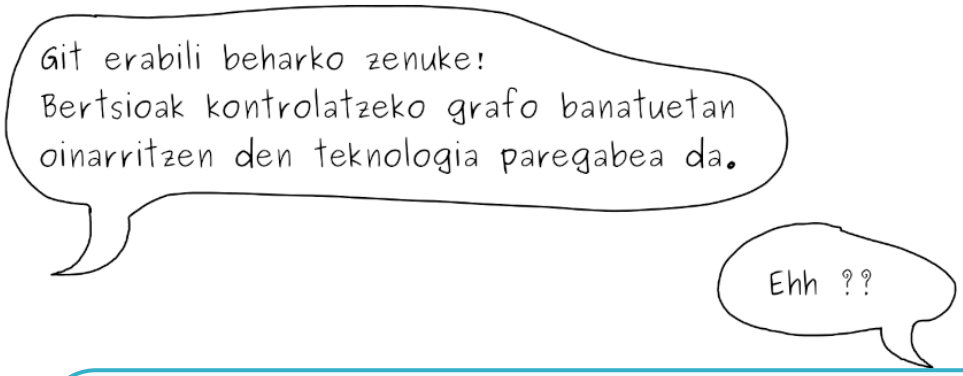


1. Hitzaurrea

Eskuliburu honek Git erabiltzeko beldurra gal dezazun du helburu. Beldurra galtzeko erremediorik onena ezagutza sendoa izatea da, eta, horregatik, eskuliburu honetan kontzeptu errazenetatik hasi eta trikimailu konplexuetara iritsiko gara. Ez dizut gezurrik esango, ordea, bidean ariketa-sorta handia landu beharko baituzu Giten mamian dauden ezaugarriak barneratzeko. Ez dago bestelako misteriorik, **Git erabiliz ikasten da**, hortaz, Git ikasi nahi baduzu denbora tarte luzea eskaini beharko diozu. Hori bai, inbertitu duzun minutu bakoitza baliagarria izango zaizula eta etorkizunean sekulako etekina aterako diozula ziurtatzen dizut.

Tesia garatzen igaro nituen urteetan hainbat software-proiektu garatu nituen, baita hainbat artikulu zientifiko idatzi ere. Garapenak taxuz egiten ari nintzen uneetan ez nintzen jabetzen bertsioak kontrolatzeko sistemak horren beharrezko izango nituenik, baina arazoak topatzean izugarri eskertzen nuen halako sistemak martxan jarri izana. Adibide zehatz bat aipatzearren, behin konturatu gabe eta sekulako arazoa izango nuela ideia zipitzik izan gabe iturburu-kodeko pusketa garrantzitsu bat ezabatu eta softwarea publikatu nuen. Kode hori garatzeko Git erabili izan ez banu hilabeteetako **lanak errepikatzera** behartuta nengoke. Ordura arte Git erabili, erabiltzen nuen, baina momentu horretan bertan jabetu nintzen zein garrantzitsua den softwarea **inkrementalki garatzea** —pausoka—, eta gehikuntza horiek **kontrolpean** izatea. Ez da bakarrik Giten bitartez segurtasun-kopiak izateko aukera errazten dela, baizik eta akatsak eta garapeneko une konplikatuak elegante saihesteko ematen digun indarrean datza Giten gakoa.

Ez zaitut Git gustuko izatera behartu nahi, aldiz, eskuliburu honetan egingo dugun bidaian parte hartzera gonbidatzen zaitut. Ez daukat zalantzarik eskuliburua amaitzean zuk zeuk behartuko duzula zeure burua Git erabiltzera.

Jorratuko ditugun gaien pintzelada bat ematearren, hasieran **bertsioak kontrolatzeko sistemen inguruan mintzatuko gara**, aurrekari batzuk azalduz kontzeptu orokorrak ulertzeko. Ondoren, Giten zentratuko gara, eta Git erabiltzeko oinarrizko definizio eta erabileran sakonduko dugu **oinarrizko aginduak eta prozedurak** azalduz. Git aztertutakoan berehala ariketak egiten hasiko gara, lehenbizi Git instalatu eta kaixo mundu motako ariketa errazak eginez, jarraian **ariketa konplexuagoetara** salto egiteko. Horrekin batera, Git **konfiguratzeko** trikimailu batzuk ikusiko ditugu, eta Git gure modura konfiguratzeko aukera izango dugu.

