Mòdul basat en regles per la gestió de laboratoris d'anàlisis clínica

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Oscar Fernàndez Castro
i dirigit per
Elena Valderrama Vallés
Bellaterra, ١٧ de Septembre de ٢٠٠٩

El sotasignat, Elena Valderrama Vallés

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

Oscar Fernàndez Castro

I per tal que consti firma la present.

Signat:

Bellaterra, 17 de Setembre de 2009

El sotasignat, Luís Suárez Novau,

Cap de projecte de l'Omega v4, de Roche Diagnostics

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

Oscar Fernàndez Castro

I per tal que consti firma la present.

Signat:

Bellaterra, 17 de Setembre de 2009

Índex

1.- Introducció

1.1.- Presentació

El següent document mostra el desenvolupament del Projecte Final de Carrera (PFC) de l'enginyeria Informàtica de la Universitat Autònoma de Barcelona (UAB).

És un projecte de conveni universitat-empresa, entre l'UAB i Roche Diagnostics S.L.

El projecte formà part del desenvolupament d'un software de gestió de laboratoris clínics que Roche Diagnostics anomenà Omega v4. Més concretament és tracta del disseny d'un mòdul basat en regles per a controlar certs aspectes d'una aplicació de gestió de laboratoris d'anàlisis clínica.

1.2.- Objectius

L'Objectiu d'aquest projecte de fi de carrera, és desenvolupar el mòdul de Rule Engine (mòdul basat en regles) , que com a part de l'Omega v4, vol dotar a un laboratori de les eines necessàries per comunicar-se amb les màquines d'anàlisis clínica i poder gestionar de la manera més senzilla tots els aspectes relacionats amb el laboratori i el seu treball diari.
Aportant les noves tecnologies, com la tecnologia web i llenguatges orientats a objectes, Roche Diagnostics vol renovar i actualitzar un dels seus productes estrelles, l'Omega 3000. L'Omega 3000 és al igual que l'Omega v4, un software per gestiona laboratoris, però al contrari que aquest portà bastants anys en el mercat, i tot i ser un producte molt complert s'ha quedat desfasat tecnologicament.

Per tant l'objectiu de desenvolupar l'Omega v4 es millorar un producte ja existent per tal de seguir mantenint la quota de mercat i demostrar que Roche continua sent una empresa líder en tecnologia, i pionera de l'àmbit sanitari.

En el cas del mòdul de RuleEngine, l'objectiu es controlar tant de manera automàtica com manual, els diferents resultats obtingut en el laboratori. Mitjançant la creació de regles, l'usuari podrà controlar els resultats obtinguts i realitzar les accions prèviament definides, sempre i quan els resultats siguin els especificats en les condicions. D'aquesta manera és el mateix sistema el que s'ocupa de controlar els resultats obtinguts.

Per exemple quan una màquina fa un anàlisis de glucosa en un tub de sang, si els resultats són lleugerament més elevats del que es recomanable, es podria usar el Rule Engine perquè afegís un comentari a la petició.

Una altre possibilitat seria, que quan l'edat del pacient estigues entre els 20 i 30 anys i fos del gènere masculí, s'afegís a la petició la prova de testosterona per tal de ser analitzada.

2.- Entorn de treball i estudi de viabilitat

2.1.- Objectiu

L'objectiu de Roche Diagnostics és oferir als seus actuals clients (o possibles clients), un nou producte amb les noves tecnologies actuals, per tal de millorar i facilitar la gestió d'un laboratori. Però al ser un software per laboratoris clínics i d'ambient molt especialitzat s'ha de donar suport tècnic al client i de correcció d'errors a mesura que van apareixent.
Tot i aquest suport continuat, arribarà un punt que els clients demanin noves funcionalitats o que l'aplicació tingui unes noves característiques que el producte actual no pugui donar, per la qual cosa s'haurà de pensar en la implementació d'un nou producte.

En el cas de Roche Diagnostics un clar exemple és l'Omega 3000, un producte que també serveix per la gestió de laboratoris clínics i que ja porta més de 10 anys en el mercat. L'Omega 3000 a arribat a un punt que el seu manteniment i desenvolupament de les noves implementacions requerides per als clients són de complexitat molt elevada i això és problemàtic tant per l'empresa com per els clients, per això es va optar per la creació d'un nou producte, l'Omega v4 que a de guarda certa compatibilitat amb l'anterior però que alhora té millores substancials com ser de tecnòloga web.

L'Omega v4 és un projecte molt gran i està dividit en diferents mòduls, cadascun amb una funcionalitat especifica. Ens trobem amb els mòduls d'Order Entry, Validation, WorkingList, Hematology i Rule Engine entre d'altres. Aquest projecte es centra en el mòdul de Rule Engine que s'utilitza per crear i executa regles per controlar el funcionament de l'Omega v4. Es podrà veure com a partir de certs paràmetres un usuari (amb permisos) pot crear un conjunt de regles molt elevat per controlar certs aspectes, com podria ser el resultat d'una mostra, l'entrada de demogràfic en una petició, etc.

En definitiva l'objectiu és observar les mancances de l'anterior producte, i els nous requeriments que demana el mercat, per crear un nou producte que agradi als clients i permeti a Roche seguir mantenint el liderat en l'àmbit dels laboratoris.

2.2.- L'empresa

2.2.1.- Historia i organització de l'empresa

Roche Diagnostics S.L. va ser fundada l'any 1896 a Basilea, Suïssa. Va començar com un petit laboratori i actualment és una de les empreses líders a nivell mundial en investigació per a la salut. Es troba en més de 20 països, com ara Espanya, Brasil, Alemanya, Japó, Argentina i molts d'altres. La seu central es troba actualment a Rotkreuz, Suïssa.

L'empresa es divideix en dos grans branques, la farmacèutica i la de diagnosis. La part de diagnosis és l'encarregada del desenvolupament del software clínic per als laboratoris. En el cas espanyol la part farmacèutica es troba a Madrid i la part de diagnosis es troba a Sant Cugat del Vallès, Barcelona.

L'estructura de l'empresa a Espanya està dividida en les següents seccions:

- Diabetis care: Serveis per al seguiment i control de la diabetis (mesuradors, software de seguiment, etc.).
- Diagnostics: Dona suport a laboratoris d'anàlisis clínic (software i hardware).
- Molecular Diagnostics: Detecció malalties mitjançant àcids nucleics amb tecnologia PCR.
- Near Patient Testing: Crear aparells fàcils d'usar perquè pacients sense experiència siguin capaços d'utilitzar.
- Applied Sciencies: S'encarrega de la investigació Biomèdica.

Roche és una gran empresa que té més de 68.000 empleats per tot el món.

Empleats per regió	
Roche global	68218
Europa	29934
Suïssa	7860
Nord-americà	21899
Llatinoamericà	4465
Àsia	10459
Japó	5988
Àfrica, Austràlia i Oceania	1461

Empleats per divisió	
Roche global	68218
Farmàcia	48049
Diagnòstica	19788
Altres	381

Roche Diagnostics Sant Cugat, on es realitza el projecte RuleEngine, dona feina a aproximadament unes 600 persones, i es desenvolupen totes les activitats de diagnosis dels països ibèrics (Espanya i Portugal) i de la regió de llatinoamericana.

2.2.2.- Situació dintre l'empresa

La part de diagnosis de Roche Sant Cugat es troben varis departaments, com ara Recursos Humans (RRHH), Màrqueting i IS (Informatic Solutions). El departament de IS és l'encarregat del disseny de les aplicacions software pròpiament dites.
Dintre de IS troben tres subdepartaments:

- Informatic Solution Development (IS-D): És el departament encarregat de la creació dels productes de software de Roche a Espanya, Portugal i Llatinoamericà. Un dels productes creats per IS-D és l'Omega v4.

- Informatic Solution Quality (IS-Q): Un producte (software) que acaba de sortir, especialment si és de l'àmbit sanitari, a de garantir un mínim de qualitat. IS-Q és el departament encarregat de verifica que els productes desenvolupats tenen aquest mínim de qualitat. Per tant és el departament que valida els productes.

- Informatic Solution Tecnology (IS-T): Aquest departament s'encarrega de donar la infraestructura i aplicacions necessàries per a poder treballar amb els software de Roche.

Per tal de treure un producte al mercat és indispensable que els tres departaments (IS-Q, IS-T i IS-D) treballin de manera conjunta i coordinada.

És dintre el departament de IS-D on s'ha desenvolupat aquest projecte.

Dintre del departament de IS-D hi ha varies aplicacions en funcionament de manera simultània, com ara el Omega 3000, l'Omega v4, PSM, Microbiologia i altres.

Actualment el grup que treballa amb l'Omega v4 usa una eina de desenvolupament anomenada Cache, i que proporciona l'empresa Nord-americana Intersystems. Cache és una base de dades post-relacional, molt robusta, amb accés directe al motor multi dimensional i amb un SQL d'alt rendiment. Cache és una eina òptima per el desenvolupament d'aplicacions web, ja que permet un ràpid desenvolupament i una alta velocitat de transaccions, així com escalabilitat i consultes en temps real sobre dades transaccionals.

ZEN és una tecnologia de creació recent que està incorporada en Cache, i en la qual es centra gran part del desenvolupament de l'Omega v4. Roche a apostat de manera molt important per aquesta nova tecnologia per considerar-la ideal per aplicacions web, ja que és de desenvolupament ràpid , i es pot crear de manera automàtica components de Javascript i HTML. Però el fet d'utilitzar una tecnologia tant nova (encara no hi ha cap programa al mercat que utilitzi ZEN) té l'inconvenient de que en certa manera en certa manera som uns 'conillets d'índies' alhora de provar aquesta nova tecnologia.

```
<LIS:LISText id = “txtValue” label=”lblValue” inputType=”STRING”/>
<LIS:LISSpacer width=”20”/>
<LIS:LISDataCombo id = “cmbSM” label=”lblSMod” OnCreateResultSet=”CreateRSSecMod”/>
```

Figura 2.1 ZEN Components

Només definint aquestes tres línies el programador pot crear d'una manera fàcil i ràpida una àrea de text, un espai de la mida desitjada, i un combo que extreu les dades de la funció indicada.

2.3.- Ús de l'aplicació

Roche Diagnostics desenvolupa aplicacions per laboratoris d'anàlisis clínica, per tant s'ha d'adaptar al funcionament dels laboratoris. De manera que hi ha usuaris que només hauran d'imprimir informes i/o donar d'alta pacients i d'altres (com el cap del laboratori) que haurà de donar d'alta les proves i configurar les regles.

Per tant l'aplicació a de tenir en compte els diferents usuaris, donant els permisos necessaris a cert usuaris depenent dels rols que tinguin assignats. A més s'ha d'intentar que cada usuari pugui realitzar les seves funcions de la manera més ràpida i simple possible.

2.4.- Alternatives

Es podria preguntar perquè s'ha escollit Cache per realitzar un projecte de tal magnitud, i més concretament perquè es va escollir ZEN, una tecnologia tant nova per realitzar un programa tant important per Roche. La raó principal és que Cache és l'eina corporativa de Roche Diagnostics, ja que Roche confia plenament en Intersystems, l'empresa que ha desenvolupat Cache, ja que amb anterioritat han cooperat en altres projectes que han tingut bons resultats.

Altres raons són:

- Portabilitat: Cache és multi plataforma, es pot executar en Windows i Unix.

- Flexibilitat: Hi ha diverses maneres d'accedir a les dades de Cache, ja sigui amb SQL, objectes o arrays. L'Arquitectura de dades unificada de Cache elimina la "diferència d'impedància" entre les definicions de les dades relacionals i dels objectes.

- Rendiment: Cache proporciona una velocitat molt alta en recuperació i modificació de dades.

- Desenvolupament web: Cache proporciona eines de desenvolupament web, que permeten una gran velocitat.

- Interoperativitat: Els objectes creats en Cache poden ser usats amb Java, C++ o qualsevol interfície COM. D'aquesta manera Cache es compatible amb entorns de desenvolupament com Visual Basic, Delphi, etc.

Un dels aspectes més problemàtics que s'han trobat al aplicar la nova tecnologia web és que al contrari dels anteriors productes de Roche Diagnostics eren de tipus visual, anteriors Omegas i PSM, això feia que el accés a les dades fos extremadament ràpid. Però tenia el inconvenient de que l'aplicació es tenia que instal·lar en cada màquina del client, i per fer modificacions aquestes es tenien que fer en cada màquina que tingues l'aplicació instal·lada. Aquest inconvenient feia perdre temps i diners a l'empresa.
Amb l'actual tecnologia web, això ja no passa, però al contrari que les aplicacions anteriors els accessos a dades són més lents, cosa que va provocar les queixes d'alguns clients, ja que estaven acostumats a les antigues aplicacions que funcionaven més ràpid.

Per tant els desenvolupadors hauran de posar especial atenció en l'optimització de l'aplicació per aconseguir que respongui en el menor temps possible.
Cal esmentar que depenent del navegador usat el temps de resposta pot variar de manera significativa. En principi l'aplicació dona suport pels navegador IE6, IE7,Firefox2 i Firefox 3.5, encara que per raons de rendiment s'aconsella utilitzar la versió més recent del Firefox (Firefox 3.0 no està suportat).

Tot i els inconvenients explicats, les seves avantatges fan de Cache una eina ideal per fer aplicacions com l'Omega v4.

2.5. - Especificacions

El producte a desenvolupar, ha de complir amb els estàndards de Roche Diagnostic i de l'Omega v4. Amb els software escollit per Roche Diagnostics per crear l'Omega v4, Cache i ZEN, s'ha de desenvolupa un mòdul basat en regles per gestionar laboratoris d'anàlisis clínica.

Per realitzar el projecte és necessari un ordinador amb un processador mitjanament potent (Pentium 4, C2D o similar), 2 GB de RAM, un disc dur de 40 GB o més, i un monitor amb resolució de 1280x960. La connexió a Internet és també un requisit indispensable.

En quan a software, Interystems Cache i ZEN són imprescindibles per el desenvolupament. El Perforce és necessari per gestionar els arxius creats, i els navegadors Internet Explorer i Mozilla Firefox per connectar-se a l'aplicació.

Per realitza proves de rendiment, seria aconsellable disposa d'un servidor, amb el software ja comentat.

2.6.- Requeriments

2.6.1.- Requeriments hardware

Com tota aplicació l'Omega v4 necessita uns recursos hardware per poder funciona de manera correcta. Al ser una aplicació altament escalable, podríem configurar els seus requisits depenent de la mida del laboratori, més concretament de quantes persones l'usaran de mode simultani.
Per tant definirem tres tipus de laboratoris, petits, mitjans i grans.

Tipus de laboratoris:

- Laboratoris petits: fins a 25 usuaris simultanis.
- Laboratoris mitjans: de 25 a 75 usuaris simultanis.
- Laboratoris grans: més de 75 usuaris simultanis.

Configuracions hardware mínima i recomanada per laboratoris petits:

Requeriments mínims	
CPU	C2D E2200
RAM	2 GB
HDD	4*36Gb Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Requeriments recomanats	
CPU	C2D 8200
RAM	4 GB
HDD	4*72 GB Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Configuracions hardware mínima i recomanada per laboratoris mitjans:

Requeriments mínims	
CPU	C2 Quad E6600
RAM	4 GB
HDD	4*36Gb Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Requeriments recomanats	
CPU	C2 Quad E6600
RAM	8 GB
HDD	4*72 GB Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Configuracions hardware mínima i recomanada per laboratoris grans:

Requeriments mínims	
CPU	C2D E2200
RAM	2 GB
HDD	4*36Gb Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Requeriments recomanats	
CPU	C2D 8200
RAM	4 GB
HDD	4*72 GB Ultra3 SCSI-3 (15k rpm)
DVD	Gravador
UPS	Opcional 1kVA
Xarxa	Ethernet 100/1000

Aquest requeriments són els aconsellats per al servidor que han de suportar múltiples accessos de forma concurrent, però per al client els requeriments són considerablement menors, degut a que a d'executar bastants menys processos que el servidor.

