Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya - BarcelonaTech

Plataforma per al monitoratge i l'explicació de violacions de clàusules de documents SLA

Projecte de final de carrera
Enginyeria Informàtica

Marc Rodríguez Navarro
14 de gener de 2014

Director: Jordi Marco Gómez
Departament: LSI

Agraïments

Sovint, la resposta a una pregunta és la cerca de la resposta en sí. Vull aprofitar aquestes línies per donar les gràcies a tots aquells que m'han ajudat a transformar aquests darrers set anys en una resposta a la meva pregunta:

Al Xavier Franch, per donar-me l'oportunitat d'entrar al projecte SALMon i al grup, i de fer aquest projecte de recerca; al Jordi Marco, per la seva inestimable orientació, atenció i paciència; i al Marc Oriol, mentor i amic.

Als companys del despatx i del grup de recerca, per tots els bons moments, i en especial al David Aguilera, per les discussions filosòfiques i les reconciliacions, i al Toni Villegas, per les vegades que ens hem rigut.

Als meus companys de projectes i classes de la carrera i amics, elit i sinèrgia incomprensiblement efectiva. Entre ells, a l'Hèctor Manrique, per ser amb mi de principi a fi del llarg viatge; al José Manuel López, per mostrar-me camins més curts; a l'Oriol Collell i el Pere Joan García, per les claus a altres futurs, i al Jordi Cidoncha, per la complementació.

To my Erasmus' friends, because Aberdeen '11 will not fall into oblivion.

A l'Oscar Mercè i el Victor Mercè, per tot.

I als meus pares, pel seu suport.

Sense tots vosaltres, res d'això hauria estat imaginable. Moltes gràcies a tots!

Índex

Índex de Figures

Índex de taules

1. Introducció

La present memòria documenta el desenvolupament del projecte de final de carrera *Plataforma per al monitoratge i l'explicació de violacions de clàusules de documents SLA*.

El projecte instancia un model de referència per a l'anàlisi i monitoratge de qualitat del servei amb SALMonADA, un sistema basat en serveis que detecta violacions de clàusules en acords de nivell de servei (SLA) en temps real, notifica als subscriptors i els explica els motius de la violació de forma entenedora. SALMonADA és la integració de dues eines: SALMon, un monitor de serveis web, i ADA, un analitzador d'acords de nivell de servei. El projecte descriu la integració de les dues eines, les modificacions a l'eina SALMon per tal de fer-ho possible, i el desenvolupament d'una interfície web d'usuari.

En aquest capítol s'introdueix l'àmbit del problema, el de la qualitat en serveis web, i a continuació es descriu el projecte en sí, sense entrar en excessiu detall en cap dels temes.

1.1. Un món de serveis

Els serveis són un dels pilars de la nostra civilització. Un servei és una comoditat o benefici que resulta intangible, consumible i no reutilitzable, el qual s'efectua entre un proveïdor que l'ofereix i un consumidor que el necessita.

El món n'és ple d'exemples: Quan hom agafa el metro, obre l'aixeta a casa, va a la biblioteca o deixa el cotxe al mecànic per a una reparació està utilitzant serveis. Aquests són tan fonamentals per al món com els béns tangibles.

El progrés de la tecnologia ha fet que haguem de considerar les màquines no només com a eines sinó com a components importants en el procés de realitzar aquests serveis. De vegades, en són components imprescindibles; algun cop fins i tot en són l'únic. En particular, la innovació en el camp de l'enginyeria informàtica ha mogut la distribució física de dades i codi a un nou paradigma, el dels serveis web, on codi i dades ja no són quelcom que cadascú té al seu disc dur, sinó un servei que qualsevol consumidor autoritzat pot fer servir de forma automàtica i remota.

Tant si els serveis són realitzats íntegrament per persones com si no, tots ells comparteixen una sèrie de característiques. Una d'elles és que tenen valor, i en una civilització capitalista com la nostra, això es tradueix en un intercanvi de servei per preu entre proveïdor i consumidor.

El conjunt de proveïdors i consumidors de serveis crea, doncs, un mercat, i en el mercat la competència sorgeix de forma natural: Els consumidors cerquen els proveïdors que ofereixin un servei de major qualitat al millor preu per assegurar el seu èxit, i de forma paral·lela els proveïdors cerquen oferir un servei de major qualitat o de menor cost que els altres per a tenir més consumidors i aconseguir així el seu propi èxit.

Per a garantir la realització amb èxit del servei amb una certa qualitat, proveïdor i consumidor es poden posar d'acord en un contracte que defineixi la realització del servei i el seu preu. Aquest contracte pot estipular que el servei s'oferirà amb una qualitat mínima determinada. No obstant, res és perfecte; les persones no en som una excepció, i les màquines menys. La

qualitat d'un servei és un paràmetre que fluctua amb el temps, i per a conèixer-la no n'hi ha prou en creure's un contracte; s'ha de monitorar l'execució del servei. És important que proveïdor i consumidor siguin conscients de la qualitat del servei, doncs una qualitat que no compleixi el contracte no és acceptable, i pot donar lloc a decisions de negoci.

1.2. La qualitat en serveis web

> *La principal característica d'un servei web és que es tracta d'una peça de software que un usuari pot utilitzar sense ser-ne propietari, és a dir, sense necessitat d'instal·lar-lo i usant-lo a través d'Internet i protocols estàndard. [Oriol]* (p. 4, traduït)

Un servei web és, en el camp de l'enginyeria del software, un servei realitzat i consumit per sistemes software[1] de forma distribuïda a través d'una xarxa, com pot ser Internet, mitjançant protocols específics coneguts tant pel sistema client com pel sistema proveïdor, el qual pot oferir dues comoditats al client: L'accés a informació posseïda pel proveïdor (distribució de dades) i l'accés a funcionalitats que el proveïdor implementa (distribució de codi i recursos).

Els serveis web fan servir un llenguatge per a la descripció de la seva funcionalitat, com ara *Web Services Description Language (WSDL)*, i un protocol per a la comunicació entre proveïdor i consumidor, com per exemple *Simple Object Access Protocol*[2] *(SOAP)*. Un usuari que vulgui usar un servei sabrà com usar-lo mitjançant un document de descripció del servei, escrit en un llenguatge de descripció com WSDL, que el proveïdor ha d'haver posat a la seva disposició (de forma pública o privada). Aquest document, entre altres coses, descriu quins són els protocols de comunicació que el servei web reconeix i accepta. L'usuari es podrà comunicar amb el servei, doncs, emprant un d'aquests protocols, mitjançant l'enviament i recepció de *missatges*[3].

Els serveis web, com la resta de sistemes software, responen a un *model de qualitat* que descriu els atributs que defineixen la seva *qualitat de servei* o *Quality of Service*[4] *(QoS)*, és a dir, com de bé fan allò que han de fer. Exemples d'aquests atributs són el temps de resposta, la disponibilitat del servei, i el compliment de la funcionalitat.

Aquests atributs són qualitatius, no quantitatius, és a dir, són atributs que es poden observar, però no es poden mesurar directament: Es pot dir que un servei compleix la funcionalitat adequadament o té una mala disponibilitat. No obstant, en el món dels serveis i en concret dels serveis web, el que ens interessa és poder comparar aquests atributs entre diferents serveis i/o establir unes garanties mesurables. Per a fer-ho, es desglossen aquests atributs en *mètriques de qualitat*, que són característiques mesurables objectivament que influeixen en els atributs del model de qualitat. Per exemple, el temps de resposta màxim d'un servei web en un període determinat és una mètrica mesurable (es pot dir que és, per exemple, de 200

---

[1] Noti's que tant el client (o consumidor del servei) com el proveïdor poden ser persones o empreses que implementin i/o mantinguin aquests sistemes software. Al llarg de la memòria, quan es parli d'algun dels dos ("consumidor" i "proveïdor"), per omissió, s'estarà fent referència tant al sistema software com a aquell qui l'implementa o manté com si fossin una sola entitat.

[2] Vegeu el capítol 3.3 Eines, plataformes i estàndards (pàgina 21).

[3] Un missatge és una cadena de caràcters que s'envia entre client i proveïdor i que conté informació com ara peticions del consumidor o respostes del proveïdor.

[4] Vegeu el capítol 2.2 Què és la qualitat del servei? (pàgina 8).

mil·lisegons), que ajuda a decidir si el temps de resposta d'un servei és bo, dolent, suficient, pitjor que el d'un altre, etc.

Per a establir unes garanties de qualitat, client i proveïdor del servei acorden unes condicions sobre aquestes mètriques, i les expressen en forma d'un acord de nivell de servei o *Service Level Agreement*[1] *(SLA)*, que no és més que un document legal signat per ambdues parts que descriu la relació entre ambdues i el servei, les condicions, i si aquestes són obligatòries o bé tenen recompenses o penalitzacions en cas de complir-se o ésser violades, i quines.

En el cas dels serveis web, aquests SLAs poden ser expressats en llenguatges estàndard com ara *Web Services Agreement*[2] *(WS-Agreement)*. Això fa que es pugui automatitzar el procés de comprovar si les mètriques de qualitat del servei mesurades compleixen o no les condicions de l'SLA.

De la mateixa manera, la mesura d'aquestes mètriques també es pot automatitzar mitjançant un *monitor*. Des dels inicis de la programació han existit els monitors, programes que serveixen per mesurar mètriques de qualitat d'altres programes o de hardware, com ara el rendiment. Tot sistema operatiu modern implementa un munt de funcions de monitoratge, com per exemple un visor de l'ús de CPU en temps real.

En el cas dels serveis web aquesta definició s'amplia, ja que no només es pot monitorar el proveïdor del servei, sinó tots els elements que influeixen en la realització del servei[3]. A més a més, la forma de monitorar les mètriques canvia, ja que parlem de software distribuït que segueix uns protocols i estàndards específics, i per tant es necessita un tipus de monitor específic. Aquests monitors, entre altres coses, monitoren l'intercanvi de missatges entre client i proveïdor, i en mesuren paràmetres com el moment de l'enviament o recepció, el contingut dels missatges o el fet que obtinguin resposta.

1.3. Descripció del projecte

Com s'ha dit, ja existeixen eines per al monitoratge de software, així com de serveis web. També existeixen eines que poden comprovar si les clàusules d'un document SLA es compleixen per una qualitat de servei mesurada.

Aquest projecte intenta cobrir la divisió entre ambdós processos, proporcionant una manera automàtica de comprovar el compliment d'un document SLA partint tan sols del propi document i de la descripció del servei, passant pel seu monitoratge i la posterior notificació de les violacions als interessats, en cas que aquestes ocorrin. D'aquesta manera, aquell qui estigui interessat en detectar en quins moments la no realització o realització pobre d'un servei trenca les condicions del contracte ho pot saber a l'instant, i actuar en conseqüència.

El projecte neix de la necessitat de detectar i comprendre les violacions d'acords entre proveïdor i consumidor en temps real, mitjançant el monitoratge automatitzat dels serveis web i la interpretació d'aquests acords de nivell de servei.

---

[1] Vegeu el capítol 2.3 Acords de nivell de servei (pàgina 13).
[2] Vegeu el capítol 3.3 Eines, plataformes i estàndards (pàgina 21).
[3] Vegeu el capítol 2.2.4 Monitoratge de mètriques (pàgina 12).

El projecte es desenvolupa en el marc de la col·laboració entre els grups de recerca ISA[1] de la US i GESSI[2] de la UPC, i comprèn, entre altres tasques, la integració de dues eines ja existents:

1. **ADA**, un servei d'anàlisi d'acords actualment en desenvolupament al grup ISA, que permet interpretar documents SLA escrits en llenguatge WS-Agreement; i
2. **SALMon**, un sistema de monitoratge i *testing*[3] de serveis web actualment en desenvolupament al grup GESSI, que permet monitorar serveis web que segueixin el protocol SOAP, així com configurar un *tester* que els invoqui com un client real a intervals de temps.

A continuació s'expliquen les necessitats noves o ja existents que han motivat el projecte, les motivacions per les quals els actors implicats han decidit cobrir aquestes necessitats, i l'objectiu concret que se n'ha derivat, això és, l'objectiu del projecte.

1.3.1. Necessitats

Quan un client cerca usar un servei, pot fer servir un o més proveïdors d'aquest servei. En molts casos, a més, client i proveïdor estableixen un contracte SLA, document que determina el preu, condicions i garanties de l'ús del servei.

En aquests casos, hom pot necessitar assegurar que les condicions d'aquests contractes es compleixen. El fet que no es compleixin significa que el proveïdor del servei no l'està realitzant tan bé com va prometre, o bé que el consumidor no l'està usant correctament. Tant si és un com si és l'altre cas, quan l'acord es viola, consumidor i proveïdor poden prendre mesures, com ara complir alguna penalització, cessar la relació contractual, o altres.

Per a comprovar el compliment de l'acord, s'ha d'interpretar el document SLA i configurar un monitor que monitori el servei. A continuació cal realitzar-ne un seguiment actiu a fi d'adonar-se en quin moment es viola una condició del document. Fer aquest seguiment actiu consisteix en comprovar que la qualitat del servei mesurada no viola cap de les condicions, i fer-ho cada vegada que la qualitat del servei canvia. Finalment, quan una violació es detecta, cal avisar a aquells qui estiguin interessats en saber-ho (per exemple, el consumidor o el proveïdor, però també una tercera part), i fer-ho de manera entenedora i acurada, incloent les clàusules violades, les causes de la violació i potser possibles efectes.

Tot aquest procés resulta un conjunt de tasques feixugues, tedioses i propenses a errors que es poden automatitzar, facilitant així la feina en gran mesura.

Existeixen diverses solucions per a l'anàlisi i monitoratge de documents SLA. Cap d'elles, però, cobreix tot el procés de principi a fi proporcionant a més una explicació de les violacions intel·ligible i completa com la que es cerca, com es veurà al capítol 3.1.2 Antecedents (pàgina 18).

---

[1] Ingeniería del Software Aplicada, del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla (US).

[2] Grup de recerca en Enginyeria del Software per als Sistemes d'Informació, del Departament d'Enginyeria de Serveis i Sistemes d'Informació (ESSI) de la Universitat Politècnica de Catalunya (UPC).

[3] Definim el testing un servei web com el fet de mesurar la qualitat del servei en un moment determinat, sense que faci falta que un client real l'usi.

Es necessita una persona experta en el format dels acords per a interpretar aquestes violacions, igual que es necessita per a redactar els acords. Malgrat això, redactar l'acord és una tasca que es fa només al principi i quan es re-negocia l'acord, cosa que ocorre relativament poc sovint, mentre que comprovar la qualitat de servei en temps real és una tasca que es pot requerir molt freqüentment.

El projecte, doncs, busca cobrir la necessitat d'automatitzar tot aquest procés. Aquesta pot ser desglossada bàsicament en tres necessitats:

1. Poder configurar un monitor de serveis web tan sols amb un document SLA. Això facilita el procés de configuració del monitor i el fa més usable.
2. Poder detectar a temps real quan una clàusula d'un contracte SLA ha estat violada. Això permet a l'usuari no haver de fer un seguiment actiu de les clàusules del contracte, per tant fa el sistema més usable.
3. Poder rebre notificacions que expliquin, de manera entenedora per a les persones, les violacions dels contractes SLA ocorregudes. Això permet facilitar el procés de negoci posterior a la detecció d'una violació, cosa que fa el sistema més escalable funcionalment.

A més, aquestes necessitats es volen cobrir d'una manera determinada, com es veurà a ll capítol 4 Anàlisi de requisits (pàgina 27).

1.3.2. Motivacions

En tot projecte s'han de tenir en compte dues motivacions: La del client que contracta la realització del projecte, i la de l'empresa que el realitza. En el cas present, els grups de recerca ISA de la US i GESSI de la UPC es poden considerar el client, i l'autor de la present memòria, l'empresa.

La motivació que ha mogut els grups ISA i GESSI vers aquest projecte és la de desenvolupar una eina que porti a la pràctica molts dels conceptes de l'estat de l'art del camp del monitoratge de la qualitat del servei. Moltes propostes i treballs[1] tracten el monitoratge de documents SLA o de serveis web i la notificació de violacions, però cap d'ells proporciona una eina que tracti el procés de principi a fi, de forma suficientment flexible[2], i a més a més que proporcioni una manera d'explicar les violacions amistosa per a usuaris humans, tal i com s'explica a [SALMonADA_JCR]. Aquesta, doncs, és una oportunitat d'implementar aquesta eina, que servirà com a suport a aquesta línia de recerca en la qual ambdós grups de recerca ja han publicat diversos articles amb èxit.

Per altra banda, la motivació que ha mogut a l'autor és la de realitzar el projecte de final de carrera en camps innovadors (i apassionants per a l'autor) com són el de l'enginyeria de serveis i el de l'enginyeria de sistemes web, per guanyar experiència en aquests camps i aprofitar l'experiència prèvia adquirida durant la col·laboració amb el grup GESSI en el projecte SALMon.

[1] Vegeu 3.1.2 Antecedents (pàgina 15).
[2] De forma canviable i que faciliti la interoperabilitat del sistema i l'agregació d'aquest en sistemes majors. Vegeu 4.3 Requisits (pàgina 16) per a més informació.

1.3.3. Objectius

L'objectiu d'aquest projecte és implementar un sistema software que permeti[1] (a) la configuració del sistema partint únicament de les dades del document SLA, (b) el monitoratge del compliment per part del servei web del contracte SLA, (c) la comprovació en temps real del compliment del document SLA i, en cas que aquest contracte vegi alguna de les seves condicions violada, (d) la notificació automàtica al consumidor sobre la violació d'una clàusula, si així ho ha sol·licitat, d'una manera entenedora, explicitant la clàusula i els motius de la seva violació.

En concret, interessa que el sistema resultant pugui monitorar serveis web descrits en contractes SLA d'un format concret, que ha de ser conforme a l'especificació WS-Agreement, que funcionin sota el protocol SOAP, i que pugui notificar al client interessat, mitjançant una interfície, informació concreta i intel·ligible per a humans sobre la violació ocorreguda (De l'estil "La clàusula: "Temps de resposta màxim < 200 mil·lisegons i disponibilitat mitjana > 90%" s'ha violat perquè el temps de resposta màxim és ara de 300 mil·lisegons"). A més, interessa que el sistema sigui canviable en tant que permeti substituir un o més dels seus components, entre ells els serveis ADA i SALMon, per d'altres de compatibles.

Aquest projecte és de tipus manteniment/evolució d'un sistema informàtic ja existent. En particular, es desglossa en tres parts:

1. **Integració de dos sistemes existents.** El sistema que es busca implementar fa ús de dos sistemes existents: Per una banda, el servei ADA permet cercar violacions de clàusules en contractes SLA. Per altra banda, el servei SALMon permet monitorar serveis web SOAP.
2. **Modificació del sistema SALMon.** Per tal de permetre la integració i plena funcionalitat del sistema resultant, fa falta afegir noves funcionalitats a l'eina SALMon, així com rectificar-ne d'altres.
3. **Generació de manuals.** Per a permetre l'ús futur del sistema, és necessari redactar una documentació completa per a l'usuari. A més, per al futur manteniment del sistema, és necessari documentar el seu funcionament i les consideracions que futurs desenvolupadors hauran de tenir en compte.

1.4. Consideracions del projecte

Aquest projecte s'ha realitzat en el marc de la recerca acadèmica, i és la implementació d'un model conceptual de referència proposat pels grups de recerca ISA i GESSI. Per aquest motiu, certes decisions de disseny han vingut donades per aquest model conceptual, malgrat que a la memòria present s'intenta traçar aquestes decisions als requisits que les han causat.