Lehen urrats horien ostean, kontzeptu konplexuagoetara salto egingo dugu, zure programatzeko gaitasuna beste maila batera eramango duten kontzeptuetara. Atal honetan **aldaketa lokalen eremua** ezagutuko duzu, seguruenik orain arte ezagutu ez dituzun aukera mordoa irekiko dizkizuna. Aldaketa lokalen eremuan garatzaileak kontrola dauka inplementatu behar duen iturburu-kodea txukuntzeko, garapenean atzera eta aurrera mugarik gabe egin baitaiteke. Eremu horretan kode-garatzaileak askatasun osoa dauka proiektu guztia hankaz gora jarriko duten probak egiteko, informazioa galtzeko **arriskurik ez** baitago. Arriskua kudeatzeko mekanismoak erabiltzean, nahi izanez gero, egindako proba guztiak ezabatu eta aurreko egoera segurura itzultzeko aukera ematen digu Gitek.

Amaieran, kodea **taldekideekin partekatzeko** Gitek eskaintzen dituen mekanismoak erakutsiko dizkizut, eta proiektuaren historia txukun manten dezazun laguntzen saiatuko naiz. Horretarako, talde handiek eta txikiek dituzten **lan egiteko fluxuen** inguruan mintzatuko natzaizu, eta **taldekide eraginkorra** izan zaitezen gomendioak emango dizkizut. Orain arte taldean lan egitea desatsegina suertatu bazaizu taldekide guztien iturburu-kodea integratzea zaila iruditzen zaizulako, Gitekin gauzak beste modu batera nola egin ikasiko duzu. Askoz ere modu seguruagoan, garbiagoan eta txukunagoan, gainera.

Eskuliburu hau amaitzean Git **uler dezazun** gustatuko litzaidake, baita haren onurak eta zailtasunak zein diren kontziente izatea; eta batez ere, zure lan egiteko modura egokitu dezazun nahiko nuke. Gitek erremintak eskaintzen dizkigu garatzaileoi, baina lan egiteko prozedurak guk geuk definitu behar ditugu. Taldean definitzen den **lan-prozesura egokitzean** datza Git erabiltzearen arrakasta, horixe bera lortzeko gaitasunak eskuratu ditzazun gustatuko litzaidake.

1.1 Eskerrak

Aspalditik eskuliburu hau idazteko gogoa izan arren, aitortu behar dut bide luzea egin behar izan dugula eskuliburuak argia ikusi duen arte. Atal hau sasibide luzea bilakatu dena zeharkatzen lagundu didaten guztiak eskertzeko erabiliko dut; ez baitira gutxi izan, bakoitzak bere moduan aisialdiko denborari uko egin eta eskuliburuari hondar aletxo bat gehitu diolako. Lehendabizi, UEUri eta bereziki UEUko euskarazuzentzaile izan den Anderri eskertu nahi nioke bere lana, egindako mota guztietako akatsak zuzentzeko pazientzia mugaezina izan duelako, Anderri esker esku artean daukazun eskuliburuak forma tekniko aparta dauka. Nire aldetik uste dut euskarazko terminologia teknikoa lantzea oso garrantzitsua dela, euskara punta-puntako teknologian sartuta dagoen hizkuntza izan dadin. Izugarrizko lana den karga hori Anderren sorbaldan sostengatu da, erabat.

Ezin nahiko eskertu Aitorri eskuliburuaren zirriborro nahasgarriak eta bukatu gabekoak irakurri eta nire buruan zeuden ideia abstraktuak ulertzeko eta ordenatzeko egin duen lan izugarriagatik. Bere kontribuzioak hain garrantzitsuak izan dira, ezen azkenean eskuliburuaren autore gisa egotea proposatu bainion. Aitorri ere eskertu behar diot Giten ez etsitzeko eta amorerik ez emateko motibatzeagatik. Eskuliburu honetan aurkituko dituzun atalik konplexuenak eta nahasgarrienak tentu handiz antolatu eta zuzendu ditu Aitorrek. Zalantzarik gabe eskuliburuaren kalitatearen erantzule nagusia da Aitor, milesker!