Els requeriments mínims per als clients són els següents:

Requeriments mínims pels clients	
CPU	Pentium4 o AMD K7
RAM	512 MB (1GB recomanat)
HDD	40GB
Xarxa	Ethernet 100/1000
Ratolí i teclat	si
Monitor	Resolució 800*600 (altament recomanades resolucions de 1024*768 o 1280*1024)
Lector de codi de barres	Opcional (depèn de l'ús de l'aplicació)

2.6.2.- Requeriments software

Els requisits de software són més semblants per al client i el servidor que en el cas del hardware. És necessari un sistema operatiu com Linux o Windows (95, 98, 2000, XP) preferiblement correctament actualitzats i un navegador d'Internet compatible amb l'aplicació com són IE6, IE7 i Firefox2, encara que es recomana utilitzar el Firefox2. D'altre banda el servidor a de tindre instal·lada la versió corresponent de Cache.

2.7.- Documentació

Una part molt important especialment en projectes de gran envergadura, és la documentació, que ha de ser clara, fiable i a d'intentar resoldre tots els possibles dubtes per tal d'ajudar el màxim possible al desenvolupador, i aquest pugui fer la seva feina de manera més ràpida i eficient.

La documentació la podem dividir en dos grans grups, la documentació funcional i la documentació tècnica.

- Documentació funcional: descriu el funcionament que a de tenir l'aplicació i quines funcions ha de realitzar.
  El client pot ajudar a donar una primera idea de com vol que sigui el producte ja que al final serà ell l'usuari final que a d'utilitzar el programa.
  Els analistes també podem aportar informació funcional després d'haver parlat amb el client i haver vist quines són les seves necessitats. Una de les avantatges de que la informació provingui dels analistes és que la informació serà més tècnica que si prové del client.
  La política de documentació de Roche Diagnostics fa que tota la documentació ha d'estar per escrit en format .doc en un dels servidors de l'empresa de manera que estigui accessible per tothom que o necessiti.

- Documentació tècnica: indica com s'ha de realitzar el projecte, les eines o metodologies que es poden utilitzar per aconseguir l'objectiu. A d'ajudar a resoldre dubtes sobre com usar l'arquitectura escollida, possibles opcions o modes d'implementació.
  Una font d'informació és la que proporciona l'empresa Intersystems a través d'Internet en la seva pàgina web, o la documentació dels cursos de formació que pot donar l'empresa als seus treballadors.

2.8.- Planificació temporal

Un dels aspectes més importants quan es vol treure un producte al mercat és el temps. És necessari fer un càlcul aproximat de quan estarà enllestit el producte per tal que els comercials de l'empresa el puguin oferir, de tal manera que preveure a quins concursos podran participar.

Totes les grans empreses, Roche inclosa, fan una planificació temporal dels temps que trigaran en realitzar cada etapa del projecte, depenent òbviament dels 'recursos' que l'empresa té disponible. D'aquesta manera els responsables del projecte fan una estimació i donen una data aproximada de quan estarà acabada una fase o el projecte en si mateix. És una estimació ja que per norma general és molt complicat conèixer al cent per cent quan s'acabarà, ja que depèn de molts factors, però el més important és que encara que no s'encerti amb exactitud, si que s'aproximi el màxim possible.

Al ser un projecte de conveni universitat-empresa, aquest es realitzarà dintre de l'empresa i per tant la planificació de l'Omega v4 i més concretament del mòdul basat en regles o Rule Engine la farà el responsable del projecte de Roche Diagnostics. A excepció clar està de la memòria del projecte que és responsabilitat exclusiva del alumne.

Nombre de tarea	Duración	Comienzo	Fin
Disseny	**12 días**	**lun 04/08/08**	**mar 19/08/08**
Disseny UC, US	9 días	lun 04/08/08	jue 14/08/08
Disseny UT	4 días	jue 14/08/08	mar 19/08/08
Implementació	**107 días**	**jue 21/08/08**	**vie 16/01/09**
Reglas	19 días	jue 21/08/08	mar 16/09/08
Accions	27 días	mié 17/09/08	jue 23/10/08
Condicions	38 días	vie 24/10/08	mar 16/12/08
Control	10 días	mié 17/12/08	mar 30/12/08
Activació	8 días	mié 07/01/09	vie 16/01/09
Validació	19 días	mié 07/01/09	lun 02/02/09
Correcció d'errors	19 días	mar 03/02/09	vie 27/02/09

Figura 2.2 Dades de la planificació temporal del projecte

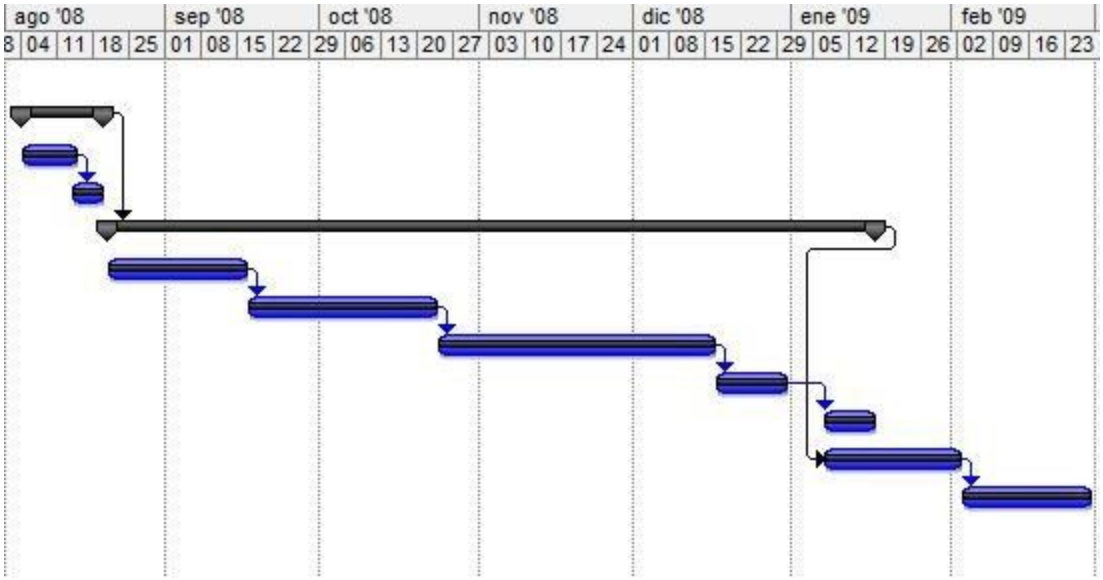


Figura 2.3 Diagrama de Gantt de la planificació temporal del projecte

2.9.- Anàlisis econòmic

Roche és una empresa multinacional que es troba en molts països, així com el seus productes. Per exemple en el cas espanyol esta implantada en més de 100 laboratoris, però també els trobem en Indonèsia, Brasil, etc.

Amb l'actual Omega v4, Roche es vol centrar principalment en el mercat de la sanitat publica i per aconseguir-ho hi ha dos maneres possibles.

- Concursos públics.
- Contractes per prestacions de serveis.

En els dos casos es venen les llicencies d'ús de les aplicacions i els serveis de postvenda per un període que oscil·la entre 3 i 5 anys.

Per tant es fa necessària una avaluació econòmica de l'Omega v4, però també ens centrarem amb l'avaluació econòmica del Rule Engine que és el que ens importa en aquest PFC.

- Anàlisis econòmic del mòdul de Rule Engine:

Cost del Personal:

Número	Rol	Cost/hora (€)	Hores	Cost Total (€)
1	Cap de projecte	60	25	1500
2	Analista funcional	2x50	2x40	4000
1	Suport Tècnic	50	10	500
1	Analista-Programador	30	800	24000
1	Tester	30	50	1500
Total:				31500

Costos fixes:

Descripció	Cost (€)
Llicencies software Intersystems (Cache)	10000
Llicencies software Microsoft	5000
Cost del personal (pcs, software, etc.)	32000
Total:	47000

Cost total del mòdul de Rule Engine:

Descripció	Cost (€)
Cost del personal	31500
Costos fixes	47000
Total	78500

El mòdul que ens interessa en aquest projecte, el Rule Engine té un cost de 78.500 euros, però s'ha de tindre en compte que el mòdul de Rule Engine no es pot vendre ell sol, sinó que és part d'un

producte més gran, per la qual cosa fa falta fer un anàlisis econòmic del producte sencer, per comprovar si realment és rentable.

- Anàlisis econòmic de l'Omega v4:

Cost del personal:

Nom	Rol	Cost/hora (€)	Hores	Cost Total (€)
1	Responsable IS	80	2400	192000
1	Responsable IS-T	65	2400	156000
1	Responsable IS-D	65	2400	156000
1	Responsable IS-Q	65	2400	156000
1	Cap de projecte	60	5800	348000
3	Analista funcional	50	17400	870000
1	Suport Tècnic	50	5800	290000
3	Analista-Programador	30	17400	522000
10	Desenvolupador	30	58000	1740000
2	Tester	30	6000	180000
Total (€):				4.610.000

Costos fixes de l'Omega v4:

Descripció	Cost (€)
Llicencies software Intersystems (Cache)	200000
Llicencies software Microsoft	100000
Cost del personal (pcs, software, etc.)	230000
Total:	530000

Cost total de l'Omega v4:

Descripció	Cost (€)
Cost del personal	4.610.000
Costos fixes	530.000
Total (€)	5.140.000

Cost d'implantació:

Cada instal·lació que es realitza en un hospital té un cost d'implantació considerablement elevat ja que es necessari uns programes i unes llicencies a més del hardware corresponent per fer funcionar

el software. Tot i que aquest cost depèn del tipus de laboratori, la seva mida, els accessoris i les màquines de les que disposen, entre altres. És calcula que el cost mig per implantació és d'uns 80.000 euros.
Apart s'ha de tindre en compte els beneficis de Roche que serà d'un 25% del cost d'implantació. Per tant el cost final és de 100.000 euros.

Conclusions:

Com es pot observar el cost total de l'omega v4 és considerablement elevat. Per la qual cosa Roche no espera tenir beneficis fins el tercer o quart any d'implantació de l'Omega. Això es degut que normalment en l'àmbit sanitari és bastant complicat fer un forat al mercat per un producte nou, ja que molts centres són reticents a canviar un producte que ja coneixen i que funciona correctament per un altre desconegut i que al ser més nou és una mica més propens a errors.

Segons els càlculs realitzats per Roche s'espera obtenir un benefici aproximat de 25.000 euros per instal·lació d'Omega v4.

Instal·lacions d'omega v4 previstes:

Any	2009	2010	2011	2012
Espanya	4	15	28	45
Regne Unit	0	0	6	14
França	0	0	6	11
Alemanya	0	0	4	9
Itàlia	0	3	13	18
Resta d'Europa	0	0	24	48
Àsia	0	0	3	8
Llatinoamericà	0	2	18	67
Total	4	20	102	220

Any	Nº instal·lacions
2009	4
2010	20
2011	102
2012	220

Com es pot veure Roche té previst que el número d'instal·lacions de l'Omega v4 creixi de manera exponencial en pocs anys. Encara que sembla que poden ser moltes instal·lacions s'ha de tenir en compte que és un producte global. Altres productes de Roche que han tingut bastant èxit, com és el PSM (programa per comunicar les màquines) tenen més de 1.200 instal·lacions arreu del món.

Per tant per sufragar els costos del projecte serien necessàries entre 200 i 210 instal·lacions per la qual cosa, com s'ha comentat anteriorment, no serà viable fins el quart any.

Vistos els càlculs anteriors arribem a la conclusió que el projecte és viable apart de necessari, ja que Roche necessitava l'Omega per seguir competint en diferents camps de l'àmbit sanitari.

3.- Anàlisis funcional i de requeriments

Els requeriments són el conjunt d'idees que el client té sobre que a de fer el software a desenvolupar, i permeten conèixer els elements necessaris per definir un projecte. Tenen un paper molt important a l'hora de produir software, ja que defineix que es vol produir, com s'ha de fer i quins passos s'han de seguir per fer-ho.

D'aquesta manera podem dir que l'anàlisi funcional serveix per crear especificacions correctes que s'escriguin d'una manera clara i sense ambigüitats el comportament que a de tenir el producte. D'aquesta manera el que es pretén és optimitzar el procés d'implementació del producte i facilitar el seu desenvolupament.

Un dels problemes més habituals que es tenen al definir l'anàlisi funcional d'un producte, és la mala comunicació que a vegades es dona entre les persones. Sovint el client i l'analista no es posen d'acord ja sigui perquè el client no a sabut descriure el producte que volia o perquè l'analista no a entès el que el client volia dir. També pot passar en diferents àmbits de desenvolupament del producte, com la comunicació analista- desenvolupador. Això es deu a que el llenguatge que utilitzem per comunicar-nos habitualment és un llenguatge molt ambigu.
Per tant d'aquí ve la importància de fer uns requeriments funcionals el més clars possibles.

Dintre de el anàlisis de requeriments el podríem dividir en quatre fases diferents:

- Reconeixement del problema: l'analista a d'estudiar diferents alternatives per solucionar el problema.
- Avaluació i síntesis: l'analista a de refinar les funcions que tindrà el software i les característiques.
- Especificació: es dona un representació del producte per mostrar-la al client.
- Revisió: s'especifiquen els criteris de validació.

3.1.- Requeriments tècnics

Aquests requeriments mostren les limitacions del sistema, així com les tecnologies necessàries per implementar el producte, ja sigui el software de programació pròpiament dit (Cache i ZEN), com el software complementari que els desenvolupador necessitin, com ara el Perforce, els navegadors etc.

3.2.- Requeriments no funcionals

Els requeriments no funcionals es refereixen a propietats del software com ara la fiabilitat, el temps de resposta, la portabilitat, etc.

Aquests requeriments sorgeixen de les necessitats de l'usuari, ja sigui per polítiques internes (confidencialitat, seguretat, etc.), o simplement per qüestions econòmiques.

Per l'Omega v4 són elements crucials tant la portabilitat con la seguretat. La portabilitat és important perquè alguns hospitals tenen sistemes basats en Linux i d'altres el tenen basat en Windows. Per altre banda al emmagatzemar dades sanitàries de caràcter privat és molt important que les persones no autoritzades no puguin accedir a les dades.

En el cas del Rule Engine, un aspecte que s'ha de tenir molt en compte és el temps de resposta, ja que al ser un mòdul que es instanciat des de els altres mòduls del quals disposa l'Omega v4, si l'execució és lenta, es ressentiria de manera notable tota l'aplicació.

3.3.- Requeriments corporatius

Roche, per assegurar-se la qualitat dels seus productes, té un requeriments estàndards propis. Malgrat que per diversos motius no es compleix sempre al 100%, és una bona eina per assegurar-se que els productes compleixin un mínim de qualitat.

3.3.1.- Requeriments corporatius (disseny)

Roche Diagnostics S.L. va decidir que tot l'Omega v4 havia de tenir un disseny específic, per tant el Rule Engine també a de complir amb aquests requeriments. Totes les pantalles han de tenir un disseny similar per ajudar a l'usuari a familiaritzar-se amb el programa i fer-lo més accessible i coherent.

3.3.1.1.- Colors

Per començar es va decidir que els colors de les pantalles serien principalment el blau i el gris. Deixant el taronja per visualitzar els objectes seleccionats per l'usuari. L'elecció d'aquests colors es va decidir per donar un aspecte sobri i seriós a l'Omega.
Per tant totes les pantalles de l'Omega v4 fan servir els mateixos css, per tenir un aspecte semblant.

3.3.1.2.- Resolució de pantalla

La mida de la pantalla i més concretament la resolució d'aquesta és un aspecte molt a tenir en compte a l'hora de realitzar el producte, ja que al tindre més o menys espai pot fer que una pantalla que a una resolució es veia correctament, al canviar-la no es vegi com seria desitjable. Per tant es va decidir que les resolucions suportades per Omega v4 serien les de 800x600, 1024x768 i 1280x1024, però donant prioritat a les dos més grans.

