGenerator</i> exekutatzeko parametro hiru pasa behar zaizkio: 1) Modeloaren karpeta <i>patha</i> . 2) Dokumentazioa sortzeko karpeta. 3) Modelo bertsioa. Probatu behar da parametro desegokiak pasatzen. Kasu honetan, parametroak ez dute behar duten informazioa.	Ongi

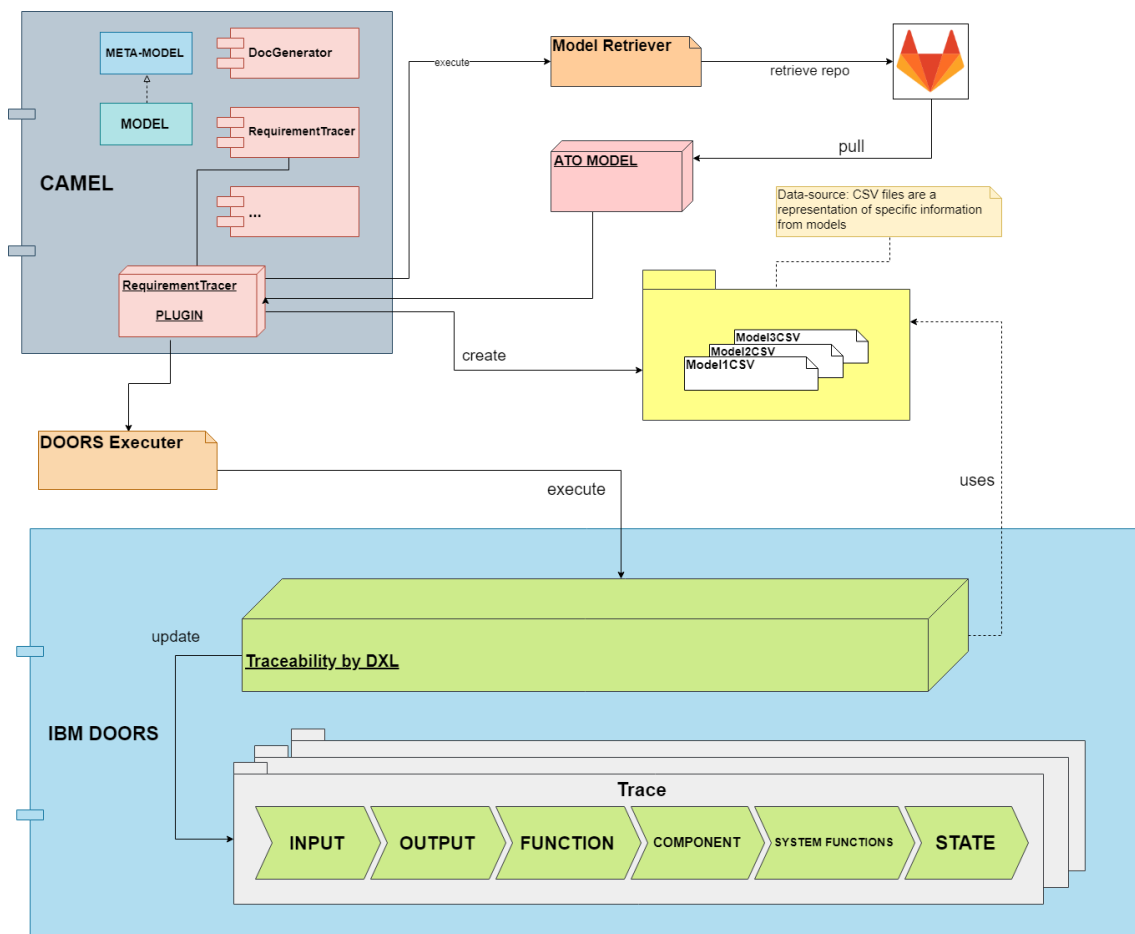
6 taula: *DocGenerator* sistemaren proben taulak.

## RequirementTracer osagaia

BETRADOK proiektuko bigarren helburua betekizunen trazabilitate automatikoa egitea izan da. Horretarako software definizioko eredutik traza jakin bat diseinatu egin da eta hori izan behar da automatikoki eguneratuko den informazioa.

## RequirementTracer: Arkitektura

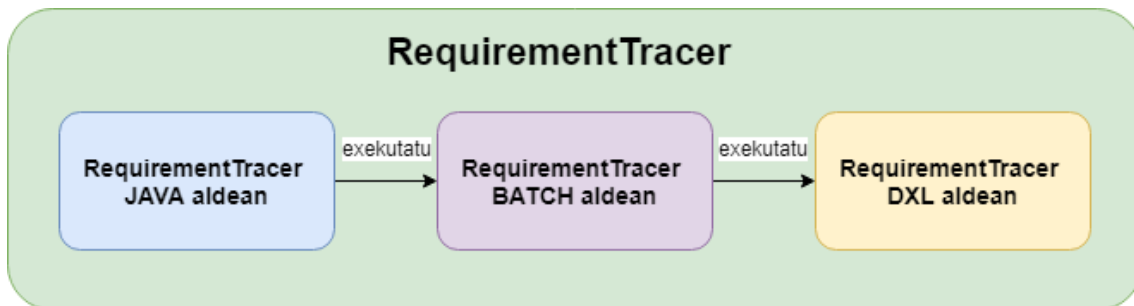
Hurrengo irudian<sup>23</sup> ikus daitekeen bezala, oso sinplea den diagrama diseinatu da. Arkitekturako diagrama ez da *UML* diagrama bat, baina oso modu sinple eta deskriptiboan adierazten da sistemaren funtzionamendu orokorra.



23 irudia: RequirementTracerreko arkitekturaren diagrama.

Hau da diagrama hobeto ulertzeko deskribapen azkar bat: lehenik jaso behar da modeloen informazioa eta hori *Git* bidez lortuko da. *Git*-i esker lortuko da proiektuko software definizio mailako espezifikazio berriena. Ondoren, *RequirementTracer* pluginak sortu egingo ditu CSV fitxategiak (trazaren araberakoa). CSV formatuko fitxategi horiek aurrerago erabiliko dira. Geroago, *RequirementTracer* pluginak *batch* formatuko script bat exekutatuko du, zeinak *DOORS* exekutatuko duen eta aurrez aurredefinituta dagoen funtzio espezifiko bat exekutatuko duen, kasu honetan *traceability* deitutakoa. Horri esker, *DOORS*eko traza eguneratuko du, aurretik sortutako CSV fitxategiak erabilia.

Aipatutako bide luze hori hiru aldetan exekutatzen da: lehenik, *plugin* bat; bigarrenik, *batch* formatuko *script* bat; eta, hirugarrenez, *DOORS* barruan exekutatzen den funtzioa. Nolabait hiru zatitan banatutako sistema da.



24 irudia: *RequirementTracer*ren antolaketa hiru aldetan.

Aurreko diagrama<sup>24</sup> kontuan hartuta, beharrezkoa den informazioa deskribatuko da alde bakoitzeko xehetasunak jakitearren.

*RequirementTracer*reko pluginari buruz:

- *Eclipse Modelling Tool 2018-12* tresnarekin garatu da.
- *Plugin* bat da.
- *Java* programazio-lengoaian idatzia da.
- Aurreko atal batean<sup>65</sup> zehaztutako bi parametroak eskatzen ditu.
- *CSV* fitxategiak sortzen ditu.
- Bukaeran, *batch* aldeko zatia exekutatzen du.

*RequirementTracer*reko *BATCH* aldeko zatiari buruz:

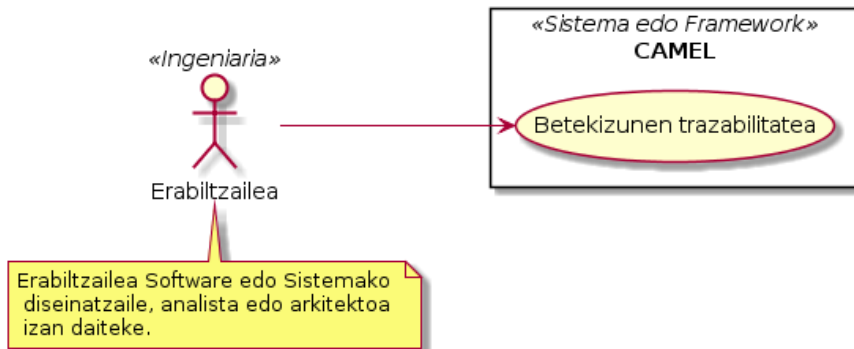
- *Plugin*ak pasatzen dizkion parametro batzuk erabiltzen ditu.
- *IBM Rational DOORS* exekutatzen du.
- *DOOR*seko funtzio espezifiko bat exekutatzen du, *DXL* aldeko funtzio nagusia dena.

*RequirementTracer*reko *DXL* aldeko zatiari buruz:

- *BATCH* aldeko zatiak pasatuko parametroak irakurtzen ditu.
- *CSV* formatuko fitxategiak irakurtzen ditu eta dagokion traza eguneratzen du.

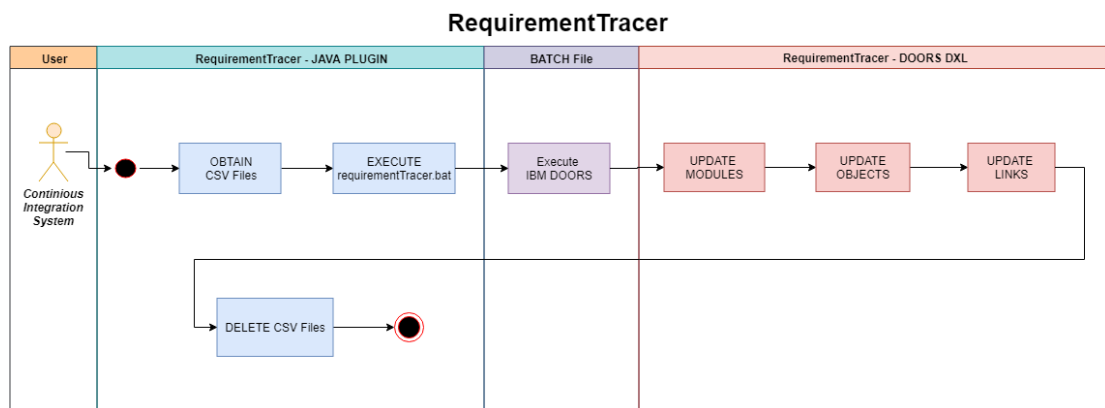
## Erabilpen Kasuen Eredua

Lehenik eta behin, sistemak bete behar dituen erabilpen kasuak definitu dira. Horretarako hurrengo irudiak erabilpen-kasuen eredua errepresentatzen du.



25 irudia: *RequirementTracer*ren erabilpen kasuen ereduak.

Horretaz aparte, *RequirementTracer*entzako ere diseinatu egin da aktibitate-diagrama. Irakurleak hobetu uler dezan zeintzuk diren jarraitu beharreko pausuak.



26 irudia: *RequirementTracer*reko aktibitate-diagrama (analisi-mailan).

### *RequirementTracer*ren erabilera: Parametroen hitzarmena

Behin baino gehiagotan aipatu den bezala, *RequirementTracer* sistema *CAMEL*ek exekutatu du komando bidez. Komando bidez exekutatzeak ahalbidetzen du *continuous integration* bidez exekutatzea. Horretaz gain, sistema exekutatzeke parametroak erabiltzen badira, proiektuekiko generiko den sistema lortu daiteke. Hurrengo taulan adierazten da exekutatzeke modua komando bidez eta beharrezkoak diren parametroak.

Komandoa	1. parametroa	2. parametroa
<i>RequirementTracer</i>	Eredu erroaren <i>patha</i> . Informazioa nondik jaso behar den erakusten duen <i>patha</i> .	Software definizioke <i>patha</i> .