Jarraitzeko, Giten eskuliburu hau irakurri eta iruzkinak eman dizkidazuen guztioi eskertu nahiko nizueke zuen kontribuzioa, oso baliagarriak izan dira niretzat zuek aipatutako kontuak, eta eskuliburua askotan berridatzi ere egin dut osoki zuek esandakoaren arabera. Uste dut zuei esker eskuliburua askoz ere ulergarriagoa eta didaktikoagoa dela. Benetan mila-mila esker eskuliburuaren bukatu gabeko bertsioak eta atalak irakurri dituzuen guztioi!

Azkenik, Willix taldea eskertu nahiko nuke, eman didaten babesa eta motibazioagatik. Talde horretan Git erabiltzen zela ikusteak motibatu ninduen aspaldi ni neu ere kontu horietan murgiltzera, haiek bezain beste ezagutu nahi nuelako bai Git baita beste teknologia asko ere. Eskuliburuaren idazketan trabatuta nengoela aurrera egiteko lagun izan nituen, loaren orduak kendu eta arazoen inguruan hitz egiteko. Biba zuek!

2. Helburu zehatzak

Informatikariek eta garatzaileek esfortzu handia eskaintzen diote kode-garapenari; horregatik, ezinbesteko dute kodea txukun, garbi eta seguru garatzea ahalbidetuko dieten tresnak ezagutzea eta erabiltzea. Git bertsioak kontrolatzeko sistemarik ezagunena eta erabiliena dugu egun, eta hori erabiltzen ikasteko denbora eta praktika behar bada ere, etorkizunean ordainetan ematen dituen **onurak, abantailak eta akatsak konpontzeko gaitasuna oso nabariak dira**. Software-garapenari dagokionez, Git erabiltzea ez da hautazkoa **kalitatezko garapena** egin nahi bada.

Eskuliburu honen helburua ez da Git osorik deskribatzeko idazlan bat izatea, baizik eta Git erabiltzeko nagi den hura motibatzeko, oinarrizko kontzeptuak erakusteko eta etorkizun hurbilean izango dituen arazoak konpontzen laguntzeko gida bat izatea. Eskuliburu hau bidelagun duela irakurleak bere programatzeko gaitasunak gorengo maila batera eramatea lortuko du, eta ezagutza horiek bere esparru profesionalean aplikatu ahal izango ditu etekin eta errendimendu handiagoa lortzeko.

Aurrean daukazun eskuliburu hau atal hauetan banatu dugu: lehen atalean hitzaurrea aurkituko duzu, eskuliburuaren laburpen motz bat egiten da bertan eta zu motibatzea du helburu; bigarren atalean eskuliburuaren atalak eta erlazionatutako kontzeptuak deskribatzen dira; behin sarrera eginda, hirugarren atalean bertsioak kontrolatzeko sistema desberdinen sailkapena egiten da, eta horien artean Git kokatu; Giten inguruko oinarrizko kontzeptuak deskribatzen eta lantzen dira laugarren eta bosgarren ataletan, eta atal horiek irakurri ostean Gitekin lanean hasteko prest izango zara. Git aurreratuari hasiera emateko, seigarren eta zazpigarren ataletan fitxategien

egoeren inguruan mintzatuko gara, eta azkenik, zortzigarren eta bederatzigarren ataletan adarkatzen ikasiko dugu, zalantzarik gabe, Giten ezaugarririk ahaltsuena dena —eta beldurgarriena—. Bukaeran, hamargarren atal berezi bat aurkituko duzu, ohiko Git arazoei konponbide on bat emateko errezetak biltzen dituena.