3.3.1.3.- Estructura de les pantalles

Com he dit amb anterioritat és molt important unificar criteris per la creació de les pantalles.

L'estructura principal de totes les pantalles és la Master-Detail.
L'estructura Master-Detail consta de dividir les pantalles en dos parts clarament diferenciables, per una banda la part Master, que és on normalment es troba una taula per mostra totes les dades provinents del servidor, així com un conjunt de botons per interaccionar amb les dades. Botons com ara afegir, esborrar, activar/desactivar, etc. El conjunt de botons varia depenent de la funcionalitat de la pantalla.

L'altre part és la zona de detall, en aquesta zona acostumen a estar tots els elements necessaris perquè l'usuari pugui utilitzar els botons de la zona Master. Per exemple, si un usuari vol afegir un registre nou, anirà a la zona de detall i seleccionarà els elements que desitja per guardar-los a la base de dades. La zona de detall també inclou els botons “acceptar” i “cancel·lar”.

Encara que l'estructura Master-Detail és àmpliament consensuada per la majoria de pantalles de l'Omega v4, hi ha unes poques pantalles que per raons de funcionalitat no era aconsellable realitzar-les amb aquesta estructura. Entre elles es troben algunes pantalles del Rule Engine, com la pantalla d'activació de regles manuals.

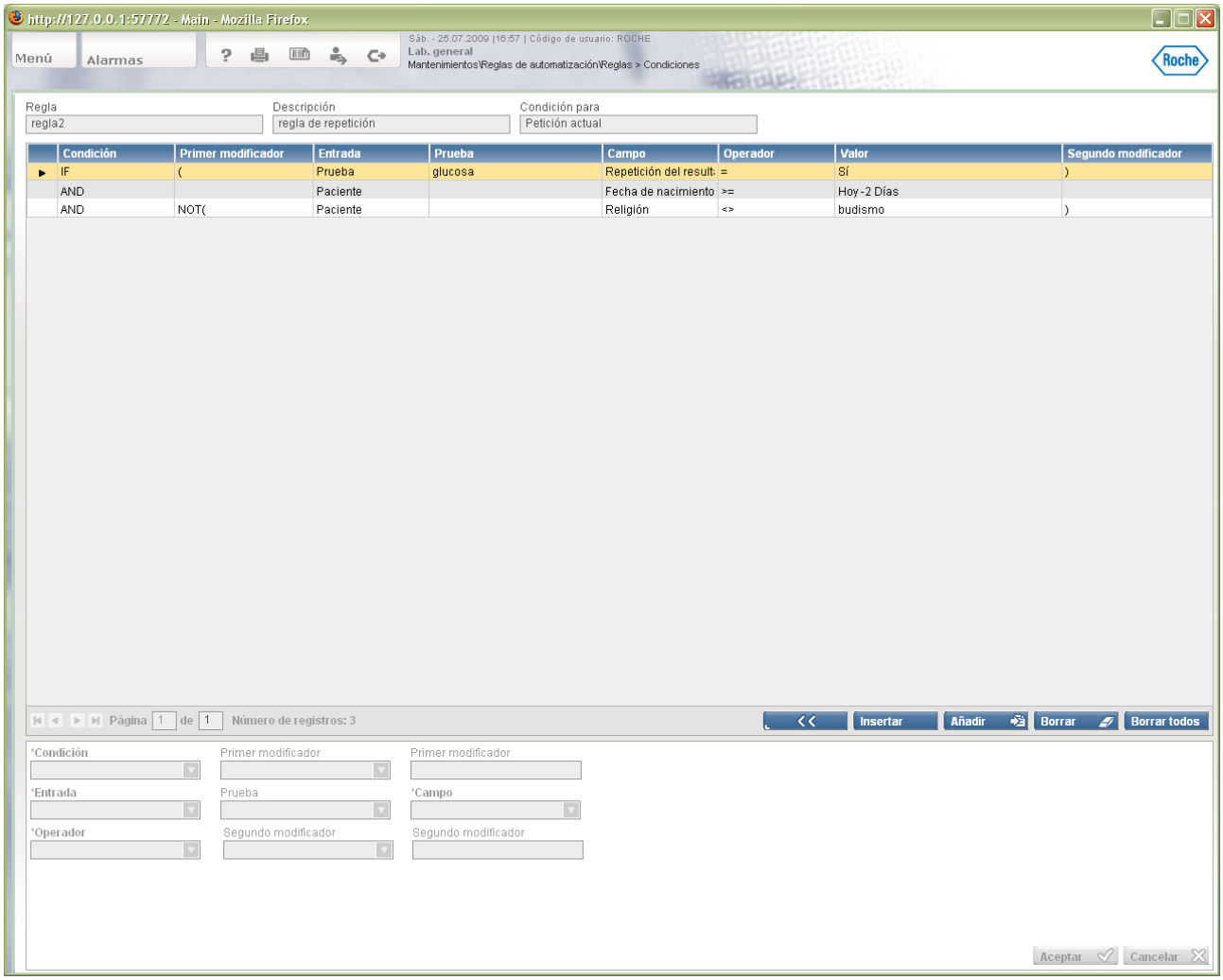


Figura 3.1. Pantalla de condicions. Pantalla de tipus Master-Detail

Figura 3.2. Pantalla d'activació de regles manuals. Pantalla sense Master-Detail

3.3.1.4.- Components de pantalla

Com es pot observar a les figures 3.1. i 3.2. els desenvolupador tenen a la seva disposició certs components per poder crear les pantalles d'una manera ràpida i eficient. Cadascun d'aquests components té propietats especifiques per a ell, però d'altres propietats d'aquests components són heretades d'altres components més genèrics.

Les principals propietats comunes són:

- hidden: perquè s'oculti o no el component.
- disabled: per habilitar o no el component.
- value: valor que tindrà el component.
- required: obligatori posar valor al component.
- width: per definir la longitud del component.

3.3.2.- Requeriments corporatius de desenvolupament

3.3.2.1.- Requeriments corporatius de desenvolupament (nomenclatura)

Roche també té una sèrie de polítiques en quant a les nomenclatures, ja sigui per les classes, els components o per les bases de dades.
L'objectiu és simplificar i aclarir a que ens estem referim en tot moment, i d'aquesta manera només fa falta veure el prefix dels noms per saber a quina part del producte fan referència.

Tipus	Prefixe
Taula	t
Component	c
Web Service	ws

A més d'aquesta nomenclatura també és fa servir una altre depenent al mòdul al qual pertany:

Mòdul	Prefixe
System	sys
ACB	acb
RuleEngine	rul
Reports Printing	rrp

Per tant d'aquesta manera totes les taules que fan referència al Rule Engine començaran pel nom “trul”, i tots els seus components per “crul”.

La resta del nom a d'especificar de manera clara l'àmbit d'acció de la informació que conte. Per exemple la taula on es guarden les definicions de les regles és diu trulRules.
Pel que fa les classes que formen les pàgines web, tenen la següent nomenclatura:

LISPages + Nom del mòdul + Nom de la pantalla

En el cas de les pantalles de Rule Engine un exemple de nomenclatura és :

LISPages.RuleEngine.Rules.cls

Els prefixes per l'àmbit de les variables és el següent:

Àmbit	Prefixe
Paràmetre	p
Global	g
Local	Sense prefixe

Prefixes pels tipus de variables:

Tipus	Prefixe
Integer	int
String	str
Boolean	bln
Object	obj
TimeStamp	ts
Array	arr

Hi ha gent que és pot preguntar perquè es important la nomenclatura i perquè Roche posa tantes restriccions en ella. La resposta és simple, per unificar el codi i fer-lo més accessible. Una cosa que s'ha de tenir en compte és que molts cops la persona que a de corregir un error o modificar part d'un codi no és la mateixa que la creat, per tant si aquesta persona sap les normes de la nomenclatura li serà molt més fàcil realitzar canvis en aquests codi.

3.3.2.2.- Requeriments corporatius de desenvolupament (Comentaris)

Els comentaris que apareixent en les pantalles són cosa del programador, així que es pot trobar pantalles molt comentades i d'altres on no s'explica casi res.
Però en el cas dels Web Services i dels components, hi ha unes normes a seguir:

- Nom de la funció: nom que té la funció.
- Descripció: que fa aquesta funció.
- Paràmetres: paràmetres que intervenen i breu descripció.
- Valor de retorn: quin tipus de valor retorna. (També s'han d'incloure els errors controlats).
- Versió: qui a creat i/o modificat la funció, la data i els canvis realitzats.

```
/**************************************************
*Function: GetFieldList
*Description: Component method that return the list of fields of trulField table
*Param: pintEntrance → trulEntrance table identifier
* Returns: Dataset object
*Version: 2009-02-10 (RedSauce OFC) New
*              2009-02-25 (RedSauce OFC)  Bug fixed for pintEntrance = 6 (always active)
/**************************************************
```

Aquesta informació es extremadament útil per saber que fa una funció i per saber quins canvis s'han realitzat recentment. La idea és que només llegint el comentari una persona que no havia vist amb anterioritat aquesta funció pugui saber perquè s'utilitza.

4.– Eines de desenvolupament

Per crear un producte és necessari dotar d'un conjunt d'eines als desenvolupadors. Començant per els llenguatges de programació amb els qual s'implementarà el producte, o els gestors d'arxius, per tal de poder tenir controlats els arxius.

Disposa d'aquestes eines pot ser indispensable, o molt aconsellable per tal de dur a terme el desenvolupament del producte.

4.1.- Perforce

En un projecte tant gran com és l'Omega v4 és indispensable una eina de gestió d'arxius.
Perforce és un repositori de gestió d'arxius multiusuari. Cada vegada que algú vol implementar un canvi en l'Omega v4, a de pujar-lo a Perforce, i gràcies a aquesta eina es poden tenir controlades les modificacions fetes sobre l'arxiu. Ja que en cada canvi queda registrada la persona que la realitzat, quan la realitzat i un comentari dels canvis que a aplicat.

A més un cop pujat el canvi, no es borra la versió anterior del document, sinó que es crea una nova conservant l'antiga. D'aquesta manera en cas que hi hagués un error, un usuari podria sincronitzar amb una versió anterior. O també mitjançant una aplicació de comparació que està inclosa en Perforce, veure amb claredat quines parts del codi han estat modificades en les diferents versions.

Perforce permet bloquejar arxius, de manera que si un usuari obre un arxiu per editar-lo ningun altre usuari podrà modificar aquest arxiu.

Totes les propietats avanç esmentades més el fet que Perforce funciona a la perfecció amb tot tipus de xarxes el converteix en una utilitat indispensable per poder dur a terme un projecte de gran envergadura en el que treballen tantes persones com és l'Omega v4.

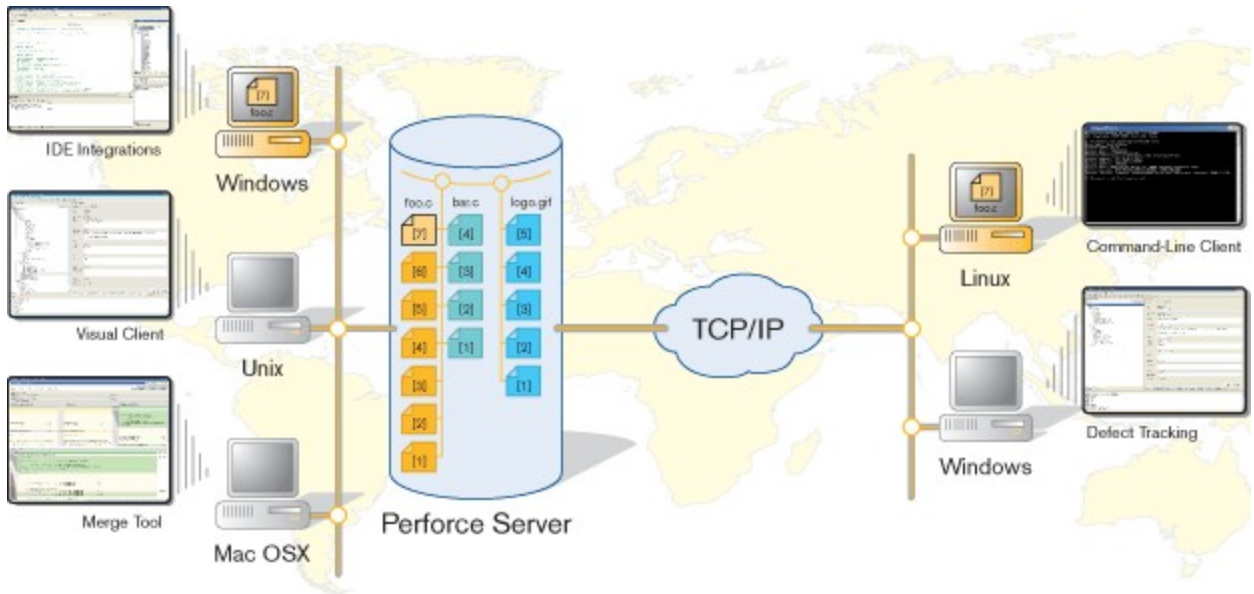


Figura 4.1 Esquema del Perforce

4.2.- ZEN

Quan al principi Roche estava investigant amb quines eines podria realitzar el seu futur projecte anomenat Omega v4, va rebre una oferta de l'empresa nord americana Intersystems per tal d'utilitzar ZEN. ZEN proporciona un conjunt d'eines que permeten als desenvolupadors crear d'una manera ràpida i fàcil aplicacions web complexes.

Utilitza una implementació ampliada de la tecnologia AJAX (Asynchronous JavaScript and XML), que permet l'ús de les capacitats de desenvolupament d'objectes d'Intersystems, i l' utilització de CSP (Cache Server Pages).
El seu funcionament es basa en components pre-creats, que s'encarreguen de crear tot el Javascript i el HTML associat. D'aquesta manera el desenvolupador només a de fer una crida al objecte, definint unes poques característiques i el ZEN s'encarregarà de crear tot el HTML i el Javascript necessari.

Avantatges d'utilitzar ZEN:

- Temps: al usar components pre-creats, es poden crear interfícies d'usuari d'una manera molt més ràpida.

- Integració amb base de dades Cache: ZEN està integrat amb la base de dades de Cache, la qual cosa permet que el client ZEN i el servidor de Cache s'intercanviïn informació.

- Ampliable: es poden aplicar css estàndard per modificar l'aparença de ZEN, així com sobreescriure alguna de les funcionalitats d'un component per tal de canviar algun dels seus comportaments.

- Independència del client: ZEN funciona sota IE6, IE7 i firefox2. Encara que pot haver-hi algunes diferencies de funcionament entre navegadors.

- Facilitat d'ús: les aplicacions realitzades amb ZEN són rapides de desenvolupar i fàcils de mantenir. Es degut a que ZEN utilitza un client estàndard de HTML.

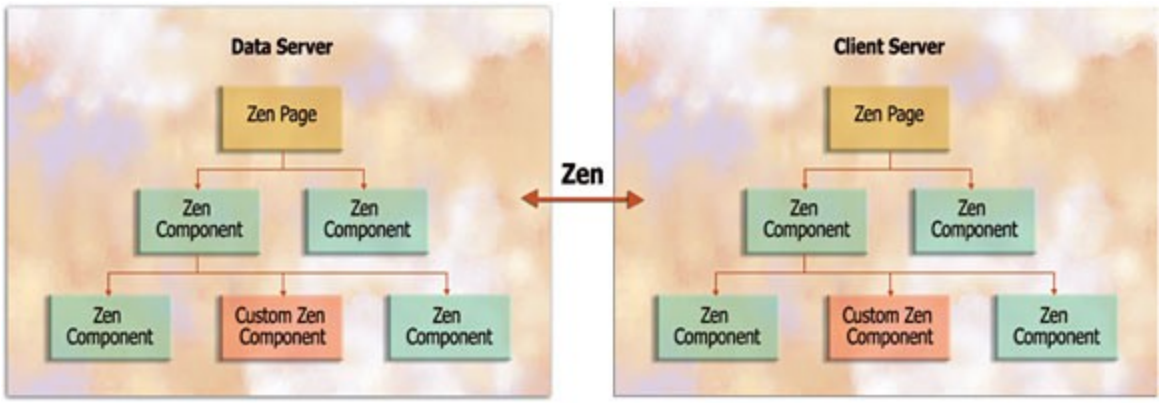


Figura 4.2. Exemple estructura pàgina de ZEN

Cada pàgina de ZEN, crea una instància del model d'objecte de la pàgina tant en client com en el servidor de dades. ZEN s'encarrega de anar sincronitzant els objectes de les dos pàgines, d'aquesta

manera s'aconsegueix un major rendiment i seguretat.