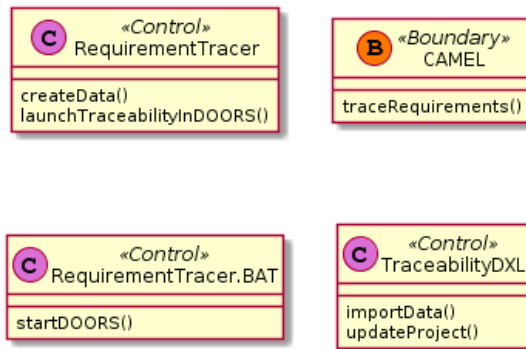
7 taula: *RequirementTracer*ren parametroen taula.

## Analisi eredu

### Klase-Diagrama

Hurrengo diagrama<sup>27</sup> klase-diagrama da (analisi-mailakoa) eta ongi erakusten du nola banatzen diren ardurak hiru zatietan zehar.

#### RequirementTracer Class Diagrams

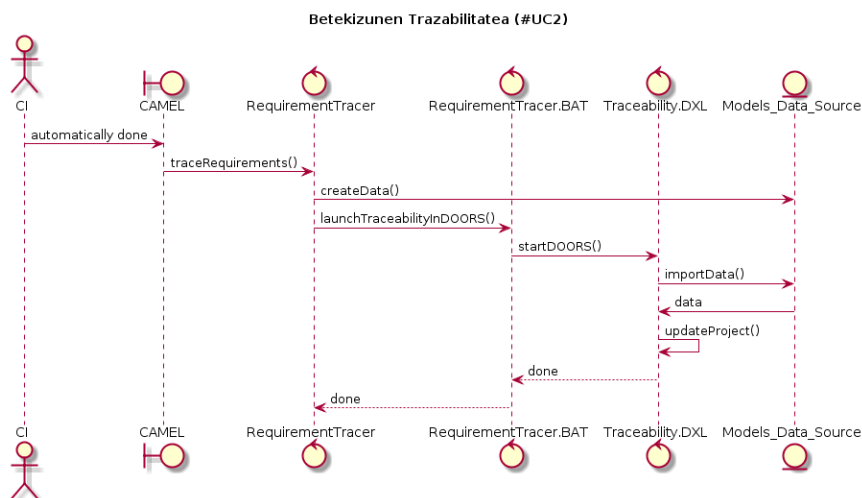


27 irudia: RequirementTracerren klase-diagrama (analisi-mailan).

Aurreko diagramatik ikus daiteke *CAMEL*en arduraz aparte, entitate bakoitzak adierazten du hiru aldetako bakoitza.

### Sekuentzia-Diagrama

Hau guztia hobeto ulertzeko asmoz, analisi-ereduko sekuentzia-diagrama ere diseinatu da, non hobeto ikusi daitezkeen entitate nagusiak eta exekututzen diren ordena argi ikusten den. Gainera, aktibitate-diagrama ere erakutsiko da, horrek ere ongi erakusten duelako hiru zati desberdinen exekuzioaren ordena.



28 irudia: RequirementTracerreko sekuentzia-diagrama (analisi-mailan).



Aurreko diagraman<sup>28</sup> ongi ikusi daiteke beti ere hiru alde edo zati desberdin daudela sistemaren exekuzioan zehar. *BETRADOK* proiektuan *RequirementTracer* deitua izan den sistema hiru alde horiek dituen sistema da eta hiru alde horiek garatu egin dira proiektuan zehar.

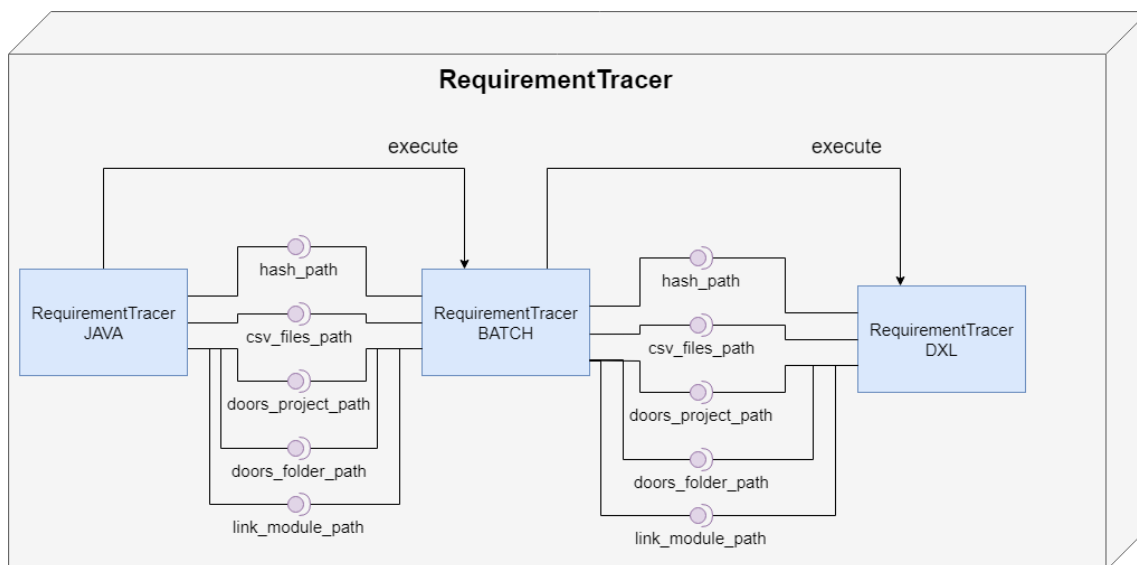
Gainera, ikusi daiteke sistema osoa exekutzeko ardura *Continuous Integration*ena (*CI*) dela. Garrantzitsua da hori aipatzea, baina egia da komando-bidez exekutzeko prestatuta dagoela, beraz, ez da bakarrik erabilgarria jarraikako integrazioaren bidez exekutzeko, beste modutan erabiltzeko ere prestatuta dago.

## Implementazioa

Horretaz guztiez aparte, implementazioaren inguruko argibideak ere eman nahi dira, kasu batzuetarako. Ez da kodea azalduko (oso luzea litzateke), baina implementazioa hobetu ulertzeko hainbat kontzeptu landu eta azalduko dira.

### Sistema barruko parametroak

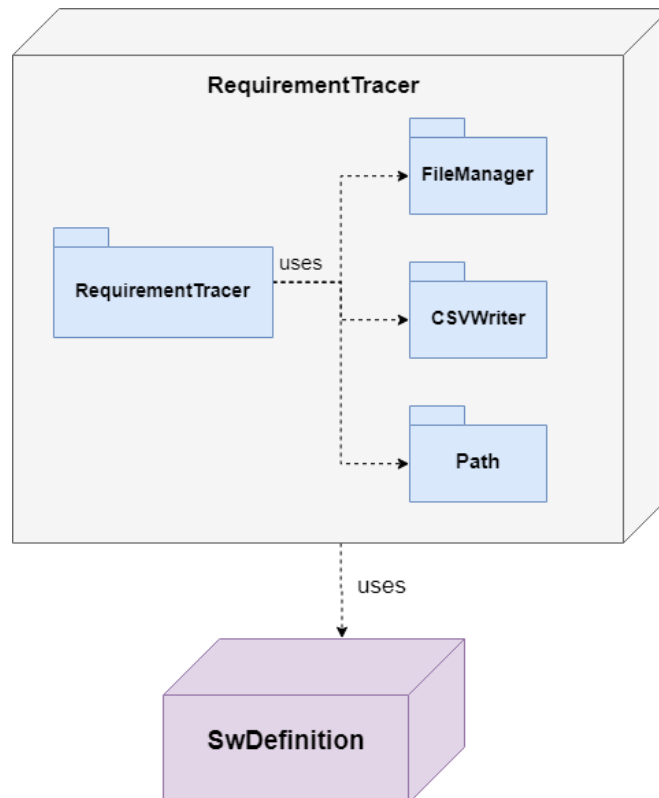
Lehenik eta behin, sistema barruko parametroak nola pasatzen diren adierazteko diagrama agertzen da. Ohartu behar da parametroak nolabait ezkerreko azpisistemetatik eskuineko azpisistemetara pasatzen direla. Gainera, parametro hauek ez dira sistema erabiltzeko erabiltzaile batek jakin beharreko argumentuak. Parametro hauek azaltzeko arrazoi bakarra zera da: erabiltzaileak jakin dezan sistema barruan zein parametro pasatzen dira.



29 irudia: *RequirementTracer* barruko parametroen eskema.

## Pluginaren pakete-diagrama

Hurrengo irudi honek adierazten ditu *RequirementTracer*reko pluginak (hots, *java* aldeko osagaia) dituen inplementazioko pakete-diagrama sinple bat. Ez daude klase espezifikoak adierazita; izan ere, aipatu bezala dokumentu honetan ez da kode-mailako argibide zehatzik eman nahi.



30 irudia: *RequirementTracer*reko pluginaren pakete-diagrama.

Hainbatetan aipatu bezala, *RequirementTracer*ren *plugin* edo *java* aldeko zati honen helburu hauek ditu:

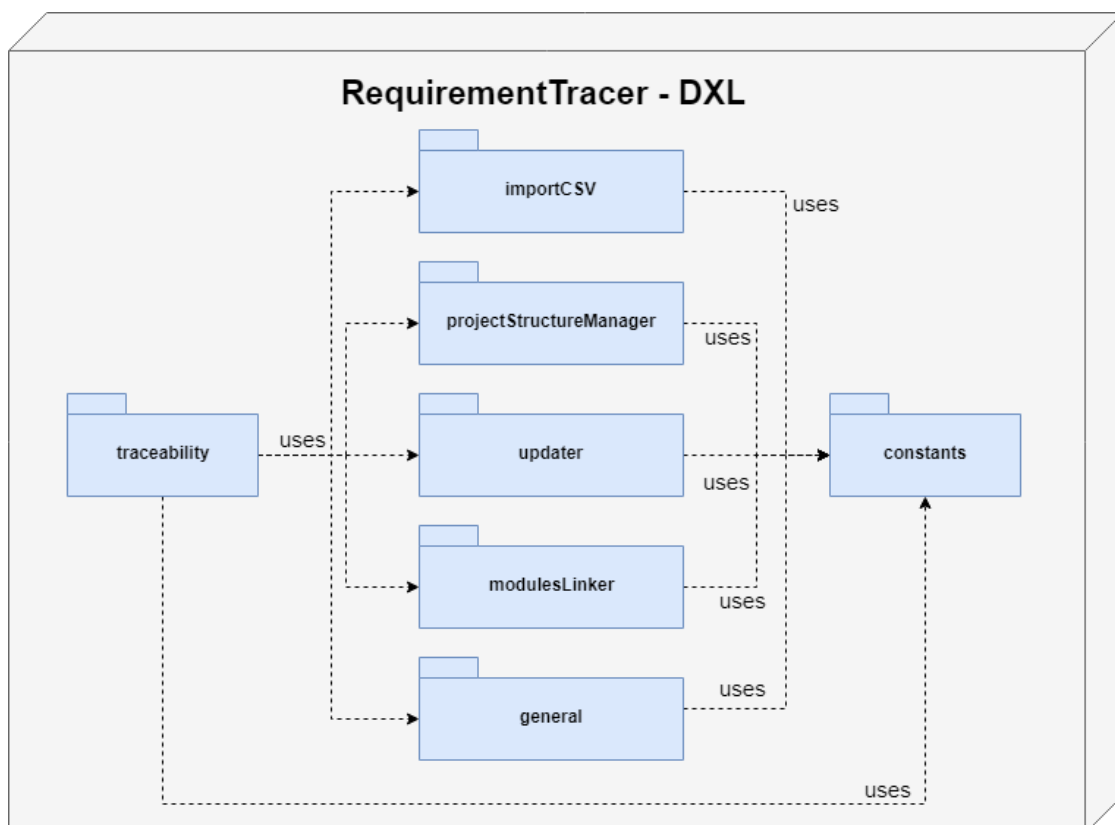
- Software definizioko datuak lortu eta *CSV* fitxategiak sortu.
- *BATCH* sistemako *scripta* exekutatu.