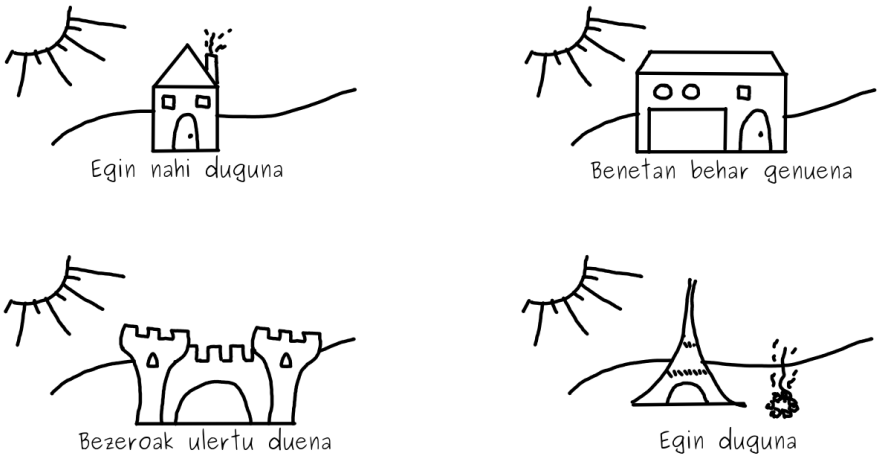
Gogoratu eskuliburu bat ez dela zertan beti hasieratik bukaerara irakurri behar, bakoitzak behar duen ataletik baizik. Prest zara bideari ekiteko?

2.1 SDMak eta Git

Softwarea garatzeko metodologia bat (SDM, ingelesez *software development methodology*) software-proiektu bat ahalik eta modurik onenean eta eraginkorrenean garatzeko jarraitu behar dugun **arau-bilduma** baino ez da. Batzuetan arauak sinpleak dira, adibidez, ezer planifikatu aurretik kodea garatzen hasten garenean; eta beste batzuetan, aldiz, arauak oso konplexuak dira eta rol, bilera eta zeregin konplexu asko kontuan hartzen dituzte. Momentu honetan garrantzitsua da konturatzea egin nahi dugun garapen bati SDM egokia esleitzea ez dela tribiala, eta, askotan, proiektuaren **arrakasta edo porrota** dakarrela erabaki horrek. Proiektu sinpleak eta konplexuak dauden modura, SDM sinpleak eta konplexuak daude. Lotura zuzena egitea da giltza, eta ez da lan makala. Oraindik gogoratzen ditut nire lehen Software Ingeniaritzako eskolak unibertsitatean, eta gogoan dut nola ez nuen ezertxo ere ulertzen.

«Zertarako behar ditut nik softwarea garatzeko arau-bilduma konplexu mordo bat, kodetzen ona banaiz?»

Gerora, lantalde handietan kodea garatzen hasi ahala hasi nintzen ulertzen talde guztia eraginkortasunez martxan jartzeko **plan bat behar dela**. Plan hori softwarea garatzeko metodologia baino ez da. Eta bizitzako beste kontu askorekin gertatzen den modura, esku artean duzun proiektu horri onurarik handiena ekarriko dion plana aukeratzea da zailena, eta, horretarako, kodetzen hasi aurretik proiektua ondo ulertu behar da: Zer egin nahi dugu? Nola egingo dugu? Zein da taldearen jomuga? Zein lerro nagusi ditugu? Zer da bezeroak behar duena? Giltza **kudeaketa-karga gutxien** gehituko dion SDMa aukeratzea da, betiere plan faltagatik proiektua burugabe geratzen ez den bitartean. Jarraian dagoen ilustrazioa adibide bat besterik ez da (2.1 irudia), baina harrituko zinateke, horrelakoak gertatzen diren benetako kasuak zenbatuko bazenitu. Software-ingeniaritzaren ikuspegitik horiek definitzea beharbeharrezkoa da, erabiliko dugun kodeketa-hizkuntza edo SDMa bera erabaki aurretik.



2.1. irudia: Garapen-plan desegokia izatearen ondorioak garapen-lanak eta beharrak dibergenteak izatea dira.