4.3.- LISComponents

Malgrat de les avantatges que ens proporciona ZEN, hi ha certes funcionalitats necessàries per a l'Omega v4 que ZEN no ens aporta. Per això a vegades s'encapsulen els components ZEN per dotar-los de noves funcionalitats. Aquests components ZEN amb noves funcionalitats són els anomenats LISComponents.

Els LISComponents són els objectes usats per definir les diferents pantalles que té l'Omega v4 entre elles les pantalles del Rule Engine.

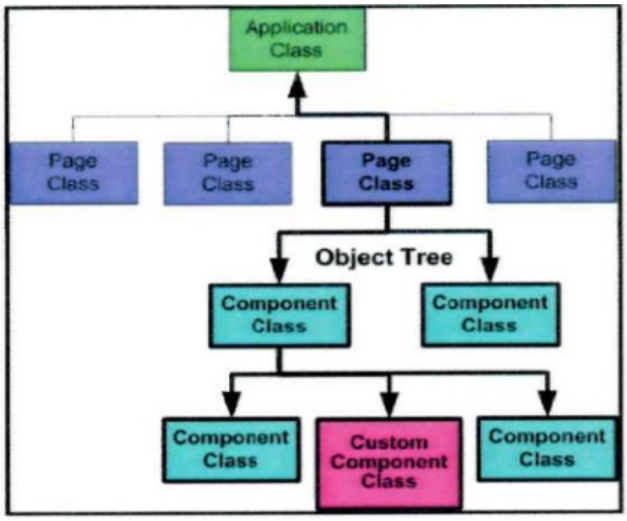


Figura 4.3. Estructura de classes (ZEN)

4.3.1.- FrameWork

Algunes de les noves funcionalitats necessàries no es poden encapsular dintre els components com es fa amb la resta de LISComponents. El que es fa és incloure la nova funcionalitat dins del Framework.

Aleshores el Framework és un grup de funcionalitats que no han pogut ser incloses dintre dels components.

Algunes d'aquestes funcionalitats són:

- Funcionalitats de paràmetres:

  - GetParameter
  - SetParameter

  Utilitzats per enviar i rebre paràmetres entre pàgines.

- Funcionalitats de tractament d'errors:

    - HasError
    - SetError
    - ClearError

- Funcionalitats extres:

    - ShowMessagePopUp

    Surt un pop-up amb informació prèviament definida per el desenvolupador.

    - ShowFileBrowserPopup

    S'obre un finestra on es mostren tots els arxius per un directori determinat. Permet escollir un arxiu i et retorna la ruta de l'arxiu.

4.3.2.- Objectes del LISComponents

Aquesta és la llista dels principals LISComponents amb les seves funcionalitats especials. Les funcionalitats heretades no s'inclouen ja que les tenen les majoria dels objectes. (Required, hidden, inputType, width, etc.).

LISTablePane: component usat per mostra les dades que es retornen del servidor. Acostuma a ser l'objecte principal de la part Master de les pantalles estàndard de tipus Master-Detail.

Les seves principals funcionalitats són:

onCreateResultSet: nom de la funció que enviarà les dades del servidor.
onDblClick: nom de la funció que s'executa al fer un doble click en un dels registres de la taula.
onselectedRow: nom de la funció que s'executa al fer un click simple sobre un dels registres de la taula.
useSnapshot: si el valor és true (false per defecte), les dades obtingudes amb la crida del OnCreateResultSet són guardades en el servidor de Cache. Permet reordena sense torna a demanar les dades del servidor.

Crida: <lis:LISTablePane>

LISColum: component usat per definir una columna. Totes les crides de LISColum han d'estar dintre de la definició d'un tablePane.

Les seves principals funcionalitats són:

header: defineix el nom de la capçalera de la columna.
colName: per especificar el nom del element amb el que es vol relacionar la columna i el retorn de les dades del servidor.
onDrawCell: nom de la funció que utilitzant HTML permet modificar la imatge mostrada per la

columna.
filterType: tipus de filtre que tindra la columna.

Crida: <lis:LISColum >

LISText: component usat per habilitar l'entrada de text per part de l'usuari.

Les seves principals funcionalitats són:

forbiddenChars: nom del mètode que indica quins són els caràcters permesos per mostra en el component.
maxlength: número màxim de caràcters que es poden introduir.

Crida: <lis:LISText>

LISDataCombo: un dels components més usats. Mostra una llista de valors prèviament obtinguda mitjançant una crida a servidor. Consta de dos valors, el camp value i el camp text.

auxColumn: per escollir quines de les columnes retornades pel servidor es vol utilitzar com auxiliar.
dropDownWidth: defineix l'amplada que tindrà el combo al desplegar-se.
dropDownHeigth: defineix l'altura que tindrà el combo al desplegar-se.
choiceColumn: la columna que es mostrarà al escollir un valor.
displayColumn: el número de columna que és mostrarà al desplegar el combo.
valueColumn: el número de columna del qual s'escollirà el valor.
editable: per poder escriure dintre el dataCombo (false per defecte).

Crida: <lis:LISDataCombo>

LISTextArea:

rows: número de files que tindrà l' àrea de text.
maxlength: número màxim de caràcters que es poden introduir.

Crida: <lis:LISTextArea>

LISFieldSet: component pensat per agrupar d'altres components. D'aquesta manera si apliques alguna propietat al fieldset, per exemple disabled = 1, es deshabilitaran tots els components que es trobin dintre del fieldset.

layout: defineix com es col·locaran els diferents elements del grup. Els possibles valors són vertical i horitzontal.
legend: petit text que mostra la capçalera.
showBar: mostra o no una barra.

Crida: <lis:LISFieldSet>

LIS:LISCheckBox: crida per crear checkbox.

onClick: nom del mètode que es cridarà al prémer el checkbox.
onChange: nom del mètode que es crida al canviar el valor del checkBox.

Crida: <lis:LISCheckBox>

LISSpacer: component usat per deixar separació entre els altre components.
Usa les comandes estàndard de width, height.

Crida: <lis:LISSpacer>

4.4.- Cache

Cache és l'eina escollida per Roche Diagnostics per implementar l' Omega v4. Al igual que el ZEN pertany a l'empresa nord americana Intersystems.
Encara que no es tant coneguda com altres aplicacions similars com poden ser Oracle o qualsevol altre base de dades relacional, si que a aconseguit fer-se un forat important en certs camps com ara la medicina, gràcies a la seva capacitat per gestionar un gran volum d'informació de forma eficient.

Intersystems Cache utilitza una base de dades post-relacional que tracte les dades com objectes continguts dintre d'arrays multidimensionals, la qual cosa evita la creació directa de les relacions entre les taules. D'aquesta manera Cache no a de reensamblar les dades de les diferents taules, i el resultat és una execució considerablement més ràpida.

Un servidor d'aplicacions de Cache té tres nivells diferents, que són el servidor de Cache ObjectScript (Mumps), el servidor d'objectes i el servidor web.

Cache ObjectScript és un potent i fàcil llenguatge de programació orientat a objectes. Està especialment dissenyat per crear de manera eficient aplicacions a les bases de dades.

Les característiques més importants són:

- Tractament de dades: es poden guarda i consultar les dades utilitzant codi SQL o directament com si d'una variable es tractes ja que té la característica que permet barrejar els mètodes d'accés a dades. Els desenvolupadors poden veure les dades com a objectes, com relacions a taules (usant SQL) o com a matrius multidimensionals.

- Integració SQL i HTML: dintre del codi de Cache object script es pot crida a codi HTML i SQL.

- Rutines: els mètodes i les definicions de les classes es guarden en arxius de dades global, i posteriorment el compilador, les transformes en una o varies rutines. Per tant podem dir que el codi és un conjunt de rutines.

El servidor d'objectes del que disposa Cache té la avantatge que a més de suporta les relacions implícites (claus primàries, etc.) també suporta les relacions explicites entre objectes. La qual cosa el fa més escalable que les bases de dades relacionals.

Funcionament detallat de Cache:

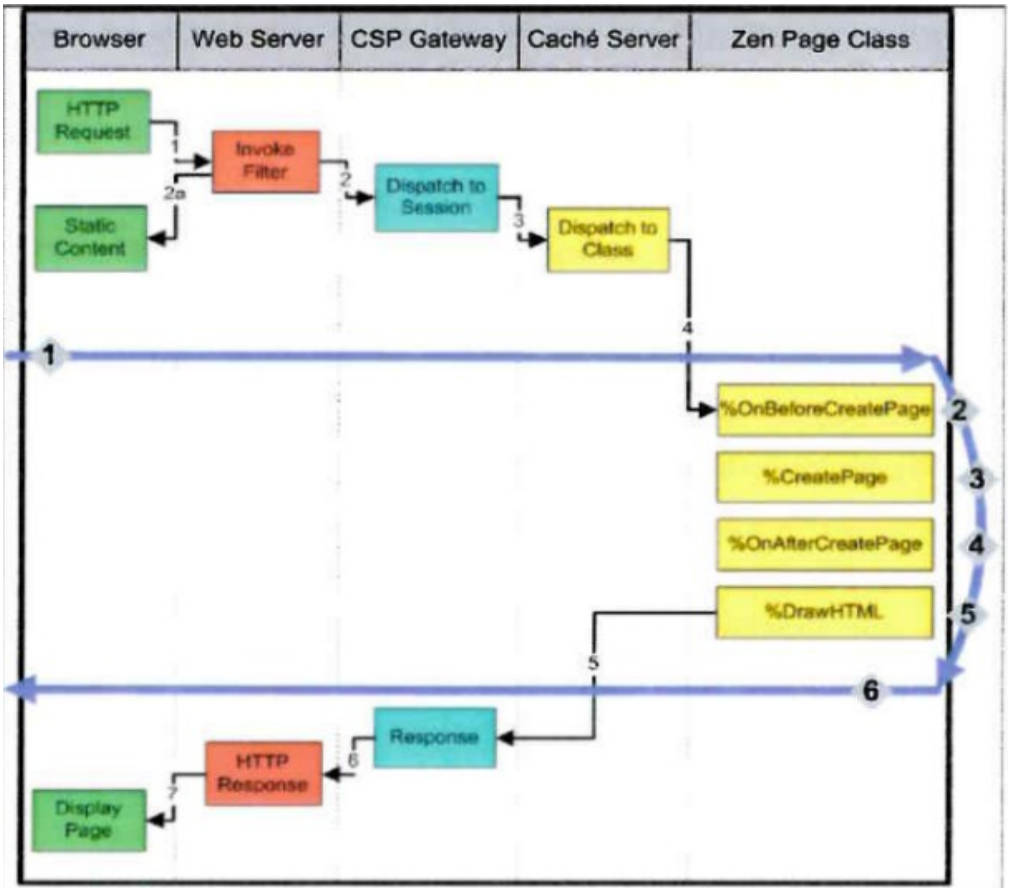


Figura 4.4 Funcionament de Cache

Passos que segueix Cache:

1. El navegador fa una petició.
2. Quan el servidor web (del client) rep una petició l'envia el CSP Gateway.
3. La petició s'envia al servidor de cache.
4. El servidor al rebre el missatge determina que vol el client i executa la següent seqüència de passos:
    4.2. Crida el mètode %OnBeforeCreatePage
    4.3. Es crea una instància de la pàgina i es crida la funció %CreatePage. %CreatePage és l'encarregada de crear els components de la pàgina i afegir-los en l'estructura.
    4.4. Es crida el metode %OnAfterCreatePage, el qual pot modificar certs punt del mètode %CreatePage.
    4.5. Es crida la funció %DrawHTML. Cadascun dels components té la seva pròpia funció %DrawHTML.
    4.6. S'envia el codi HTML al client.
5. El CSP Gateway rep la resposta del servidor.
6. El CSP Gateway envia el codi generat al servidor web.
7. El servidor web envia la resposta al navegador i aquest la visualitza.

5. Metodologia de desenvolupament (Model en V)

Apart de les eines per desenvolupar un producte, també és molt important la metodologia utilitzada. Roche Diagnostics ha estat usant durant molt anys la metodologia de desenvolupament que s'anomena model en V. El qual a demostrat ser altament eficient per a projectes de gran envergadura.

L'objectiu principal del Model V es aconseguir una producció de software de la màxima qualitat. El Model V es un procés iteratiu dividit en quatre fases principals i deu subfases.

- Fase d'anàlisi: La idea bàsica del projecte. Pregunta que és necessita i de quina manera.
    - Inicialització: Determinar objectius.
    - Requeriments: Parlar amb el client per saber que vol exactament.
- Fase de viabilitat: S'estudia si el projecte és viable.
    - Anàlisis: S'analitza si el projecte es pot realitzar. Es comença la documentació del projecte.
- Fase de de implementació: Es comença a implementar el projecte.
    - Disseny: Es diu com a d'actuar internament el producte.
    - Programació: S'escriu el codi del producte.
    - Testeig: Seguint la documentació prèviament definida, es comprova si el producte té el mateix comportament. S'actualitza la documentació tècnica.
    - Prova d'integració: Es crea una versió alpha. Si es necessari es modifiquen els Use Cases.
    - Prova de sistema: Es prova el sistema. S'obté la versió beta.
- Fase de finalització: Es donen els últims retocs abans de llençar el programa.
    - Avaluació / Acceptació: Es documenta la validació realitzada.
    - Versió de llançament: El programa ja es pot posar en el mercat.

L'Omega v4 i més concretament el Rule Engine han seguit la metodologia del Model en V. Per el cas del Rule Engine es una metodologia excel·lent ja que lo primordial és la qualitat del producte, i en el cas del Rule Engine, que es dedica a l'àmbit sanitari, hi ha certs errors que no es poden permetre. Per tant s'ha de garantir una qualitat elevada del producte.