Aquest projecte, igual que tots, és una col·laboració entre un client que en demana la realització i una empresa que el realitza. En el cas present, ambdues parts han participat activament en el disseny i la construcció del sistema, amb responsabilitats separades. A la secció 4.4 Abast (pàgina 30) s'explica aquesta divisió de responsabilitats, però la resta de la memòria està plantejada des del punt de vista exclusiu de l'autor de la memòria, i per tant aquells mòduls i responsabilitats que no apliquen a aquest es deixen fora de l'abast del projecte i es tracten

[1] Noteu que l'objectiu (a) respon a la necessitat 1, els objectius (b) i (c) a la 2, i l'objectiu (d) a la necessitat 3.

com una caixa negra, explicant com funciona i com s'integra, però no com està feta ni quin procés de desenvolupament va seguir.

Finalment, cal notar que tots els diagrames i manuals de la memòria són en anglès, doncs el projecte s'ha desenvolupat en un grup de recerca de projecció internacional que sol comptar amb la participació de membres no catalanoparlants, i per tant resulta d'interès poder disposar d'aquest coneixement en anglès per a futur ús i referència.

1.5. Organització de la memòria

La present memòria està organitzada en tretze capítols, i descriu el desenvolupament del projecte en les etapes del anomenat "cicle de vida tradicional" del software, també conegut com a model en cascada [CWeis], per facilitar-ne la comprensió.

En l'actual capítol es dona una visió superficial del projecte tot descrivint-ne les necessitats del món real que interessa cobrir, les motivacions per a dur a terme el projecte, els objectius que té el projecte que es deriven d'aquestes necessitats i els detalls sobre l'estructura de la memòria.

Als capítols 2 i 3 s'aprofundeix en el context del projecte, tot explicant conceptes necessaris per a la comprensió d'aquest i el context social, tecnològic i científic en el qual s'ha executat. Es descriuen antecedents tecnològics, els dos projectes que han estat integrats, i l'entorn i eines amb els quals s'ha dut a terme el projecte.

Al capítol 4 s'analitzen els paràmetres del projecte en detall. De l'objectiu es deriva l'abast del projecte, se separa en paquets o mòduls de treball que gaudeixen de certa independència, i s'expliquen els requisits del projecte i els riscs que caldrà tenir en compte.

A partir de l'anàlisi del problema fet al capítol 4, es deriva un pla de projecte concret, del qual es parla al capítol 5, junt amb la metodologia que s'ha seguit per dur-lo a terme.

Als capítols 6, 7, 8 i 9 s'explica la realització del projecte tot detallant i explicant els diferents artefactes generats seguint la metodologia, els problemes trobats, solucions preses i alternatives contemplades.

Al capítol 10 s'analitza com s'ha desencadenat la realització del projecte al llarg del temps, i es fa un estudi econòmic del cost que hauria representat el projecte si aquest hagués estat realitzat per una empresa real.

Finalment, als capítols 11 i 12 es parla del treball futur per a seguir completant el sistema, i les conclusions que s'han derivat de la seva realització, i al capítol 13 hi podem trobar la bibliografia de les referències emprades al llarg de tota la memòria.

1.6. Convencions

Les convencions de format de text de la memòria són:

Itàlica

Noms de classes, components del sistema, nous conceptes que s'estan definint o que es definiran més endavant, referències a elements de llistes i taules.

Negreta

Noms dels elements d'una llista.

"Cita" o informació [Referència] (número de pàgina, traduït?)

Cites o informació extretes de referències bibliogràfiques que es poden consultar al capítol **¡Error! No se encuentra el origen de la referencia. ¡Error! No se encuentra el origen de la referencia.**. De vegades s'indica la pàgina de la referència per comoditat. "Traduït" indica que la cita és textual i ha estat traduïda d'una altra llengua, i qualsevol diferència que hi pugui haver és deguda a la traducció.

2. Qualitat del servei

En aquest capítol es descriu tot el que cal saber sobre la qualitat del servei per a la comprensió d'aquest projecte. Primer es presenta el context en el qual sorgeix el concepte, que és el de les arquitectures orientades a serveis. A continuació s'explica què és la qualitat de servei, com es defineix i s'analitza, com es mesura, i finalment com es controla: Què són i com són els acords de nivell de servei. Una petita conclusió lliga el tema amb l'objectiu del projecte.

2.1. El context: Arquitectures orientades a serveis

Un servei web és, com ja s'ha dit, un sistema software accessible remotament per altres sistemes mitjançant interfícies públiques i protocols estàndard de descripció i comunicació. Tot servei web té un proveïdor que en permet l'ús i consumidors que l'utilitzen quan el necessiten. Per a usar-lo, el consumidor empra un protocol de comunicació que, mitjançant missatges o altres tipus de peticions (per exemple, crides remotes o peticions HTTP), crida a operacions o recursos concrets del servei.

Un servei web, a més, pot ser consumidor d'altres serveis web. Això crea un nou paradigma de l'enginyeria del software que és el de les arquitectures orientades a serveis o *Service Oriented Architectures (SOA)*. Una arquitectura orientada a serveis és una arquitectura software en la qual els components són serveis, i sovint serveis web.

En una arquitectura SOA, els serveis web poden usar altres serveis web en el que s'anomena una *composició de serveis*, teixint així una xarxa de dependències entre ells. Les arquitectures SOA creen composicions de serveis de dues maneres diferents: *Orquestració* i *coreografia* de serveis.

Per una banda, l'*orquestració* de serveis consisteix en un servei web central que utilitza la resta de serveis web, i conté la lògica de negoci que defineix la funcionalitat del sistema i la integració dels altres serveis amb aquest (vegeu la Figura 1).

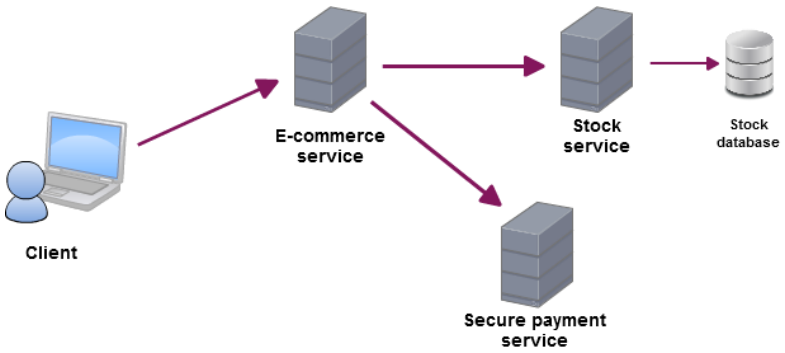

Figura 1: Exemple d'arquitectura orientada a serveis amb orquestració.

Per altra banda, la *coreografia* de serveis és una topologia no centralitzada, en la qual diversos serveis s'usen entre ells i aconsegueixen funcionalitats que ells sols no tindrien per a benefici del conjunt, però sense haver-hi cap punt de vista que sigui conscient del funcionament global i gestioni tot el sistema.

En les arquitectures SOA sol ocórrer que diversos serveis realitzen funcions similars o iguals. Es diu que aquests serveis implementen la mateixa definició de servei. A tall d'exemple, existeixen molts serveis web que realitzen prediccions climatològiques, com ara els que proveeixen AEMET (Agencia Estatal de Meteorología) o Yahoo. Això és un concepte important en la composició de serveis, ja que permet al consumidor escollir entre diversos proveïdors de servei. Diferents serveis poden tenir diferent cost, diferent qualitat del servei o altres pros i contres. Però com es pot saber quins serveis tenen més o menys qualitat? Què és aquesta "qualitat del servei"?

2.2. Què és la qualitat del servei?

La *qualitat del servei* o *Quality of Service (QoS)* és, segons l'estàndard ISO 8402[1] [ISO8402] (traduït), "la totalitat de trets i característiques d'un producte o servei que influeixen en la seva aptitud per satisfer necessitats explícites o implícites". La qualitat del servei és una apreciació subjectiva de com de bé un servei web compleix certes característiques que hom considera desitjables dels serveis web. És un concepte nascut en el paradigma de l'enginyeria de serveis que hereta del concepte de qualitat del software, i que cal no confondre amb el concepte de qualitat del servei en el camp de les xarxes de computadors[2].

Moltes característiques desitjables dels sistemes software ho són també dels serveis web, com ara l'eficiència o la facilitat d'ús, però d'altres no tenen rellevància en el camp dels serveis web, com ara la facilitat d'instal·lació (els serveis web ja estan instal·lats en els servidors del proveïdor i s'usen remotament), i fins i tot hi ha noves característiques que no eren considerades en els sistemes software, com ara la capacitat de col·laboració en orquestracions o coreografies de serveis (un concepte inherent a les arquitectures SOA).

2.2.1. Els models de qualitat

Aquestes característiques desitjables, normalment anomenades *atributs de qualitat*, són difícils de classificar. Per a poder definir el concepte de qualitat del servei i així mesurar-lo, els experts construeixen els anomenats *models de qualitat* o *Quality Models*. "Un model de qualitat del software és un conjunt estructurat d'atributs de qualitat del software" [Oriol] (pàgina 21, traduït). El seu objectiu és agrupar aquests atributs de qualitat en una jerarquia.

Existeixen múltiples models de qualitat per a la qualitat del servei de serveis web. Tots ells intenten definir la jerarquia completa d'atributs de qualitat que composen la QoS. Tots ells ho fan des de punts de vista diferents i amb resultats diferents, i encara no existeix cap consens.

Un exemple de model de qualitat força complet és Web Service Quality Model (WSQM) [WSQM], un model desenvolupat pel consorci OASIS (Organization for the Advancement of Structured Information Standards) i basat en l'estàndard ISO9126[3]. A continuació es dóna una visió general d'aquest model de qualitat, en base al resum que es pot trobar a [Oriol] (pàgina 23).

---

[1] Un estàndard ISO és un conjunt de normes de fabricació de productes o serveis, comercialització o comunicació promogut per l'Organització Internacional de l'Estandardització.

[2] En el camp de la telefonia i les xarxes de computadors, la qualitat de servei es refereix al conjunt d'atributs de qualitat imputables a una connexió, com són el temps de resposta, la pèrdua de senyal, la relació senyal-soroll, la diafonia, l'eco, les interrupcions, la freqüència de resposta, els nivells de volum, etcètera.

[3] L'estàndard ISO9126 conté un model de qualitat per a sistemes software en general.

El WSQM és un model que separa la QoS en tres parts: *Quality Factors* o factors de qualitat, que descriu les característiques i atributs de qualitat per a serveis web; *Quality Activities* o accions de qualitat, que defineix les accions realitzades durant el cicle de vida d'un servei web relacionades amb la seva QoS; i els *Quality Associates* o associats de qualitat, que descriu les persones i organitzacions involucrades en les accions de qualitat.

De les tres seccions, la que resulta rellevant per al projecte és la de factors de qualitat. Aquesta descriu una jerarquia dels atributs de qualitat tal com segueix:

- **Qualitat de valor de negoci:** Atributs que avaluen el grau d'adequació a l'objectiu de negoci. Alguns són el cost, la idoneïtat, el retorn de la inversió i la reputació.
- **Qualitat de mesura de nivell de servei:** Atributs quantitatius que poden ser percebuts pel consumidor d'un servei web i mesurats durant l'ús d'aquest. Els divideix en dos camps:
  - **Rendiment:** Com de ràpid un servei pot respondre les peticions. Es mesura amb el temps de resposta de les invocacions i el nombre màxim de peticions que pot processar per unitat de temps.
  - **Estabilitat:** Com d'estable i continu és un servei oferint la seva funcionalitat. Es mesura amb el temps que està disponible (el servei està mantingut i llest per ser usat), accessible (el servei respon conforme ha rebut una petició, encara que no la respongui) i amb el nombre de respostes correctes que dona.
- **Idoneïtat per als estàndards:** Atributs que indiquen la capacitat per comunicar-se amb altres serveis web en diferents plataformes i sistemes.
- **Qualitat del procés de negoci:** Indicadors de rendiment que representen la funcionalitat per a la col·laboració entre dos o més serveis web. Com a exemples podem trobar la facilitat d'orquestració o coreografia i la capacitat per gestionar els missatges de forma fiable.
- **Qualitat de gestió:** Indicadors de capacitat per a gestionar un servei web de forma consistent. Per exemple, la capacitat per saber l'estat del servei o el valor de les mètriques de rendiment i estabilitat, i la capacitat de configuració del servei web.
- **Qualitat de seguretat:** Atributs que indiquen com de protegit està el servei web davant missatges no autoritzats i atacs de creació o destrucció de dades. Alguns d'ells són la confidencialitat, la gestió d'autenticació i l'auditoria d'esdeveniments.

2.2.2. Les mètriques de qualitat

Tot model de qualitat descriu la QoS com una jerarquia d'atributs de qualitat. Aquests atributs de qualitat són no quantificables per definició: Fan una definició qualitativa i no quantitativa del servei, doncs són una apreciació subjectiva i condicional del comportament del servei web. A tall d'exemple, un usuari impacient o que necessiti resposta urgent percebrà com a molt lent un servei web que hagi trigat un segon a respondre una petició concreta, mentre que un altre que no ho consideri important el percebrà com a suficientment ràpid.

Això no obstant, hi ha certs atributs de qualitat que es veuen influenciats per característiques reals mesurables i quantificables, com per exemple la percepció del temps de resposta es veu influenciada pels temps de resposta de cada petició o per la mitjana de temps de resposta dels últims dies. Aquestes característiques quantificables s'anomenen *mètriques de qualitat*.

Els atributs de qualitat (i, per extensió, les seves mètriques) es poden classificar des de diferents perspectives. Des d'una perspectiva de *domini*[1], podem distingir [Oriol] (pàgina 25, traduït):

- **Atributs independents del domini:** Aquells que es poden aplicar a qualsevol tipus de servei web. Per exemple, el temps de resposta, la disponibilitat i el cost.
- **Atributs dependents del domini:** Aquells que només apliquen a certs dominis. Per exemple, en el cas del domini de la predicció meteorològica, la precisió de la predicció.

Des d'una perspectiva de l'obtenció de la mesura, les mètriques també es poden classificar en tres categories, d'acord amb [Zeng]:

- **Mètriques anunciades pel proveïdor:** Mètriques proveïdes pel proveïdor del servei. Són subjectes a les decisions o percepcions del proveïdor. Un exemple n'és el preu anunciat del servei.
- **Mètriques qualificades pel consumidor:** Mètriques calculades en base a l'avaluació i retroacció dels consumidors del servei, que és subjectiva als consumidors. Un exemple clar en seria la reputació del servei.
- **Mètriques observables:** Mètriques calculades en base a esdeveniments operacionals del servei monitorat, que són objectives tant pel proveïdor com pel consumidor. Com a exemples tenim mètriques de temps de resposta i de disponibilitat.

L'última distinció a notar és, des d'una perspectiva del càlcul de mesures, la divisió de les mètriques segons [AKeller] en:

- **Mètriques de recurs o bàsiques:** Aquelles que poden ser obtingudes directament dels recursos del proveïdor del servei, com ara els servidors o les aplicacions desplegades.
- **Mètriques compostes o derivades:** Aquelles que són calculades a partir de diversos valors de mètriques bàsiques o compostes seguint un cert algorisme. Per exemple, valors mitjans en intervals de temps, mínims, màxims, o funcions més complexes de diferents mètriques.

Malgrat que la distinció entre ambdues sembla a priori prou clara, no ho és[2].

El propòsit d'aquest projecte és el de desenvolupar un sistema software que assisteixi el procés de gestionar la QoS de forma automàtica, per tant, de tots els atributs de qualitat, només interessen aquells que es poden mesurar automàticament, és a dir, són influenciats per mètriques mesurables automàticament. En el cas del model de qualitat WSQM, aquests són principalment els atributs de *Qualitat de mesura de nivell de servei*, i potser algun altre d'altres camps, com ara el cost en circumstàncies determinades.

[1] El domini d'un servei web és el camp semàntic que ocupa la funcionalitat del servei web. Per exemple, els camps del càlcul de divises, la predicció meteorològica o les campanyes de màrqueting per correu electrònic són diferents dominis de serveis web.

[2] Vegeu 6.3 Model conceptual (pàgina 19).

Com s'explica a la secció següent, el model WSQM, malgrat ser bastant complet, és insuficient pel que fa a l'expressivitat sobre la mesura de mètriques per al projecte present. Com s'ha vist, l'objectiu d'aquest projecte són els atributs que tenen mètriques el procés de mesura de les quals es pot automatitzar. Per a automatitzar aquest procés fa falta un software monitor, que mesuri el valor d'aquestes mètriques durant el funcionament del servei. Per a fer-ho, fa falta saber quines mètriques són aquestes i com es poden monitorar, i per a això fa falta veure com funcionen els serveis web.

2.2.3. Funcionament dels serveis web

Un servei web té una interfície que es pot descriure en un document WSDL i que pot rebre i enviar *missatges*.

Un *missatge* és una unitat d'informació que pot ser de dos tipus, petició i resposta, i conté:

- En el cas d'una petició, les dades sobre quina petició es fa al servei web, i totes les dades necessàries per a respondre-la que el servei no tingui.
- En el cas d'una resposta, les dades sobre quina petició s'està responent, i la informació de la resposta.

La comunicació amb missatges pot ser síncrona (una petició espera una resposta) o asíncrona (s'envien peticions i respostes sense un ordre estricte).

El document WSDL descriu quines peticions es poden fer al servei, quines dades s'han d'incorporar al missatge, si respon de forma síncrona, i quina informació incorpora la resposta. En altres paraules, és el contracte d'una interfície amb diferents operacions, i per a cadascuna d'elles té la seva firma (nom, paràmetres i tipus de retorn).

2.2.4. Monitoratge de mètriques

Monitorar una mètrica significa obtenir el valor d'aquesta mètrica de forma contínua (detectant en temps real el canvi de valor) o discreta (calculant el valor en el moment que succeeix un esdeveniment).

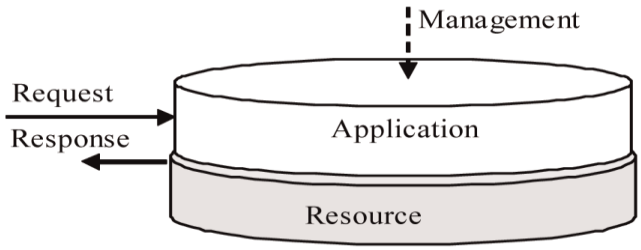

Figura 2: Model Mill, que captura els elements que s'han de monitorar en un servei web segons [QWang].

Per entendre què es pot mesurar d'un servei web, es poden classificar les mètriques de diferents maneres. Un exemple de classificació és el model *Chinese Stone Mill* (molí de pedra xinès) que proposa [QWang], que planteja un servei web com un sistema tancat de cinc parts (vegeu la Figura 2):

- **Request Message (missatge de petició):** Els missatges de petició són l'entrada d'informació als serveis web, i determinen el seu funcionament a molts nivells. Re-

bre moltes peticions en un moment de diferents clients pot afectar al rendiment; l'ordre en què es rebin o la seva correctesa poden ser paràmetres decisius per determinar la qualitat del servei; etcètera.