*RequirementTracer* javan idatzitako pluginaren patroiak hauek izan dira:

- Klase estatikoen erabilera. Memoria ez kargatzeko eta ahal den heinean memoria ez erabiltzeko, klase estatikoak erabili izan dira gehienetan.
- Klase abstraktuen erabilera. Klase bat baino gehiagok metodo berdinak erabili behar badituzte, klase abstraktu eta interfazeak diseinatu dira. Erabaki horri esker, kodea mantengarria eta hedagarria da.

## DXL aldeko fitxategi-Diagrama

*RequirementTracer* sistemako hirugarren aldea *DXL* bidez idatzita dagoen zatia da, *IBM Rational DOORS* barruan exekutatzen dena. Beraz, fitxategi hauek (*RequirementTracer*reko) alde edo zati hau *IBM Rational DOORS* barruko karpeta-sistemetan gorde behar da. Hurrengo irudiak fitxategi guzti horien dependentziak eta izenak adierazten ditu. Aurreko ataletan aipatu bezala, ez da kodearen azalpenik emango.



31 irudia: *RequirementTracer*reko *DXL* aldeko antolaketa.

*RequirementTracer* *DXL* aldeko patroiak hauek izan dira:

- Moduluetan banatzea funtzio nagusiak. Kode nagusia handia da eta *DXL*k ez du beste programazio-lengoaiek eskaintzen duten pakete banaketarik, beraz, printzipioz kode osoa fitxategi bakar batean idatzi daiteke. Hala eta guztiz ere, kodea irakurgarria izateko eta mantengarria izan dadin, funtzio desberdinetan banatu egin da kodea.
- Memoria ez kargatzeko, sortzen diren estruktura bereziak bukaeran ezabatu egiten dira (*Skip* listak gehienbat).

## Xehetasun teknikoak

Datu gisa, 2019ko uztailean lortu egin da sistema osoa exekutatzea PC simple batean 45 segundotan. Denbora aldetik oso eraginkorra da soluzioa; izan ere, PC ahaltuago batean aurreikusten da denbora horren erdian exekutatuko dela.

Hurrengo estekan inplementazioa ikus daiteke. Kode osoa da:

- [RequirementTracer](#). 2019. Jon Legarda.

### *RequirementTracer*: Probak

Hurrengo taulak ongi laburbiltzen du *RequirementTracer* gainean egindako software proba guztiaren informazioa. Taula orokorra da, ez dira probak egikaritzeko komandoak zehaztu. Hala ere, ongi definitzen dute zein kasu dauden frogatuta.

Proba Titulua	Deskribapena	Emaitza
<i>SwDefinition</i> eko iturri modeloko objektu baten editatzea.	Iturri gisa jokatzeko duen proba pilotuko modulu batean objektu baten informazioa editatzea frogatzen da.	Ongi
<i>SwDefinition</i> eko iturri modeloko objektu baten borratzea.	CSVan borratuta baldin badago objektu bat, moduluan borratzen ongi borratzen den frogatzen da.	Ongi
<i>SwDefinition</i> eko iturri modeloko objektu baten gehitzea.	CSVan elementu berri bat gehitu baldin bada, <i>IBM Rational DOOR</i> Seke moduluan ere objektua gehitu den ala ez frogatzen da.	Ongi
<i>SwDefinition</i> eko helburu modeloko objektu baten editatzea.	Helburu gisa jokatzeko duen proba pilotuko modulu batean objektu baten informazioa editatzea frogatzen da.	Ongi
<i>SwDefinition</i> eko helburu modeloko objektu baten borratzea.	CSVan borratuta baldin badago objektu bat, helburu (target) gisako moduluan ongi borratzen den ala ez frogatzen da.	Ongi
<i>SwDefinition</i> eko helburu modeloko objektu baten gehitzea.	CSVan elementu berri bat gehitu baldin bada, <i>IBM Rational DOOR</i> Seke helburu (target) moduluan ere gehitu den objektua ala ez.	Ongi
Objektuen Linken (interno eta externoen) eguneraketa.	CSVa manipulatu eta konektatuta egon behar liratekeen objektuen ezabaketak linken ezabaketak ere dakarren ala ez. Berezko konportamenduaren arabera, eguneratu behar da objektua eta haien linkak, linken iturria jatorritzat hartuz.	Ongi

Proba Titulua	Deskribapena	Emaitza
Moduluetako objektuan editatu, borratu eta gehitu aldi berean.	Aurreko proba-kasu guztiak aldi berean frogatzen da. CSVetan objektuak borratu, editatu eta gehitzen dira jakiteko proiektu osoa ondo editatzen den.	Ongi
<i>IBM Rational DOORS</i> erako instantzia sortzea ( <i>start</i> ) argumenturik pasa gabe.	<i>RequirementTracer</i> paketea hiru azpisisematan banatuta dago. <i>BATCH</i> aldean dagoen scriptaren bidez <i>IBM Rational DOORS</i> exekutatu behar da, beti argumentu batzuk pasata.	Ongi
<i>IBM Rational DOORS</i> erako instantzia sortzea ( <i>start</i> ) argumenturik gutxiagirekin.	<i>RequirementTracer</i> paketea hiru azpisisematan banatuta dago. <i>BATCH</i> aldean dagoen scriptaren bidez <i>IBM Rational DOORS</i> exekutatu behar da, beti argumentu batzuk pasata.	Ongi
<i>IBM Rational DOORS</i> erako instantzia sortzea ( <i>start</i> ) argumentu gehiagorekin.	<i>RequirementTracer</i> paketea hiru azpisisematan banatuta dago. <i>BATCH</i> aldean dagoen scriptaren bidez <i>IBM Rational DOORS</i> exekutatu behar da, beti argumentu batzuk pasata.	Ongi
<i>RequirementTracer</i> osoaren exekuzioa lokalean.	<i>RequirementTracer</i> paketea hiru azpisisematan banatuta dago. Hiru horien artean konektatu eta komunikatu behar dira exekuzioa ondo joan dadin eta nahi den emaitza ateratzeko. Kasu honetan, lokalean egiten da, hots, ez <i>Continious Integration</i> aldean.	Ongi
<i>RequirementTracer</i> osoaren exekuzioa <i>Continious Integration</i> en.	<i>RequirementTracer</i> paketea hiru azpisisematan banatuta dago. Hiru horien artean konektatu eta komunikatu behar dira exekuzioa ondo joan dadin eta nahi den emaitza ateratzeko. Kasu honetan, <i>Continious Integration</i> ekin batera egiten da.	Ongi

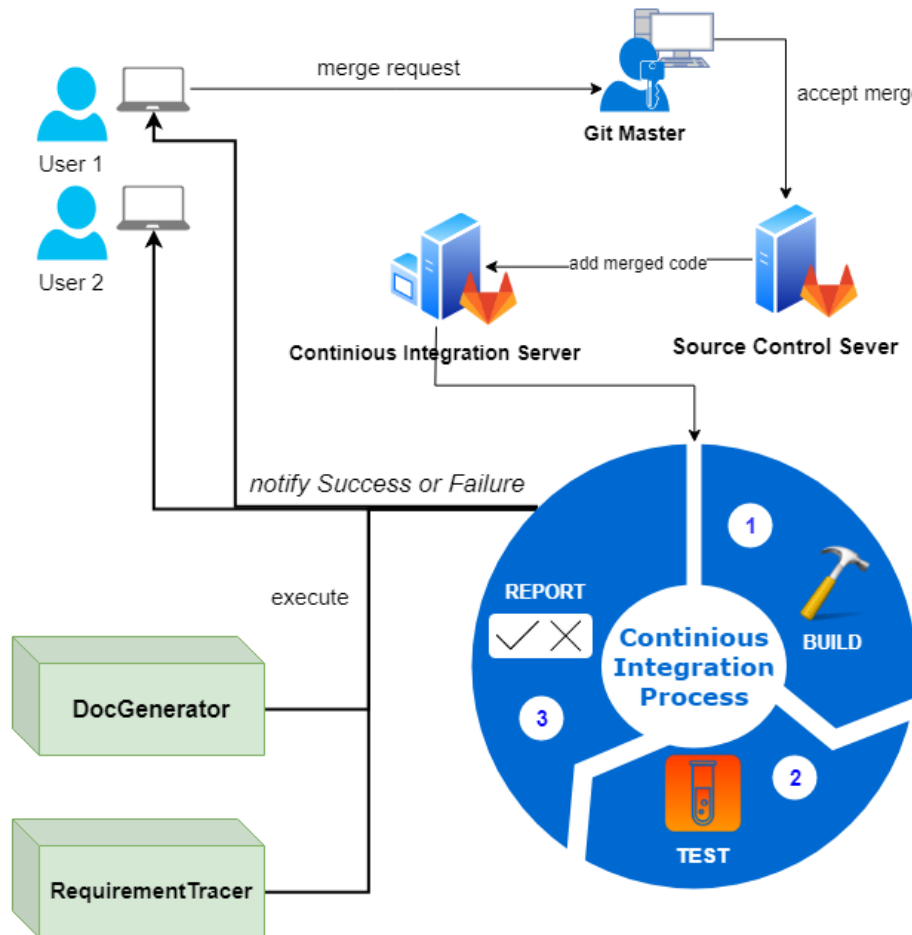
8 taula: *RequirementTracer*ren proba kasuen taula.

## Sistemen Hedapena: *Continious Integration*

Hurrengo pausua bi sistemak (*DocGenerator* eta *RequirementTracer*) *CAMEL*ekin nola integratu diren azaltzea da.

Integrazioa *Continious Integration* bidez egin da. Horretarako bi sistemak *CAMEL*ek komando bidez exekutatuko ditu (horregatik prestatuta dago parametroen erabilera). Jarraikako integrazioari esker exekutatu, ez da beharrezkoa erabiltzaile batek bi sistemak eskuz exekutatzea eta, beraz, automatikoki daukagu bi sistemen artean. Hurrengo irudian<sup>32</sup> agertzen da integrazio-mota horren eskema bat.

## Integration of DocGenerator & RequirementTracer on Continuous Integration



32 irudia: Continuous Integratiorekin exekuzioaren eskema.

Integrazioa nola implementatu ez da BETRADOK proiektuaren barruan sartzen; izan ere, proiektu honetan zehaztu egiten dira bi sistemak exekutatzeko parametroak zeintzuk diren eta nola erabili behar diren. *Continuous Integration* bidez *CAMEL*ek exekutatzearen implementazioa ez da proiektu honen irismenean sartzen, ezta helburuetan ere.

### Ondorio orokorra

Proiektuaren bukaeran bi sistemak, *DocGenerator* eta *RequirementTracer*, hasiera batetik definitutako betekizunak betetzen ditu. Hurrengo paragrafoetan definituko da bi sistema horien irismena, ikuspegi teknikitik ikusita.

Alde batetik, *DocGenerator* gai izango da softwareko arkitektura (*software architecture*) eta software diseinuko (*software design*) dokumentazio formala sortzen, aurrez definitutako bi txantiloiak eta beharrezkoa den informazioa erabilita. Software arkitekturako dokumentu bat ateratzen du exekuzio bakoitzeko eta, momentu honetan, berrogeita hamar dokumentu software diseinurako, bat *ATO*ko osagai bakoitzeko. Gainera, dagoeneko *continious integration* bidez exekutatzen da. Horrek esan nahi du *ATO*aren inplementazioan aldaketa bat dagoenean eta aldaketa hura baieztatu egiten denean, bi sistema horiek exekutatu egiten direla automatikoki. Horren alde ona zera da: momentu oro *ATO*aren dokumentazio eguneratuta lortzen. Horretaz aparte, sistema mantentzeko eta hobetzeko dokumentazioa ere du. Ondorioz, lortu egin dira aurrez zehaztutako helburuak.