Behin jomuga eta egingo dugun ibilbidea finkaturik daudela, hurrengo pausoa garapen-metodologia aukeratzea da: SDMa, zeinak garapen-prozesua definituko baitu. Besteak beste, SDMak pertsonen rolak eta kategoriak, baliabideen kudeaketa, bileren antolakuntza, bileren maiztasuna, arazoen kudeaketa, komunikatzeko bideak, etab. definitzen ditu, eta garapen-proiektua bizirik dagoen bitartean taxuz jarraitu behar zaio. Era horretan, **behar desberdinak dituzten proiektuak dauden heinean, SDM desberdinak ere definitzen dira**, esaterako: ikasleek askotan erabiltzen duten heroiaren metodologia, azkeneko gaua seko lokartuta geratu arte guztia programatzen saiatzeko helburua duena; ur-jauzi metodologia, proiektua pausoka eta atzera bueltarik egin gabe garatzeko helburua duena; metodologia iteratiboa, proiektuaren gehikuntzak zikloka egiteko helburua duena; prototipatuak, bezeroari eredu bat erakustea helburu nagusi dutenak; moldagarriak edo *agile* deritzonak, bezeroaren ezusteko beharren aldaketetara moldatzeko prestatuta daudenak...

Eskuliburu hau ez da software-ingeniaritzaren ingurukoa, badira primerako materialak kontu horiek guztiak azaltzen dituztenak, eta horregatik, ez dut gai horretan asko sakondu nahi. Baina bada aipatu behar dudan kontu garrantzitsu bat gutxienez Gitekin lotuta dagoena, eta hori azaldu beharrean nago: garapen inkremental dinamikoa. Termino horrek bi kontu desberdini egiten die erreferentzia jatorrian: **garapen inkrementalari eta garapen dinamikoari**.

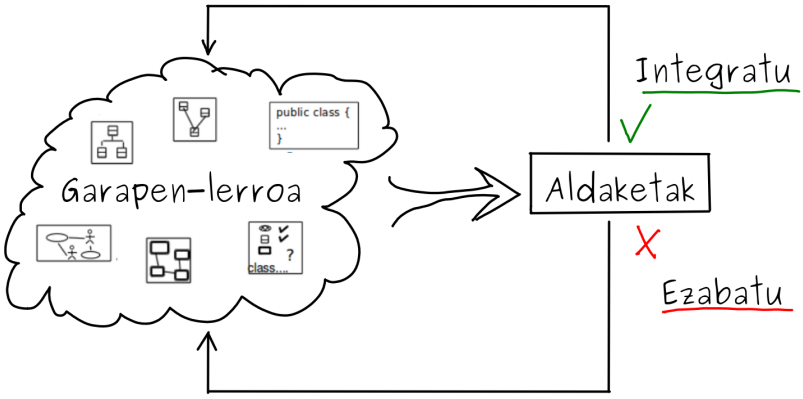
Informatikaren esparruan ohikoa da teknologiak beharren arabera aldatzen joatea, azkar gainera. Era horretan, softwarea garatzeko metodologiak ere eguneratzen

joan dira beharrak eskatu ahala. Urteen esperientziek garbi utzi dute garapen-plan interesgarrienak **dinamikoak eta inkrementalak** direla, eta hori frogatzen duten autore eta material asko dago.

Garapena dinamikoa izateak plan bat erabat finkatuta uztea ezinezkoa dela esan nahi du, hau da, software-proiektuak **bizirik daudela, eta denboran zehar bezeroen nahiak aldatu egiten direla**. Askotan bezeroak ez dira pertsona teknikoak, eta zer egin daitekeen ez dakitelako, edota haien beharrak ere aldatzen direlako, askotan proiektuak ez dira definitzen diren modura amaitzen. Hori onartu beharra dagoen errealitate bat da, eta kode-garatzaile eraginkorrak izan nahi badugu, bezeroen behar-aldaketei aurre egiteko prest egon behar dugu. Lerro honekin defenda daiteke gaur egun horren erabiliak diren metodologia arinak, *agile* deiturikoak, adibidez: *Scrum* edo *Kanban*.