- **Response Message (missatge de resposta):** Els missatges de resposta són el que rep el consumidor del servei, i per tant són l'únic mitjà pel qual el consumidor pot percebre la qualitat del servei directament. L'article diferencia tres tipus de mètriques, de forma no exclusiva, que es poden extreure dels missatges de resposta:
  - **Availability (disponibilitat):** Mètriques que determinen si el consumidor pot rebre el missatge de resposta, sigui correcte o no.
  - **Correctness (correctesa):** Mètriques que determinen si la resposta a la petició del client és correcta. Solen ser dependents de la implementació.
  - **Efficiency (eficiència):** Mètriques que determinen si el consumidor pot rebre la resposta a una petició sota certes restriccions, com ara un límit de temps o de precisió.
- **Application (aplicació):** El funcionament intern de l'aplicació també influeix en la qualitat d'un servei web. L'estat intern de l'aplicació és interessant de conèixer perquè pot haver-hi lògica de negoci errònia o altres tipus de problemes.
- **Resource (recurs):** El funcionament del servei web depèn, en última instància, dels recursos de baix nivell sobre els que es desplega: CPU, memòria, sistema de fitxers, capacitat de la xarxa i altres.
- **Management Operation (operacions de gestió):** Molts serveis web disposen d'una interfície de gestió separada de la d'usuari, que permet a l'administrador configurar certs paràmetres. Aquests esdeveniments també poden afectar al funcionament del servei i s'han de tenir en compte.

Per a monitorar una mètrica cal tenir una *sonda* (també anomenada sensor o instrument de mesura). Una sonda és una entitat que observa el comportament d'una part de l'espai d'estats d'una aplicació [BASchroeder], per exemple un conjunt de variables. Una sonda pot ser accionada de dues maneres:

- Per un canvi en l'entitat que observa (*Monitoring*) o
- Per una petició del sistema monitor (*Testing)*.

En ser accionada, genera un esdeveniment que serà capturat i tractat pel monitor.

Es pot veure un exemple de classificació de les sondes a [QWang], on es separen en dos tipus segons la seva localització respecte al sistema (vegeu la Figura 3):

- **Interceptor-based (interceptores):** El codi de la sonda és extern a l'aplicació, i observa de manera no intrusiva les interaccions de l'aplicació amb l'entorn (usuaris, altres aplicacions...). L'aplicació pot no ser conscient d'aquest tipus de sondes, però no poden monitorar l'estat intern de l'aplicació si aquesta no el transmet.
- **Instrumentation-based (desplegades):** El codi de la sonda és imbricat dins el codi de l'entitat a observar. Aquest tipus de sonda pot monitorar qualsevol part de l'aplicació. El desplegament de la sonda, però, ha de ser en temps de compilació.

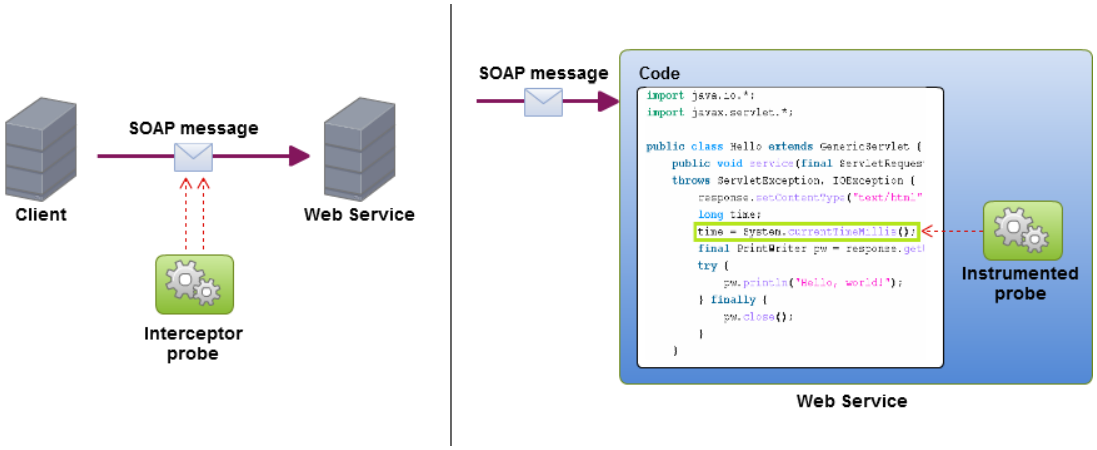

Figura 3: Exemple que il·lustra la diferència entre sondes interceptores i desplegades. Les primeres no necessiten modificar el codi intern del sistema monitorat, però no poden mesurar mètriques que les desplegades sí que poden.

En el cas dels serveis web, les sondes interceptores poden monitorar missatges de petició i de resposta, i possiblement operacions de gestió, però per monitorar l'aplicació i el recurs fan falta sondes desplegades.

Les sondes interceptores monitoren entitats que en sí mateixes són esdeveniments (els missatges de petició i resposta, per exemple). A causa d'això, el testing és diferent en aquest tipus de sondes. Monitorar un esdeveniment en un moment arbitrari no és possible perquè pot ser que en aquell moment l'esdeveniment ja hagi succeït o encara hagi de succeir. Per a solucionar-ho, quan s'acciona la sonda s'ha de produir un esdeveniment artificial. En els serveis web, el cas més usual és el d'enviar un missatge de petició al servei des del sistema monitor i rebre'n la resposta, com si fos un consumidor artificial del servei.

2.3. Acords de nivell de servei

Sabent quines mètriques es poden conèixer, i com es poden monitorar, es pot establir un mètode per controlar-ne els valors. Aquest mètode és la negociació d'acords de nivell de servei.

Un *acord de nivell de servei* o *Service Level Agreement (SLA)* és un acord entre dues parts, és a dir, proveïdor i consumidor, on estableixen quin servei s'ofereix i quines prioritats, responsabilitats i garanties tindrà cadascuna de les parts al respecte. El proveïdor del servei, en particular, pot tenir responsabilitats i oferir garanties envers la qualitat del servei.

Els acords de nivell de servei s'expressen en forma de documents SLA, que poden ser contractes vinculants a nivell legal o acords informals, i poden ser expressats en llenguatge col·loquial o en llenguatges estàndard de documents SLA, com per exemple WS-Agreement.

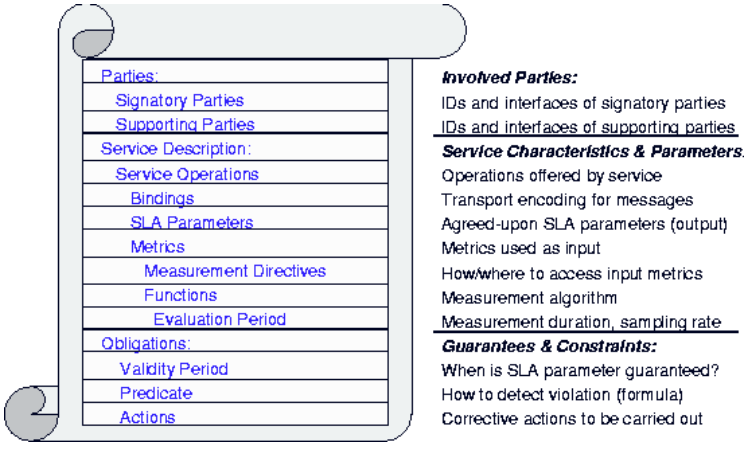

Figura 4: Estructura general d'un SLA tal i com apareix a [AKeller2].

L'estructura general d'un document SLA, segons [Ludwig] i [AKeller2], es composa de tres seccions (vegeu la Figura 4):

- **Parts signants:** Descriu, per a cada part signant (i terceres parts involucrades), la seva identificació i les propietats tècniques, com són l'adreça i la definició de la interfície.
- **Descripció del servei:** Descriu les característiques del servei (protocols i operacions) i les mètriques que se'n poden monitorar. Les mètriques, al seu torn, són descrites per:
    - **Nom, tipus i unitat:** Tota mètrica té un nom que la descriu, un tipus de valor (numèric, booleà, percentual, etc.), i una unitat en la qual s'expressen les mesures (mil·lisegons, bytes, errors, missatges al mes, etc.).
    - **Directives de mesura:** Instruccions de com monitorar la mètrica, com ara la direcció URI d'un programa, un missatge d'un protocol o una comanda per invocar programes que la monitorin.
    - **Funcions:** L'algorisme o fórmula matemàtica que descriu com calcular la mètrica en base a altres mètriques, en cas que sigui composta.
    - **Període d'avaluació:** El període i la freqüència amb els quals s'hauria de monitorar la mètrica.
- **Obligacions:** Garanties i restriccions sobre els valors de les mètriques. Una obligació pot ser de dos tipus:
    - **Objectiu de nivell de servei o *Service Level Objective (SLO)*:** També anomenat clàusula. Representa una promesa respecte a l'estat d'alguna mètrica. Un SLO es separa en:
        - **Període de validesa:** Conjunt d'intervals de temps durant els quals aplica.
        - **Predicat:** Condició booleana sobre el valor de la mètrica (p.ex. menor que 100).
    - **Acció o *Action Guarantee*:** Mesures a prendre quan el predicat d'un SLO no es compleix (p.ex. informar a una part signant, pagar una recompensa o pagar una penalització).

Es diu que una clàusula d'un document SLA es viola quan el predicat d'un SLO esdevé fals. Per a comprovar si és fals, cal monitorar les mètriques que apareixen al predicat de l'SLO.

En cas que sorgeixi una violació en un document SLA, pot ser convenient explicar:

- Quines clàusules han estat violades,
- quina ha estat la causa de la violació (quines mètriques han fet fallar el predicat, i quin és el valor conflictiu), i
- quins són els possibles causants (p.ex. una manca de disponibilitat pot ser deguda a problemes de xarxa, mentre que un baix temps de resposta pot ser degut a una saturació del servei, entre d'altres).

2.4. Monitoratge i explicació de violacions de documents SLA

Com influeix tot això en aquest projecte? L'objectiu d'aquest projecte és, com ja s'ha dit, el de poder monitorar aquests documents SLA, i en cas que un o més dels seus SLO siguin violats, oferir totes aquestes explicacions a les parts interessades.

Per a fer això cal realitzar una plataforma que sigui capaç de monitorar totes les mètriques que poden aparèixer als SLOs del document (vegeu la Figura 5), i que es poden monitorar de forma automàtica per a serveis web d'una arquitectura SOA. Aquestes mètriques poden ser de Request i Response i també de l'estat intern del servei web, i per tant ha de permetre fer-ho mitjançant sondes interceptores i desplegades. Aquest monitoratge s'ha de poder fer tant en temps real (Monitoring) com sota demanda (Testing). Val a dir, però, que cal afegir restriccions al format dels documents SLA per tal que això sigui factible.

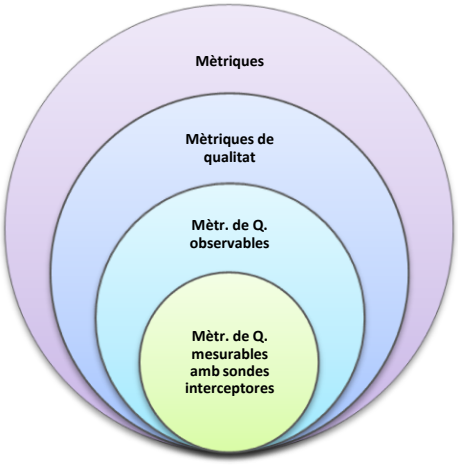

Figura 5: Una classificació de les mètriques. Un dels objectius del projecte és fer que SALMon pugui monitorar mètriques de qualitat, doncs inicialment només mesura algunes d'entre aquelles mesurables amb sondes interceptores.

En els següents capítols s'explica amb més detall com es vol fer, i què s'ha de fer exactament, començant pel punt de partida: Antecedents i context del projecte.

3. Context del projecte

Aquest capítol explica el context en el qual neix el projecte. Primerament descriu els antecedents del projecte i com va sorgir l'oportunitat. Després es descriuen les eines ja existents que s'han estès i aprofitat per al desenvolupament del projecte, junt amb l'entorn de desenvolupament que s'ha usat i les eines i plataformes de tercers que han estat necessàries per a dur-lo a terme.

3.1. Situació inicial

L'oportunitat de realitzar aquest projecte va néixer quan l'autor de la present memòria ja treballava en el projecte SALMon del grup de recerca GESSI. A continuació es detalla el context del punt de partida del projecte.

3.1.1. Col·laboració

La idea del projecte neix en el context de la xarxa europea Software Services and Systems Network (S-Cube)[1], que es va fundar de l'1 de març del 2008 fins al 29 de febrer del 2012, i va servir d'impuls per als grups de recerca per investigar sobre el món dels serveis web i establir contactes entre ells.

Entre aquests grups hi havia els grups GESSI i ISA, que van descobrir que compartien interessos comuns i tocaven temes similars, i van decidir començar una col·laboració i veure com podien integrar esforços. Així va sorgir la idea de desenvolupar el projecte SALMonADA, el nom del qual és un acrònim dels noms dels projectes sobre serveis web dels dos grups, SALMon i ADA, respectivament.

La idea era crear un model de referència[2] que proporcionés una arquitectura software de referència per a una plataforma per al monitoratge de documents SLA, i demostrar la seva viabilitat instanciant el model de referència amb una implementació funcional d'aquesta plataforma: SALMonADA.

Durant el desenvolupament del projecte, els grups ISA i GESSI van publicar amb èxit dos articles científics que parlen del projecte: "SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents" [SALMonADA_PESOS] i "Comprehensive Explanation of SLA's Violations at Monitoring Time" [SALMonADA_JCR].

3.1.2. Antecedents

L'anàlisi d'antecedents s'ha separat en dos vessants: Eines comercials i eines en el camp de la recerca.

Per a l'anàlisi d'antecedents en eines comercials s'ha emprat el motor de cerca Google amb les consultes "sla monitor" i "sla violation monitor". Els resultats s'han pogut classificar en tres categories:

[1] http://www.s-cube-network.eu/

[2] Un *model de referència* és un marc de treball abstracte o una ontologia específica d'un àmbit que consisteix en un conjunt de conceptes interrelacionats clarament definits produït per un o més experts amb la intenció de fomentar la comunicació entenedora.

- Els **rellevants**, que han estat quatre: op5, Uptrends, Oracle Service Bus i Zyrion Traverse. De tots ells, només Oracle Service Bus és capaç de monitorar serveis web pròpiament dits, mentre que els altres són monitors de xarxa i aplicacions distribuïdes a un nivell de protocol més baix.
- Els **articles de recerca**, que s'han considerat irrellevants perquè es tracten a la segona part de l'anàlisi.
- Els **irrellevants**, compostos per resultats que no presentaven cap eina, i per eines que només monitoraven a nivell IP, que s'anomenen monitors de IP SLA, i que queden fora de l'abast del problema ja que tracten el concepte de "qualitat de servei" en comunicació en xarxes, un concepte paral·lel al que es tracta aquí però més restringit.

D'entre els rellevants, es pot observar que, tot i que tots ells presumeixen de poder monitorar SLAs, cap d'ells considera el SLA com un document independent de l'eina (vegeu la Taula 1), sinó més aviat un conjunt de regles que es poden configurar a l'eina, i que en cas que alguna es deixi de complir, l'eina notifica a l'usuari. Per tant, cap de les eines comercials analitzades és capaç d'interpretar un document SLA en cap format i extreure'n aquestes clàusules automàticament.

Per a l'anàlisi d'antecedents en eines s'ha partit de l'anàlisi de l'estat de l'art de l'article Comprehensive Explanation of SLA Violations at Monitoring Time [SALMonADA_JCR]. Malgrat tot, l'anàlisi dels articles no ha entrat en l'abast d'aquesta memòria, ja que cap d'ells sembla tenir una eina implementada accessible a l'anàlisi.

La taula presentada en el mateix article s'ha fet servir de base per a l'anàlisi, doncs l'article presenta el model de referència que aquest projecte busca instanciar amb SALMonADA, i per tant els objectius del projecte i de l'article són exactament els mateixos.

		Functionality				**Structure**	
PROPOSALS	Availability	Supported SLAs	Monitor config.	Monitoring results	Explanation of violations	Architecture elements	Architecture structure
op5	Commercial tool (open source)	Absence SLA	Not SLA	No	Detection	Monitor and Analyzer in 1 component	CBS
Uptrends	Commercial tool	Absence SLA	Not SLA	Through API	Detection and explanation. H-U.	[N/A]	[N/A]
Oracle Service Bus	Commercial tool	Absence SLA	Not SLA	Through API and query lang.	Detection and explanation. H-U.	Monitor and Analyzer separated	SOA
Zyrion Traverse	Commercial tool	Absence SLA	Not SLA	Through API	Detection and explanation. H-U.	Monitor and Analyzer separated	CBS

Taula 1: Anàlisi d'antecedents.

3.1.3. ADA

Agreement Document Analyzer[1] (ADA, vegeu Figura 6) és una eina desenvolupada pel grup de recerca ISA de la US que permet analitzar documents SLA sobre serveis web SOAP expressats

[1] http://www.isa.us.es/ada/modules/portalWFInterface/init.php

en el llenguatge WS-Agreement, un llenguatge de descripció de SLAs basat en eXtensible Markup Language (XML)[1].

Figura 6: Logotip de la plataforma ADA.

Les seves dues funcions bàsiques són:

- Detectar si un document SLA té errors i
- comprovar si es pot establir un acord entre dues parts que respecti ambdós documents SLA.

En el primer cas, s'analitzen les condicions dels SLOs que conté un document SLA que ADA anomena *plantilla*, i si hi ha contradiccions o restriccions que fan que l'SLA no pugui ser respectat amb independència del context, aleshores ADA les mostra.

Per al segon cas, ADA compara un document SLA que anomena *oferta* contra la plantilla, i si no hi ha cap cas en què ambdós documents puguin ser respectats simultàniament, ADA pot explicar perquè.

ADA es pot usar de quatre maneres diferents: Mitjançant una interfície web per a usuaris finals, com a servei web SOAP, mitjançant *OSGi*[2], i com a biblioteca per al llenguatge de programació Java.

Un dels grans beneficis d'ADA és un format de fitxer propi que s'anomena *WS-Ag4people*, que és un llenguatge de text pla per a definir documents SLA escrits en WS-Agreement de forma entenedora per als humans i senzilla d'editar. La interfície web permet visualitzar els documents en aquest format.

3.1.4. SALMon

Service Level Analyzer Monitor[3] (SALMon, vegeu la Figura 7) és una eina desenvolupada pel grup de recerca GESSI de la UPC que permet monitorar i fer testing de serveis web, així com analitzar posteriorment la qualitat del servei.

SALMon considera que cada servei web es composa de diferents operacions, i cada operació d'un servei web concret té mètriques que seran monitorades.

---

[1] XML és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C), que és llegible per humans i per programes. Avui dia és un estàndard molt estès i per tant resulta fàcil de tractar.
[2] Open Services Gateway initiative (OSGi) és una plataforma per a Java que permet el desplegament automàtic de diferents components software de manera remota.
[3] http://gessi.lsi.upc.edu/salmon/web/index.jsp

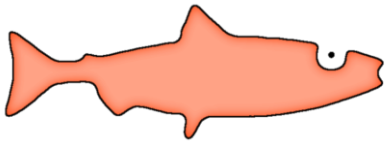

Figura 7: Logotip de la plataforma SALMon.