Bestetik, *RequirementTracer* sistemak *CAMEL* eta *IBM Rational DOORS*en arteko elkarrekintza osoa egiteko gai da. Lehenik eta behin, dagokion traza jarraitzeko datuak bildu egiten dira eta automatikoki *DOORS* irekitzen du, zeinek datu horiek erabilita automatikoki eguneratzen du aurretik diseinatutako traza. Bestalde, *DocGenerator*rekin gertatzen den bezala, *continious integration*ni esker exekutatzen da. Horri esker, kode berria gehitutakoan oso denbora motzean jakin daiteke zein betekizun ez den ongi betetzen. Sistema hau borobiltzeko asmoz, sistema mantentzeko eta hobetzeko dokumentazioa dauka, diseinuaren inguruko hainbat argibideekin eta zenbait gomendioekin. Laburbilduz, lortu egin dira hasiera batean planteatutako helburuak.





## 12 Arrisku Analisia

Atal honetan proiektuan zehar identifikatutako arriskuei inguruko analisia egiten da. Proiektuaren fase guztietan identifikatutako arriskuak zerrendatzen dira eta horiek gertatzeko probabilitatearen inguruan azterketa egiten da. Gainera, arriskuak gertatzekotan, arazo horiei aurre egiteko estrategiak definitzen dira.

### Arriskuen Identifikazioa

Hurrengo taulan zerrendatu egiten dira proiektuaren hasieran eta proiektuaren gauzatzean identifikatu diren arriskuak.

Arrisku ID	Arriskua	Deskribapena
R01	<i>DocGenerator</i> : Beste bide egokiago bat aurkitzea.	<i>DocGeneratoren CAMEL -&gt; Rhapsody -&gt; Word</i> bidea (aurreikusitako bidea) ez izatea %100 bideragarria eta estrategia aldatzea.
R02	<i>DocGenerator</i> : <i>CAMEL -&gt; Rhapsody</i> transformazioan informazio galera.	<i>DocGeneratoren CAMEL</i> etik <i>Rhapsody</i> -rako bidean, derrigorrez informazioa galtzea.
R03	<i>DocGenerator</i> : Mantenerako denbora.	<i>DocGenerator</i> ren bi kode nagusi egongo dira, transformazio bakoitzeko bat. Beraz, bi kode mantendu beharko dira eta kostu handia izan dezake bi kode desberdin mantentzea gauza baterako.
R04	<i>DocGenerator</i> : exekuzio denbora altua.	<i>DocGenerator</i> exekutatze denbora oso altua izatea eta ordu bat edo gehiago behar izatea dokumentazioa sortzeko.
R05	<i>DocGenerator</i> : Word dokumentuak manipulatzeko APIen falta.	<i>DocGeneratoren</i> Word dokumentuak manipulatzeko APIa beharrezkoa da, dokumentazio hura sortzeko. API gutxi egotea, mantenerik ez izatea eta kontsistenteak ez izatea. <i>Open Source</i> motakoak ez izatea.
R06	<i>DocGenerator</i> : Diagramak sortzeko <i>PlantUML</i> APIa eskasa izatea	Gerta daiteke <i>PlantUML</i> ez izatea nahikoa jorratu eta sortu nahi diren diagrama guztiak sortzeko.
R07	<i>RequirementTracer</i> IBM Rational <i>DOORS</i> rekin menpekotasuna	<i>RequirementTracer</i> paketea <i>DXL</i> n programatu behar da <i>IBM Rational DOORS</i> en. Dependentsia hura oso altua izan daiteke.

Arrisku ID	Arriskua	Deskribapena
R08	<i>RequirementTracen</i> objektuak ondo ez eguneratzea eta exekuzio bakoitzean egindako link esternoak desagertzea	<i>RequirementTraceren</i> gerta daiteke Objektuak eguneratzerakoan kanpoko link esternoak (manualki sartzen diren horietakoak) "galtzea".
R09	<i>RequirementTracer</i> ez exekutatzea, Sare arazoak direla eta.	<i>RequirementTracer</i> ek beharrezkoa du barne sarean kokatu dagoen <i>IBM Rational DOOR</i> Seko instantzia bat sortzea; eta horretarako sareko makina espezifiko batera baimena eta atzigarritasuna izan behar du.
R10	<i>RequirementTracer</i> erako mantenu falta, esperientzia ezagatik.	<i>RequirementTraceren</i> <i>DXL</i> bidezko programazioa egiten da, non <i>Skip</i> motako estrukturak lantzen diren, adibidez. Esperientzia eta formazio pixka bat behar da <i>DXL</i> ko aldea mantentzeko.

9 taula: Identifikatutako arriskuen zerrenda.

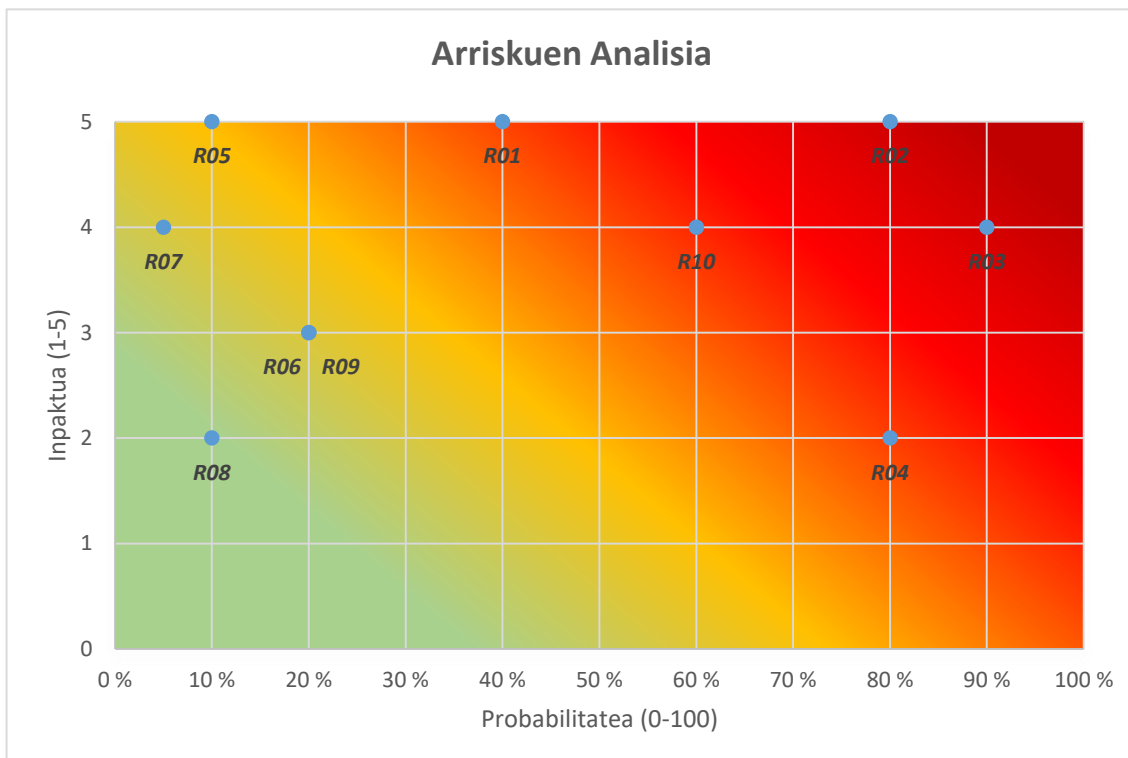
## Arriskuen Anlisi Kuantitatibo eta Kualitatiboa

Hurrengo taulan aurreko atalean zerrendatu egin diren arriskuak gertatzeko probabilitatea eta horiek proiektuaren garapen osoan zehar izango luketen inpaktua zehazten dira. Arrisku horiek zerrendatzeko arriskuaren identifikazio-zenbakia erabili da.

Arrisku ID	Probabilitatea (%)	Inpaktua (1-5)
R01	40	5
R02	80	5
R03	90	4
R04	80	2
R05	10	5
R06	20	3
R07	5	4
R08	10	2
R09	20	3
R10	60	4

10 taula: Arriskuen probabilitatea eta inpaktua adierazteko taula.

Hurrengo grafikoan aurrez erabilitako bi parametro horien lotura egiten da eta kolore bidez adierazi egiten da zein arrisku diren garrantzitsuak direnak kontrolatzeko eta arintzeko edo ezabatzeke.



1 grafikoa: Arriskuen probabilitatea eta inpaktuaren arteko lotura.

## Arriskuei Erantzuteko Plangintza

Lehenik eta behin, definitutako arrisku bakoitza arintzeko edo aurre egiteko estrategia azaltzen da. Kasu honetan, arriskuen zerrenda ordenatuta dago garrantziaren arabera: garrantzitsuenak direnetik garrantzi txikia duten arriskuak arte.

Arrisku ID	Arriskua Arintzeko Estrategia
R02	Beste estrategia orokorra bilatu edo arazoa konpontzeko estrategia espezifikoa bilatu.
R03	Beste estrategia orokorra bilatu. Derrigorrez.
R04	Exekuzioaren paraleloko prozesamendua aztertu.
R10	Proiektua bukatu baino lehen, software honen mantenuaz ( <i>RequirementTracer</i> ) arduratuko den pertsonari "Produktuaren dokumentazioa" dokumentua prestatzea; baina formazio pixka bat eman ere, inplementazio eta diseinuaren nondik norakoak azaltzeko asmoz.

Arrisku ID	Arriskua Arintzeko Estrategia
R01	Beste estrategia horren bideragarritasun-analisi bat egin. Erabaki zuzena den edo atzera buelta.
R05	<i>Word</i> manipulatzeko <i>API</i> a sortu (kostu oso altua) edota beste formatu bat erabili.
R06	<i>PlantUML</i> z aparte, beste <i>API</i> espezifikoa bat erabiltzea diagrama konkretu batzuentzako edota diagrama guztietarako.
R09	<i>Continuous Integration</i> ez arduratzen den makinak sarerako atzigarritasun eta baimena ziurtatzea beste modu bat erabiliz. Beste konponbidea: Makina horretan <i>IBM Rational DOORS</i> instalatzea eta instantzia lokala sortzea, hots, nahiz eta modulu orokorrean aldatu gauzak, exekutatzeko diren <i>DXL</i> funtzioak lokalean egotea.
R07	Beste erraminta sortu edo <i>Open Source</i> motako beste erremintak bilatu.
R08	Beste erraminta sortu edo <i>Open Source</i> motako beste erremintak bilatu.

11 taula: Arriskuak mitigatzeko estrategia garrantziaren arabera ordenatuta.

Horretaz aparte, beharrezkoa da aipatzea *OpenUP* metodologiarekin eta *PMBOK* gidarekin zehaztutako atala izan dela, eta horiei esker bi dokumentu garrantzitsu idatzi direla proiektuan zehar arriskuak kudeatu eta analisia egiteko garaian. Aipatutako bi dokumentu horiek hauek dira:

- [Arrisku zerrenda](#). 2019. Jon Legarda.
- [Arriskuen Kudeaketa](#). 2019. Jon Legarda.

## 13 Proiektuaren Antolaketa eta Kudeaketa

Atal honetan proiektuaren antolaketa eta kudeaketaren inguruan hitz egiten da. *BETRADOK* proiektua antolatu eta kudeatzeko ***Project Management Institutek gomendatutako PMBOK gida erabili da proiektuan zehar***. Hori erabiltzearen arrazoi nagusia proiektuan profesionaltasun punturik ahalik eta handien lortzea izan da.

### 13.1 Antolaketa

Proiektuaren antolaketaren inguruan ez da gauza handirik esan behar. Proiektuaren egilea izango da atera diren frente guztiei aurre egin dion langilea. **Jon Legarda izan da proiektuaren kudeaketa osoa eraman duena**. Kudeaketarako atalean ongi zehazte da kudeaketa-mota guztietan hartutako erabakiak.