Metodologia horiek azken urteetan oso erabiliak izan dira, **aldaketak modu naturalean onartzeko prestatuta daudelako**. Metodologia horiek rol eta zeregin desberdinak definitzen dituzte, eta proiektua bere bizi-ziklo guztian zehar aktibo manten dadin dute helburu. Gogoratu, normalean bezeroa izaten da software-proiektua diruz mantentzen duena, horregatik, bezeroa gurpilaren barruan integratzea, haren iritziak kontuan hartzea, eta maiztasun altuarekin prototipoak erakustea behar-beharrezkoa da proiektua hutsaren pare gera ez dadin. Horretarako, garatzaileak ez ezik, garapen-prozesuari kontrola jartzeko eta bezeroarekin harremanak izateko *rol* desberdinak ere definitzen dira, baita bilerak egiteko prozedurak eta maiztasunak ere. Garapena **dinamikoa** izateak esan nahi du kodea azkar aldatzeko mekanismo bat behar dugula, hau da, probak egiteko modu merke bat behar dugula. Edozein momentutan garapen-lerro bat utzi eta beste erabilera-kasu bat inplementatzera mugitzeko gaitasuna daukagula, inolaz egindako lana galdu gabe eta egindako proba guztiak gure garapen-lerro nagusietan eragin negatiborik izango ez dutela konfiantza osoarekin.

Garapena inkrementala izateak software-proiektua **helburuz helburu** garatuko dugula esan nahi du, baina kontzeptu hori askotan ondo ulertzen ez delakoan nago. Garapena inkrementala izateak ez du esan nahi aplikazioaren modulu edo erabilera-kasu bat bukatu behar dugula hurrengoa hasi ahal izateko. Horren kontrara, ematen ditugun **pausoak neur daitezkeela** esan nahi du. Hau da, egiten ditugun gehikuntzak neur eta ebalua daitezkeela, eta bukaerako helburura iristeko gertuago gaudela neur dezakegula. Software-ingeniaritzan erabilera-kasuek edota erabiltzaile-historiek *INVEST* izan behar dutela esaten dugu: **I***ndependent*, balio desberdina ematen diotenak bukaerako produktuari; **N***egotiable*, bezeroarekin eztabaidagarriak direnak; **V***aluable*, bukaerako produktuari aurrerapen esanguratsu bat ematen diotenak; **E***stimable*, garapen-taldeari denboran eragingo dion lana aurresateko gai direnak; **S***izeable*, garapen-taldeari eragingo dion baliabideen higadura-maila neur-



2.2. irudia: Garapen dinamikoak software-produktua **malgutasunez eraikitzea** eskatzen du. Garatzen ari garen erabilera-kasuen artean mugituz, probak eginez eta edozein momentutan lana galduko ez dugula ziurtatuz.

garri dutenak; eta **T***estable*, probatu eta bezeroari egiaztatuta entregatu daitezkeenak. Ikusten duzun modura, ezaugarri horiek guztiak goi-mailakoak dira eta garatzen ari garen produktua neurtzeko edota bezeroarekin dugun harremana neurtzeko baliagarriak dira. Goi-mailako ezaugarri horiek neurtu ahal izateko, **garapen-mailan neurketa hori ahalbidetuko digun teknologia izatera behartzen gaitu**, esaterako: bertsioak kontrolatzeko sistemak —BKSak—.

Garapen inkrementalaren adibide bat ematearren, demagun ekosistema baten simulazio-software bat eraiki nahi dugula. *Agile* kontzeptua ondo barneratua ez duen garatzaile batek garapena honela egitea diseinatuko luke seguru asko, urjauziaren SDMan oinarrituta: lehenbizi ekosistemako elementuen diseinua, gero ekosistemako kudeatzailearen garapena, ondoren ekosistemako elementuak; hala nola otsoak, ardiak, erleak, loreak, ura... bukatzeko proba guztiak egin eta amaitu. Gutxi gorabehera garapenak 2.3 irudian laburbildu den eskemari jarraituko lioke.