En l'instant d'inici del projecte, l'arquitectura de SALMon (vegeu Figura 8) consta de tres serveis web que comparteixen una mateixa base de dades: *Analyzer* (analitzador), *Monitor* i *Proxy*.

- El servei Analyzer permet registrar condicions sobre mètriques una a una, i configura el servei Monitor per a que monitori les mètriques.
- El servei Monitor permet registrar serveis web, les seves operacions que es volen monitorar i les mètriques que se'n vol obtenir. També permet obtenir tots els valors monitorats fins al moment d'una operació, i els missatges de petició i resposta de cada crida SOAP.
- El servei Proxy[1] permet enviar un missatge SOAP a un servei web concret si es troba registrat pel Monitor, i en calcula i emmagatzema la QoS amb les mètriques que el Monitor hagi registrat. També permet la simulació de valors de QoS per a mètriques concretes.

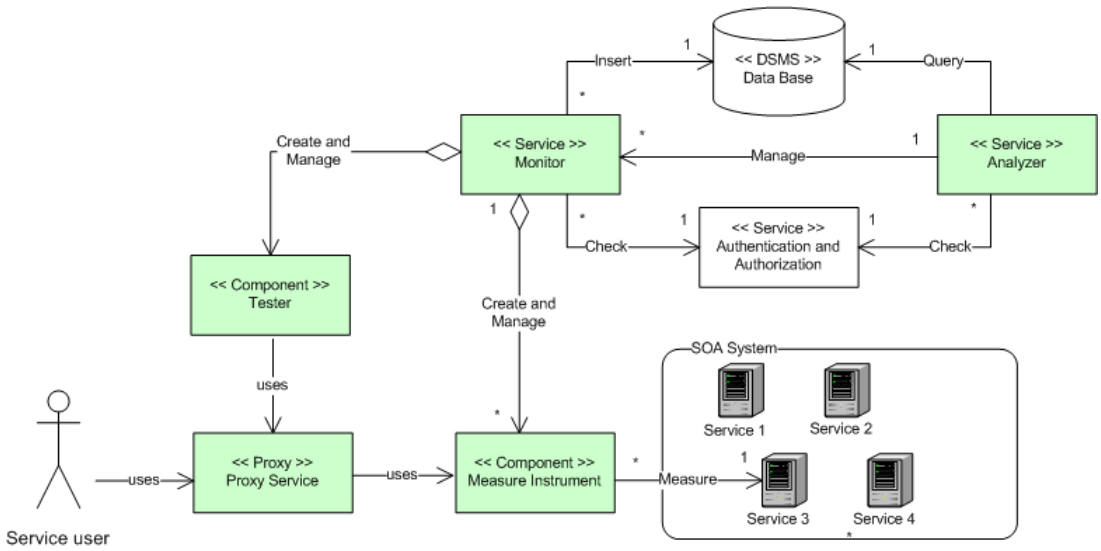

Figura 8: Arquitectura bàsica de SALMon com es descriu a l'article [SALMon].

En l'instant inicial, SALMon permet el monitorat únicament de serveis web SOAP. Les mètriques que és capaç de monitorar són tres mètriques bàsiques i les seves derivades:

- **ResponseTime** o temps de resposta: El temps que passa entre que el servei Proxy envia el missatge al servei web monitorat i en rep la resposta.
- **Availability** o disponibilitat: Cert si el servei web monitorat retorna una resposta funcional (és a dir, que ha arribat a començar a executar la operació) dins d'un

[1] El servei *Proxy* (anglès per a "representant") rep aquest nom perquè actua com a intermediari entre el client i el proveïdor, i idealment les parts que desconeguin la presència de software monitor al sistema no se n'han d'adonar, com si es tractés d'un servidor proxy HTTP. Vegeu 7.1.4 Model de desplegament (pàgina 27).

temps màxim escollit per l'administrador de SALMon en registrar el servei web a monitorar.

- **RoundTripTime**[1]: El temps que passa entre que el servei Proxy demana el document WSDL del servei web monitorat i el rep. Aquesta mètrica resulta útil en cas que el document WSDL sigui disponible al mateix servidor que el servei, i serveix com a temps de resposta del servidor sense tasques costoses.

Les mètriques que SALMon deriva d'aquestes són: Per al ResponseTime i el RoundTripTime, "mínim històric" (és a dir, el mínim d'entre tots els valors de la mètrica per a aquella operació), "màxim històric" i "mitjana absoluta"; i per a la Availability, "percentatge de temps de disponibilitat", "percentatge de temps de no disponibilitat" i "temps mig de recuperació".

En el punt de partida, SALMon té un únic tipus d'interfície, que és com a arquitectura orientada a serveis web SOAP, i per tant s'ha de configurar i fer servir mitjançant missatges SOAP.

3.2. Entorn de desenvolupament

El projecte s'ha desenvolupat en un entorn dividit en quatre àrees: Producció, pre-producció, local i desenvolupament. En totes elles, el conjunt de serveis web de l'arquitectura s'executa damunt d'un mateix servidor *Apache Tomcat* (diferent per a cada entorn), mentre que es fan servir dos servidors *Apache HTTP Server* per a una interfície web i un *Enterprise Service Bus*. Vegeu 3.3 Eines, plataformes i estàndards a continuació per a més informació de tots ells.

Cadascun dels quatre entorns té una funció (vegeu la Figura 9):

- L'entorn de **desenvolupament** va consistir en l'entorn d'edició de l'ordinador portàtil de l'autor de la memòria, amb sistema operatiu Windows 7. Serveix per implementar el sistema. Una vegada implementat, es desplega en local.
- L'entorn **local**, que fou l'entorn de desplegament de l'ordinador portàtil de l'autor de la memòria. Serveix per fer proves unitàries i generals del sistema. Un cop el sistema supera aquestes proves, es desplega a pre-producció. Si es detecta un problema, se soluciona a l'entorn de desenvolupament.
- L'entorn de **pre-producció** que es va usar va ser un servidor Tomcat dedicat en un entorn Ubuntu 10.04.4 (Lucid Lynx) del Laboratori de Càlcul de la UGDSI privat. L'entorn de pre-producció simula tan fidelment com sigui possible l'entorn de producció real, i serveix per desplegar-hi el sistema i fer-hi proves realistes sense donar accés al usuari final ni comprometre el funcionament actual del servei de producció. Si es detecta un problema a pre-producció, es torna al desenvolupament per a reparar-lo.
- L'entorn de **producció** va consistir en un altre servidor de-

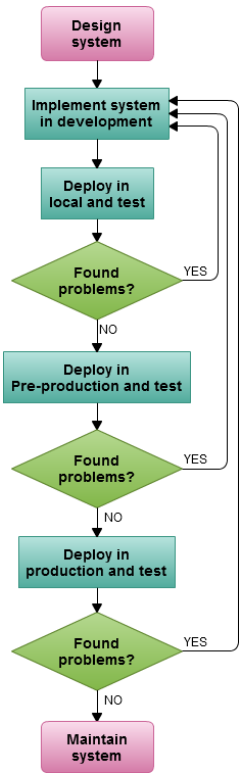

Figura 9: Cicle de vida del desenvolupament.

[1] És incorrecte dir que el RoundTripTime és una mètrica d'un servei "A". En realitat és la mètrica ResponseTime de la operació HTTP GET d'un servei web REST "B" que ofereix el WSDL del servei "A".

dicat del Laboratori de Càlcul de la UGDSI d'iguals característiques al de pre-producció, accessible públicament. Producció serveix per oferir el servei a l'usuari final, i per tant qualsevol sistema que s'hi desplegui ha d'estar provat al màxim. Això no obstant, la realitat sempre supera les expectatives, i un sistema desplegat a producció també es considera subjecte a proves. Qualsevol problema que es detecti a producció fa tornar al desenvolupament per a solucionar-lo.

A més, es va disposar d'un servidor *Subversion* dedicat del Laboratori de Càlcul de la UGDSI, que es va fer servir com a dipòsit (en anglès, "*repository*") del codi i eina per al desplegament del sistema més eficient.

3.3. Eines, plataformes i estàndards

A continuació es llisten les eines que s'han fet servir per desenvolupar el projecte (especificar, dissenyar, implementar i provar), les plataformes sobre les quals s'ha construït el sistema, i els estàndards que aquest empra.

Eclipse

Un dels entorns de desenvolupament integrats més famosos. Inclou eines per al desenvolupament d'aplicacions en llenguatge Java, malgrat tenir extensions per altres llenguatges, com ara per provar i depurar el codi. També s'integra a la perfecció amb altres plataformes, com Axis2 i Tomcat. La major part del codi del sistema s'ha escrit i depurat sobre Eclipse. La versió usada ha estat la 3.3.2 per a Microsoft Windows XP x86-32, anomenada Eclipse JEE Europa Winter.

http://www.eclipse.org/

Figura 10: Logotip d'Eclipse.

jEdit

Un editor de fitxers de text lleuger i pràctic. Reconeix i dóna color a la sintaxi d'un gran nombre de llenguatges, i permet comparar fitxers a pantalla partida. S'ha usat en el projecte per a l'edició de fitxers de text variats, com ara fitxers script en Ruby, documents SLA, WSDL i altres documents XML.

http://www.jedit.org/

Figura 11: Logotip de jEdit.

Cacoo

Un editor en línia de models i diagrames variats. És senzill i de gran potència, però no aporta cap eina de verificació de models software ni de generació de codi, com faria una eina CASE. S'ha usat per a gran part dels diagrames i figures de la memòria.

https://cacoo.com/

Figura 12: Logotip de Cacoo.

Dropbox

Un servei d'emmagatzematge i intercanvi de fitxers en el núvol. S'ha usat per a disposar dels fitxers del projecte que no fossin codi a diferents ordinadors, i mantenir-ne a més una còpia de seguretat.

https://www.dropbox.com/

Figura 13: Logotip de Dropbox.

Secure Shell Client

Un terminal del protocol SSH per a Microsoft Windows que permet establir connexions remotes amb aquest protocol per línia de comandes o mitjançant una interfície gràfica. S'ha fet servir per a desplegar el sistema a producció i pre-producció.

Toad for MySQL

Una eina de gestió de bases de dades MySQL mitjançant una interfície gràfica. S'ha fet servir per a gestionar les bases de dades als entorns de desenvolupament i local.

http://www.toadworld.com/

Microsoft Word 2007

Un processador de text de format propietari. S'ha usat per a l'edició i maquetat de la memòria.

Microsoft Project 2010

Un gestor de projectes de format propietari. S'ha usat per a l'edició dels diagrames de Gantt i per gestionar el projecte.

Trace Modeler 1.6.6

Un senzill editor de diagrames de seqüència de format propietari, amb interfície gràfica d'usuari d'estil "drag and drop" (arrossegar i deixar anar). S'ha usat per editar els diagrames de seqüència de la memòria.

http://www.tracemodeler.com/

Apache Axis2

Software intermediari[1] per a la recepció i enviament de missatges SOAP i la generació de codi per a serveis web SOAP i clients SOAP a partir de documents WSDL. Igual que Tomcat, també es pot fer servir com a un servidor d'aplicacions web independent. S'ha fet servir per a generar les interfícies de tots els serveis web del sistema i les classes que actuen de client dels serveis web. La versió usada ha estat la 1.3.

Figura 14: Logotip d'Axis2.

http://axis.apache.org/axis2/java/core/

Apache Synapse

Un *Enterprise Service Bus*[2] (ESB) que pot actuar com a proxy d'un servei web i permet gestionar els missatges que rep. Entre altres coses, permet replicar-los, modificar-los, i canviar-ne la destinació. Incorpora un servidor Apache Axis2. S'ha usat com a sonda interceptora entre el client i proveïdor. Per a més detalls vegeu 7.1.1 Arquitectura dirigida pels esdeveniments (pàgina 67). La versió usada ha estat la 1.2.

Figura 15: Logotip de Synapse.

http://synapse.apache.org/

[1] Un *software intermediari* ("middleware" en anglès) és una capa de software que es troba entre el sistema operatiu i l'aplicació, i que simplifica les operacions de comunicació i integració entre aplicacions.

[2] Un *ESB* és una arquitectura que facilita la integració de diversos sistemes que es comuniquen mitjançant protocols diferents. Funciona mitjançant un bus o canal central que comunica els diferents sistemes, tradueix els missatges entre protocols i els redirigeix seguint unes regles configurades. Sovint, quan es parla de ESB, es parla del software que implementa aquest canal central.

Apache Tomcat

Un servidor web i contenidor de *servlets*[1]. Tots els serveis web del sistema s'han desplegat sobre Tomcat mitjançant Axis2. La versió usada en local ha estat la 5.5. La versió usada en producció ha estat la 6.0.24.

Figura 16: Logotip de Tomcat.

http://tomcat.apache.org/

MySQL

Un sistema gestor de bases de dades relacionals de codi obert que funciona amb llenguatge SQL. S'ha usat per a totes les bases de dades relacionals del projecte, que són totes excepte un sistema de fitxers.

Apache DBCP

Una biblioteca Java d'Apache Commons continguda dins d'Apache Tomcat que implementa *pools*[2] de connexions a bases de dades. S'ha usat per controlar la concurrència d'accés a les bases de dades de tot el sistema.

http://commons.apache.org/proper/commons-dbcp/index.html

WebSocket

Protocol d'intercanvi de missatges entre un client i un servidor HTML mitjançant una sola connexió bidireccional. Elimina les restriccions de l'arquitectura client-servidor, permetent fer *Server Push*[3]. S'ha usat per implementar la publicació de violacions a la interfície web.

http://www.websocket.org/

HTML5

La cinquena versió del popular llenguatge de marcatge d'hipertext, HTML, el qual s'usa a les planes web per tal de donar informació al navegador sobre com mostrar la plana. S'ha usat per al desenvolupament de la interfície web perquè HTML5 disposa de reconeixement natiu de la tecnologia WebSocket.

Figura 17: Logotip de WebSocket.

Jetty v8.1

Contenidor d'aplicacions web Java del grup Eclipse. És similar a Tomcat, i té suport natiu de la tecnologia WebSocket. Pel fet de ser Java, permet una fàcil integració amb SOAP mitjançant Apache Axis2, característica necessària per comunicar-se amb els serveis web del sistema. S'ha usat per al desenvolupament de la interfície web mitjançant servlets convencionals.

Figura 18: Logotip d'HTML5.

http://www.eclipse.org/jetty/

[1] Un *servlet* és una classe Java que defineix el comportament d'un servidor web en rebre peticions HTTP.
[2] Un *pool* és un magatzem d'algun tipus de recurs software, com ara connexions a una base de dades, que emmagatzema instàncies ja inicialitzades d'aquest recurs i les manté preparades per a fer servir, en comptes de crear-ne i destruir-ne sota demanda.
[3] Tècnica mitjançant la qual un servidor envia informació a un client sense que aquest l'hagi hagut de demanar.

WS-Agreement

Especificació d'un protocol de serveis web per acordar contractes entre dues parts, tals com un proveïdor i un consumidor d'un servei, usant una extensió del llenguatge XML subjecta a un esquema concret.

http://www.ogf.org/documents/GFD.107.pdf

SOAP

Protocol estàndard del W3C[1] per a l'intercanvi d'objectes entre aplicacions mitjançant missatges XML compostos d'unes capçaleres i un cos, dirigits a operacions. El protocol està pensat per què les operacions i tipus de missatges acceptats per un servei siguin definits en WSDL.

http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

[1] *World Wide Web Consortium*, l'organisme encarregat de decidir els estàndards en el món web.

4. Anàlisi de requisits

Aquest capítol fa un anàlisi dels requisits del projecte, començant per analitzar els paràmetres genèrics que descriuen el projecte, i descriu a partir dels requisits l'abast del projecte en detall.

Els requisits s'han establert en base als objectius del projecte mitjançant un procés d'anàlisi iteratiu amb el client durant tot el projecte. S'ha analitzat el procés de negoci inicial i com ha de quedar automatitzat pel nou sistema, i s'han extret els requisits d'aquest anàlisi i de reunions amb el client on s'ha intentat garantir que tots els requisits quedaven reflectits i no n'apareixien d'innecessaris o redundants.

4.1. Planificació de projectes

Es pot definir un projecte com l'acció d'assignar uns certs recursos durant un temps determinat a treballar per assolir un objectiu concret [PGPSI]. Partint d'aquesta definició, ens adonem que un projecte té tres paràmetres clars que en determinen la seva viabilitat: Objectiu, recursos i temps.

En el món hi ha necessitats ja existents, i d'altres poden ser creades. Una necessitat es transforma en una oportunitat quan es disposa de temps, recursos i un objectiu concret que cobreix la necessitat.

Per determinar si un projecte és viable cal realitzar un anàlisi d'aquesta oportunitat. Un anàlisi d'oportunitat és un estudi sobre els paràmetres d'un projecte i el context (espacial, temporal, cultural...) en el qual ha sorgit l'oportunitat de realitzar-lo que intenta determinar si aquest projecte serà factible o no. Normalment, en una empresa, les oportunitats de negoci són analitzades i, en cas que el projecte es consideri factible, s'elabora un pla de projecte i, si tot va bé, s'inicia.

Un anàlisi d'oportunitat consisteix, primerament, en determinar l'objectiu del projecte i quin temps i recursos són necessaris per dur-lo a terme, i donar una prioritat al projecte. Si, dins de la cartera de projectes (més conegut en anglès com a *"project portfolio"*) de l'empresa el projecte pot rebre aquests temps i recursos sense prendre'ls a d'altres projectes amb més prioritat, aleshores és un bon candidat.

Cal tenir en compte, a més, l'ordre de prioritat dels paràmetres objectiu, recursos i temps, ja que hi ha una regla no escrita que diu que, en tot projecte, un dels tres paràmetres com a mínim fallarà per escassetat (no s'arribarà a complir tot l'objectiu, es necessitaran més recursos o més temps). Cal saber de bon començament, doncs, quin és el més important i quin el que s'està més disposat a sacrificar.

A continuació cal analitzar la situació del punt de partida, com s'ha fet al capítol anterior, l'objectiu destí i la utilitat o necessitat del projecte. Si es considera que pot tenir èxit, és potencialment viable.

Un dels mètodes per a jutjar si un projecte pot tenir èxit o no és la *Regla de les tres U* [PGPSI]:

- **Útil:** Un projecte ha de respondre a una idea útil, és a dir, que cobreix una necessitat, ja existent o nova.
- **Utilitzable:** Un projecte ha de generar un producte o servei utilitzable, és a dir, que tingui una qualitat que permeti que un usuari el pugui usar satisfactòriament.
- **Utilitzat:** Finalment, un projecte té èxit si acaba sent utilitzat per la gent (com més, millor), independentment de si ha provocat guanys o pèrdues econòmiques. Per a ser utilitzat, un producte o servei ha de ser útil i utilitzable, i ha de fer-se un lloc en el món (i aquesta sol ser la part més complicada).

4.2. Anàlisi d'oportunitat

Els paràmetres del projecte són els següents:

- **Objectiu:** Automatitzar el procés de negoci de detecció i notificació de violacions d'un SLA (vegeu la secció 4.2.1 Procés de negoci tot seguit).
- **Recursos:** Els recursos infraestructurals s'expliquen al capítol 3.2 Entorn de desenvolupament. Els recursos econòmics no apliquen, ja que és un projecte de final de carrera, però es podria considerar com a fita superior el preu màxim per hora (que seria el cas del cap de projecte) de les hores dels crèdits del projecte, que són $37{,}5\ crèdits \times 20 \frac{hores}{crèdit} \times \frac{60€}{hora} = 45000€$.
- **Temps:** Es disposa de temps fins al dia 18 de juny del 2013.