Egia da proiektuan zehar hainbatek lagundu egin dutela eta zuzendari moduan beste langileak egon direla (enpresan eta unibertsitatean). Hala ere, Jon Legarda izan da antolaketa osoa eraman duena.

### 13.2 Kudeaketa

Proiektuaren memorian behin baino gehiagotan aipatu den bezala, proiektuaren kudeaketa ***PMBOK gida-liburuaren arabera egin da***. Hori dela eta, liburu horrek aipatzen dituen kudeaketa-motak deskribatzen dira atal honetan. Kasu batzuetan, kudeaketa-mota jakin bat memoriako beste atal zehatz batean aipatu da. Kasu horietan, soilik erreferentzia egingo da.

### Integrazioaren kudeaketa

Integrazioaren kudeaketarik esker, proiektuan zehar eman diren prozesu desberdinak identifikatu, definitu, konbinatu, bateratu eta koordinatzeko eman beharreko pausuak eta zehaztu beharreko prozesuak kudeatzen dira.

### Proiektu eratze aktaren garapena

Proiektuaren hasierako akta edo kontratua proiektuaren egileak, enpresak eta unibertsitateak sinatutako akordioa da, non zehaztu egiten diren ordu kopuru estimazioa eta proiektuaren hasierako helburuak. Erreferentziaz:

[\*Gradu Amaierako Lana CAF Signalling enpresan egiteko akordioa UPV-EHUrekin.\*](#) (2019ko otsailaren 4a). *CAF Signalling*, Euskal Herriko Unibertsitatea, Jon Legarda Gonzalez.

### Proiektuko kudeaketa planaren garapena

Proiektuaren plangintza eramateko *OpenUP* metodologiaren "*Project Plan*" erabili da, helburuak definitu eta irisgarritasuna definitzeko.

- [\*Project Plan\*](#). 2019ko otsaila. Jon Legarda Gonzalez.

### Proiektuaren garapena zuzendu eta kudeatu

Proiektua aurrera doan heinean zuzentzeko eta kudeatzeko ardura hiru aktore garrantzitsuenena izango da: Jon Legarda Gonzalez, *CAF Signalling* eta unibertsitateko tutorea. Proiektua epe motzetan zuzentzeko eta kudeatzeko (arriskuak gutxituz) *OpenUP* eta *SCRUM* metodologiak konbinaketa egin da eta hurrengo dokumentuan antolaketa hura ikus daiteke:

- [\*Iteration Plan\*](#). 2019. Jon Legarda Gonzalez.

### Irismen Kudeaketa

Proiektuaren irismena kudeatzeko planteatutako helburuen eta denboraren arteko erlazioa bilatu egin da. Hainbat pausu jarraitu dira hura definitzeko.

### Betekizunen Bilketa

Betekizunak bildu egin ziren proiektuaren zuzendaria, zuzendariordea eta egileak egingako bilera batean. Bilera hartan definitu egin ziren sistemak bete beharreko betekizunak.

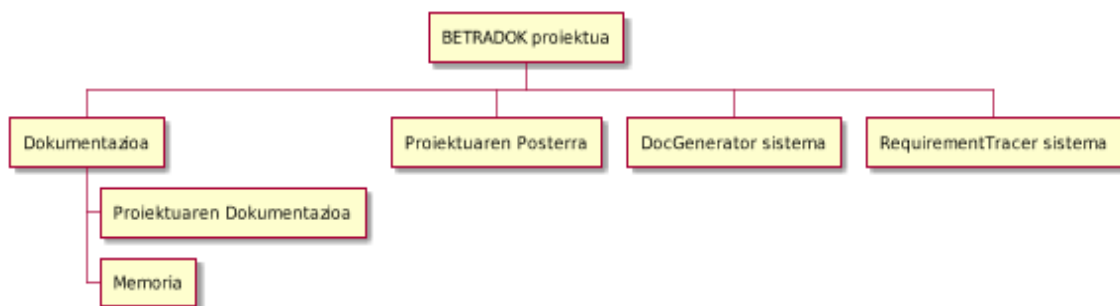
### Bisioa definitu (*Project Vision*)

Proiektuaren bisioa izan zen lehenengo definitu egin zen gauzetako bat. Bisioa ondo ulertzeko, bisio dokumentu estandarizatua bete egin zen:

- [\*Bisioa\*](#). 2019. Jon Legarda.

### Proiektuaren LDE diagrama

Hurrengo LDE diagramak zehazten ditu zein emangarri entregatu behar diren proiektuan.



2 grafikoa: Proiektuaren LDE diagrama, beharrezko entregak erakusteko.

## Epeen kudeaketa

Epeen kudeaketa azaltzeko hobeto denboraren planifikazioa<sup>87</sup> atalera jotzea.

## Kostuen kudeaketa

Kasu honetan, kostu ekonomikoetaz ari gara eta horiek nola kudeatu. Denbora-kostuak ez dira atal honetan analizatzen.

### Kostuak Kudeaketa Planifikatzea

Hasiera batetik proiektuan egon zitezkeen kostu ekonomikoak aurreikusiak zeudela onartu egin zen, hau da, dagoeneko proiektu osorako ordainketa guztiak (software, hardware, osagai, lizentziak...) eginak zeudela. Hori dela eta, pentsatu zen kostuak ez zutela plangintza eta kontrol berezirik behar. Ahal den heinean, erabili beharreko software konponente edo programak (hurrengo ataletan aipatutakoez gain) libreak izan behar ziren (*Open Source* edo software libre). Bukaeran ikusi da aurreikusitako hura egia zela, eta ez da behar izan kostu ekonomikoen kudeaketa berezirik egitea.

### Kostuen estimazioa

Proiektuan ingeniari informatiko baten beharra eta horrek dakarren kostu ekonomikoaz aparte, proiektua gauzatzeko bi software berezi eta libreak ez direnak behar izango direla aurreikusi zen:

- *IBM Rational Rhapsody*. Horren kostua: 800€ (BEZa barne).
- *IBM Rational DOORS*. Horren kostua: 800€ (BEZa barne).

## Aurrekontua espezifikatu

Aurrekontua<sup>96</sup> beste atal jakin batean zehazten da.

## Kostuen Kontrola

Kostu gehiagoren kontrola ez da eraman behar izan, hasiera batetik egindako aurreikusketak zuzenak izan zirelako.

## Kalitatearen kudeaketa

Kalitatearen kudeaketa<sup>26</sup> beste atal batean egiten da.

## Giza-baliabideen kudeaketa

Giza baliabideak kudeatzea ere oso garrantzitsua izan da proiektua aurrera eramateko garaian; izan ere, proiektua taldean gauzatu da, nahiz eta proiektu pertsonala izan.

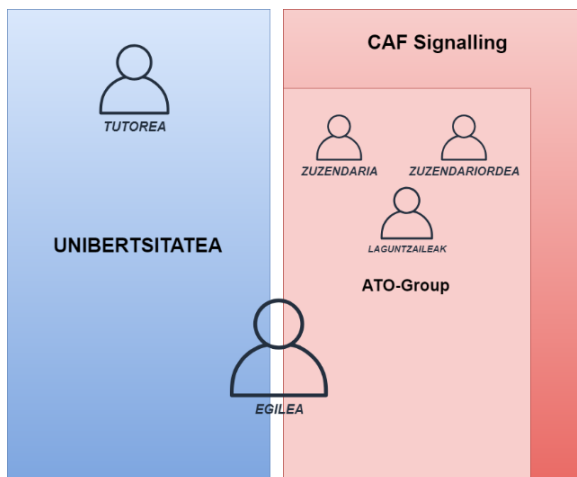
## Giza baliabideen kudeaketaren plangintza

Proiektuaren ezaugarriak direla eta, hauek dira proiektuan parte hartu duten giza baliabideen rolak:

- Proiektuaren zuzendaria: proiektua planteatu eta zuzentzen duen pertsona da. Egilea gidatu behar du proiektua arrakastatsua izan dadin, helburuak definituz. Rol hau enpresako kide batek bete du.
- Proiektuaren zuzendariordea: zuzendariarekin batera egilea proiektuan zehar gidatu behar du. Kasu honetan, egilea gidatu behar du lan kudeaketa egokia izan dezan. Rol hau enpresako kide batek bete du.
- Proiektuaren laguntzaileak: proiektuan zehar laguntzen duten pertsonak dira. Zuzenean edo zeharka proiektuan interesa agertzen eta laguntzaile rola hartzen duten pertsonak. Rol hau enpresako kide batek bete du.
- Unibertsitateko tutorea: proiektu hau Gradu Amaierako Lana izanik, beharrezkoa da pertsona batek unibertsitate esparruan egilea gida dezala. Horren helburua izango da unibertsitatearen eskakizunak betetzen direla ziurtatzea. Rol hau unibertsitateko (Euskal Herriko Unibertsitatea) kide batek betetzen du.
- Proiektuaren egilea: proiektua gauzatu behar duen pertsona da. Rol hau unibertsitateko ikasle batek betetzen du.



Proiektuaren berezitasunak kontuan hartuta, giza baliabideen kudeaketan bi azpimultzo bereizten dira: unibertsitatea eta enpresa (*CAF Signalling*). Proiektu hau *CAF Signalling* enpresan kokatzen da, *ATO-Group* deituriko lan-taldean hain zuzen ere.



33 irudia: Proiektuko giza baliabideen eskema orokorra.

## Taldekieeen Eskuratzea

Hauek dira aurreko azpiatalean definitutako rolak bete dituzten pertsonak, haien ardurekin batera:

- Proiektuaren zuzendaria: Mikel Larramendi Arburua.
- Proiektuaren zuzendariordea: Jon Goya Mendiluce.
- Proiektuaren laguntzaileak: Iñaki Berriotxo Gabiria.
- Unibertsitateko tutorea: Juan Manuel Pikatza Atxa.
- Proiektuaren egilea: Jon Legarda Gonzalez.

## Proiektuaren Taldea Zuzentzea eta Garapena

Proiektuaren lan-taldea zuzentzea, zuzendariei egokitu zaie (tutorea kontuan hartu gabe). Enpresan ere bi zuzendariak (zuzendaria eta zuzendariordea) dira *ATO-Group* taldea zuzentzen eta kudeatzen dituzten pertsonak.

Horretarako enpresan *ATO-Group*ek metodologia jakin bat definituta du lan egiteko: *Scrum* metodologia. Hori dela eta, metodologia hau ere hainbat lekuetan aipatu egin da.

Gainera, proiektuaren egilea arduratu behar izan da proiektuaren alde tekniko eta alde akademiko erlazionatzeaz, bai enpresako kideekin komunikatuz, bai unibertsitateko kidearekin komunikatuz.

## Komunikazioen kudeaketa

Komunikazioen kudeaketari begira, atal honetan definitzen dira zein pausu jarraitu diren barneko eta kanpoko interesatu eta kideen artean komunikatzeko. Ahalik eta komunikazio hoberena edukitzeko lehenengo interesatuak identifikatu dira; ondoren, komunikazioak planifikatu dira; eta bukatzeko, informazioa banatzeko pausuak zehaztu dira.

## Interesatuen Identifikazioa

Proiektu honen interesatuak hauek izan dira:

- *CAF Signalling*: enpresa hau da proiektuan interesatu dena eta, aldi berean, bezero eta proiektu “egiletzat” har daitekeena.
- *ATO-Groupeko* langileak: lan-talde honetako langileek erabiliko dute “*DocGenerator & RequirementTracer*” proiektuko soluzioa; beraz, haiek ere interesatuta daude, beren lana erraztu dezakeen erraminta sortu delako proiektu honetan.
- Proiektuaren egilea: pertsona honi interesatzen zaio proiektua GrAL gisa aurkeztu eta defendatzeko.