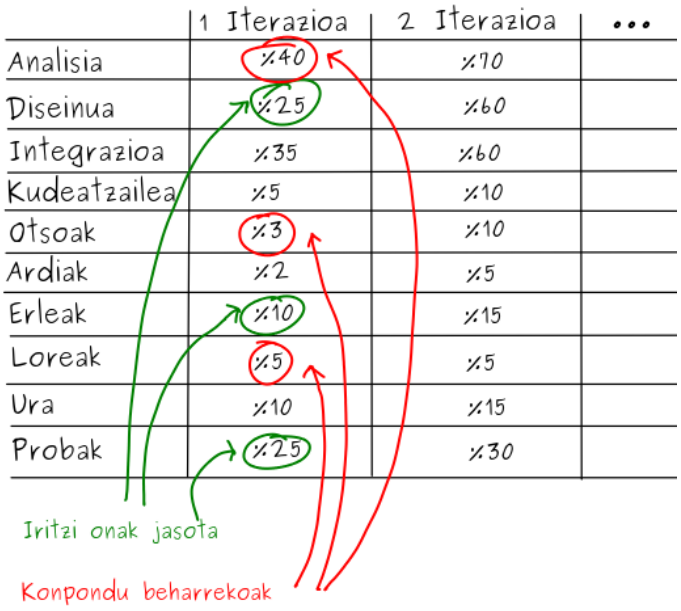
Aldiz, *agile* metodologia barneratua duen garatzaile baten plana erabat desberdina izango litzateke: lehenbizi elementu guztien hasierako diseinu funtzional bat egingo luke, gero oinarrizko proba batzuk, eta proba horien emaitzen arabera hurrengo iterazioa diseinatuko luke, era horretan, **iterazio bakoitzean bukaerako helburura pixkanaka-pixkanaka** hurbilduko litzateke. Garatzaile honek 2.4 irudian dagoen eskemari jarraituko lioke garapena egiteko.

Bi garapen-planen diagramak aztertzen baditugu, erraz ikus daiteke lehen plana **sekuentziala** dela, eta, ondorioz, **aldaketak integratzeko arrotza**. Aldiz, bigarren planean hasierako unetik dago ekosistema funtzional bat, nahiz eta bukatu gabeko

2.3. irudia: Ekosistema baten garapenaren diseinua ur-jauzi motako SDM baten bitartez.

prototipo bat izan. Kasu horretan, garapen inkrementala ondo gauzatu da, eta une oro dago **produktu erabilgarri bat bezeroari erakusteko**. Garapenak zuzen jarraitzeko ezinbestekoa da **bezeroa iterazioen parte** bilakatzea. Trantsizio horietan guztietan bezeroaren iritzia izateak proiektua onartzera eramango du bezeroa. Kontrara, lehen kasuko plan sekuentzialean bezeroak ekosistema funtzionala bukaeran bakarrik ikusiko duenez —zeren ekosistemaren zati isolatuak bere horretan ez dira funtzionalak—, proiektuaren onarpena bakarrik bukaerako unean gauzatuko da.

Bezeroaren iritziak garapen-iterazioetan integratzeko, **garapen-lerroen artean azkar mugitzeko aukera** ematen digun teknologia bat erabili behar da; lana ez galtzeko moduko probak eta esperimentuak egiten utziko diguna, eta proba egokiak eraginkortasunez **integratzeko** aukera emango diguna. Proiektuaren amaiera oso momentu kritikoa izan daiteke bezeroak bere iritzirik eman ez badu garapen guztian zehar, gustatu ezean proiektu guztia baztertzera jo dezakeelako. Bezeroaren iritziak



2.4. irudia: Ekosistema baten garapenaren diseinua *agile* motako SDM baten bitartez.

eta aldaketak azkar integratzen ahalbidetuko digun SDMa eta behar horiek asetuko dituen BKSa erabiltzea behar-beharrezkoak dira.