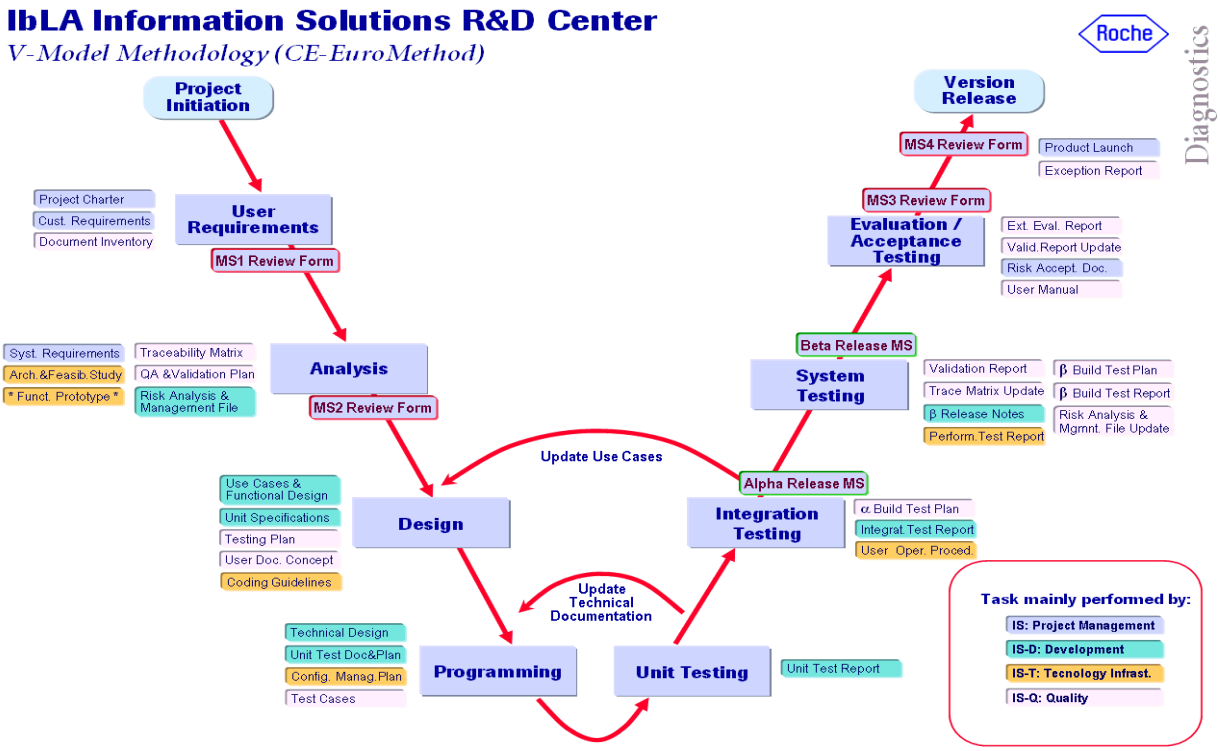


Figura 5 Model en V

6. Rule Engine

L' Omega v4 es un producte que s'encarrega de gestionar laboratoris d'anàlisis clínica de tot tipus, ja siguin grans o petits, o tinguin moltes o poques màquines d'anàlisis clínica.

L'Omega es un programa molt gran que es divideix aproximadament en uns 20 submòduls, cadascun d'aquest submòduls s'encarrega de realitzar unes tasques determinades dintre del laboratori.

Per exemple el mòdul de General Master Files s'encarrega dels manteniments, des de definir les proves, el tubs o les mostres que posteriorment s'utilitzaran. El mòdul de PatientHandling s'encarrega de controlar quan es dona d'alta a un pacient i del seu manteniment.

El Rule Engine és el mòdul encarregat d'executar unes accions prèviament definides en cas que s'obtingui el resultat desitjat. Dit d'una altre manera, el Rule Engine és l'encarregat de la creació, la gestió i l'execució de les regles.

Un exemple de regla seria:

Regla 1 = Al afegir proves en la pantalla de registre de peticions
if (Prova – Glucosa - valor resultat = 10)
llavors Afegir comentari - “Valor glucosa correcte”

Aquesta regla el que fa, és quan s'afegeix una prova en una pantalla determinada de l' Omega v4, comprova que existeixi aquesta prova i que tingui valor = 10. En cas afirmatiu afegirà un comentari a la petició que contingui aquesta prova.

6.1 Regles

Si el Rule Engine és un sistema basat en regles per controlar certs aspectes de l'Omega v4, el primer que ens hem de preguntar és que es una regla.

De manera genèrica, una regla es pot considerar com un conjunt d'una o varies accions amb una o varies condicions i els seus modes d'activació corresponents.

Les regles només poden ser definides per un usuari amb certs privilegis, normalment usuaris administradors o similars, això es degut a que les accions definides dintre de les regles poden realitzar canvis importants en les dades del laboratori, i a vegades irreversibles.

Al entrar dintre del menú principal, en la pestanya de manteniments, entre d'altres es pot trobar la pantalla de definició de regles. En aquesta pantalla l'usuari podrà introduir tots els camps necessaris per definir les diferents parts de les quals constà una regla.
Primerament l'usuari ha d'introduir camps com el nom de la regla i la seva descripció, que si ve no afecten al funcion de la regla serveixen per identificar-la posteriorment. Altres camps com l'aplicació si que poden afectar de manera significativa el resultat d'executar una regla, ja que limiten el seu àmbit d'aplicació. Per exemple si tens una petició de laboratori general i una altre d'urgències, que compleixen les condicions d'una regla, aquesta només avaluarà la petició que

pertany a l'aplicació per la qual a estat definida.

Dintre de la pantalla de regles l'usuari podrà accedir a les pantalles de definició d'accions i/o condicions.

Parts d'una regla:

- Nom: Nom de la regla.
- Descripció: Breu descripció de la regla.
- Aplicació: Aplicació on la regla s'executarà (Laboratori general, urgències etc).
- Tipus: A quins elements afecta la regla. (Actualment només a les peticions).
- Estat: Regla activada o desactivada.
- Ordenació: Número per ordena les regles al mostrar-les per pantalla.
- Modes d'activació: Una regla pot tindre varis modes d'activació, però com a mínim a de tenir-ne un.
- Acció: Una regla a de tenir una o més accions.
- Condició: Una regla a de tenir una o més condicions.

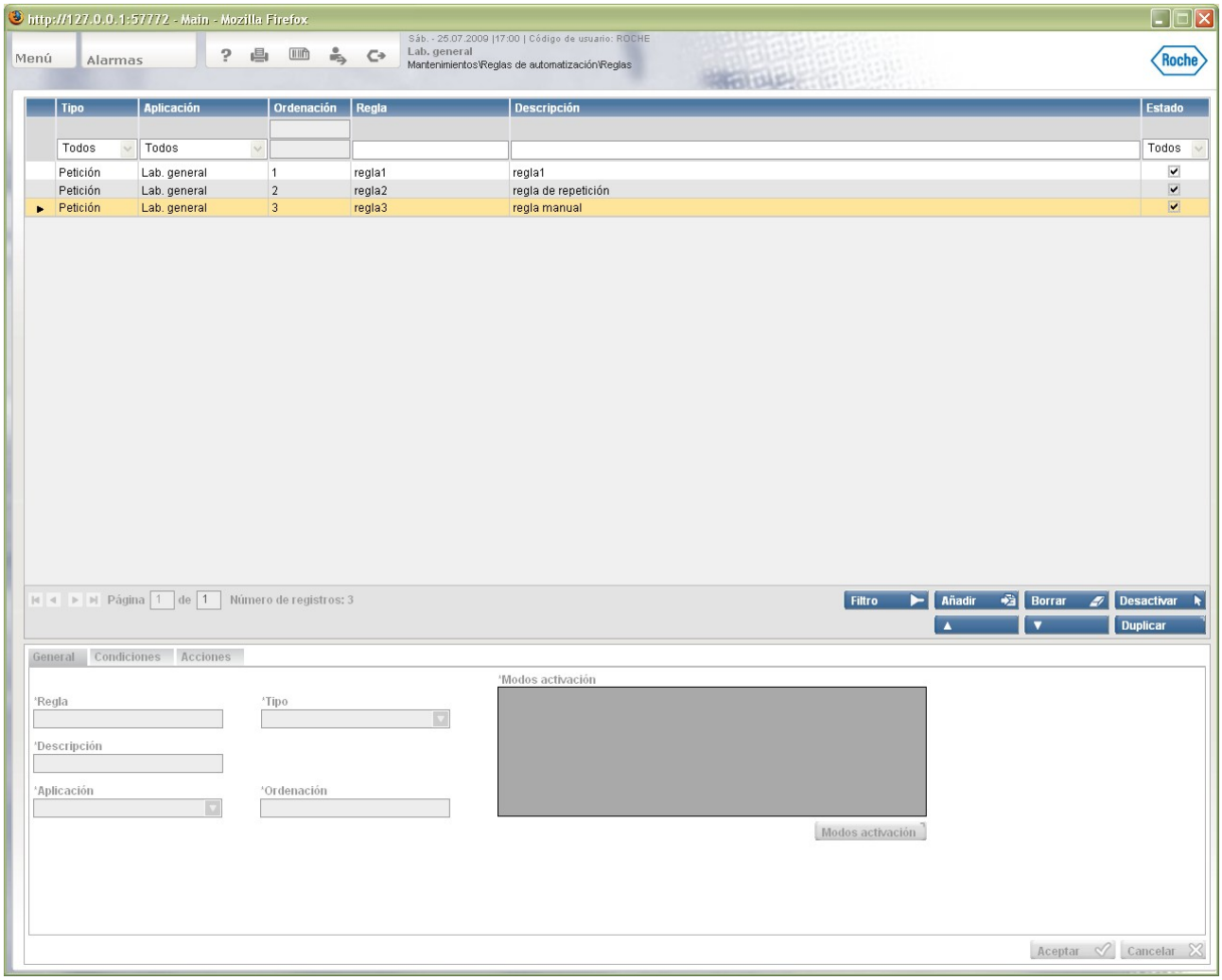


FIGURA 6.1 Pantalla de creació de les regles.

6.1.1 – Modes d'activació

Els diferents modes d'activació que és poden escollir al definir una regla determinaran quan serà 'disparada aquesta regla'. De modes d'activació hi ha molts diferents, com ara al entrar una petició, al validar un resultat, etc. Però bàsicament és poden dividir en dos grans grups:

- Manuals:
    - Manual: L'usuari serà l'encarregat d'executar la regla. Haurà d'anar a la pantalla de control de regles i executar-la.
    - Un cop al dia: Aquesta regles seran executades un cop al dia per un programa que s'executa a una hora determinada. Aquest programa funciona en background, per la qual cosa l'usuari no en notarà l'execució.
- Automàtiques:
    - Dintre els modes d'activació automàtics, es divideixin en varis subgrups, com al afegir una prova, al registrar resultat o al afegir un demogràfic.
      Cadascun d'aquest subgrups té a la seva vegada varis modes d'activació.

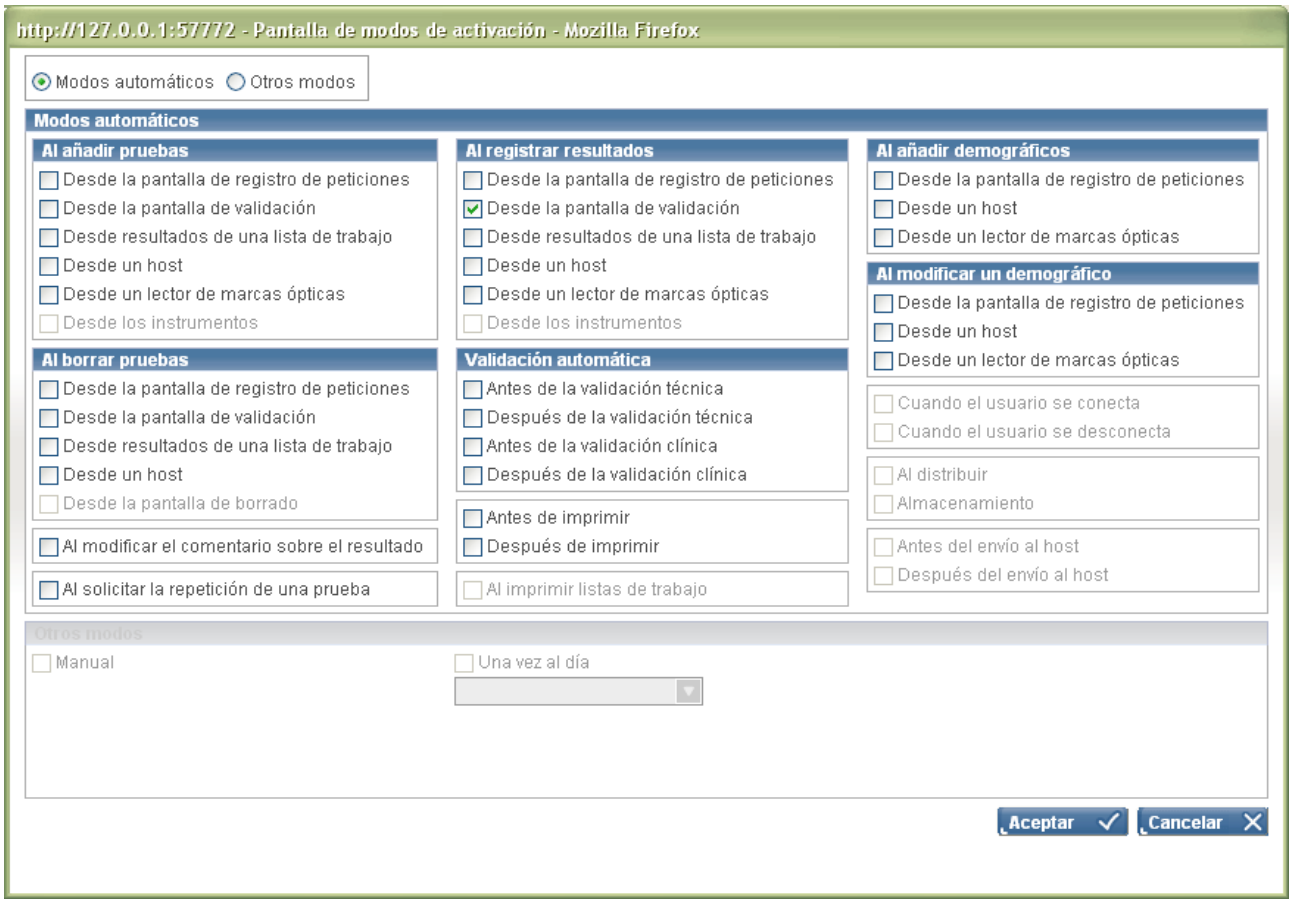


FIGURA 6.2 Pantalla de modes d'activació

L'usuari podrà escollir més d'un mode d'activació, sempre i quan tots ells pertanyin al mateix grup.

Si un usuari escull un mode d'activació automàtic no podrà escollir també el mode d'activació manual i viceversa.

Com he comentat anteriorment , els modes d'activació són els que diuen quan s'ha d'executar una regla i quan no. Això es fa posant diferents crides a la funció d'execució del Rule Engine repartides entre els diferents mòduls de l'Omega v4. Cadascuna d'aquestes crides té un identificador propi.

Llavors el que fa el Rule Engine és buscar si hi ha alguna regla activa, que tingui aquest identificador en els seus modes d'activació. Si en trobar alguna l'executarà i sinó passarà de llarg i continuarà amb l'execució de codi del mòdul on es troba.

Per tant, al escollir els modes d'activació, el que es fa es dir-li a la regla quin són els identificadors, de les crides als

Òbviament, les crides a la funció d'execució del Rule Engine només apliquen a les regles automàtiques, ja que les manuals les ha d'executar expressament l'usuari.

A les regles d'execució de tipus manual, més concretament les d'un cop al dia, l'usuari pot escollir a quina hora vol que s'executin les regles. Per norma general s'acostuma a escollir una hora amb poc volum de feina per evitar sobrecarrega l'aplicació.

6.1.2 - Accions

Un dels objectius del Rule Engine es dona la màxima llibertat possible a l'usuari per crear tots els tipus d'accions que vulgui. Això a la practica fa que sigui un gestor de regles molt potent, però a la vegada també la persona que crear les regles ha de tenir un mínim de coneixements, ja que es poden definir accions que entrin en conflicte amb els modes d'activació. Per exemple
Al crear una regla l'usuari a de definir almenys una acció.

Les principals parts d'una acció són:

- Mode : Quan s'ha d'executar l'acció

    - Llavors: s'executa si es compleix la condició.
    - Sinó: s'executa quan no es compleix la condició.

    No són obligatori cap dels dos, però almenys a d'estar un dels modes.

- Acció: Descriu que farà la regla.

    - Esborrar
    - Afegir
    - Repetir

- Validar
- Afegir resultat
- Imprimir
- Tancar petició

- Objectiu: L'element sobre el qual serà realitzada l'acció. El valor d'aquest camp varia segons l'acció escollida, però normalment són petició, prova o pacient.

- Selecció: Serveix per especificar el valor de l'acció – objectiu. Per exemple quan Acció = Afegir i objectiu = prova, es pot escollir quina prova afegir, entre totes les definides en el sistema.
  Aquest camp s'activarà depenent dels valors acció – objectiu, sempre que estigui actiu serà obligatori escollir-ne un valor.