## Komunikazioen Plangintza

BETRADOK proiektu honetan, JLEk bezeroaren baitan lan egin du azken finean. Hori dela eta, ez da “bezero-saltzaile” erlazio bateko komunikazio kudeaketa hain formala egin behar izan.

Proiektuan zehar aipatutako berezitasun jakin hura komunikazio-mota “informala” izan da. *CAF Signalling*, aldi berean bezero eta JLEren lan-taldea izanik, beste kideekin komunikazioan ez da beharrezkoa izan informe edo beste komunikazio berezirik. Gainera, unibertsitateko tutorearekin komunikazioa ere “informala” izan da.

## Informazio Banaketa

Komunikazioa gauzatzeko hauek izan dira erabilitako kanalak:

- Korreo bidezko komunikazio informala.
  - *CAF Signallingen IBM Notes* aplikazioa.
  - Unibertsitateko tutorearekin *Gmail* edo *EHU-Korreo*a baliatuta.
- Ahozko komunikazio informala: bilera presentzialak eta beharrezkoak diren gaiak jorratu, komunikazioa estua izateko.

Komunikazioaren maiztasunaren inguruan, bi zati bereizten dira: batetik, enpresako taldekideekin maiztasuna (aldi berean, bezerotzat jo daitekeena) eta, bestetik, unibertsitateko tutorearekin maiztasuna. Taldekideekin komunikazio-maiztasuna oso altua izan da, egunerokoa. Unibertsitateko tutorearekin, ordea, noizean behin egin dira bilerak. Horretaz aparte, korreoz maiztasun handiko komunikazioa burutu da tutorearekin.

## Arriskuen kudeaketa

Arriskuen kudeaketa ez da hemen aipatuko; izan ere, [arriskuen analisia](#) egin den atalean aipatzen da arriskuak nola kudeatu diren.

## Erosketen kudeaketa

"Erosketen kudeaketa" dokumentu honetan, proiektuan zehar egon daitezkeen erosketak kudeatzeko estrategia definitzen da.

## Erosketen Plangintza

Proiektu hau (beste atal askotan aipatu bezala) *CAF Signalling* enpresan gauzatu da, beraz, erosketa horiek planifikatzeko *CAF Signalling* kontuan eduki behar da. Hala eta guztiz ere, proiektu osoan zehar ez da behar izan erosketa berezirik egin.

Erabili den software edo baliabide informatiko gehienen helburua *Open Source* motakoa izatea izan da. Memoria eta dokumentazioan erabili diren baliabide guztiak lizentzia libreak dauzkate. Ondorioz, ez da egon kostu ekonomikorik.

## Erosketen burutzea, administrazioa eta itxiera

Aurrez esandakoa kontuan hartuta, erosketarik ez da egin, beraz, ez da horien burutze, administrazio edo itxiera faserik kudeatu.

## Interesatuen kudeaketa

BETRADOKen interesa duten *stakeholderrak* kudeatzeko ere plangintza zehatza eraman da proiektuan zehar.

## Interesatuaren Identifikazioa

Proiektu honetan zuzenean edo zeharka arrazoi desberdinengatik interesa eduki duten interesatuak honako hauek dira (proiektuaren egilea zerrenda honetatik kanpo geratu da):

- *CAF Signalling*: interes zuzena daukan aktorea da. *CAF Signalling*en barruan bere langileak sartzen dira.
- *CAF Signalling*-en bezeroak. Zeharkako interesatuak dira; izan ere, proiektuak *CAF Signalling* emanaraziko dizkion aurrera pausuak beren bezeroei soluzio hobeak eskaintzen lagunduko dio enpresari.
- Euskal Herriko Unibertsitateko Informatika Fakultatea: proiektua GrAL gisa aurkezteagatik, zuzeneko interesatu gisa jo daiteke.

## Interesatuaren Kudeaketako Plangintza

Interesatuaren kudeaketako plangintza honetan soilik *CAF Signalling* eta EHUko Informatikako Fakultatea kontuan hartuko dira.

### *CAF Signalling*

Enpresako kideekin egunero 5 minututako bilera egin da “*sprint daily*” bileretan. Horretaz aparte, beharrezkoa denean bilerak egingo dira proiektuaren jarraipena egiteko.

### EHUko Informatika Fakultatea

EHUko Informatika Fakultatearekin soilik proiektuaren defentsa izango da kontaktu zuzena egongo den momentua.

## Interesatuaren Parte Hartzearen Kudeaketa

Interesatuarekin, batez ere *CAF Signalling*ekin, parte-hartzea kontrolatzeko bilerak egin dira eta "derrigorrezko" asistenteak egon dira bilera horietara deituta.

## Interesatuaren Parte Hartzearen Kontrola

Interesatuak ongi erantzun dute JLEren bilera deialdietara eta ez da egon arazorik.

## 14 Denbora Planifikazioa

Atal honen helburua da proiektuaren denbora aurre-planifikazioa azaltzea, mugarri garrantzitsuak zehaztea eta proiektuaren bukaeran egondako desbideraketak eta bukaerako planifikazioa ahalik eta zehatzen azaltzea. Denboraren jarraipena egiteko *RedMine* tresna erabili da. Tresna horretan irekitzen joan dira atazak eta azpiatazak eta egunero eguneratzen joan da lanean ibilitako denbora.

Proiektuaren hasieran erabaki garrantzitsu batzuk hartu ziren:

1. Proiektua hasi baino lehen ez dira azpiataza gehienak sortuko.
2. Azpiatazak iterazioka edo beharrezko momentuan sortuko dira; izan ere, oso zaila da aurreikustea ezjakintasun handia duen proiektu batean zein atazetan egingo den lan hemendik hilabete batzuetara. Enpresan ere horrela lan egiten dute eta aproposena iruditu zitzaion egileari, aipatutako erabakia hartuz.
3. Ordu kantitatea kontuan hartuko da, gutxi gora-behera eta epeak zehaztuko dira, baina ataza bakoitzari estimatutako orduak ez dira %100ean zehaztuko.

Hurrengo azpiataletan zehaztuko da proiektuaren mugarri garrantzitsuenak; geroxeago, plangintzan kontuan hartutako LDE diagrama; ondoren, proiektuan zehar planifikatu diren lan-atazak eta azpi-atazak (bai memoria bai proiektu teknikoan) eta bukatzeko *Gantt* diagrama bat erakusten duena zein ibilbide zehazki eman den.

### Mugarri garrantzitsuak

Proiektuaren planifikazioarekin hasteko gauzarik garrantzitsuena da bezeroarekin adostutako mugarri garrantzitsuak biltzea eta kontuan hartzea. Proiektu honen bezero gisa har ditzakegu *CAF Signalling* eta EHUko Informatika Fakultatea, proiektua aurkeztuko den gunea.

Hurrengo taulak laburbiltzen ditu mugarri garrantzitsuak eta haien deskribapenak.

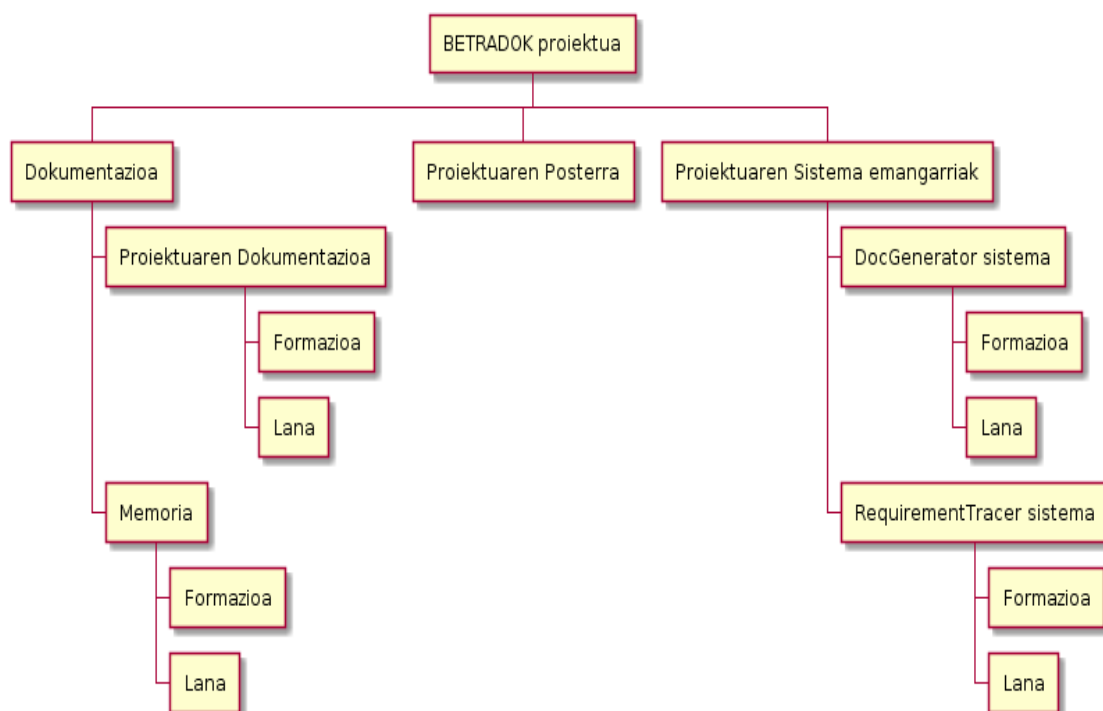
Mugarri	Deskribapena
2019.02.04	Proiektuaren hasiera.
2019.07.05	Proiektuaren bukaera <i>CAF Signallingen</i>
2019.07.24	Proiektuaren matrikulazioa unibertsitatean
2019.09.08	Proiektuaren entrega (Memoria eta Posterra) unibertsitatean
2019.09.12-15	Proiektuaren defentsa (aurkezpena) unibertsitatean

12 taula: Mugarri garrantzitsuen taula.

Aurreko taularen epeak proiektu hasieran markatutakoak izan ziren. Epe guztiak bete egin dira proiektuan zehar, bat izan ezik. **CAF Signallingekin akordio batera iritsi zen hango data atzeratzeko irailera arte** (badaezpada) eta proiektua ahalik eta hobekien bukatzeko.

## Lan-Ataza Nagusien LDE diagrama

Hurrengo irudiak adierazten du nola antolatu diren atazak emangarrien arabera eta jakinda bakoitzak derrigorrez formazio edo heziketarako denbora bat behar izan dela derrigorrez.



34 irudia: Emangarrien arabera LDE diagrama.

Kontuan hartu behar da ataza-nagusi bakoitzak derrigorrez behar izan duela formaziorako denbora bat. Azken finean, *OpenUPri* buruz irakurtzea eta ulertzea, *CCII* estandarra, teknikoki lortu beharreko ezagutza guztiak...denbora behar dute formakuntzarako. Hori dela eta, formakuntza eta benetako lan-teknikoa dena banatu egin da.

Proiektuaren hasieran aurreikusi zen **proiektu osoa 700 ordukoa** izango zela. Hori aurreko irudian zehaztutako ataza-nagusiak egiteko, bai formakuntzan, bai benetako lanean. Horretaz aparte, estimatu egin zen, gutxi gora-behera, ataza-nagusi horiei eskaini behar izango ziren orduak. Hurrengo taulan ongi adierazten dira eta banatzen dira ordu horiek.

Formazio eta Lan orduen Estimazioa		
Lana	<i>DocGenerator</i>	200
	<i>RequirementTracer</i>	150
	Memoria	50
	Dokumentazioa	50
	<b>Guztira</b>	<b>450</b>
Formazioa	<i>DocGenerator</i>	100
	<i>RequirementTracer</i>	100
	Memoria	10
	Dokumentazioa	40
	<b>Guztira</b>	<b>250</b>
<b>Guztira</b>	<b>700</b>	

13 taula: Formazio eta lan-orduen banaketa (I).