El paràmetre més important és el temps, doncs no es disposa de pròrroga possible. Seguidament ve l'objectiu, ja que és menys rígid perquè es poden retallar funcions del sistema, i finalment el paràmetre més sacrificable i el que segurament fallarà són els recursos, doncs és possible permetre's dedicar-hi més hores.

Es pot comprovar que l'objectiu cobreix una necessitat real en l'àmbit de la recerca ja que automatitza una part del procés de negoci que encara no ha automatitzat ningú (vegeu el capítol 3.1.2 Antecedents), i probablement també en l'àmbit de la industria, ja que l'arquitectura SOA, els serveis web SOAP i la comprovació de la qualitat del servei mitjançant documents SLA són camps que tot just estan creixent. Per tant, el projecte és potencialment útil.

També es pot veure que el projecte és viable, ja que diferents problemes de l'objectiu ja han estat atacats per separat pels projectes ADA i SALMon, i aquest és un projecte d'integració i evolució d'aquestes. Fent una metàfora, es pot dir que és el pas final de l'algorisme de dividir i vèncer.

4.2.1. Procés de negoci

El procés de negoci real es divideix en tres casos d'ús (vegeu la Figura 19): Acordar un SLA entre client i proveïdor, monitorar la realització del servei, i detectar violacions del SLA. El projecte busca automatitzar del tot els casos d'ús segon i tercer, tot oferint una forma comprensible d'explicació de les violacions.

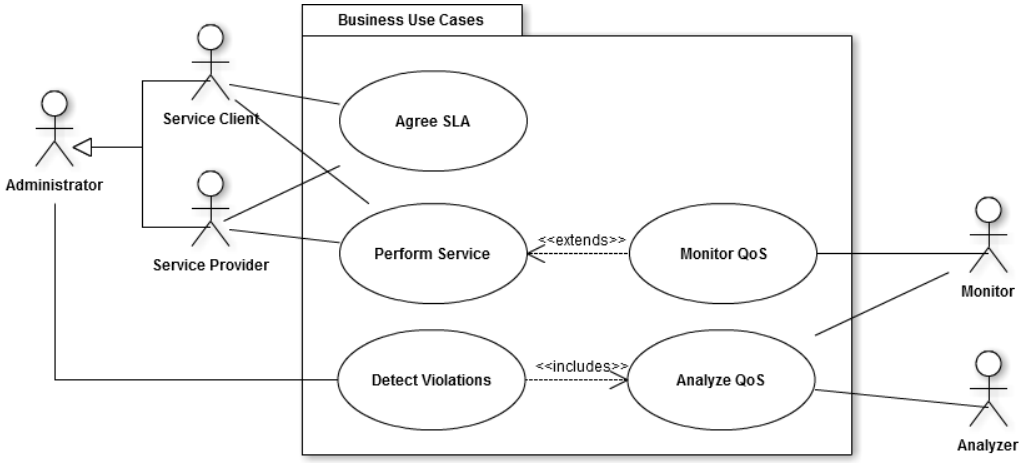

Figura 19: Casos d'ús de negoci.

Els actors en aquest model poden ser perfectament persones, però es necessita fer que tant el Monitor com l'Analitzador siguin, a més, components software. Noti's que tant el client com el proveïdor poden fer d'administrador, fins i tot simultàniament, vigilant el compliment del SLA.

En concret, els passos a automatitzar són (vegeu la Figura 20):

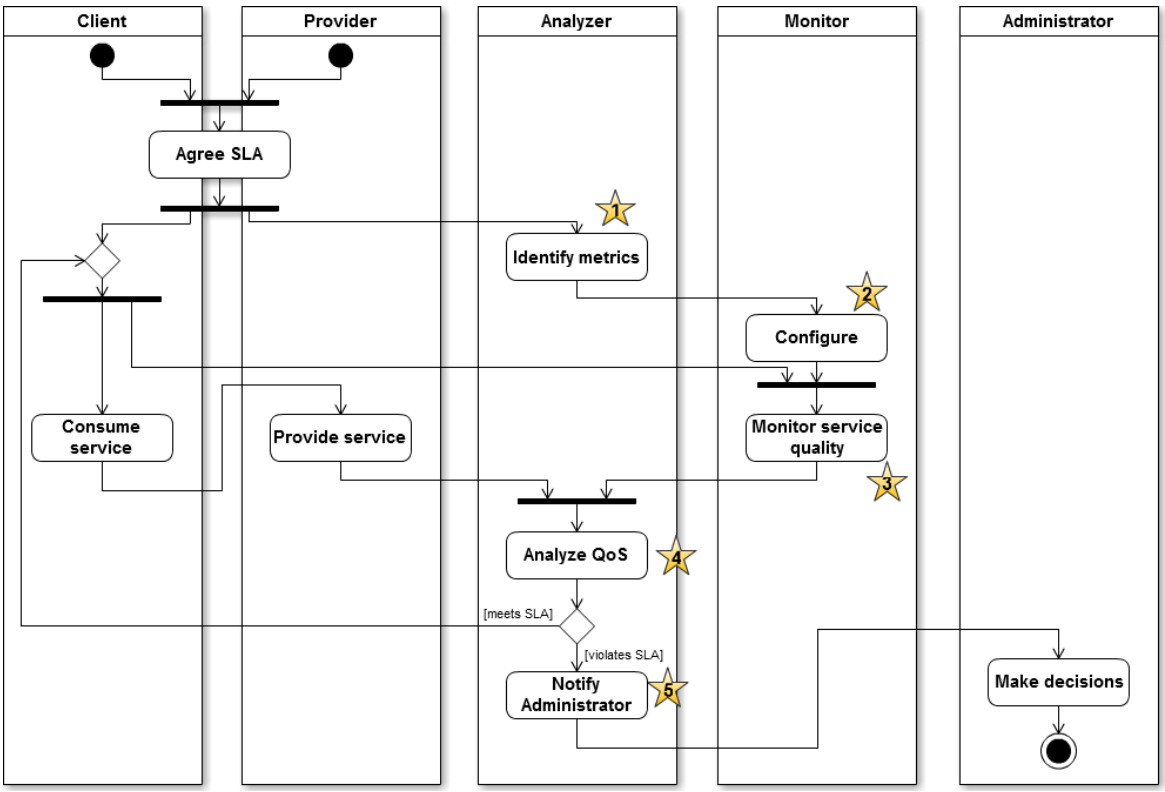

Figura 20: Diagrama d'activitat de negoci.

1. **Identificar mètriques:** Té com a entrada un SLA i com a sortida la informació necessària per a monitorar el servei i quines mètriques cal monitorar (dades de configuració).

2. **Configurar monitor:** Té com a entrada les dades de configuració i com a resultat un monitor configurat i actiu.
3. **Monitorar QoS:** Comença amb la detecció d'esdeveniments produïts per sondes i té com a sortida valors de mètriques.
4. **Analitzar QoS:** Té com a entrada valors de mètriques i el SLA i com a sortida la informació sobre les condicions violades, si n'hi ha.
5. **Avisar a l'administrador:** Té com a entrada informació sobre condicions violades, i com a resultat notifica a l'administrador de les violacions de forma entenedora.

A més, cal proporcionar un mètode per afegir noves mètriques al sistema, ja que un SLA pot incloure mètriques que el sistema no sigui capaç de monitorar.

4.3. Requisits específics

A continuació es detallen els requisits que s'han detectat per al projecte. La llista es desglossa en una secció de requisits funcionals i quatre de requisits no funcionals seguint l'estàndard IEEE 830-1993 d'especificació de requisits [IEEE830].

Els requisits es descriuen en fitxes que són una simplificació de la plantilla Volere d'especificació de requisits [Vol]. Aquestes fitxes inclouen:

- **Identificador:** El número de cada requisit és un identificador únic.
- **Descripció:** Significat del requisit.
- **Justificació:** Motius pels quals apareix el requisit. Traça a objectius o necessitats descrits prèviament, si n'hi ha.
- **Condició de satisfacció:** Mesura que fa possible provar si el sistema compleix el requisit.
- **Satisfacció del client:** Grau de satisfacció del client si el sistema compleix el requisit, en escala d'1 a 5, essent 1 "desinteressat" i 5 "molt satisfet".
- **Insatisfacció del client:** Grau d'insatisfacció del client si el sistema no compleix el requisit, en escala d'1 a 5, essent 1 "poc importa" i 5 "molt decebut".
- **Dependències:** Requisits que influeixen o depenen d'aquest.
- **Conflictes:** Requisits que no es poden satisfer si se satisfà aquest.

4.3.1. Requisits d'interfícies externes

Segons l'estàndard, aquests requisits han d'incloure tots els mètodes d'entrada i sortida del sistema, descrivint la seva font, format, i la informació que transmeten.

Requisit 1.1			
Descripció			
Un usuari podrà carregar un document SLA.			
Justificació			
El sistema ha de complir l'objectiu (a), configurar el sistema a partir d'un SLA (vegeu 1.3.3 Objectius).			
Condició de satisfacció			
El sistema té una interfície que permet enviar un document SLA.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	1.2, 1.3, 5.4	Conflictes:	-

Requisit 1.2			
Descripció			
El sistema reconeixerà documents SLA que contenen la informació necessària i suficient per a monitorar un servei web.			
Justificació			
El sistema ha de complir l'objectiu (a), configurar el sistema a partir d'un SLA (vegeu 1.3.3 Objectius), per tant ha de reconèixer documents SLA que contenen tota aquesta informació en sí mateixos (sense necessitar altres documents o fonts d'informació).			
Condició de satisfacció			
El sistema pot monitorar un servei web només amb la informació proveïda en un SLA que conté: La direcció (*endpoint*[1]) d'un servei web, la direcció (*SOAP Action*[2]*, URI*[3]...) de tota operació del servei web que s'ha de monitorar i la descripció dels intervals de temps en els quals cal monitorar el servei.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	5.4	Conflictes:	-

Requisit 1.3			
Descripció			
El sistema reconeixerà documents SLA que contenen la informació necessària i suficient sobre les mètriques que s'han de monitorar.			
Justificació			
El sistema ha de complir l'objectiu (a), configurar el sistema a partir d'un SLA (vegeu 1.3.3 Objectius). Un SLA conté SLOs que descriuen condicions sobre mètriques. Cal que aquestes mètriques estiguin ben definides, i cal que el sistema pugui reconèixer aquesta definició.			
Condició de satisfacció			
El sistema pot monitorar mètriques només amb la informació proveïda en un SLA que conté, per a cadascuna: Nom, tipus, unitat, i directiva de mesura: Funció matemàtica, direcció d'un servei web que la calculi o identificador únic que la identifiqui com a ja definida en el sistema.			
Satisfacció del client:	3	Insatisfacció del client:	4
Dependències:	1.4, 5.4	Conflictes:	-

Requisit 1.4			
Descripció			
Un usuari podrà definir com calcular mètriques noves.			
Justificació			
El sistema ha de complir l'objectiu (a), configurar el sistema a partir d'un SLA (vegeu 1.3.3 Objectius), i aquest SLA pot contenir mètriques que el sistema actual no pot monitorar.			
Condició de satisfacció			
El sistema té una operació que permet definir noves mètriques en temps d'execució d'alguna d'aquestes maneres com a mínim: Indicant la direcció d'un servei web que calcula la mètrica,			

[1] Un *Endpoint* és la direcció del punt d'entrada d'un servei web. Normalment és una URL que permet accedir-hi mitjançant protocols web (HTTP, REST, SOAP...).

[2] Per al cas del protocol SOAP, un *SOAP Action* és un URI que identifica únicament una operació dins un servei web.

[3] Un *Universal Resource Identifier* (URI) és una cadena de caràcters que segueix un format estàndard i que serveix per identificar de manera única un recurs (una dada, operació, document, definició...) que pot ser remot. Les direccions URL de les planes web són un cas concret d'URI.

indicant la formula matemàtica de la mètrica o indicant l'algorisme de càlcul de la mètrica.			
Satisfacció del client:	4	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

Requisit 1.5			
Descripció			
El sistema detectarà missatges SOAP de petició i resposta com a esdeveniments monitorables.			
Justificació			
El sistema ha de complir l'objectiu (b), monitorar el servei web (vegeu 1.3.3 Objectius). Això es pot fer per mitjà de sondes interceptores o desplegades. Es demana que el sistema permeti com a mínim detectar missatges de petició i resposta, ja que són necessaris per a calcular les mètriques natives del requisit 2.3.			
Condició de satisfacció			
Un missatge per a un servei web un SLA del qual s'està monitorant pot ésser detectat pel sistema, tingui efectes o no.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	2.2, 3.1, 6.3	Conflictes:	-

Requisit 1.6			
Descripció			
El sistema tindrà una interfície de notificació automàtica de violacions.			
Justificació			
El sistema ha de complir l'objectiu (d), notificar automàticament al consumidor de la violació d'una clàusula (vegeu 1.3.3 Objectius).			
Condició de satisfacció			
El sistema té una interfície que mostra informació sobre cada violació ocorreguda en temps real (instants després que aquesta succeeixi).			
Satisfacció del client:	2	Insatisfacció del client:	5
Dependències:	1.7	Conflictes:	-

Requisit 1.7			
Descripció			
Les violacions ocorregudes es mostraran de forma incremental.			
Justificació			
Per cobrir de forma apropiada la necessitat 2, que l'usuari no hagi de fer un seguiment actiu de les clàusules del contracte (vegeu 1.3.1 Necessitats), fa falta que l'usuari pugui veure, en qualsevol moment, totes les violacions que han ocorregut, no només l'última.			
Condició de satisfacció			
En un moment determinat, l'usuari pot veure a la interfície totes les violacions ocorregudes fins al moment.			
Satisfacció del client:	3	Insatisfacció del client:	4
Dependències:	-	Conflictes:	-

4.3.2. Requisits funcionals

Els requisits funcionals defineixen les accions fonamentals que ha de realitzar el software en acceptar i processar les entrades i generar les sortides. Se solen escriure en forma futura, com ara "El sistema farà..." amb connotació d'obligatorietat (traducció de la forma anglesa "shall").

Inclouen validació d'entrades, seqüència de les operacions, resposta a situacions anormals, efectes dels paràmetres i relació de les sortides amb les entrades.

Requisit 2.1			
Descripció			
El sistema no acceptarà documents SLA incorrectes.			
Justificació			
Un document SLA incorrecte no permet un funcionament normal del sistema. És important informar a l'usuari d'això, o altrament pot pensar que la falta d'informació és una absència de violacions del document SLA.			
Condició de satisfacció			
Introduir un SLA incorrecte fa saltar un error que informa a l'usuari que l'SLA és incorrecte. Es considera un SLA incorrecte aquell que no contingui la informació descrita als requisits 1.2 i 1.3, i també aquell que no respecti l'estàndard WS-Agreement o el llenguatge específic de l'aplicació que es derivi de WS-Agreement.			
Satisfacció del client:	3	Insatisfacció del client:	2
Dependències:	1.2, 1.3, 5.4	Conflictes:	-

Requisit 2.2			
Descripció			
El sistema monitorarà tot esdeveniment detectat que sigui susceptible de ser objecte d'una QoS (qualitat del servei) que violi un SLA prèviament introduït.			
Justificació			
El sistema ha de complir l'objectiu (b), monitorar el servei web (vegeu 1.3.3 Objectius), per a poder saber quina QoS ofereix.			
Condició de satisfacció			
Si l'usuari introdueix un SLA, el sistema *intenta calcular* a partir d'aquell moment la QoS de les mètriques que descriu l'SLA de tots els esdeveniments detectats del servei web que descriu l'SLA. "*Intenta calcular"* vol dir que si el sistema disposa de l'algorisme o funció matemàtica que descriu el càlcul de la mètrica, en calcula el valor, i si només disposa de la direcció d'un servei que ho calcula, li envia una petició a tal efecte.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	2.3, 2.6, 2.8	Conflictes:	-

Requisit 2.3			
Descripció			
El sistema comprovarà en temps real el compliment del SLA.			
Justificació			
El sistema ha de complir l'objectiu (c), comprovar el compliment del SLA (vegeu 1.3.3 Objectius), per a poder detectar violacions d'aquest. Fer-ho a temps real interessa a l'usuari perquè és capaç de reaccionar al moment de la violació.			
Condició de satisfacció			
La mesura d'un nou valor de QoS ve sempre succeïda per comprovar que tots els SLOs del contracte se segueixen satisfent.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	2.4	Conflictes:	-

Requisit 2.4
Descripció

El sistema notificarà tota violació d'un SLO.

Justificació

El sistema ha de complir l'objectiu (d), notificar automàticament al consumidor de la violació d'una clàusula (vegeu 1.3.3 Objectius).

Condició de satisfacció

Quan es mesura un nou valor d'una mètrica i hi ha una condició sobre aquesta mètrica d'un o més SLOs que no es compleix, el sistema notifica l'usuari de tots els SLOs que s'han violat.

Satisfacció del client:	3	Insatisfacció del client:	5
Dependències:	-	Conflictes:	-

Requisit 2.5

Descripció

El sistema registrarà tot error de comunicacions o intern en un o més fitxers de *log*.

Justificació

El sistema ha de ser mantenible, i en un sistema altament distribuït com aquest, resulta molt complicat conèixer els motius d'una fallada si no es disposa d'un registre.

Condició de satisfacció

Tota petició rebuda a un dels serveis del sistema i tot esdeveniment detectat fan que el sistema dugui a terme les accions esperades (respondre la petició adequadament, monitorar) o bé genera una entrada en un o més registres d'errors amb informació sobre el moment i el tipus d'error sorgit.

Satisfacció del client:	2	Insatisfacció del client:	2
Dependències:	2.6	Conflictes:	-

Requisit 2.6

Descripció

Per a cada esdeveniment detectat i cada mètrica que es pugui i s'hagi de monitorar, el sistema desarà un valor d'aquella mètrica i esdeveniment.

Justificació

El sistema ha de complir l'objectiu (c), comprovar el compliment del SLA (vegeu 1.3.3 Objectius). Per a fer-ho, només cal que desi els valors relatius a mètriques que es troben en SLOs que involucren la operació concreta del servei concret que s'està monitorant.

Condició de satisfacció

Quan el sistema monitora un missatge, per a cada mètrica susceptible de violar una condició sobre aquell servei i operació, el sistema o bé desa un valor o bé genera una entrada del registre d'errors.

Satisfacció del client:	2	Insatisfacció del client:	4
Dependències:	2.7	Conflictes:	-

Requisit 2.7

Descripció

A falta d'informació, el sistema pressuposarà que no s'ha violat cap terme del SLA.

Justificació

El fet de detectar una violació del SLA és afirmar que el consumidor o el proveïdor d'un servei han trencat el seu contracte. En cas de ser un contracte amb veracitat legal, no es pot fer tal acusació sense proves.

Condició de satisfacció

En cas de manca de dades suficients per afirmar-ho, malgrat que un servei hagi violat el SLA, el sistema no informa de cap violació.