- Tipus d'execució:
  - Única: Només s'executa l'acció una vegada.
  - Repetida: Pot executar l'acció més d'una vegada.

La pantalla d'accions és altament dinàmica, depenent dels valors principals escollits, aniran apareixen un conjunt de camps per acabar d'especificar l'acció.
Per exemple, si l'usuari introdueix l'acció Afegir – Petició, es crearan un conjunt de camps com Aplicació, Número de seqüència inicial, Número de seqüència final i Comentari en la nova petició.

Per cada acció definida, el Rule Engine té una funció que aplica els canvis sol·licitats en l'acció. D'aquesta manera quan el Rule Engine llegeix l'acció definida per l'usuari, i decideix quina és la funció que ha d'executar per tal de dur a terme l'acció desitjada.

Com que les accions poden modificar dades importants, és molt necessari que per cada acció realitzada quedi constància de que a fer l'acció, quan i quin usuari la executat. D'aquesta manera es pot saber de manera immediata que a passat i qui és el responsable.

L'usuari per defecte del Rule Engine és el “SYSTEM_RULE_ENGINE”. Però en el cas de les regles manuals serà qui l'hagi executat.

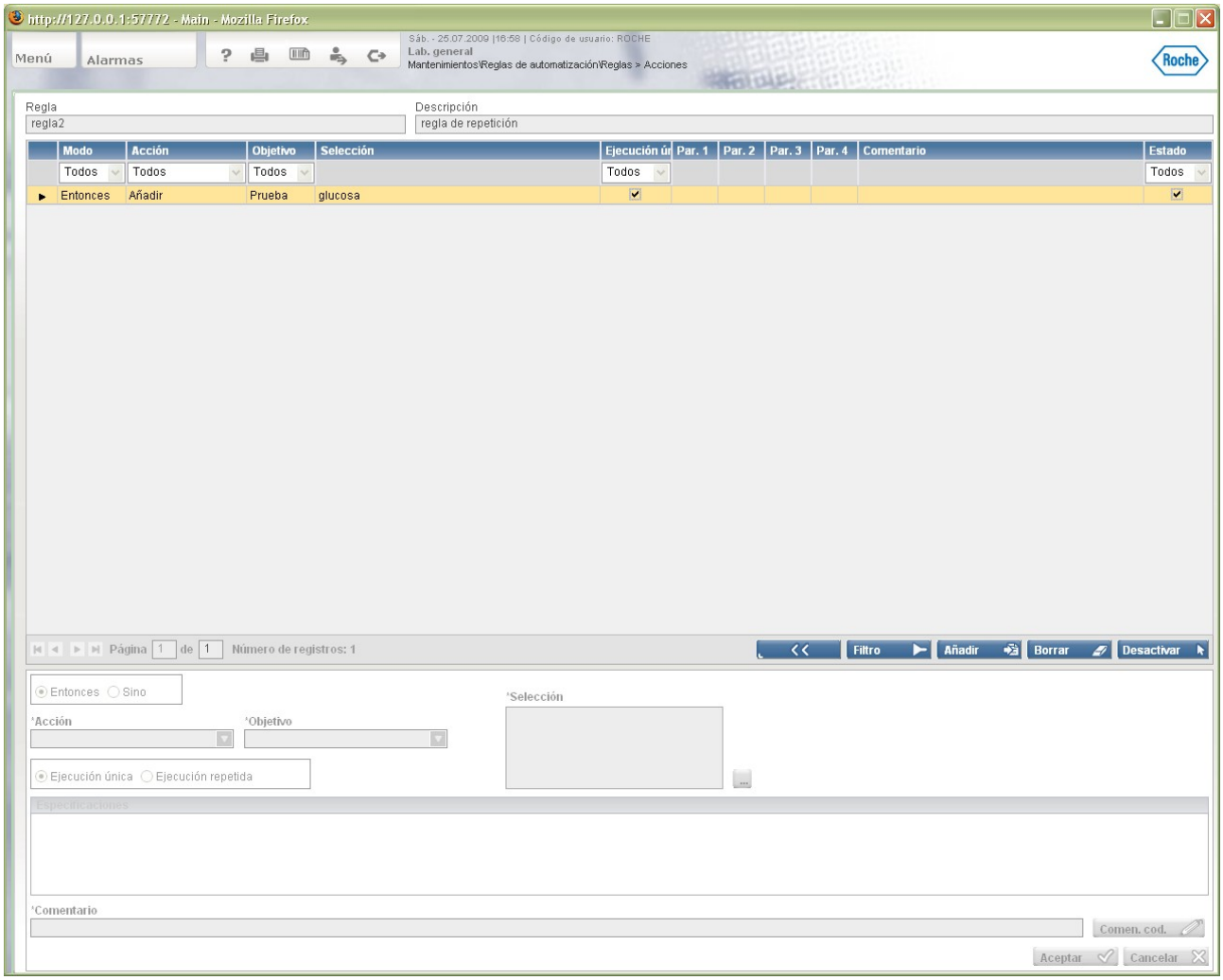


Figura 6.3 Pantalla de definició d'accions.

6.1.3 - Condicions

Les condicions definides dintre d'una regla són les encarregades de dir si l'acció associada a aquesta regla s'ha d'executar o no.

Al entrar en la pantalla de condicions l'usuari a de definir una o més condicions. Encara que l'usuari introdueixi varies condicions s'ha de tenir present que cada línia de condició s'avalua de manera individual, i el resultat d'una condició no afectarà a les altres.

Al final el conjunt de condicions retornarà un resultat que dependrà dels resultats de cada condició per separat i dels operadors binaris. Els possibles resultats d'avaluar un o més condicions són 0 o 1.

També hi ha un valor intermedi, el 2, que significa que la condició a avortat l'avaluació. Això es pot donar quan li falten valors als camps que a d'avaluar. En aquest cas surt immediatament de l'avaluació de condicions i passa a avaluar la següent (en cas que hi hagi).

Quan totes les línies de condicions han estat avaluades es realitzen les operacions binaries. Prèviament es substitueixen els avortaments (2) per 0 per evitar errors.

Un exemple seria un conjunt de tres condicions:

1 AND (1 OR 0)

En cas que les condicions retornin un 1 l'acció associada a la regla s'executarà, en cas de retorna un 0 no es realitzarà cap acció.

Una condició consta de les següents parts:

- Condició:
    - IF : (només la primera condició).
    - AND: Operador aritmètic x*y. (de la segona condició cap endavant).
    - OR : Operador aritmètic x+y. (de la segona condició cap endavant).
- Primer modificador:
    - “NOT(“ : Inverteix el resultat obtingut per la condició.
    - “(“ : parèntesis d'obertura.
    - “C” : Elimina l'anterior paràmetre.

    No es obligatori posar el camp primer modificador.
- Entrada: A quin tipus de dades fa referència.
    - Petició : mostrarà els camps relacionats amb les peticions.
    - Prova: mostrarà els camps relacionats amb les proves.
    - Pacient: mostrarà els camps relacionats amb els pacients.
    - Sempre cert: es desactivaran tots els demés camps.
- Prova: Només s'activarà quan el valor d'entrada sigui = prova
    - Totes: S'ha de complir en totes les proves de la petició.
    - Alguna: S'ha de complir en almenys una prova de la petició.
    - Prova especifica: l'usuari escull la prova que vol entre les definides en els manteniments.
- Camp: Els valors mostrat en aquest combo canviaran depenent del valor escollit en Entrada. Per exemple, si s'ha escollit Entrada = pacient, en el Camp apareixeran demogràfics de pacient, com poden ser sexe, nom, religió etc.

- Operador: “=”, “<>”, “>=”,”<=”, existeix, pertany. Els operador que apareixen aniran variant depenent del Camp escollit.

  Un cop escollit l'operador es mostraran entre 1 i 4 objectes que estaven ocults, i que ajudaran a acabar d'implementar la condició.

  Si un usuari ha escollit pacient – nom, li apareixerà una àrea de text que li permetrà escriure el nom que vulgui. En canvi si escull petició – data de registre, li apareix un calendari per escollir la data que vulgui.

- Segon modificador:

  - “)“ : parèntesis de clausura.
  - “C”: eliminar l'anterior paràmetre entrat.

  No es obligatori.

La construcció de les condicions és un procés complexa, on intervenen molts elements diferents. Té l' avantatge que són altament configurables per part del usuari, però aquesta llibertat que se li dona a l'usuari, també es pot torna contra ell, ja que pot realitzar condicions o conjunts de condicions que no tenen massa sentit.

A l'hora de construir una condició, els camps més importants són l'entrada, el camp i l'operador. Al escollir una o altre entrada s'escull quin dels principals fluxos de construcció es farà servir, d'aquesta manera hi ha quatre fluxos de construcció principals (petició, prova, pacient i sempre cert). El valor del “camp” diu on s'han de buscar les dades hi ha quines taules s'han d'anar a buscar. L'operador filtre els resultats indesitjats.

D'aquesta manera una condició com:

if (pacient – Religió = budisme)

acaba generant un sql com:

```
SELECT rOrders->rPatients->Religion
FROM tOrderTests
WHERE rTests->MicroType = 0
AND rOrders = 200011
AND rOrders->rPatients->Religion = 80      (80 és el codi que té el budisme)
```

Si la consulta retorna algun valor la condició retornarà un 1, per contra si retorna buit la condició retornarà 0.

Com he dit abans, hi ha quatre fluxos principals per la construcció de condicions, però hi ha un conjunt reduït de condicions que no poden ser avaluades per aquests fluxos. Cadascuna d'aquestes condicions rep un tractament especial al ser avaluada.

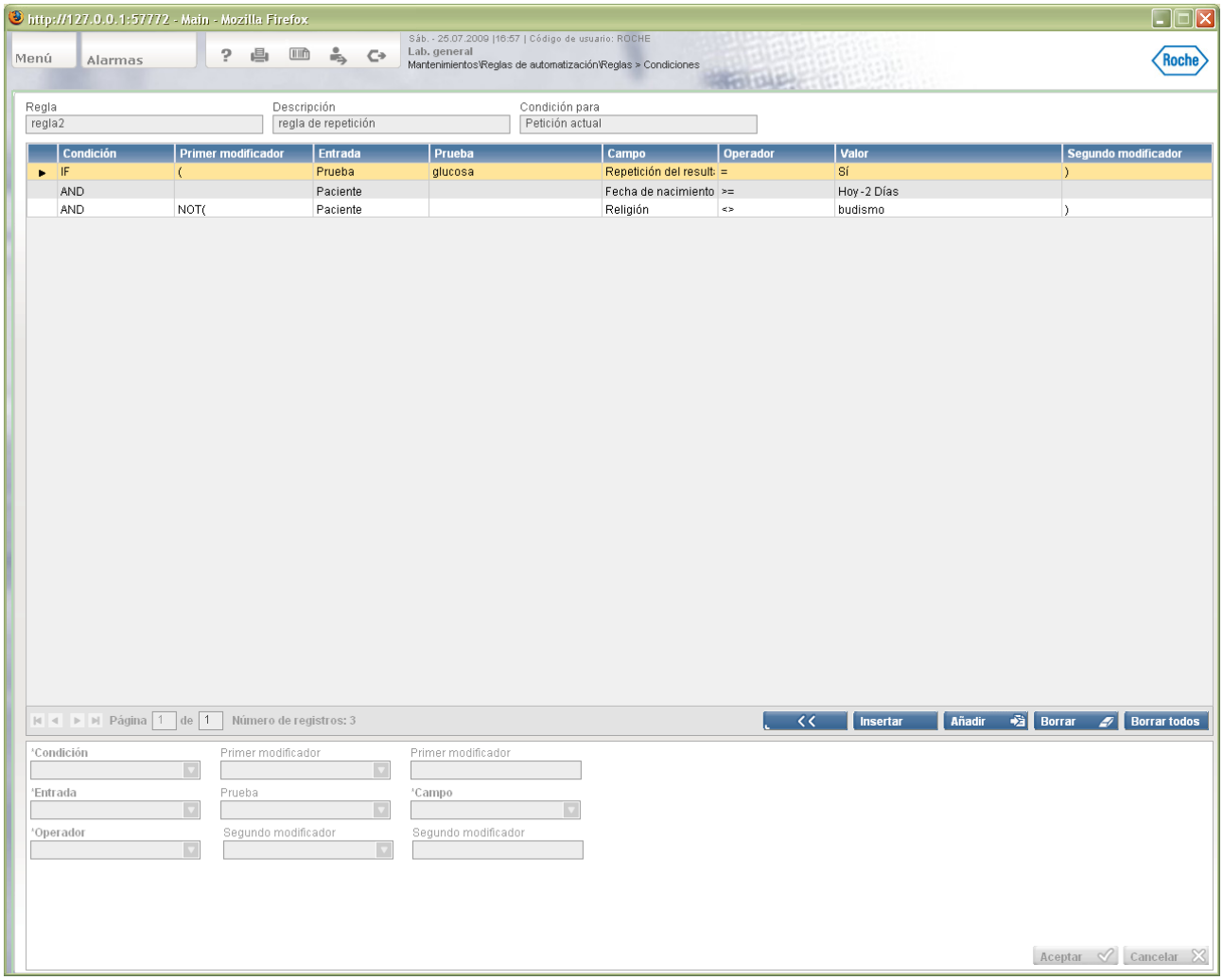


FIGURA 6.4 Pantalla de condicions

6.2. - Rules Control

La pantalla de Rules Control és on es poden veure totes les regles definides del Rule Engine. Es pot visualitzar el nom de la regla i l'estat en el que es troba.

Es una pantalla molt útil perquè es pot visualitzar des de quan està en funcionament una determinada regla hi ha quants elements a afectat. A més és des de l'únic lloc on es pot activar una regla manual.

Els camps que es poden visualitzar són:

- Nom de la regla: serveix per identificar-la, es tracta del nom entrat en la pantalla de definició de regles.

- Rang de peticions: Automàtic o manual.

- Procés:

    - Preestablert (en el cas de ser una regla automàtica).
    - Manual
    - Un cop al dia: regla manual que s'executa a l'hora prèviament definida.

- Data inicial: data de quan es va activar per últim cop la regla.

- Data final: Aquest camp estarà buit quan la regla es trobi en execució. Quan estigui parada mostrarà quan es va desactivar la regla.

- Accions realitzades / Peticions afectades / Peticions avaluades: Aquesta columna mostra la feina realitzada per la regla des de que es va activar per última vegada.

    - Accions realitzades: mostra el número d'accions que a executat la regla.
    - Peticions afectades: el número de peticions que han sofert algun canvi causat per la regla.
    - Peticions avaluades: el número de peticions que la regla a avaluat. Independentment del resultat obtingut per les condicions de la regla.

    Aquest camp es reinicia al activar la regla. Posant els comptadors a 0.

- Mode:

    - Treballant : regla activa.
    - Stop: regla parada.

- Estat:

    - Actiu : la regla està activa.
    - Desactivada: la regla està desactivada.

    L'estat de la regla només es pot modificar en la pantalla de definició de regles.

Funcionalitats de la pantalla:

- Informe: mostra les accions realitzades per la regla.

- Llençar regla: Si la regla és automàtica l'activarà. En cas de ser de tipus manual s'accedirà a la pantalla d'activació de regles.

- Parar: Desactiva una regla. Aquest botó només aplica en el cas de les regles automàtiques, ja que les regles manual es desactiven automàticament un cop acabada l'execució.

- Parar tot: Deté l'execució de totes les regles actives del sistema.

- Filtre: activar /desactivar el conjunt de filtres de la pantalla.