Hurrengo irudiak datu berdinak erakusten ditu, baina perspektiba desberdina batekin, non hobeto ikus daitekeen emangarri bakoitzarentzat estimatutako orduak.

Formakuntza eta Lan-orduen Estimazioa			
	Lana	Formazioa	Guztira
<i>DocGenerator</i>	200	100	300
<i>RequirementTracer</i>	150	100	250
Memoria	50	10	60
Dokumentazioa	50	40	90
<b>Guztira</b>	<b>450</b>	<b>250</b>	<b>700</b>

14 taula: Formazio eta lan-orduen banaketa (II).

Erakutsitako datuen artean garrantzitsuena hau da: **450 ordu**. 450 dira benetan proiektuan lan egindako orduak, memoria, dokumentazioa eta bi soluzioen egikaritzea bateratzen duen ordu-kantitatea. Beste 250 orduak izan dira formakuntza-orduak eta ezin dira proiektuaren barne hartu. Hala eta guztiz ere, orduen jarraipena egiterakoan, eta beraz, batera egin dira kontrol desberdina egin gabe.

Hurrengo taulak adierazten du proiektuan zehar sortutako lan-ataza guztiak eta ataza horietan emandako orduak. Lan-ataza horiek izan dira aipatutako *RedMine* tresnan sortutakoak eta aipatu bezala ez da formakuntza eta benetako lanaren kontrola egin. Gainera, ingelesez dago taula; izan ere, enpresan ingeleza erabiltzen da atazetarako.

Lan-Atazak eta Orduak		
Atazaren IDa	Ataza	Lan egindako Orduak
6637	[PARENT] Setting Up of the environment for the whole project	24
6629	[PARENT] Project (GrAL/TFG) documentation & Tasks Managing	151,75
6630	> Structure of documentation web-page	12,5
6631	> <i>OpenUP</i> methology documentation	48,5
6633	> Research part of documentation	3,5
6632	> Budget part of the documentation	2
6634	> Attachment part of documentation	33,25
6635	> Memory part of the documentation	40
7253	> Correct and improve attachment, budget, research documents.	12
7247	[PARENT] Project Poster	5
6748	[PARENT] Automatic Generation of Documentation	339
6636	> Transformation <i>CAMEL</i> -> <i>Rhapsody</i>	140
6588	> Find out <i>APIs</i> to manipulate Word/RTF/Odt documents in Java	6,5
6589	> Pilot Test to work on manipulating Word documents based on templates	48
6746	> Create a table to relate Documentation needs and <i>CAMEL</i> elements.	24,5
6924	> <i>SwArchitecture's</i> documentation: Sections 4.1., 4.1.3. (BASIC)	14
6925	> <i>SwArchitecture's</i> documentation: Sections 4.1.6., 4.1.7. (BASIC)	14,5
6926	> <i>SwArchitecture's</i> documentation: Section 4.2. (BASIC)	4,5
7054	> <i>SwDesign</i> documentation: Obtain one document per Component	8
7085	> Iterate correctly <i>SwDesign</i> model and obtain the correct Components	1,5
7086	> Prepare documents templates with specific info.	3,5
7087	> <i>SwDesign</i> documentation: Section 3.3. (BASIC)	2
7089	> <i>SwDesign</i> documentation: Section 3.3.1, 3.3.2., 3.3.3. (BASIC)	2,5
7090	> <i>SwDesign</i> documentation: Section 3.3.4., 3.3.5. (BASIC)	1,5
7099	> <i>SwDesign</i> documentation: Section 4 (BASIC)	9,5
7122	> Restructure code for setting Paths by arguments	1,5
7128	> Implement personalization of the Documents depending models	4,5
7157	> Implement <i>DocGenerator</i> for <i>CAMEL 2</i>	2,5



Lan-Atazak eta Orduak		
Atazaren IDa	Ataza	Lan egindako Orduak
7158	> Implement changes on DocGenerator for <i>SwDesign</i> documentation.	16
7159	> Implement changes on <i>DocGenerator</i> for <i>SwArchitecture</i>	28,5
7160	> Refine image addition for Documents.	5,5
6747	[PARENT] Requirements Traceability -> <i>DOORS-CAMEL</i>	237
5833	> <i>CAMEL-&gt;DOORS</i> by CSV - Pilot Test	92,5
6749	> Creation of CSV files and automatic export to <i>DOORS</i>	0
6749	> Subsystem's CSV files creation. [CHANGED NAME]	5,5
6829	> Subsystem formal modules creation (traces model n1)	14,5
6834	> Refactorization & import of CSVs data to <i>DXL</i> structures	8
6835	> Subsystem link modules creation (traces model n1)	4
6921	> Bug related to null Object when getting a module element	8
6922	> Subsystem's trace-model n1: relate Output with Outputs	4
6923	> Changes on structure for the trace-model n1	15,5
6952	> Bug with null Modules and creation of Objects	23
6969	> Decrease on execution time	9,5
6998	> Comment the code	4
7000	> Definitive changes on trace-model structure	22,5
7001	> Retrieve last model from <i>GitLab</i> via <i>BATCH</i> file	8
7022	> Integrate <i>DXL</i> program in <i>DOORS</i> Folder	7,25
7031	> Ideas to refactor <i>DXL</i> code (traceability. <i>DXL</i> )	1,5
7069	> Adapt <i>RequirementTracer</i> form Camel 3 to Camel 2	2
7115	> Prepare <i>RequirementTracer</i> for CI	5,5
7143	> Upload version of <i>RequirementTracer</i> from <i>CAMEL 2</i> to <i>CAMEL 3</i>	1,75

15 taula: Lan-Ataza eta orduen taula.

**Oharra!** Posterra egitearen ideia EHUko Informatika Fakultateak ateratako deialdi berria da. Proiektua egin aurretik, ataza hori ez zegoen. planifikatuta (ataza zenbakia 7247). Gainera, ez da kontsideratuko proiektuaren parte 6637 zenbakidun ataza.

Ondorioz, guztira **732.75** ordu lan egin dira. Beraz, desbiderapena ez da hain handia izan.

Kontuan hartu behar da ez direla ondo azaldu ataza bakoitzeko xehetasunak; izan ere, dokumentu honen helburua laburpen bat egitea da eta ez da egokia datu guzti-guztiak azaltzea.

Hori dela eta badago beste dokumentu espezifiko bat ataza bakoitzeko informazio zehatzagoa ematen duena:

- [Lan-atazen zerrenda](#). 2019. Jon Legarda.

## Lan-Atazen banaketa Iterazioetan

Lan-ataza guzti horiek pixkanaka-pixkanaka garatzen joan da egilea proiektuan zehar. Hurrengo taulan adierazten dira proiektuko hasieratik bukaeranoko iterazio guztiak eta iterazio horien epeak.

Iterazio Kodea	Iterazioa	Epea
IT1	Iterazioa 1	otsailaren 4tik otsailaren 18ra
IT2	Iterazioa 2	otsailaren 19tik martxoaren 4ra
IT3	Iterazioa 3	martxoaren 5etik martxoaren 25era
IT4	Iterazioa 4	martxoaren 26tik apirilaren 8ra
IT5	Iterazioa 5	apirilaren 9tik apirilaren 24ra
IT6	Iterazioa 6	apirilaren 25etik maiatzaren 6ra
IT7	Iterazioa 7	maiatzaren 7tik maiatzaren 20ra
IT8	Iterazioa 8	maiatzaren 21etik ekainaren 3ra
IT9	Iterazioa 9	ekainaren 4tik ekainaren 19ra
IT10	Iterazioa 10	ekainaren 20tik uztailaren 1era
IT11	Iterazioa 11	uztailaren 2tik uztailaren 15era
IT12	Iterazioa 12	uztailaren 16etik abuztuaren 1era

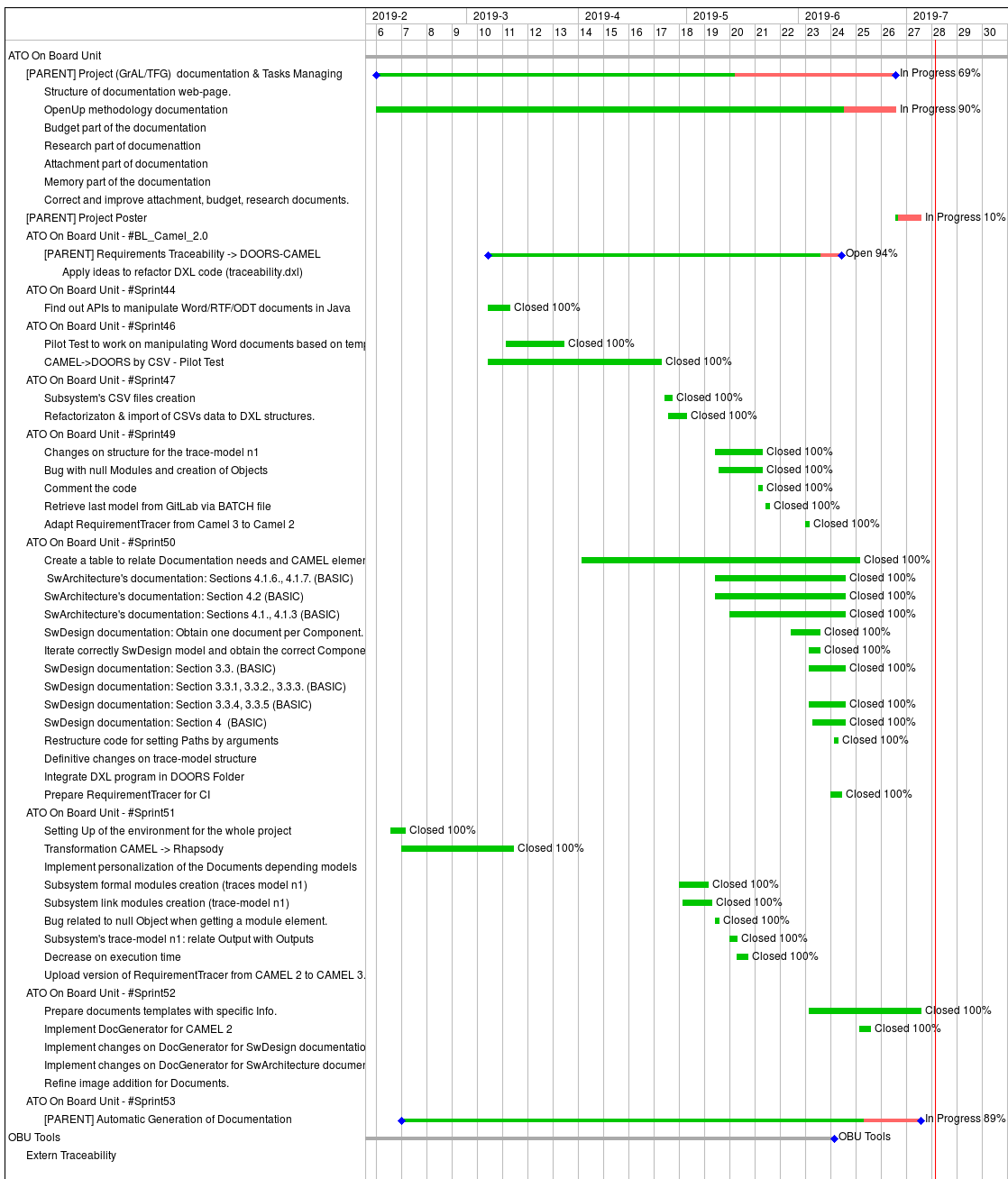
*16 taula: Iterazio banaketaren taula.*

Horretaz aparte, interesgarria litzateke aipatzea nola banatu egin diren ataza guztiak iterazio desberdinetan zehar. BETRADOK proiektuaren memoriaren parte den atal honen helburua ez da %100ean zehaztea iterazio bakoitzean zeretan lan egin den azaltzea, garrantzitsua ez den informazio zatia delako. Hori kontuan hartuta, proiektuan zehar proiektu plana eta iterazio planak egin dira. Bi dokumentu garrantzitsu horien erreferentzia honako hau da:

- [Proiektu plana](#). 2019. Jon Legarda.
- [Iterazio plana](#). 2019. Jon Legarda.