Satisfacció del client:	4	Insatisfacció del client:	4
Dependències:	-	Conflictes:	-

Requisit 2.8
Descripció
El sistema podrà calcular, nativament, les mètriques ResponseTime, Availability, i les seves derivades, d'operacions de serveis web SOAP.
Justificació
Aquestes mètriques són altament comunes i resulta convenient que el sistema les pugui calcular nativament. Per a més informació sobre les mètriques, vegeu 3.1.4 SALMon (pàgina 20). Com s'explica a la secció 3.1.4 SALMon (pàgina 20), la mètrica RoundTripTime no existeix com a tal. Com que no és necessària per al projecte ja que és una mètrica exclusiva dels serveis de tipus REST, i és possible implementar-la gràcies al requisit 5.4, es deixa fora de l'abast.
Condició de satisfacció
El sistema pot calcular els valors de les mètriques ResponseTime i Availability, i les mètriques derivades mínim històric, màxim històric i mitjana absoluta del ResponseTime i percentatge de temps disponible, percentatge de temps caigut i temps mig de recuperació de la Availability.

Satisfacció del client:	1	Insatisfacció del client:	3
Dependències:	4.2, 6.1	Conflictes:	5.2

4.3.3. Requisits de rendiment

Aquesta secció especifica els requisits numèrics sobre la interacció de l'usuari o d'altres sistemes software amb el sistema. Aquests poden ser el nombre d'usuaris que ha de suportar, la quantitat d'informació que ha de poder gestionar, el nombre d'operacions que ha de poder realitzar en un interval de temps, etcètera.

Requisit 3.1
Descripció
El procés de detecció d'esdeveniments no hauria d'interferir amb el rendiment dels serveis monitorats.
Justificació
Si el procés de detecció interfereix amb el rendiment dels serveis, la qualitat del servei que percep el consumidor es veu afectada.
Condició de satisfacció
L'increment del temps de resposta d'un servei monitorat amb el monitor en local davant del temps de resposta del servei sense el monitor és constant (per tant, independent de la mida del missatge) i menyspreable.

Satisfacció del client:	4	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

Requisit 3.2
Descripció
El sistema suportarà múltiples usuaris simultanis.
Justificació
Una arquitectura SOA pot rebre peticions simultànies de diferents usuaris, per tant el sistema ha de ser capaç de gestionar-les sense efectes inesperats, és a dir, com si cada petició simultània s'hagués produït sola.

Condició de satisfacció			
Dues peticions simultànies al sistema es comporten com si estiguessin aïllades, és a dir, el sistema és segur en quant a concurrència.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	-	Conflictes:	-

Requisit 3.3			
Descripció			
El sistema podrà emmagatzemar una gran quantitat de valors de mètriques.			
Justificació			
Les mètriques que apareixen en un SLA poden ser històriques (per exemple, la mitjana històrica) o de molts valors. Com que un servei web pot rebre diverses peticions al minut, és necessari un compromís mínim, per al cas d'aquesta versió no comercial del sistema, de que es podran emmagatzemar valors durant mesos o setmanes. Aquest compromís s'ha fixat en l'espai necessari per a un any de monitoratge a raó de cent crides al minut. Tenint en compte que el valor d'una mètrica decimal de doble precisió, el tipus més freqüent (doncs el ResponseTime és molt freqüent i és decimal), és de 8 bytes com a mínim a la majoria de sistemes gestors de bases de dades, es necessita: $$\frac{8\ B}{1\ valor} \times \frac{100\ valors}{1\ minut} \times \frac{525948,766\ minuts}{1\ any} = 401,27\ MB/any$$			
Condició de satisfacció			
El sistema reserva 400 megabytes o més per a valors de mètriques.			
Satisfacció del client:	1	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

4.3.4. Requisits lògics de la base de dades

Aquests són els requisits lògics sobre qualsevol informació que hagi de ser persistent. Poden ser sobre el tipus d'informació que s'ha de desar, la freqüència d'accés, les entitats que han d'aparèixer i les seves relacions, restriccions d'integritat o d'altres.

Requisit 4.1			
Descripció			
El sistema desarà, per a cada petició monitorada, el missatge de petició i el missatge de resposta.			
Justificació			
El contingut dels missatges és susceptible de ser necessari per al càlcul de certes mètriques.			
Condició de satisfacció			
Tota petició interceptada pel monitor és representada a la base de dades amb el seu missatge de petició i també amb el seu missatge de resposta, si en té.			
Satisfacció del client:	3	Insatisfacció del client:	4
Dependències:	-	Conflictes:	-

Requisit 4.2			
Descripció			
El sistema desarà, per a cada esdeveniment detectat, les dades necessàries per calcular, com a mínim, les mètriques natives del sistema (Response Time, Availability i derivades).			
Justificació			
Per a poder fer una auditoria de les violacions d'un SLA, és necessari disposar de les dades			

que han portat a aquell resultat. El sistema és més fidedigne si guarda aquestes dades com a prova de que les violacions detectades eren reals.

Condició de satisfacció

Tot missatge detectat és representat a la base de dades amb la marca de temps del moment de la detecció, la seva relació amb altres missatges de petició o resposta, i la informació de disponibilitat del missatge de resposta.

Satisfacció del client:	3	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

Requisit 4.3

Descripció

El sistema desarà, per a cada valor d'una mètrica, informació sobre el moment en què es va monitorar i el servei i operació monitorats.

Justificació

Les clàusules d'un SLA poden incloure sovint condicions sobre aquests paràmetres, de manera que és important poder seleccionar els valors en funció d'aquests paràmetres.

Condició de satisfacció

Es poden fer consultes a la base de dades seleccionant els valors d'una mètrica en funció del servei i la operació o en funció de períodes de temps.

Satisfacció del client:	2	Insatisfacció del client:	3
Dependències:	4.4	Conflictes:	-

Requisit 4.4

Descripció

El sistema desarà, per a cada valor d'una mètrica, informació sobre el missatge de petició.

Justificació

El client demana explícitament que es puguin filtrar els valors en funció de les dades del missatge de petició. Per exemple, seleccionar tots els temps de resposta d'una operació quan s'ha cridat amb un missatge de petició igual a un missatge concret.

Condició de satisfacció

Es poden seleccionar tots els valors d'una mètrica monitorats en peticions amb missatges de petició idèntics a un missatge concret. Dos missatges de petició es consideren idèntics si contenen la mateixa cadena de caràcters.

Satisfacció del client:	4	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

Requisit 4.5

Descripció

El sistema acceptarà mètriques de qualsevol tipus.

Justificació

Per a que el sistema no restringeixi la creació de possibles mètriques en un futur, es demana que accepti mètriques amb valors de qualsevol tipus: Enters, decimals, percentuals, booleans, alfanumèrics, etcètera.

Condició de satisfacció

El sistema permet configurar una nova mètrica de qualsevol tipus, i els valors es monitoren, desen i comparen de forma adequada.

Satisfacció del client:	3	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

4.3.5. Restriccions de disseny

Aquest apartat descriu les restriccions de disseny que poden ser causades per altres estàndards, per limitacions hardware o altres motius.

Requisit 5.1			
Descripció			
El sistema serà desenvolupat com a arquitectura SOA.			
Justificació			
Això permet integrar el sistema en qualsevol arquitectura SOA de forma fàcil. Fa que el sistema sigui interoperable i és un requisit natural perquè els dos sistemes que s'integren (ADA i SALMon) són arquitectures SOA en sí mateixos.			
Condició de satisfacció			
Tota funcionalitat del sistema es pot iniciar mitjançant una crida a un servei web.			
Satisfacció del client:	3	Insatisfacció del client:	5
Dependències:	3.2, 5.2, 6.2	Conflictes:	-

Requisit 5.2			
Descripció			
El sistema s'adherirà als estàndards SOAP i WSDL.			
Justificació			
Per a ser acceptat per la comunitat i interoperable, el sistema ha d'adherir-se als estàndards.			
Condició de satisfacció			
Tots els serveis que componen el sistema són serveis web SOAP i tenen un document WSDL accessible per l'usuari que en descriu les interfícies.			
Satisfacció del client:	3	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

Requisit 5.3			
Descripció			
El sistema podrà monitorar serveis web que s'adhereixin a l'estàndard SOAP.			
Justificació			
SOAP és un estàndard molt estès actualment, i això permet monitorar un gran nombre de serveis. La versió inicial de SALMon pot monitorar serveis web SOAP, per tant el sistema final hauria de ser capaç de conservar aquesta capacitat.			
Condició de satisfacció			
El sistema monitora serveis web SOAP.			
Satisfacció del client:	3	Insatisfacció del client:	3
Dependències:	6.1	Conflictes:	-

Requisit 5.4			
Descripció			
El sistema reconeixerà SLAs que s'adhereixin al estàndard WS-Agreement.			
Justificació			
WS-Agreement és un dels estàndards per a llenguatges de descripció de SLAs.			
Condició de satisfacció			
El sistema és capaç de monitorar un servei web a partir d'un SLA escrit en WS-Agreement.			
Satisfacció del client:	3	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

Requisit 5.5			
Descripció			
El sistema usarà els sistemes ja existents ADA i SALMon.			
Justificació			
Un dels objectius del projecte és integrar els sistemes ADA i SALMon (vegeu 1.3.3 Objectius).			
Condició de satisfacció			
Quan un usuari estableix que s'ha de monitorar un SLA, el procés de negoci involucra crides als sistemes ADA i SALMon.			
Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	5.1	Conflictes:	-

Requisit 5.6			
Descripció			
El sistema és no intrusiu amb el client i el proveïdor.			
Justificació			
El sistema té més facilitat de col·laboració amb altres sistemes i serveis i és més portable si, per a començar a fer-lo servir, tant client com proveïdor no han de fer canvis (o n'han de fer els mínims) en el seu sistema per permetre-ho.			
Condició de satisfacció			
Els clients d'un servei web monitorat poden no ser conscients de la presència del monitor (no han de fer cap canvi al codi) si el proveïdor ho és, i el proveïdor d'un servei web monitorat pot no ser conscient de la presència del monitor d'un client si el client ho és.			
Satisfacció del client:	4	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

4.3.6. Atributs de qualitat

Altres requisits de qualitat del sistema, com ara de disponibilitat, seguretat, mantenibilitat o portabilitat.

Requisit 6.1			
Descripció			
El sistema podrà ser fàcilment extensible per a monitorar serveis web que no s'adhereixin a l'estàndard SOAP.			
Justificació			
Hi ha molts serveis web que segueixen protocols i estàndards diferents de SOAP, com per exemple REST, i el sistema ha de poder ser estès en un futur per a cobrir aquests sectors.			
Condició de satisfacció			
El disseny final del sistema no ha de patir cap canvi que no sigui addició per a implementar el monitoratge de serveis web no SOAP, com ara REST.			
Satisfacció del client:	5	Insatisfacció del client:	2
Dependències:	-	Conflictes:	-

Requisit 6.2			
Descripció			
El sistema serà canviable a nivell de components monitor i analitzador seguint un protocol orientat a documents.			
Justificació			

Per assegurar la interoperabilitat i respectar la filosofia SOA, la funcionalitat de monitorar serveis web ha d'estar físicament desacoblada de la d'analitzar els documents SLA. Un protocol orientat a documents (és a dir, on un missatge conté un document) facilita aquesta canviabilitat perquè un document conté informació en un grau de granularitat molt baix.

Existirà per tant un document que conté les dades de configuració del monitor i la qualitat del servei monitorada que s'anomenarà *Monitoring Management Document* (MMD).

Condició de satisfacció

El component que s'ocupa de monitorar els serveis web és canviable per un altre de compatible, i el component que s'ocupa de comprovar si un SLA ha patit cap violació, també. El pas d'informació entre ambdós components es fa mitjançant un tipus de document específic.

Satisfacció del client:	5	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

Requisit 6.3

Descripció

El sistema serà transparent pel que fa a la informació dels missatges detectats.

Justificació

El monitoratge no pot comprometre la funció del servei monitorat, doncs el client percebria aquest canvi com una fallada del servei, i per tant la qualitat de servei percebuda variaria.

Condició de satisfacció

Tot missatge de petició arriba al proveïdor amb la mateixa informació que quan surt del consumidor del servei, i tot missatge de resposta arriba al consumidor amb la mateixa informació que quan surt del proveïdor, tant si el servei web està essent monitorat com si no. Dos missatges contenen la mateixa informació si produeixen els mateixos efectes i reacció en el seu destinatari.

Satisfacció del client:	1	Insatisfacció del client:	5
Dependències:	-	Conflictes:	-

Requisit 6.4

Descripció

Un usuari podrà carregar de forma senzilla un document SLA.

Justificació

El sistema ha de ser usable. L'estàndard per a carregar fitxers en una aplicació són tres clics (Obrir carpeta, seleccionar fitxer, i carregar-lo).

Condició de satisfacció

El sistema té una interfície que permet enviar un document SLA en tres clics o menys.

Satisfacció del client:	4	Insatisfacció del client:	3
Dependències:	-	Conflictes:	-

4.4. Abast

A partir dels requisits i els objectius, s'ha determinat, de forma iterativa, l'abast del projecte com segueix, prenent decisions de disseny que es discutiran al capítol 7 Disseny.

4.4.1. Què permet el sistema

El sistema permet a l'usuari introduir un document SLA en un format específic fixat i conforme a l'estàndard WS-Agreement a través d'una interfície web senzilla.

Mitjançant això, el sistema emmagatzema el document SLA i en genera un document amb la informació necessària per a configurar un monitor de serveis web. Aquest document conté les dades de configuració i a més la qualitat del servei monitorada fins al moment, i s'anomena *Monitoring Management Document* (MMD). Els MMDs estan subjectes a una especificació fixa.

A continuació, el sistema configura el servei web monitor del SALMon mitjançant el document MMD obtingut a partir del SLA.

A partir d'aquest punt, el sistema monitora totes les crides SOAP que els clients facin passar per una sonda interceptora del sistema que actuarà com un proxy (representant) SOAP del servei real, calculant-ne totes les mètriques possibles que s'hagin de tenir en compte per poder garantir el compliment del SLA. Tal esdeveniment farà que el sistema comprovi el compliment del SLA. En cas que un o més SLOs no es compleixin, el sistema avisarà a l'usuari mitjançant la interfície web de les violacions, les clàusules violades i les causes, de forma entenedora.

El sistema permet la detecció d'altres tipus d'esdeveniments monitorables, amb la condició que la notificació dels esdeveniments incorpori tota la informació necessària per al monitoratge d'aquests, mitjançant una interfície SOAP que rep les notificacions dels esdeveniments.

El sistema permet monitorar clàusules SLO sobre mètriques predefinides de la forma menor que, major que, igual, o diferent, sobre un únic rang temporal continu. Aquestes mètriques predefinides són el ResponseTime, la Availability i les seves derivades mínim històric, màxim històric i mitjana absoluta del ResponseTime i percentatge de temps disponible, percentatge de temps caigut i temps mig de recuperació de la Availability.

El sistema permet a clients web subscriure's a notificacions sobre la qualitat del servei monitorada i a notificacions sobre les violacions ocorregudes d'un SLA concret.

El sistema permet consultar mitjançant aquestes notificacions la qualitat del servei sobre una mètrica derivada d'una mètrica numèrica segons les expressions mínim històric, màxim històric i mitjana absoluta, i d'una mètrica booleana segons les expressions percentatge de temps cert, percentatge de temps fals i temps mig de recuperació a cert.

El sistema també permet configurar com es calculen noves mètriques no definides mitjançant l'ús de serveis web externs a mode de "plug-ins" o connectors, mitjançant una operació SOAP que afegeix les dades necessàries per a poder usar un d'aquests serveis web i saber quina mètrica calcula.

El sistema permet, per tant, configurar el càlcul de totes les mètriques observables tal i com es descriuen al capítol 2.2.2 Les mètriques de qualitat, ja siguin observables mitjançant sondes

interceptores o desplegades, i permet monitorar mètriques que no són necessàriament mètriques de qualitat (vegeu la Figura 21).

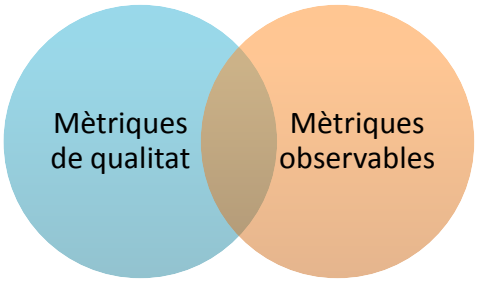

Figura 21: Relació de mètriques. SALMon només podia monitorar un subconjunt de la intersecció.

4.4.2. Què NO permet el sistema

Per a les següents observacions, noteu que "l'usuari" es refereix a una persona, no un software, i per tant és implícit que l'usuari interactua amb el sistema només a través de la interfície web i no a través de peticions a serveis web.

El sistema no permet a l'usuari consultar la qualitat del servei a través de la web. Tampoc permet fer-ho a través d'un servei web de forma diferida, únicament pot rebre notificacions en temps real.

El sistema no permet a l'usuari visualitzar els documents SLA introduïts, ni en format original ni en format WSAg4People (vegeu 3.1.3 ADA, pàgina 20).

El sistema no verifica que els serveis web afegits com a instruments de mesura de mètriques a mode de connectors funcionin. Com a conseqüència, el sistema només calcula les mètriques predefinides del sistema i les d'aquells instruments de mesura que funcionin en cada moment que es monitora el servei.

El sistema no permet monitorar crides SOAP que el client del servei no forci a passar a través del Monitor.

El sistema no permet el càlcul d'altres mètriques no descrites, ni predefinides ni derivades, que no es calculin mitjançant un instrument de mesura extern.

El sistema no permet fer Testing dels serveis web monitorats.

El sistema no permet monitorar documents SLA que el component ADA no reconegui.

4.4.3. Components del sistema

En el moment d'inici del projecte, el sistema es separa en components lògics segons mostra la Figura 8 (vegeu el capítol 3.1.4 SALMon, pàgina 20). Per assolir l'objectiu tot acomplint els requisits es necessita un sistema format pels components lògics que mostra la

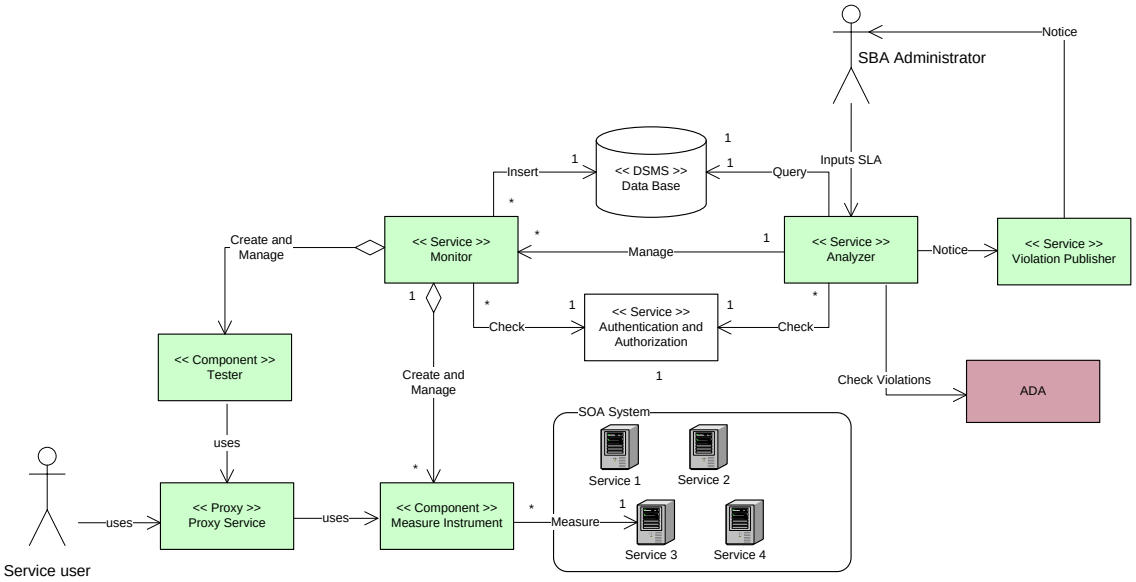

Figura 22: Arquitectura lògica de SALMonADA.