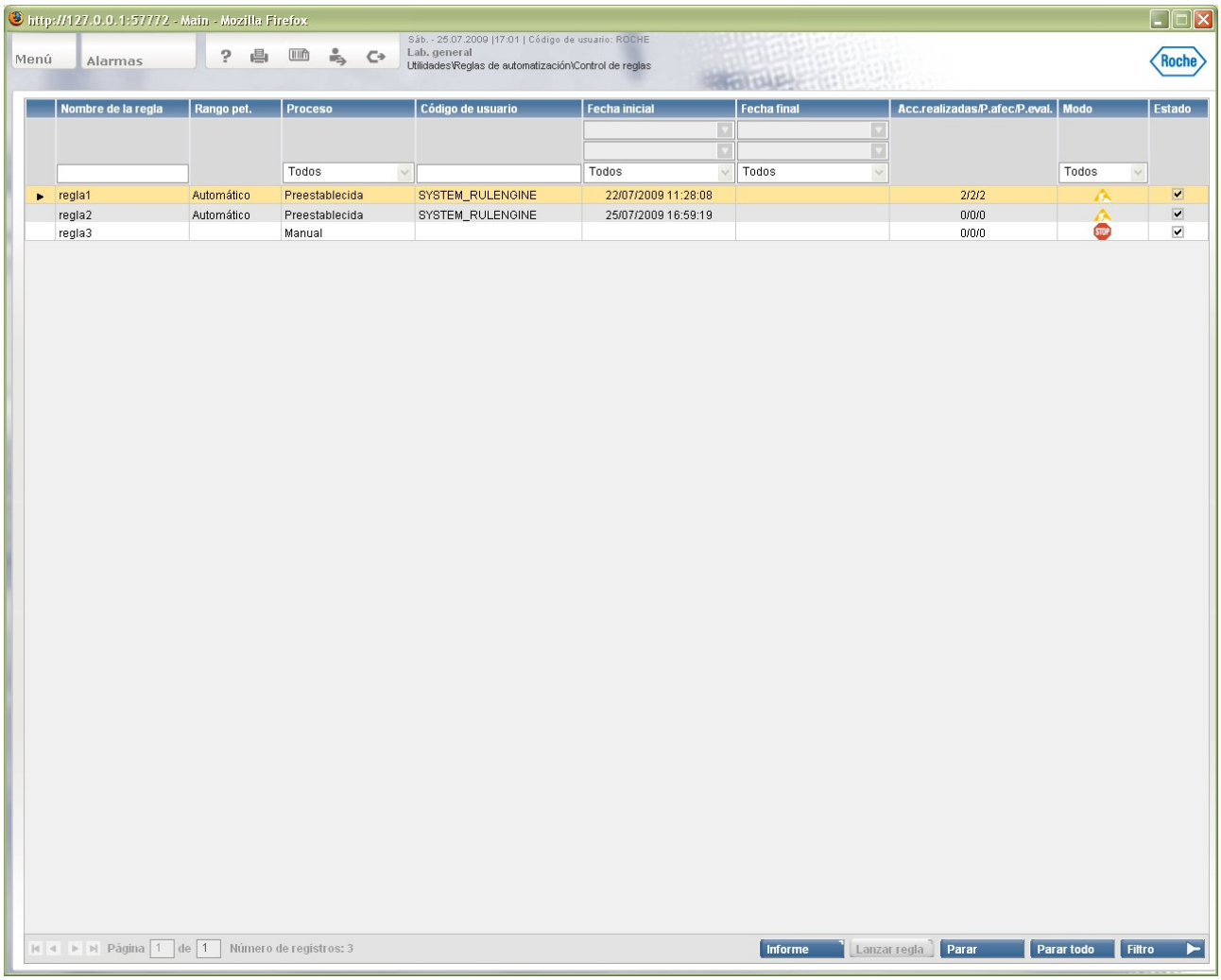


Figura 6.5 Pantalla de Rules Control

6.2.1 – Activació de regles

Dintre de la pantalla de Control de regles, quan es llençar una regla manual, s'accedeix a la pantalla d'activació de regles.

Dintre d'aquesta pantalla es pot definir el rang de peticions sobre el qual s'executarà la regla, així com el rang de dates de quan es van definir les peticions.

Aquests paràmetres es fan servir per obtenir la llista de les peticions sobre les quals s'executarà la regla.

A diferencia de les regles automàtiques que sempre treballen amb peticions obertes, al activar una regla manual, pots escollir si vols treballar amb peticions obertes o tancades. Per norma general un usuari normal no podria treballar amb una petició tancada, però al Rule Engine si que pot tractar amb elles.

Figura 6.5 Pantalla d'activació de regles manuals

En els laboratoris no s'utilitzen massa les regles manuals, ja que la majoria fa servir les regles automàtiques. Però tot i així, es un requisit que va demanar el client.

6.2.2 – Informes

La pantalla d'informes és simplement una pantalla informativa, serveix perquè l'usuari pugui consulta d'una manera ràpida quines són les accions que a realitzat una determinada regla.

Entre d'altres coses, pot veure quina és la petició avaluada, en quina data s'ha avaluat i quina acció a realitzat (abreviatura de l'acció). A més de veure quin és l'element afectat (també es visualitza l'abreviatura de l'element).

Per exemple en la figura 6.6 es pot veure com la regla anomenada regla1a realitzat l'acció RTA (afegir prova), i que la prova afegida a estat la glucosa.

Figura 6.6 Pantalla d'informes

La pantalla té dos funcionalitats diferents, la primera és eliminar el contingut visualitzat, de tal manera que no tornarà a sortir. En realitat el contingut no és borra, ja que com he comentat amb anterioritat el Rule Engine realitza accions que afecta de manera important les dades del laboratori, i per motius de seguretat no es poden eliminar. El que es fa en realitat es posar el flag de llegit a 1 per no tornar-lo a mostra.

La segona funcionalitat és sortir sense esborrar, de tal manera que les dades queden intactes i posteriorment un altre usuari les podrà consultar. S'ha de tenir en compte que per reactivar una regla, abans s'ha d'esborrar el contingut de la pantalla d'informes, en cas contrari no deixaria reactivar la regla.

6.3 - Com funciona el Rule Engine.

Un cop vist com es defineixen les diferents parts de les quals està definida una regla, s'ha de veure com ho fa el sistema per executa les regles.

Per començar s'ha de posar una crida al Rule Engine, amb un identificador de mode d'activació. Com que cada mode d'activació té un identificador únic s'haurà de posar més d'una crida en cas que es vulgui crida al Rule Engine per més d'un motiu.

Un cop el codi del programa arribi on es troba la crida amb el mode d'activació escollit, buscarà si hi ha alguna regla activa amb aquell mode d'activació, i que pertanyi a la mateixa aplicació de la regla.

Òbviament si no en troba cap, sortirà de la crida del Rule Engine i seguirà de llarg. Però si en troba alguna, executarà les seves condicions. Per moltes línies que tingui la condició al final només acaba retornant un booleà, per tant en cas que torni un 1 (true) buscarà les accions definides en aquella

regla i comencin per "llavors" i les executarà. Si per el contrari les condicions retornen un 0 (false) buscarà si té alguna acció del tipus "sinó" i en cas que la tingui també l'executarà.

Si s'executa l'acció definida per la regla, es guardarà un registre en una de les taules de Rule Engine per guardar constància de la seva execució. D'aquesta manera es pot mirar en qualsevol moment que ha fet el Rule Engine i quan.

D'altre banda tenim les regles manuals, aquestes regles un cop definides es quedaran esperant a que un usuari amb permisos les activi. Al activar-les s'avaluaran les seves condicions, si es el cas es realitzaran les accions, i al igual que les automàtiques quedarà registrat els canvis realitzats. Un cop executades es tornen a parar automàticament.

6.4 – Taules del Rule Engine

Per entendre una mica més el Rule Engine, es necessari saber quines taules el formen i quina funcionalitat aporten:

- trulActionObjectives: guarda les relacions entre el camp acció i el camp objectiu (explicat en la pantalla d'accions).
- trulActions: guarda el contingut de les accions definides per l'usuari.
- tTMPrulActions: guarda el contingut de les accions mentre l'usuari les està editant. Un cop es salvi la regla, aquesta taula es buidarà i el seu contingut es guardarà en la taula trulActions.
- trulActionTypes: conté els diferents tipus d'accions.
- trulActivationModes: conté els modes d'activació, i la seva dependència segons els tipus de regles.
- tTMPrulActivationModes: guarda els modes d'activació mentre s'estan editant, al igual que amb la temporal d'accions al guarda la regla es esborrarà el contingut de la taula i s'emmagatzemarà en la trulActivationModes.
- trulActivationModesActionTypesObjectives: serveix per definir quins modes d'activació poden anar amb unes determinades accions. (actualment no s'utilitza).
- trulActivationModesRules: relaciona les regles amb els seus mòduls d'activació.
- trulActivationThreads: Es la taula en la que es basa la pantalla de Rules Control, bàsicament guarda l'estat de les regles.
- trulConditions: emmagatzema les condicions definides per l'usuari.
- tTMPrulConditions: emmagatzema les condicions definides per l'usuari mentre aquest està modificant-les.
- trulConditionsLines: emmagatzema línia a línia les condicions definides
- tTMPConditionsLines: mentre l'usuari esta editant les condicions, per tal de no esborrar les que actualment estan definides, es guarden de manera temporal en aquesta taula. Al guarda la regla, el contingut d'aquesta taula es s'esborrarà i les seves dades passaran a la taula trulConditionLines.
- trulDeletedTests: en cas de que el Rule Engine elimini alguna prova, es guarda un registre en aquesta taula perquè quedi constància.
- trulEntrance: guarda el valor del combo “Entrada” de la pantalla de condicions.
- trulField: guarda els valors dels combo “Camp” de la pantalla de condicions. Aquesta és una

taula molt important en la construcció de les condicions. Ja que el mòdul de condicions “co.rul.crulConditions” utilitza el codi sql incrustat en aquesta taula

- trulFieldType: Conté els principals tipus de dades que s'utilitzen en la pantalla de condicions.
- trulMaster: emmagatzema informació que pot ser utilitzada per qualsevol pantalla de Rule Engine.
- trulRequest: es guarden les regles que s'han d'executar en background. Normalment es tracta de les regles amb el mode d'activació “un cop al dia”.
- trulRules: guarda les accions definides per l'usuari.
- tTMPrulRules: guarda les regla definides per l'usuari mentre les està modificant.
- trulRulesTypes: guarda el tipus de regla
- trulSelection: guarda el contingut del “selector” de la pantalla d'accions.
- tTMPrulSelection: emmagatzema temporalment el contingut del “selector” de la pantalla d'accions mentre l'usuari no guardi la regla en la base de dades.
- trulSpecifications: serveix per decidir quins dels camps dinàmics de la pantalla d'accions s'han de mostra i quins no.
- trulVariables: guarda la definició de les variables. No s'utilitzarà fins el Omega v4.1
- trulVariablesLines: guarda la definició de cada línia de cada variable. No s'utilitzarà fins el Omega v4.1
- tOrderMarks: es guarda un registre de las accions realitzades per el Rule Engine.
- tAffectedTests: relaciona les proves afectades per el Rule Engine amb les accions realitzades.

Apart de les taules que es defineixen en el Rule Engine, aquest utilitza moltes més per realitzar les accions o avaluar les condicions. Aquestes taules tot i que originalment pertanyen a d'altres mòduls de l'Omega v4 són molt importants per el Rule Engine.

6.5. - Desenvolupament

Primerament s'han de decidir les eines per desenvolupar el projecte, en aquest cas Cache i ZEN, que són les eines escollides per l'Omega v4. Tot seguit s'ha de fer un disseny d'acord amb el que s'espera que realitzi l'aplicació. En aquest cas, el més important era que el gestor sigues el més potent possible, altament configurable i de fàcil utilització.

En la part de disseny, un dels grans encerts del disseny va ser la realització de taules temporals per guarda les dades de les regles quan s'estan editant. D'aquesta manera si per qualsevol cosa l'usuari decideix sortir sense guarda els canvis realitzats, recuperarà immediatament les dades que tenia abans de l'edició.

Per contra un dels errors, va ser la manera com es guardaven les accions realitzades per el Rule Engine. En un inici les taules truCalls i truActionHistory, s'encarregaven d'emmagatzemar les dades, però tot i que el funcionament de l'aplicació era correcta es va observar que el volum de dades d'aquestes taules creixia de manera alarmant. Per la qual cosa en un re-disseny posterior, que encara que igualment guarda les accions realitzades, no desborda les taules. Les noves taules per guarda les dades són tOrderMarks i tAffectedTests.

Un dels altres inconvenients que es van trobar va ser el de sincronismes. En varies pantalles hi ha

vairs components dinàmics (creats en temps d'execució), i amb alguns navegadors IE6, IE7 i firefox 3.0 no acabaven de funcionar correctament. Finalment en alguns casos, va decidir-se usar components estàtics, que la pantalla mostra en el moment adequat.

La resta del desenvolupament va anar segons les previsions (veure apartat 2.8 Planificació temporal), tot i que finalment es va acabar amb unes dos setmanes de retard respecte al que estava previst.

6.6 – Fase de proves

Un mòdul que controla els resultats obtinguts per l' Omega v4, i que té la capacitat de modificar les dades de l'aplicació, no pot contenir cert tipus d'errors. Més tenint en compte que és un producte enfocat a l'àmbit sanitari, on els errors poden provocar problemes molt greus.

Per aquesta raó l'Omega v4 i en especial el Rule Engine han de passar un control de qualitat molt estricte per tal d'assegurar-se que cert errors no es poden produir.

El Rule Engine es un mòdul especialment difícil de validar, ja que el número d'accions i especialment el de condicions, és massa elevat per provar-les en la seva totalitat. La millor manera per validar el Rule Engine és validar un grup de cadascuna de les principals branques d'execució del Rule Engine. També avaluar els casos especials amb la màxima cura possible.

La validació del Rule Engine es pot dividir en tres fases diferents:

- Validació bàsica desenvolupador:

  Aquí el desenvolupador comprova que les dades es guarden correctament, que les pantalles realitzades segueixen el comportament d'escrit pel cas d'ús.
  Seguint la casuística de la pantalla no es detecta cap error.

- Validació desenvolupador:

  El desenvolupador comprova de manera exhaustiva que la construcció dels diferents tipus de condicions i/o accions es correcta, i que l' interacció amb els altres mòduls es realitza satisfactòriament.

- Validació IS-Q:

  Com s'ha comentat anteriorment existeix un departament que s'encarrega de validar l'Omega v4 per tal d'assegurar la màxima qualitat del producte, i detectar i corregir en la mesura de lo possible en nombre més elevat d'errors.

  El personal de IS-Q comprovarà que el funcionament de l'aplicació sigui el especificat en la documentació, i que no hi hagi cap error.

  Es comprovarà amb exhaustivitat, des de les pantalles fins la interconnexió entre els diferents mòduls que formen l'aplicació.

Quan un membre de l'equip de IS-Q detecta un error, fa una breu descripció, i el classifica segons el seu nivell de perillositat. Aquesta informació s'emmagatzema utilitzant un programa que s'anomena ClearQuest.

6.6.1. - ClearQuest

ClearQuest és un programa que serveix per validar altres programes. Els usuaris poden accedir remotament per visualitza o emmagatzema informació (errors trobats) i classificar-los segons paràmetres com la perillositat, el mòdul al que pertanyen, la persona que la trobat i la persona encarregada de corregir-lo.

Els errors són classificats en els següents estats:

- Registrat: entrada descripció de l'error, però encara no té cap persona assignada per solucionar-lo.
- Assignat: hi ha una persona encarregada d'arreglar-lo.
- Implementat: S'ha corregit l'error i s'ha de torna a validar.
- Validat: l'error està solucionat.