## Proiektuko bukaerako Gantt diagrama

Hurrengo grafikoak<sup>3</sup> adierazten du aipatutako ataza horietan lan egin diren egunak.



3 grafikoa: Atazen Gantt diagrama.

**Oharra:** diagrama hura eta gero memoriaren idazketako zati handia ere egon da, beraz, ezin da hartu proiektuko %100ean bukaerako *Gantt* diagrama bezala.

## Desbiderapenak

Proiektuan zehar hainbat desbiderapen suertatu egin dira. Hasiera batean tutorearekin adostu bezala, proiektuaren defentsa eta entrega irailean izan behar zen eta horrela izan da. Beraz, mugarri garrantzitsuenak ez du jasan inolako aldaketarik. Oso positibotzat jotzen den ezaugarria da.

Horretaz gain, proiektuaren garapen osoan zehar desbideraketa batzuk egon dira. Hurrengo zerrendan adierazita daude desbiderapen horiek:

- Estrategia aldaketa *DocGeneratoren*. Hasierako estrategia izan zen lehenik *Rhapsodyko* proiektu bat automatikoki sortu modeloko elementu guztiekin eta gero *Rhapsodyko* proiektutik dokumentazioa sortu. Horrek bi pausu behar zituen eta, beraz, bi inplementazio (edo bi kode). Prozesu hura arintzeko dena pausu batean egitea lortu egin da.
- CAF Signallingen hitzarmen luzaketa. Egileak denbora gehiago behar zuenez inplementaziorako hitzarmena luzatu egin zen. Hala ere, horrek bukaeran ez du izan desbiderapen nabarmenik; izan ere, aurreikusita zegoen denbora jakin bat horretarako.
- Momentu jakinetan lehentasunezko lanak. Momentu jakin batzuetan enpresako lankideek eskatuta, ataza konkretu bati egin dio kasu handiagoa egileak. Horrek ez du desbiderapenik ekarri.

## 15 Aurrekontua

Dokumentu honen helburua proiektuaren kostu ekonomikoa bezeroarentzat deskribatzea eta xehatzea da. Kostu ekonomiko hura, proiektuaren betekizunak argi geratu direnean eta tamaina argi geratu denean kalkulatu da.

Nahiz eta aurrekontu bat guztiz objektiboa ezin den izan, eta beraz, subjektibotasunaren menpe dagoen, lau irizpide erabili dira aurrekontua kalkulatzeko eta ahalik eta objektiboen izan dadin.

### Orokortasunak

Aurrekontua sortzerako garaian, irizpide jakin batzuk hartu dira kontuan, ahalik eta profesionalen eta errealen den aurrekontua sortzeko asmoz. Irizpide horiek *ALI (Asociación de Titulados Universitarios Oficiales en Informática)* elkarteak banatutako irizpide bat da eta hauek dira:

1. Giza Baliabideak: barne zein kanpo-kostuak, ordu kantitatearekin batera.
2. Erabilitako erremintak edo tresnak.
3. Egindako *testing* teknikoen kostua.
4. Auditoretza baten ziurtagiriaren kostua.

Proiektu honen kasuan, 3 eta 4 puntuetan ez da ezer berezirik jaso; izan ere, ez da modu berezian probatu (proba orokorrak eta proiektuaren barne egin direnetaz aparte) eta auditoretza baten ziurtagiririk ez da eskatu.

Horretaz aparte, aurrekontuan 450 ordutako lana bildu egin da, aurreikusi delako beste ordu guztiak formazio edo heziketa-orduak direla eta ez dira sartu aurrekontuan. Gainera, ordu bakoitzeko ingeniariaren barne-kostua 41,32€-ko parametroa kalkulatu da, Euskal administrazioak horrelako lanetan ordaintzen duen kantitatearen pare.

Hurrengo orrialdean aurrekontua<sup>17</sup> biltzen duen taula dago.

## Aurrekontu-Taula

Hurrengo taula honetan proiektuari dagokion aurrekontua biltzen du:

PARTIDA		PARAMETROAK				Totala (BEZik gabe)	Totala (BEZa barne)
Giza Baliabideak		Lana (ordu kopurua)	Barne Kostuak (BEZik gabe)	Lana (ordu kopurua)	Kanpo Kostuak (BEZik gabe)		
		450	41.32 €	0	0	18.594	22.500
Erramintak		Lizentzia (€)		Lizentzia Mantenua (urtero)			
1	<i>Apache POI</i>	0		0		0	0
2	<i>Eclipse Modelling Tool 2018-12</i>	0		0		0	0
3	<i>Git</i>	0		0		0	0
4	<i>GitLab</i>	0		0		0	0
5	<i>GraphViz</i>	0		0		0	0
6	<i>IBM Rational DOORS</i>	800		9600		1600	1920
7	<i>IBM Rational Rhapsody</i>	800		9600		1600	1920
8	<i>Java 8</i>	0		0		0	0
9	<i>PlantUML</i>	0		0		0	0
10	<i>RedMine</i>	0		0		0	0
11	<i>Sublime Text 3</i>	0		0		0	0
12	<i>SVN</i>	0		0		0	0
13	<i>Tortoise</i>	0		0		0	0
14	<i>Word (Microsoft Office)</i>	0		0		0	0
Erramintak - Totala						3200	3840
Testing Teknikoak		Lana (ordu kopurua)	Barne-kostuak	Besteak			
		0	0	0		0	0
Ziurtaginerako Auditoretza		0				0	0
<b>TOTALA</b>						<b>21.794 €</b>	<b>26.340€</b>

17 taula: Aurrekontuaren taula.

Ondorioz, proiektuaren prezioa **26.340€** dira (BEZa sartuta dagoelarik).

## 16 Oinarrizko Dokumentuen Ordena

Dokumentazio luze honek ekar dezake une jakin batean inkongruentziak egotea. Proiektuaren garapena denbora luzean eman da eta gerta daiteke dokumenturen batean baieztapenen bat egitea eta beste dokumentu batean baieztapen hura kontrajartzea; dokumentu jakin batean emandako datu bat beste batean desberdina izatea etab.

Hori dela eta, kontuan hartu behar da egileak idatzitako **memoria osoa izango dela lehenik kontuan hartu beharreko informazioa** aipatutako inkongruentzia edo koherentzia faltak egotekotan.

Memoria osoa dokumentu askoren bilketa da. Hura izan da azkenekoz idatzitako dokumentua, beraz, irakurleak horri egin beharko dio kasu zalantzarik izanez gero.

Hala eta guztiz ere, proiektuaren egilea edozein momentutan prest egongo da edozein zalantza argitzeko proiektuaren edo dokumentazioaren inguruan. Berarekin kontaktuan jartzeko hurrengo bi korreo elektronikoak erabiltzen ditu:

- [jonlegarda002@gmail.com](mailto:jonlegarda002@gmail.com)
- [jlegarda002@ikasle.ehu.eus](mailto:jlegarda002@ikasle.ehu.eus).

## ERANSKINAK

Eranskinak ez dira dokumentu honetan idatzi; izan ere, sorta handiko dokumentua aterako litzateke eta ez du merezi, memorian dagoen edukiaren xehetasun teknikoak biltzeko erabili dira proiektuaren garapenean zehar eta pertsonal teknikoei zuzenduta daude.

Hori dela eta, erabaki da memoriaren eranskinak, sistemaren espezifikazioa, aurrekontua eta ikerlanak biltzen dituzten dokumentuak esteka bidez atzigarri edukitzea BETRADOKen webgunean (aurreko ataletan aipatu den bezala):

<http://betradok.000webhostapp.com/>

### I. Memoriaren Eranskinak

#### A1: Sarrerako Dokumentazioa

- [Eranskinak 1: Sarrerako Dokumentazioa](#). 2019. Jon Legarda.

#### A2: Anlisi eta Diseinua

- [Eranskinak 2: Anlisi eta Diseinua](#). 2019. Jon Legarda.
  - [Arkitektura Koadernoa – I](#). 2019. Jon Legarda.
  - [Arkitektura Koadernoa – II](#). 2019. Jon Legarda.

#### A3: Tamaina eta Esfortzu Estimazioa

- [Eranskinak 3: Tamaina eta Esfortzu Estimazioa](#). 2019. Jon Legarda.

#### A4: Kudeaketa Plana

- [Integrazioaren Kudeaketa](#). 2019. Jon Legarda.
- [Irismenaren Kudeaketa](#). 2019. Jon Legarda.



- [Epeen Kudeaketa](#). 2019. Jon Legarda.
- [Produktuaren Kostuen Kudeaketa](#). 2019. Jon Legarda.
- [Kalitate Kudeaketa](#). 2019. Jon Legarda.
- [Giza Baliabideen Kudeaketa](#). 2019. Jon Legarda.
- [Komunikazioen Kudeaketa](#). 2019. Jon Legarda.
- [Arriskuen Kudeaketa](#). 2019. Jon Legarda.
- [Erosketen Kudeaketa](#). 2019. Jon Legarda.
- [Interesatuen Kudeaketa](#). 2019. Jon Legarda.

## A5: Segurtasun Plana

- [Eranskinak 5: Segurtasun Plana](#). 2019. Jon Legarda.

## A6: Beste Eranskinak

- [Hedapena](#). 2019. Jon Legarda.
  - [Produktuaren Dokumentazioa](#). 2019. Jon Legarda.
  - [Erabiltzaile Dokumentazioa I](#). 2019. Jon Legarda.
  - [Erabiltzaile Dokumentazioa II](#). 2019. Jon Legarda.
  - [Backout Plana](#). 2019. Jon Legarda.
  - [Hedapen Plana](#). 2019. Jon Legarda.
  - [Azpiegitura](#). 2019. Jon Legarda.
- [Garapena](#). 2019. Jon Legarda.
  - [Build](#). 2019. Jon Legarda.
  - [Garatzaileen Probak](#). 2019. Jon Legarda.
  - [Diseinua - I](#). 2019. Jon Legarda.
  - [Diseinua - II](#). 2019. Jon Legarda.
- [Ingurunea](#). 2019. Jon Legarda.
  - [Proiektuak definitutako prozesua](#). 2019. Jon Legarda.
  - [Tresnak](#). 2019. Jon Legarda.
  - [Garapen Kasua](#). 2019. Jon Legarda.
- [Inplementazioa](#). 2019. Jon Legarda.
  - [Inplementazioa – I](#). 2019. Jon Legarda.
  - [Inplementazioa – II](#). 2019. Jon Legarda.
- [Test](#). 2019. Jon Legarda.
  - [Proba kasuak](#). 2019. Jon Legarda.
  - [Proba Log-a](#). 2019. Jon Legarda.
  - [Proba Script-a](#). 2019. Jon Legarda.



## II. Sistemaren Espezifikazioa

Kasu honetan, eranskinen kasuan bezala, esteka bidez atzigarri daude atal honetan jorratutako dokumentuak. Dokumentu hauek *OpenUP* metodologiak gomendatutako atal ditu eta haren txantiloiak erabili egin dira.

- [Glosarioa](#). 2019. Jon Legarda.
- [Bisioa](#). 2019. Jon Legarda.
- [Betebeharren Espezifikazioa I](#). 2019. Jon Legarda.
- [Betebeharren Espezifikazioa II](#). 2019. Jon Legarda.
- [Erabilpen Kasuen Eredua](#). 2019. Jon Legarda.
- [Erabilpen Kasuak](#). 2019. Jon Legarda.

## III. Aurrekontua

- [Aurrekontua](#). 2019. Jon Legarda.

## IV. Ikerlanak

- [Ikerlanak](#). 2019. Jon Legarda.