El sistema consta de diferents components lògics. Els tres components principals són *ADA*, SALMon (el *Monitor*), i l'*Analyzer*, que actua de pont entre ambdós. A més, cal un component que serveixi per notificar les violacions ocorregudes, d'ara endavant el *Violation Publisher*, i cal notar la presència de les sondes de monitoratge (a la figura apareixen en forma del *Proxy* de SALMon) i instruments de mesura, que poden ser afegits de forma modular al sistema.

4.4.4. Paquets de treball

El projecte s'ha desglossat en cinc paquets de treball per a una millor comprensió i gestió de l'abast. Els cinc paquets són:

1. **SLA:** En aquest paquet s'engloben totes les tasques relacionades amb l'anàlisi i gestió dels documents SLA que són independents de la resta de paquets. Totes les tasques d'aquest paquet han estat dutes a terme pel grup ISA i per tant no formen part de la feina d'aquest projecte, malgrat ser part vital pel seu funcionament.
2. **Analyzer:** Aquest paquet comprèn les tasques relacionades amb la integració de les dues eines ADA i SALMon, la definició formal dels documents intermedis i l'especificació general del sistema.
3. **Violation Publishing:** Paquet amb les tasques de desenvolupament de la interfície d'usuari i les eines necessàries per a la notificació de violacions en temps real.
4. **SALMon:** Aquest paquet conté les tasques de modificació i evolució de l'eina SALMon per tal de permetre la seva integració amb ADA i el suport de noves funcionalitats necessàries per al sistema.
5. **Documents:** Paquet per a les tasques de redacció de manuals diversos i la pròpia memòria del projecte.

4.4.5. Desglossament de tasques

A continuació es desglossen les tasques dels cinc paquets del projecte (vegeu la Taula 2), amb un índex de granularitat relativament baix (les tasques podrien ser subdividides molt més).

Per a cada tasca s'indica la descripció, les tasques predecessores, l'equip encarregat i, en cas de ser una tasca afegida com a mitigació d'un risc, el risc que mitiga. En cas de ser del grup ISA, la tasca es mostra ombrejada en gris per indicar que l'autor de la memòria no ha pres part en el seu desenvolupament. En els casos que el grup és GESSI, l'autor hi ha pres part i el projecte considera els altres membres del grup com a "client" contractant, i per tant tota la feina s'ha realitzat conduïda per les seves necessitats i requisits.

A més, cal anotar que el model de referència que SALMonADA intenta implementar ha estat formalitzat conjuntament pels grups ISA i GESSI, però l'autor del projecte només va intervenir en la seva concepció en quant al disseny de l'arquitectura i anàlisi de viabilitat.

	WBS	Task Name	Predecessors	Team
1	**1**	**Plataforma per al monitoratge i l'explicació de violacions de clàusules de documents SLA**		
2	**1.1**	**SLA**		
3	1.1.1	Definir WS-Agreement-MD	22	ISA
4	**1.1.2**	**Parser SLA**		**ISA**
5	1.1.2.1	Dissenyar Parser SLA	3	ISA
6	1.1.2.2	Implementar Parser SLA	5	ISA
7	1.1.2.3	Testejar Parser SLA	6	ISA
8	**1.1.3**	**Repositori SLA**		**ISA**
9	1.1.3.1	Dissenyar Repositori SLA	5	ISA
10	1.1.3.2	Implementar Repositori SLA	9	ISA
11	1.1.3.3	Testejar Repositori SLA	10	ISA
12	**1.1.4**	**Compilador SLA a MMD**		**ISA**
13	1.1.4.1	Dissenyar Compilador SLA a MMD	5	ISA
14	1.1.4.2	Implementar Compilador SLA a MMD	13	ISA
15	1.1.4.3	Testejar Compilador SLA	14	ISA
16	**1.1.5**	**Verificador MMD**		**ISA**
17	1.1.5.1	Dissenyar Verificador MMD	22;3	ISA
18	1.1.5.2	Implementar Verificador MMD	10;17	ISA
19	1.1.5.3	Testejar Verificador MMD	18	ISA
20	**1.2**	**Analyzer**		
21	**1.2.1**	**Parser MMD**		**ISA/GESSI**
22	1.2.1.1	Definir MMD		ISA/GESSI
23	1.2.1.2	Implementar Parser MMD	22	ISA
24	1.2.1.3	Especificar temporalitats MMD		GESSI
25	1.2.1.4	Modificar Parser MMD	23	GESSI
26	1.2.1.5	Testejar temporalitats	25;34	GESSI
27	1.2.2	Especificar sistema		GESSI
28	1.2.3	Dissenyar Analyzer	27;3	GESSI
29	**1.2.4**	**Components Analyzer**		**GESSI**
30	1.2.4.1	Implementar Repositori MMD	28	GESSI
31	1.2.4.2	Especificar Interfície Analyzer	28	GESSI
32	1.2.4.3	Implementar SALMon Configurer	23	GESSI
33	1.2.4.4	Implementar prototip client ADA		GESSI
34	1.2.4.5	Implementar Analyzer	18;31;32;30;33	GESSI
35	1.2.5	Testejar Analyzer	34	GESSI
36	1.2.6	Desplegar Servei Analyzer	35	GESSI
37	**1.3**	**Violation Publishing**		
38	**1.3.1**	**Violation Publisher**		**GESSI**
39	1.3.1.1	Dissenyar Violation Publisher	34	GESSI
40	1.3.1.2	Implementar Violation Publisher	39	GESSI
41	**1.3.2**	**Interfície Web**		**GESSI**
42	1.3.2.1	Recerca frameworks amb Server push		GESSI
43	1.3.2.2	Especificar Interfície Web	27	GESSI
44	1.3.2.3	Dissenyar Interfície Web	43;34	GESSI

45	1.3.2.4	Implementar Interfície Web	44;42	GESSI
46	1.3.2.5	Testejar Interfície Web	45	GESSI
47	**1.4**	**SALMon**		
48	**1.4.1**	**Publisher**		**GESSI**
49	1.4.1.1	Dissenyar Publisher	27	GESSI
50	1.4.1.2	Implementar Publisher	49	GESSI
51	1.4.1.3	Testejar Publisher	50	GESSI
52	1.4.2	Dissenyar Monitor	69	GESSI
53	**1.4.3**	**Modificar disseny mètriques**		**GESSI**
54	1.4.3.1	Pas a mètriques abstractes BD	52	GESSI
55	1.4.3.2	Pas a mètriques abstractes codi	54	GESSI
56	1.4.3.3	Pas a mètriques abstractes interfície	52	GESSI
57	**1.4.4**	**Afegir tractament d'esdeveniments**		**GESSI**
58	1.4.4.1	Pas a esdeveniments BD	54	GESSI
59	1.4.4.2	Pas a esdeveniments codi	58	GESSI
60	1.4.4.3	Pas a esdeveniments interfície	52	GESSI
61	1.4.4.4	Pas a esdeveniments proxy	60	GESSI
62	**1.4.5**	**Afegir filtratge de valors**		**GESSI**
63	1.4.5.1	Pas a filtres BD	54	GESSI
64	1.4.5.2	Pas a filtres codi	63	GESSI
65	1.4.5.3	Pas a filtres interfície		GESSI
66	1.4.6	Implementar mètriques predefinides	55;64	GESSI
67	1.4.7	Testejar Monitor	57;62	GESSI
68	**1.4.8**	**Measure Instruments**		**GESSI**
69	1.4.8.1	Especificar Interfície Measure Instrument		GESSI
70	1.4.8.2	Implementar prototip Measure Instrument	69	GESSI
71	1.4.8.3	Testejar prototip Measure Instrument	55;59;70	GESSI
72	**1.5**	**Documents**		
73	1.5.1	Manual d'usuari		GESSI
74	1.5.2	Manual de desenvolupador		GESSI
75	1.5.3	Memòria		GESSI
76	1.5.4	Informe previ		GESSI
77	1.5.5	Gestió del projecte		GESSI

Taula 2: Desglossament de tasques.

Noti's que el fet que el grup ISA tingui un nombre dramàticament més baix de tasques és degut a que el detall d'aquesta part del sistema no és rellevant per a aquesta memòria; en cap cas és indicatiu de la quantitat de feina.

4.5. Anàlisi de riscs

Un risc és la possibilitat que succeeixi un inconvenient durant el desenvolupament del projecte. Tot risc té una probabilitat de succeir i un impacte, que és el cost, no necessàriament econòmic, que té per al projecte si succeeix.

Si un risc és important, bé perquè és probable que succeeixi, bé perquè el seu impacte seria greu, cal analitzar-lo i assignar-li una mitigació i una contingència.

- La **mitigació** és el conjunt d'accions que cal dur a terme per reduir la probabilitat de que el risc succeeixi o l'impacte que tindria. Cal assignar-los recursos inicials, doncs formen part de l'estratègia del projecte, s'hauran de realitzar.
- La **contingència** és el conjunt d'accions que caldria realitzar si el risc succeís per a poder pal·liar els seus efectes. No s'hi assignen recursos inicials, ja que si tot va bé no s'haurà d'aplicar, però cal estar preparat per a fer-ho.

	Riscs	Probabilitat	Impacte
1	No poder fer server push	2	3
2	Canvis de requisits freqüents	4	3
3	Pla massa optimista en temps	3	4
4	Dificultats amb la interfície d'ADA	3	1
5	Modificacions a la BD inestables	2	4
6	Retard de funcionalitats ISA	1	2
7	Retard de funcionalitats GESSI	2	2

Probabilitat	Impacte	Grau
Molt alta	Crític	4
Alta	Considerable	3
Significativa	Moderat	2
Baixa	Assumible	1

Taula 3: Resum de riscs.

A continuació s'expliquen els riscs que s'han detectat per aquest projecte junt amb la mitigació i/o contingència que se n'ha preparat (vegeu la Taula 3):

Risc 1: No poder fer server push

Descripció

La notificació en temps real mitjançant una web requereix una tècnica anomenada *server push*[1]. El server push no resulta una tècnica trivial, i implementar-lo de zero és una tasca massa feixuga per encabir-se en aquest projecte.

Mitigació

Cercar una plataforma de desenvolupament web que doni facilitats per a implementar el server push, i usar-la per a desenvolupar la interfície web.

Contingència

Desenvolupar un altre tipus de notificacions en temps real per a l'usuari. Per exemple, a través de correu electrònic.

Probabilitat:	Significativa	**Impacte:**	Considerable

Risc 2: Canvis de requisits freqüents

Descripció

Com que es tracta d'un projecte de recerca que participa de col·laboracions i presenta articles sobre diferents aspectes sovint, és fàcil que es doni el cas que el client (el grup de recerca) necessiti funcions específiques en moments específics, i modifiqui els requisits del projecte. Aquestes modificacions poden ser substitutives o additives, i en el segon cas poden afectar al calendari del projecte greument.

Mitigació

Realitzar el projecte seguint una metodologia àgil que permeti canvis en les funcions demanades durant el desenvolupament. Mantenir un ordre de prioritat en les funcions requerides, amb la qual el client estigui d'acord.

Contingència

Si el canvi afecta al calendari, eliminar de l'abast del projecte alguna funció de prioritat mínima.

Probabilitat:	Molt alta	**Impacte:**	Considerable

[1] *Server push* (en anglès "empènyer des del servidor") és una tècnica per trencar l'arquitectura client-servidor, en la qual el servidor sempre respon a una petició d'un client, i no pot fer res si un client no n'ha fet cap. El server push fa que el servidor pugui enviar dades al client sense que aquest hagi de demanar-les. Existeixen múltiples implementacions diferents del server push, cadascuna amb els seus pros i contres.

Risc 3: Pla massa optimista en temps			
Descripció			
Degut a la poca experiència en planificació de projectes, el cap de projecte (l'autor) pot estimar les hores requerides per les tasques de forma massa optimista. Això pot causar retards i manca de temps i desestabilitzar el desenvolupament del projecte.			
Mitigació			
Consultar estimacions de projectes anteriors. Augmentar en un marge de seguretat les estimacions de les hores de totes les tasques.			
Contingència			
Eliminar de l'abast del projecte alguna funció de prioritat mínima.			
Probabilitat:	Alta	**Impacte:**	Crític

Risc 4: Dificultats amb la interfície d'ADA			
Descripció			
La interfície SOAP del servei d'ADA és complexa perquè tracta amb documents SLA, les violacions i les seves explicacions amb referències a SLOs concrets, etcètera. El sistema necessita saber quin format de resposta fa servir perquè és un servei desenvolupat per grup ISA i el sistema necessita mostrar l'explicació de les violacions a l'usuari.			
Mitigació			
Fer un estudi previ de la interfície. Implementar una prova de concepte que demani si hi ha violacions en un SLA que es viola i en llegeixi l'explicació.			
Contingència			
Posar-se en contacte amb el grup ISA i demanar-los una explicació o exemples.			
Probabilitat:	Alta	**Impacte:**	Assumible

Risc 5: Modificacions a la BD inestables			
Descripció			
L'arquitectura del sistema SALMon és poc extensible pel que fa a l'adició de noves mètriques. Per a permetre l'addició de noves mètriques com indica el requisit 1.4, és necessari realitzar canvis estructurals crítics a la base de dades de SALMon. Tals canvis són grans, i difícils de depurar, i poden paralitzar el desenvolupament.			
Mitigació			
Realitzar els canvis progressivament, en blocs independents. Aquests blocs canvien una part de la base de dades, mantenint la compatibilitat amb la resta, però afegeixen temps de desenvolupament. En concret, primer s'implementarà l'abstracció de mètriques, i després, la gestió d'esdeveniments. Per a més informació, vegeu el capítol 6.3 Model conceptual.			
Contingència			
En cas que un dels blocs paralitzi el desenvolupament de totes maneres, eliminar les funcions que aporta aquell bloc de l'abast del projecte i cancel·lar el desenvolupament d'aquell bloc concret.			
Probabilitat:	Significativa	**Impacte:**	Crític

Risc 6: Retard de funcionalitats ISA
Descripció
Com a qualsevol projecte desenvolupat per múltiples persones o equips, existeixen dependències entre tasques d'uns i altres. Si una tasca es retarda, la següent també, i si és d'un altre membre o equip, aquest queda ociós a l'espera de la dependència.

Mitigació			
Planejar les tasques que depenguin de tasques del grup ISA per a desenvolupar més tard.			
Contingència			
Posar-se en contacte amb el grup ISA i demanar-los l'especificació de la interfície requerida. Aturar aquella tasca i seguir desenvolupant altres tasques, si és possible.			
Probabilitat:	Baixa	**Impacte:**	Moderat

Risc 7: Retard de funcionalitats GESSI			
Descripció			
Vegeu la descripció del risc 6.			
Mitigació			
Donar màxima prioritat a les funcions requerides pel grup ISA.			
Contingència			
Implementar un *stub*[1] que els serveixi per a seguir el desenvolupament.			
Probabilitat:	Significativa	**Impacte:**	Moderat

Per poder analitzar quins són els riscs més preocupants s'ha calculat un gràfic d'importància dels riscs (vegeu la Figura 23). El càlcul dona igual pes a la probabilitat i a l'impacte segons els graus que s'han definit a la Taula 3.

Com es pot veure, els riscs més preocupants són el risc 2 i el risc 3, "Canvis de requisits freqüents" i "Pla massa optimista en temps". Es donarà màxima prioritat a les seves mitigacions, doncs.

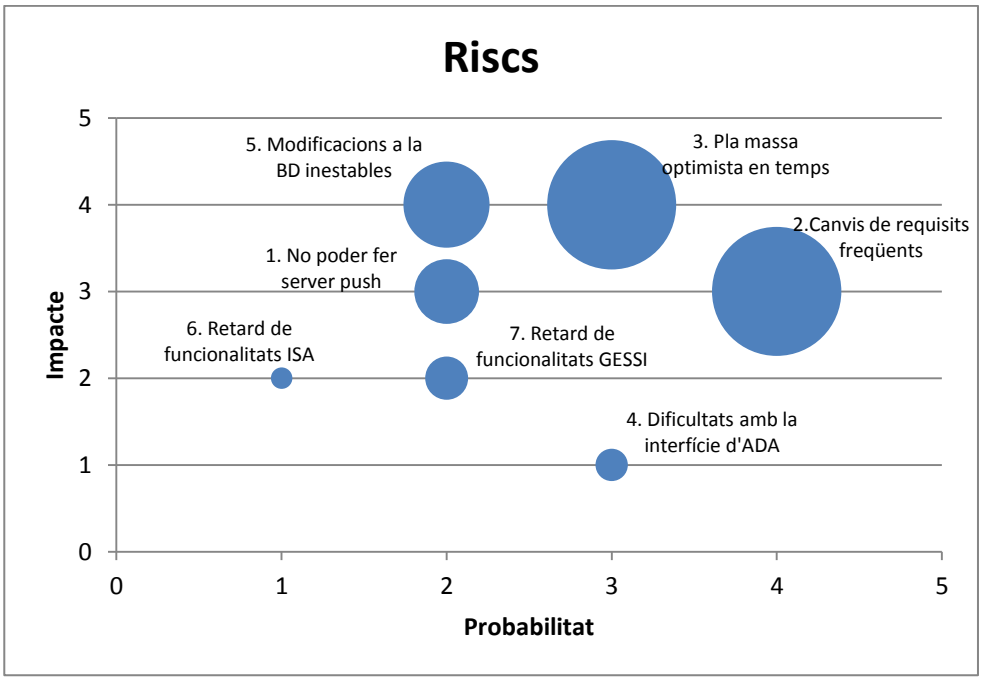

Figura 23: Gràfic d'importància dels riscs.

[1] Un *stub* és un component o tros de codi que representa un component complex, compartint-ne la interfície però essent mancat de tota lògica o procés. Es fa servir en proves software per suplir components mancants sense que això afecti l'estructura software de la resta del sistema.

Si això fos un projecte gran amb desenes de riscs, probablement s'haurien de descartar les mitigacions per als riscs de menor importància, com el 4, el 6 i el 7, però aquest és un projecte petit i totes les mitigacions s'han dut a terme en major o menor mesura, i el pla de projecte així ho contempla.

5. Pla de projecte

Haver identificat els riscs i conèixer l'abast del projecte proporciona una visió suficient per poder elaborar un pla de projecte, l'objectiu del qual és planificar una estratègia per al desenvolupament del projecte, una metodologia i una agenda de tasques, que intenti atènyer els objectius tot evitant els riscs.

Per a fer això, s'ha seleccionat una metodologia com s'explica a continuació, i s'ha desenvolupat una estratègia a seguir que es concreta tot seguit.

5.1. Metodologia

Una metodologia és una guia d'activitats i documents a realitzar durant el desenvolupament d'un projecte que en faciliten la seva gestió i intenten minimitzar la probabilitat de "xocar" amb els riscs. Les metodologies aconsellen una estructura, però en cap cas garanteixen l'èxit del projecte, ni tampoc són un llibre d'instruccions a seguir pas a pas. Podem pensar en una metodologia com un recull de consells pràctics.