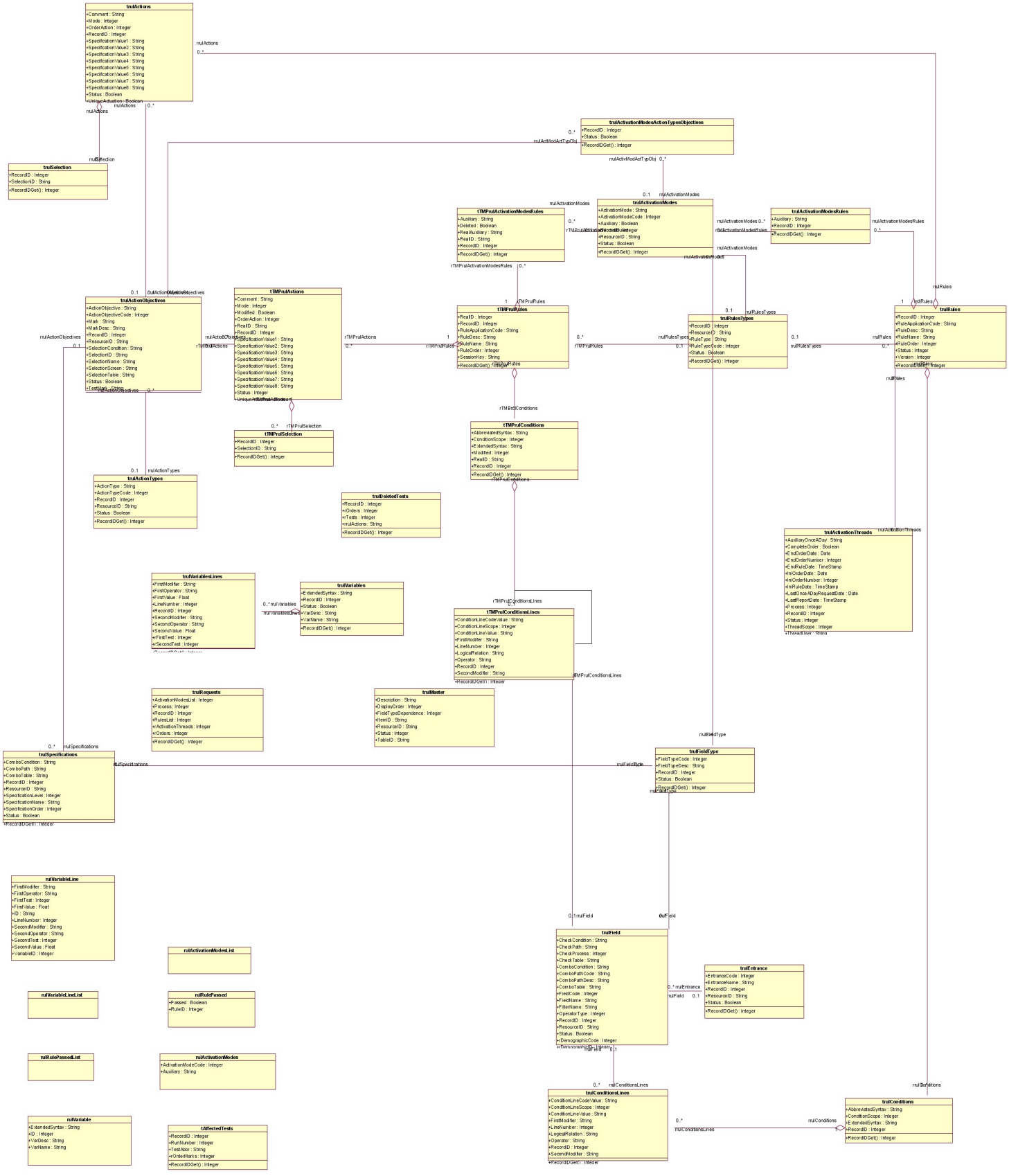


Figura 6.7 Diagrama UML del modul de Rule Engine

7. - Conclusions:

7.1 - Objectius:

Al principi es diu que els objectius d'aquest projecte eren, per part de l'empresa, actualitzar el software que l'empresa té el mercat, per tal de consolidar-se i referma la seva posició.

Actualment l'Omega v4 s'esta començ a'n a instal·lar a varis hospitals espanyols, i està tenint bastant bona acollida. Els clients sembla'n bastant contents amb el nou producte i les noves tecnologies que proporciona. Tot i que encara és d'hora per avaluar si el producte tindrà èxit i complirà les previsions fetes per Roche Diagnostics, tot apunta que serà així. Per tant es pot considera que l'objectiu de l'empresa s'està complint.

En el cas del Rule Engine ha demostrat ser un potent i eficaç gestor de regles per controlar els diferents resultats de l'Omega v4. Es altament configurable i es poden definir un gran número de regles diferents per gestionar l'aplicació. Es pot considerar que el Rule Engine a complert plenament amb els objectius plantejats en un principi.

7.2 - Futures ampliacions

Degut a que la data de llançament de l'Omega v4 estava molt pròxima es va decidir, que certs aspectes del Rule Engine, com són la definició de variables, o la seva futura ampliació per tal de poder-se comunicar amb un nou programa anomenat eFlow (per comunicar màquines) quedaven temporalment ajornats.

Això demostra que el Rule Engine tindrà en els mesos següents varies ampliacions que el faran encara més complert del que és actualment.

8. -Bibliografia:

Referències d'internet:

- Intersystems: http://en.wikipedia.org/
- Cache i ZEN: http://www.intersystems.com
- Cache: http://www.mkm-pi.com/mkmpi.php?article1882
- Cache: http://www.idg.es/pcworld/InterSystems-Cache-4.0_Base-de-datos-post-relacion/art123311.htm
- Definicions: http://www.cvc.uab.es

Documents de Roche:

- Arquitectura: Estàndards de desenvolupament
- Guia de programació ZEN
- Guia d'estils de LIS

Omega v4

Rules Conditions

Creación	**Nombre**	**Rol**	**Dpto.**	**Firma**	**Fecha***
	Oscar Fernandez	SW Development Programer	IS-D		

Revisión	**Nombre**	**Rol**	**Dpto.**	**Firma**	**Fecha***
	Luís Suárez	Technical Project Leader	IS-D		
	Mar Llobet	Validation Team Leader	IS-Q		

Aprobación	**Nombre**	**Rol**	**Dpto.**	**Firma**	**Fecha***
	Ricard Presas	Head of SW Development	IS-D		

*El formato de la fecha consiste en 2 dígitos, el mes en 3 letras y el año en 4 dígitos (DD/MMM/AAAA)

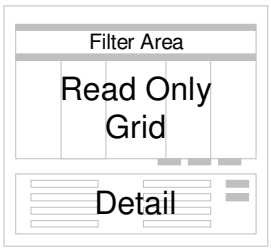
Validez:	La versión vigente del documento está disponible en el repositorio de la documentación de proyecto. Debe comprobarse la validez de las copias impresas con la documentación del repositorio.

Cambios	**Versión**	**Fecha***	**Nombre**	**Razón**
	1	19/05/2008	Oscar Fernandez	Versión Final

CONDITIONS

Description

The Conditions page allows the user to inquire and administrate conditions related to a rule. In this page user can perform maintenance operations such as adding, modifying and deleting conditions as well as checking errors in the conditions. This page is thrown from the “Rules” page.



Divisions:

- Top zone
  - Fields
    - ✓ Rule
      - ➢ Read only and filled by the system
    - ✓ Description
      - ➢ Read only and filled by the system
    - ✓ Condition for
      - ➢ Read only and filled by the system
  - Grid
    - ✓ Fields
      - ➢ Condition.
      - ➢ 1st Modifier.
      - ➢ Entry.
      - ➢ Test.
      - ➢ Field.
      - ➢ Operator.
      - ➢ Value.
      - ➢ 2nd Modifier.
    - ✓ No Sorting fields
    - ✓ No Filtering fields
- Detail
    - ✓ Fields
      - ➢ Condition.

- Combo Box (And / Or)
- By default value: and
- The first cell (in the first row) of this field will always have an "If" value so that the expression starts with this value (it will not be editable)
- Mandatory

➢ 1st Modifier.

- Combo Box
- When the user enters in this cell, a combo box with the modifiers is open (Modifiers: **Not (/ (/ C**).
- The selected modifier is added (to the first modifier text cell on the right) already existing modifiers in the cell. If the user selects the "C" modifier, the last modifier existing in the cell is deleted
- First modifier text cell is a read only cell.

➢ Entrance.

- Combo Box
- Enabled only if the Condition field is informed.
- If the "Type" field of the Rule is "ICA Rule", the possible values of this field are only "Demographic" and "Test"
- Mandatory

➢ Test.

- Combo Box
- Enabled only if the Entrance field is informed.
- Possible values: All, any, customize. If the user selects the value of customize, the system will open a pop-up to select the desired test from a list (the "Test selector" pop-up). If the "Type" field of the Rule is "ICA Rule", the "Test selector" pop-up will charge only the tests related to the value selected in the "Instrument" field in the "Rules" screen.
- Mandatory

➢ Field.

- Combo Box
- Enabled only if the Entrance field is informed.
- This combo depends on the "Entrance" field
- Mandatory

➢ Operator.

- Combo Box
- Enabled only if the Field field is informed.
- This combo depends on the "Field" field

- If “Operator” field value is >, <, >=; <=, or [the “Value” field will be Text box (except date format).
- Mandatory

➢ Value.

- Combo Box or Text Box.
- Enabled only if the Operator field is informed.
- It depends on the “Entrance”, “Field” and “Operator” fields
- When the required format of the “Value” field is a date, two combo box and one textbox will appear. First combo box with date values (like today, last12hours etc). And the second combo box with time units like weeks, months, years etc.
- If the “Operator” field’s value is “ ▪”, the “Value” field will not be editable
- If the “Operator” field’s value is “[”, the “Value” field will always be a Text Box
- If the “Field” field’s value is “Group Name”, a text box and a button will appear. If the button is clicked a pop-up will be open to select a group.
- If the “Field” field’s value is “Age” a text box and a combo box with time units (weeks, years etc) will appear.
- If the “Field” field value is type numeric, “Value” must be numeric, is the type is time “Value” must be and hour etc.
- Mandatory when the “Operator” value is not “ ▪ ”

➢ 2nd Modifier.

- Combo box
- When the user enters in this cell, a combo box is open and show the modifiers. (Modifiers: **) / C**)
- The selected modifier is added to the already existing modifiers in the second modifier text cell (in the right). If the user selects the “C” modifier, the last modifier existing in the cell is deleted
- Second modifier text cell is read only.

Initial state

The top zone fields (“Rule”, “Description” and “Condition for”) will be filled out by the system with the data of the selected rule in the “Rules” page. The other top zone fields will be loaded with the data of the rule if defined. The grid will be loaded with the expressions of the selected condition radio button. In case no condition fields are defined, the “Condition” grid will have one value at the “Condition” cell in the first row (an “If” value), the rest of this row will be empty.

Actions

- Add an Expression [button]
  - This button carries out the creation of a new expression in the Conditions grid.
  - The detail zone will be activated in order to allow the user to introduce the data of the new expression.
- Insert an Expression [button]

- This button carries out the creation of a new expression in the Conditions grid.
- The detail zone will be activated in order to allow the user to introduce the data of the new expression.
- When the user clicks the Insert button and after finishing filling all mandatory fields and clicking Ok button, then an additional expression row appears above the selected row (where the cursor is focused) of the Conditions grid.
- If the selected row is the first one and after finishing filling all fields and clicking Ok button, then the system will add a row at the beginning with the "Condition" value of "If". And the "Condition" cell of the second row will be set to "And".

- Insert existing Expressions [button]
  - This button copies expressions of other rules conditions to the Conditions grid.
  - This action opens the "Conditions selector" pop-up.
  - The selected expression rows are copied above the selected row of the "Condition" grid. If the selected row is the first one, then the system will set the value of the "Condition" cell of the selected row to "And" and will set the value of the "Condition" cell of the new first row to "If".
- Remove an Expression [button]
  - This action removes a selected expression.
  - When the user clicks the Remove button, then the selected row is deleted from the "Condition" grid (if the selected row is the first one, the "Condition" cell of the second row will be set to "If").
  - The "Condition" grid is updated.
- Remove all Expressions [button]
  - This action removes all expressions.
  - When the user clicks the Remove button, then all expressions are deleted from the "Condition" grid.
  - The "Condition" grid is updated.
- Select a test [customize value in the "Value" field combo]
  - This action is only available if the "Entrance" and "Field" fields are "Test".
  - This action opens the "Test selector" pop-up when the user selects the combo value of "Customize" in the "Value" field.
- Accept [button]
  - Apply validations of the fields. If the system detects that any mandatory field is empty, a message will be shown to the user about this fact and the fields to fill in. Then the user accepts the message and is allowed to fill in the rest of fields.
  - Additional validations must be checked.
- Cancel [button]
  - The user will be prompted to confirm the action ("*Do you want to discard changes and return to the "Rules" screen?*").
    - ✓ If the user doesn't cancel the action by pressing 'No' button, the message is closed and the user can continue adding/changing data.
    - ✓ If the user cancels the action by pressing 'Yes' button, the message is closed, the system returns to the "Rules" screen with no modifications.

- Return [button]
  - This action closes the “Conditions” page and the system returns to the “Rules” page.
  - If the Check Conditions are not ok the user will be prompted to confirm the action (“Do you want to exit without saving?”)
- Check Conditions [Automatic]
  - The system checks if there are mistakes in the conditions expressions of both conditions fields (when adding or modifying a condition line):
    - ✓ If there are mistakes, the system shows where mistakes were made in red and tells the user to correct them showing a message.
    - ✓ If there are not mistakes, nothing will be done.
- Always true condition [select the “Always true” value in the “Entrance” field combo of the grid]
  - The user selects the value of “Always true” in the “Entrance” field combo of the first row (the other rows will not have this value in the “Entrance” field combo).
  - All the other fields of the grid are set to empty (except the “Condition” and the “Entrance” fields of the first row) and not editable (except the “Entrance” field of the first row)
  - The “Insert existing expressions”, “Add” and “Insert” buttons are not available so that no more rows can be added.
  - The condition field shows the “Always true” expression.

Additional Validations

- The user can not return to the rules page if the conditions expressions have not been checked successfully (if an error in the expressions appear, the expressions will not be stored).

Detailed description

This page follows the standard definitions for the maintenance pages.

- Events table:

Event	*Web Service*	*Returned error*
Load Page	To load the grid use **ws.rul.wrulConditions.GetConditionLinesList-Temp()** To load Entry combo use **ws.rul.wrulConditions.GetEntrancesList()** To obtain rules ID **ws.rul.wrulConditions.AddConditionTemp()**	
Accept button click	When inserting a new line call **ws.rul.wrulConditions.AddCondition-LineTemp()** When modify a rule **ws.rul.wrulConditions.ModCondition-LineTemp()**	

	When deleting a line call **ws.rul.wrulConditions.DelConditionLineTemp( )** When deleting all lines call **ws.rul.wrulConditions.DelCondition-LinesTemp()**	
Close button click	**ws.rul.wrulConditions.SaveConditionTemp()** to rebuild the condition from the set of condition lines that have been modified.	
Validate button click	**ws.rul.wrulConditions.CheckBrackets()** to return the list of brackets that are not paired off.	
When select-ing a value in "Entry" combo	Populate "Field" combo using **ws.rul.wrulConditions.GetFieldsList()** When value in "Entry" combo is "Test", load "Constraint" combo by calling **ws.rul.wrulRules.GetMasterValuesList**("CON-DITION_CONSTRAINT") When value in "Entry" combo is "Variable", load "Constraint" combo by calling **ws.rul.wrulVariables.GetVariablesList()**	
When select-ing a value in "Field" combo	Populate "Operator" combo using **ws.rul.wrulConditions.GetOperatorsListRe-sultset ()** Populate "Value" combo using **ws.rul.wrulConditions.GetValuesList()** **ws.rul.wrulConditions,GetFieldTypeCode()**	
Double click action	Obtain testname **sp.spCondition.spTranslatedLineScope**	
Load dy-namic combos	**Ws.sys.wMaster.GetMasterList('LISTSDATE')** to load dynamic date combos **Ws.sys.wMaster.GetMasterList('RE_COND)** To load dynamic combo for field = 'age' **Ws.sys.wMaster.GetMasterList('FUNC_TEST)** To load dynamic combo units	

Actions that involve RoundTrip:

Action
Load Page
Accept button click
Close button click

When selecting a value in “Entry” combo
When selecting a value in “Field” combo