Softwarea garatzeko metodologien ur handietan gehiago sakondu gabe, **Git BKS da benetako garapen inkremental dinamikoa egiteko aukera paregabea emango digun tresna**. Izan ere, software-proiektu orotan egiten ditugun gehikuntzek Giten historian izango dute isla. Hau da, edozein proiektutan egiten ditugun aurrerapenak neurtzeko gai izango gara, eta dinamikoak izateko gaitasuna historian zehar mugitzeko ahalmenak emango digu, taldean dauzkagun garapen-lerro desberdinak kudeatuz eta behar direnetan integrazioak eginez. Are gehiago, *agile* metodologietan aski ezagunak diren erabiltzaile-historiak —garatu behar diren erabilera-kasuak— Giten adar gisa garatzen dira. Erabiltzaile-historia horiek direnez proiektua handitzera eramaten duten ezaugarriak, bukatuta dauden erabiltzaile-historiak proiektuaren mamian txertatzean gertatzen dira gehikuntzak.

Beste hitz batzuetan esanda: **Giten adarretan dauden erabilera-kasuak lerro nagusian integratzean egiten du aurrera proiektuak**. Oraindik kontzeptu hauek guztiak ulertzen ez baditugu ere, garapen inkrementala eta dinamikoaren oinarria dira. Garapen mota horri, garapen **ez-lineala** eta **adarkatua** deritzo.

2.2 Git ikasten hasteko gomendio batzuk

Jarraian Git ikasteko aholku batzuk zerrendatzen dira, nahiko generikoak izan arren, Git ikasten hasteko baliagarriak izango zaizkizunak.

I

Git doako BKS bat da, ez daukazu ezer galtzeko hori erabiltzen hasteagatik. Hala eta guztiz ere, Git tresna konplexua izanik zure ezagutza-mailara egokitzen den erabilera bat egiten saiatu behar duzu. Denborarekin eta praktikarekin behar berriak agertzen joango dira, eta Giten trebatzen joango zara hein berean. Hasiera batean Git erabiltzea eta BKS batekin lan egitera ohitzea izan behar du helburua, denborarekin etorriko da espezializazioa.

II

Ez egin edozein aldaketaren konfirmazioa, saiatu antolatua izaten. Gogoratu software-garapenaren lerroan zein garrantzitsua den esanguratsuak diren pausoak ematea, pauso horiek zentzuzkoak eta erlazionatuak izan behar dute etorkizunean zer egin dugun gogoratu edo ulertu nahi badugu. Pauso txikiak ematea ere lagungarria da garapen-taldea osatzen duten beste kideek guk egindakoak hobeto ulertzeko.

III

Eguna bukatzean ez da konfirmazio bat egin behar, lasai, konfirmazioak ez dira erabiltzen egin dugun lan kopurua neurtzeko. Arazoak izan ditzakeen edozein software zatiren konfirmaziorik ez da egin behar, bukatu gabeko lanak edota akatsak izan ditzaketen kode-fitxategiak ez ditugu gure taldekideen artean partekatu nahi.

IV

Esperimentuak eta probak Giten adarkatzeko baliabideekin edo garapen pribatuen eremuan egin, probak ondo ateratzen badira integratuko ditugu garapen nagusiaren lerroan, eta nahi izanez gero ezabatu eta berriro garbi hasteko aukera izango dugu. Taldekideak proba esperimental horietan sortzen den kodearekin ez kontaminatzea garrantzitsua da. Kalitatezko kodea partekatzea da gure helburua.

V Saiatu maiztasun altuarekin sinkronizatzen zure taldekideekin, horrek akatsak eta gatazkak azkar azaleratzea ekarriko du.

VI Erabili Giteko etiketa eta konfirmazio-mezu esanguratsuak. Kodea ondo dokumentatuta mantentzea garrantzitsua da taldeak kohesio handia izateko.

VII Adostu zure taldekideekin nola erabiliko duzuen Git, zenbat adar sortuko dituzuen eta zer maiztasunekin sinkronizatuko zareten.

VIII BKS bat ez da Back-up zerbitzu bat. Egia da kodea hodeian gordeta geratzen dela, baina Git back-up sistema gisa erabiltzeko intentzioa baduzu, ez zara erabilera batere ona egiten ari.

IX Prest egon gauza berriak ikasteko. Git sistema konplexua da, eta urteak behar dira menderatzeko. git help agindua oso maiz erabiltzen ari bazara, seinale ona.