Existeixen diferents tipus de metodologies. N'hi ha que són rígides, que exigeixen un alt nivell de documentació en uns formats molt específics, cosa que resulta útil en grans empreses amb un alt nombre de projectes grans, i també amb projectes que han de garantir uns estàndards de qualitat molt elevats. N'hi ha d'altres que són més àgils, que treuen importància a la formalitat i la quantitat de la documentació a favor d'altres trets, com l'alta interacció amb el client, el disseny subjecte a canvis, o el desenvolupament ràpid de funcionalitats aprofitables immediatament.

La mitigació del risc 2 obliga a fer servir una metodologia àgil. Les metodologies àgils, en general, valoren més la interacció de l'equip, el software funcional, la col·laboració amb el client i la resposta al canvi. Això resulta útil en projectes que, com aquest, estan subjectes a canvis de requisits freqüents, doncs l'important no és desenvolupar el que en un bon principi es va establir com a objectiu, sinó proporcionar al client allò que realment necessita.

S'han valorat tres alternatives possibles de metodologies àgils, a saber, OpenUP, Agile UP i Scrum. S'han triat en funció del coneixement previ de metodologies àgils. Heus aquí l'anàlisi que se n'ha fet:

- **OpenUP**, o Open Unified Process, és una metodologia oberta basada en la coneguda metodologia pesada Rational Unified Process (RUP). Un benefici d'usar OpenUP és el fet de conèixer RUP per haver-la fet servir a l'assignatura de Projecte d'Enginyeria del Software i Bases de Dades. Un dels inconvenients més grans de fer servir una metodologia és la falta d'experiència amb ella, i això fa més planera la corba d'aprenentatge. Per la resta, OpenUP és una metodologia àgil bastant rígida.
- **Agile UP**, o Agile Unified Process, és una altra metodologia basada en RUP. A diferència d'OpenUP, fa servir una cua amb prioritat d'activitats pendents que pot resultar útil per gestionar els riscs 2, 3 i 5. A priori sembla menys pesada en documentació.
- **Scrum**, probablement la metodologia àgil més famosa, es basa en l'equip i els rols diferenciats, donant importància a la interacció de l'equip entre ells i amb el client.

Fa servir una cua amb prioritat d'activitats pendents i una petita cua d'activitats a realitzar en cada iteració, cosa que resulta útil per gestionar els riscs 2, 3 i 5. Tot i que resulta perfecta per equips petits, per al projecte que ens ocupa és poc útil, ja que l'equip és molt distribuït i la metodologia s'aplica només per l'autor del projecte i el client, mentre que el grup ISA segueix la seva pròpia metodologia de forma independent.

Finalment s'ha seleccionat Agile UP per la familiaritat prèvia amb RUP i la presència de la cua de tasques amb prioritat. Tot i així, el que s'ha aplicat no és Agile UP pura, sinó una instància de la metodologia simplificada i adaptada al projecte, com es fa sovint.

5.1.1. Agile UP

Agile UP és una metodologia que es divideix en quatre fases: Inici, Elaboració, Construcció i Transició. A l'inici es determina l'abast del projecte, les característiques, el pla de treball, etcètera, i es comença l'especificació del sistema a construir. A l'elaboració es completa l'especificació i es realitza el disseny del sistema. A la construcció s'implementa, tot canviant l'especificació i el disseny quan sigui necessari, i a la transició es completa el desplegament, es valida el sistema, s'integra amb el client (entrenant als usuaris si fa falta, assegurant que funciona amb un ús real) i es tanca el projecte.

Cada fase d'aquestes es divideix en curtes iteracions enfocades a cobrir casos d'ús concrets, de manera que es desenvolupin funcions usables en comptes de capacitats intangibles fins al final del desenvolupament.

Un dels principis d'Agile UP és fer servir documentació de la forma *Just Barely Good Enough* (amb prou feines suficientment bona); només els models necessaris per a entendre com realitzar el sistema, com ara el diagrama de classes, diagrames de seqüència sols de les funcions crítiques, etcètera, i fer-los en forma d'esborranys. És més ràpid usar una pissarra o un full en brut per esbossar les idees que fer servir eines CASE. La idea és que un gruix de models que pateixen canvis durant el desenvolupament incrementa el cost de gestió de la documentació, que per altra banda ha de servir per accelerar, i no frenar.

En el projecte s'ha modelat de la forma Just Barely Good Enough, però com que és un projecte de final de carrera i necessita una memòria amb gran detall del sistema, s'ha tractat la memòria com una tasca més del projecte i s'han refinat els models a posteriori.

5.2. Pla de treball

L'estratègia del projecte consisteix en dos passos: Primerament, desenvolupar un prototip funcional de SALMonADA que resulti capaç de monitorar documents SLA amb les mètriques que permet SALMon en un principi; el segon i últim pas és evolucionar i perfeccionar tot el sistema per a fer-lo concordar amb el disseny desitjat.

Es desitja tenir aquest primer prototip per tal de poder donar suport empíric a l'article [SALMonADA_PESOS]. A tal fi, les tasques necessàries són el disseny de l'arquitectura, l'especificació dels documents SLA i MMD, el tractament de documents SLA i el seu anàlisi per detectar violacions, la implementació dels components *Analyzer* i *Publisher*, i la modificació de SALMon per a integrar ambdós components.

L'estratègia per aquesta part és començar amb l'especificació i disseny del sistema en general, tot seguit desenvolupar i integrar el component *Publisher*, i finalment desenvolupar i integrar el component *Analyzer*, integrant-lo també amb *ADA*, la modificació del qual seria paral·lela per part del grup ISA.

Per a completar el sistema a partir del prototip, és necessari modificar SALMon per permetre la gestió d'esdeveniments i l'addició de noves mètriques en temps d'execució (vegeu el capítol 6.3 Model conceptual, pàgina 58), la implementació del component *Violation Publisher* i la implementació de la *interfície web*.

L'estratègia planejada és la de començar amb l'evolució del *Monitor* de SALMon, modificant la base de dades i el codi en dos blocs independents:

- Primer, permetre l'addició de noves mètriques en temps d'execució, mantenint la resta de funcions compatibles amb el canvi.
- A continuació, permetre la gestió d'esdeveniments, mantenint la resta de funcions compatibles.

Si cap d'aquests blocs falla en reeixir repetidament, s'abandona (permanent o temporalment) el desenvolupament d'aquella funció per evitar la paralització del desenvolupament. Això és un risc probable perquè els dos blocs es basen en canvis estructurals verticals del model conceptual del sistema i per tant de la capa de dades, la qual encapsula gran part de la complexitat operacional del sistema.

A continuació, es desenvoluparà el *Violation Publisher* i tot seguit la *interfície web*, i s'integraran al sistema. Aquests components serviran per fer les proves de funcionalitat definitives. Finalment es redactaran els manuals d'usuari i desenvolupador.

Durant tot el projecte i de forma paral·lela es redactarà aquesta memòria.

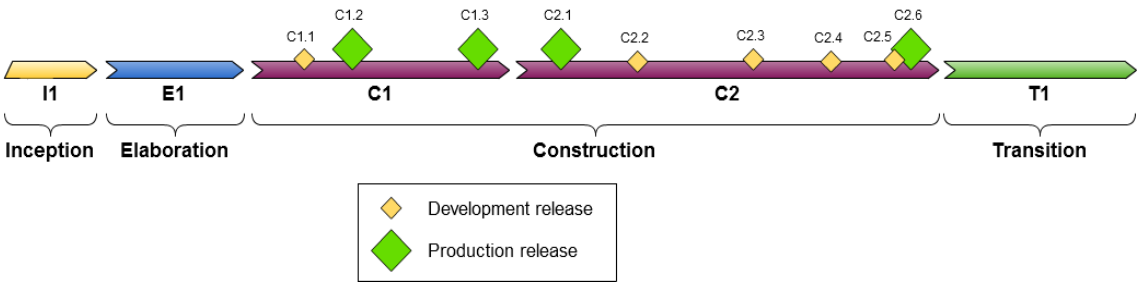

Figura 24: Cronograma de l'estratègia del pla de projecte.

El pla, d'acord amb la metodologia AgileUP, s'ha dividit en una fase de concepció per a l'anàlisi del problema i desenvolupament del pla de projecte, una d'elaboració per al disseny del sistema, una de construcció per al desenvolupament del primer prototip i una segona fase de construcció per al desenvolupament de la versió final, i una fase de transició per al traspàs de coneixements i tancament del projecte. A més, la construcció contindrà els següents llançaments (vegeu la Figura 24):

C1.1. Llançament del component *Publisher*.

C1.2. Llançament de la versió 0.1 del component *Analyzer*: Només emmagatzema SLAs i configura el *Monitor*.

C1.3. Llançament de la versió 0.2 de l'*Analyzer*: Gestiona dades de qualitat en MMDs. Prototip llest.

C2.1. Llançament de la versió 1.1 del component *Proxy*: Interfície orientada a esdeveniments i interacció no intrusiva.

C2.2. Llançament de la versió 1.2 dels components *Monitor* i *Proxy*: Permeten la creació de noves mètriques.

C2.3. Llançament de la versió 1.3 dels components *Monitor* i *Proxy*: Arquitectura dirigida per esdeveniments.

C2.4. Llançament de la versió 1.0 de l'*Analyzer*.

C2.5. Llançament del component *Violation Publisher*.

C2.6. Llançament de la *interfície web*. Versió final del sistema.

A continuació (vegeu la Figura 25) es pot apreciar un diagrama de Gantt que resumeix aquesta planificació segons les tasques descrites a la secció 4.4.5 Desglossament de tasques (pàgina 43), que ja contemplen les mitigacions per als riscs del projecte.

WBS	Task Name	Predecessors	Team	Work	Start	Finish
1	**Plataforma per al monitoratge i l'explicació de violacions de clàusules de documents SLA**			**965 hrs**	**Tue 22/01/13**	**Tue 18/06/13**
1.1	**SLA**			**0 hrs**	**Wed 23/01/13**	**Thu 31/01/13**
1.1.1	Definir WS-Agreement-MD	22	ISA	0 hrs	Wed 23/01/13	Thu 24/01/13
1.1.2	**Parser SLA**		**ISA**	**0 hrs**	**Thu 24/01/13**	**Tue 29/01/13**
1.1.3	**Repositori SLA**		**ISA**	**0 hrs**	**Fri 25/01/13**	**Wed 30/01/13**
1.1.4	**Compilador SLA a MMD**		**ISA**	**0 hrs**	**Fri 25/01/13**	**Wed 30/01/13**
1.1.5	**Verificador MMD**		**ISA**	**0 hrs**	**Thu 24/01/13**	**Thu 31/01/13**
1.2	**Analyzer**			**236 hrs**	**Tue 22/01/13**	**Tue 28/05/13**
1.2.1	**Parser MMD**		**ISA/GESSI**	**32 hrs**	**Tue 22/01/13**	**Tue 14/05/13**
1.2.2	Especificar sistema		GESSI	70 hrs	Tue 22/01/13	Mon 04/02/13
1.2.3	Dissenyar Analyzer	27;3	GESSI	20 hrs	Mon 04/02/13	Wed 06/02/13
1.2.4	**Components Analyzer**		**GESSI**	**96 hrs**	**Thu 07/02/13**	**Tue 28/05/13**
1.2.5	Testejar Analyzer	34	GESSI	12 hrs	Tue 14/05/13	Thu 16/05/13
1.2.6	Desplegar Servei Analyzer	35	GESSI	6 hrs	Thu 16/05/13	Thu 16/05/13
1.3	**Violation Publishing**			**160 hrs**	**Fri 17/05/13**	**Thu 13/06/13**
1.3.1	**Violation Publisher**		**GESSI**	**50 hrs**	**Fri 17/05/13**	**Mon 27/05/13**
1.3.2	**Interfície Web**		**GESSI**	**110 hrs**	**Tue 28/05/13**	**Thu 13/06/13**
1.4	**SALMon**			**321 hrs**	**Fri 15/02/13**	**Wed 24/04/13**
1.4.1	**Publisher**		**GESSI**	**53 hrs**	**Fri 15/02/13**	**Tue 26/02/13**
1.4.2	Dissenyar Monitor	69	GESSI	42 hrs	Wed 27/02/13	Wed 06/03/13
1.4.3	**Modificar disseny mètriques**		**GESSI**	**67 hrs**	**Wed 13/03/13**	**Thu 28/03/13**
1.4.4	**Afegir tractament d'esdeveniments**		**GESSI**	**58 hrs**	**Thu 07/03/13**	**Fri 05/04/13**
1.4.5	**Afegir filtratge de valors**		**GESSI**	**47 hrs**	**Fri 05/04/13**	**Mon 15/04/13**
1.4.6	Implementar mètriques predefinides	55;64	GESSI	8 hrs	Mon 15/04/13	Mon 15/04/13
1.4.7	Testejar Monitor	57;62	GESSI	16 hrs	Tue 16/04/13	Thu 18/04/13
1.4.8	**Measure Instruments**		**GESSI**	**30 hrs**	**Tue 26/02/13**	**Wed 24/04/13**
1.5	**Documents**			**248 hrs**	**Tue 22/01/13**	**Tue 18/06/13**
1.5.1	Manual d'usuari		GESSI	10 hrs	Fri 14/06/13	Mon 17/06/13
1.5.2	Manual de desenvolupador		GESSI	10 hrs	Mon 17/06/13	Tue 18/06/13
1.5.3	Memòria		GESSI	200 hrs	Thu 14/03/13	Tue 16/04/13
1.5.4	Informe previ		GESSI	12 hrs	Mon 11/03/13	Tue 12/03/13
1.5.5	Gestió del projecte		GESSI	16 hrs	Tue 22/01/13	Tue 28/05/13

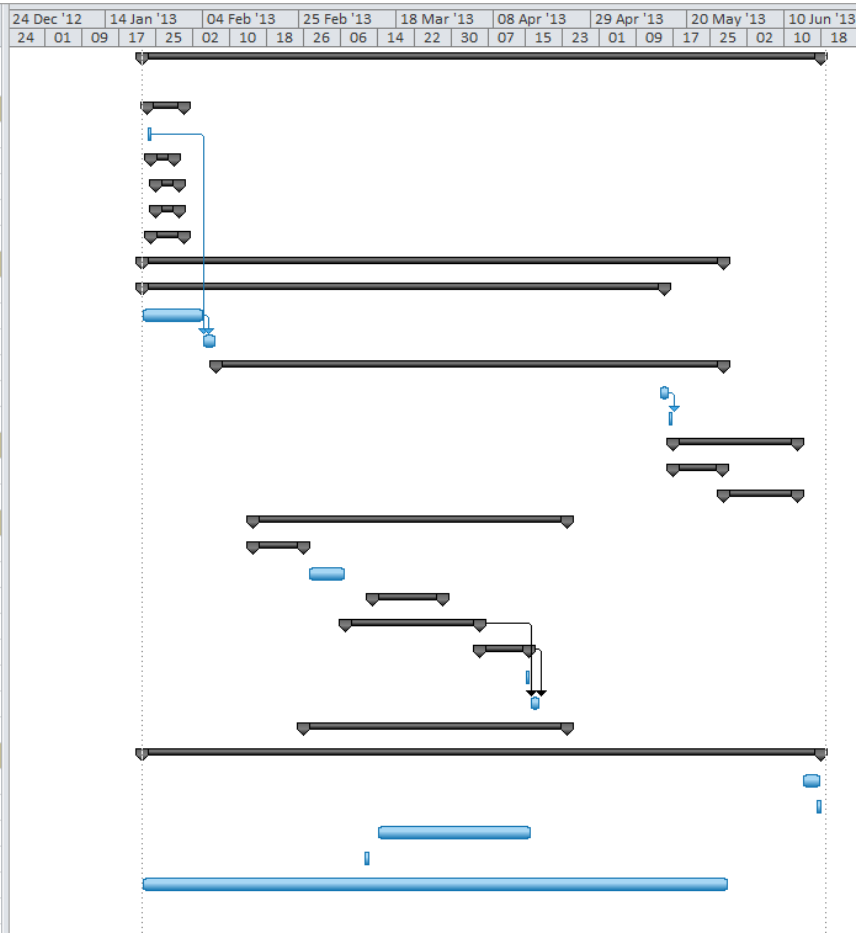

Figura 25: Diagrama de Gantt que mostra la planificació als estadis inicials del projecte.

6. Especificació

En aquest capítol s'analitza el problema en profunditat a partir de l'anàlisi de requisits fet amb anterioritat. Primerament s'analitza el model d'objectius del context del sistema. A partir d'aquí, s'extreuen els casos d'ús del sistema i se'n fa el model conceptual, i finalment es modela el comportament amb seqüències d'interacció de l'usuari amb el sistema.

6.1. Model d'objectius

Un model d'objectius modela, per a cadascun dels actors i rols de l'entorn, els objectius que el mouen en relació al problema i de què depèn per aconseguir cadascun dels objectius. És útil comprendre l'entorn del problema perquè ens ajuda a entendre el problema, com solucionar-lo i què espera cadascun dels actors de la solució.

Aquest model està fet en el llenguatge i* (pronunciat en anglès "eye star"), que forma part de l'estàndard internacional User Requirements Notation (URN) [Yu01]. És un Strategic Rationale (SR) Model (model de raonament d'estratègia), que descriu els objectius de cada actor en relació amb els altres actors, i és per tant orientat a actors.

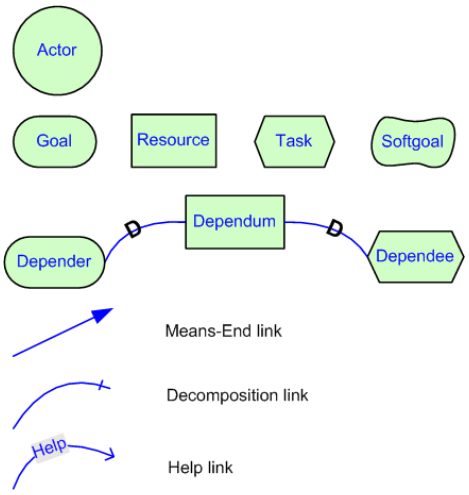

Figura 26: Llegenda de i*

Els models SR es composen de diversos elements. Aquest és un model simple que només fa servir els següents elements (vegeu la Figura 26), que es poden definir així [iStarWiki]:

- **Actors:** Entitats actives que realitzen accions per aconseguir objectius fent servir els seus coneixements.
- **Objectius ("Goals"):** Desitjos d'un actor. Saber si un objectiu es compleix és una observació objectiva.
- ***Softgoals*:** Objectius tals que saber si es compleixen és una apreciació subjectiva de l'actor.
- **Tasques ("Tasks"):** Una acció específica que l'actor desitja dur a terme.
- **Recursos ("Resources"):** Entitats físiques o informació que l'actor desitja tenir.
- **Relacions:** S'utilitzen quatre tipus de relacions:
  - **Dependència:** Un desig d'un actor ("Depender") depèn d'un altre element ("Dependee") per aconseguir quelcom necessari pel seu compliment ("Dependum").
  - **Relació Mitjà-Fi ("Means-End link"):** Un element relacionat amb un objectiu com a Mitjà-Fi és un mitjà per aconseguir aquest objectiu.
  - **Descomposició ("Decomposition link"):** Una tasca es descompon en elements més concrets que expressen com s'ha de dur a terme i què es necessita per dur-la a terme.
  - **Ajuda ("Help link"):** Una contribució positiva a un softgoal que no és suficient per aconseguir-lo.

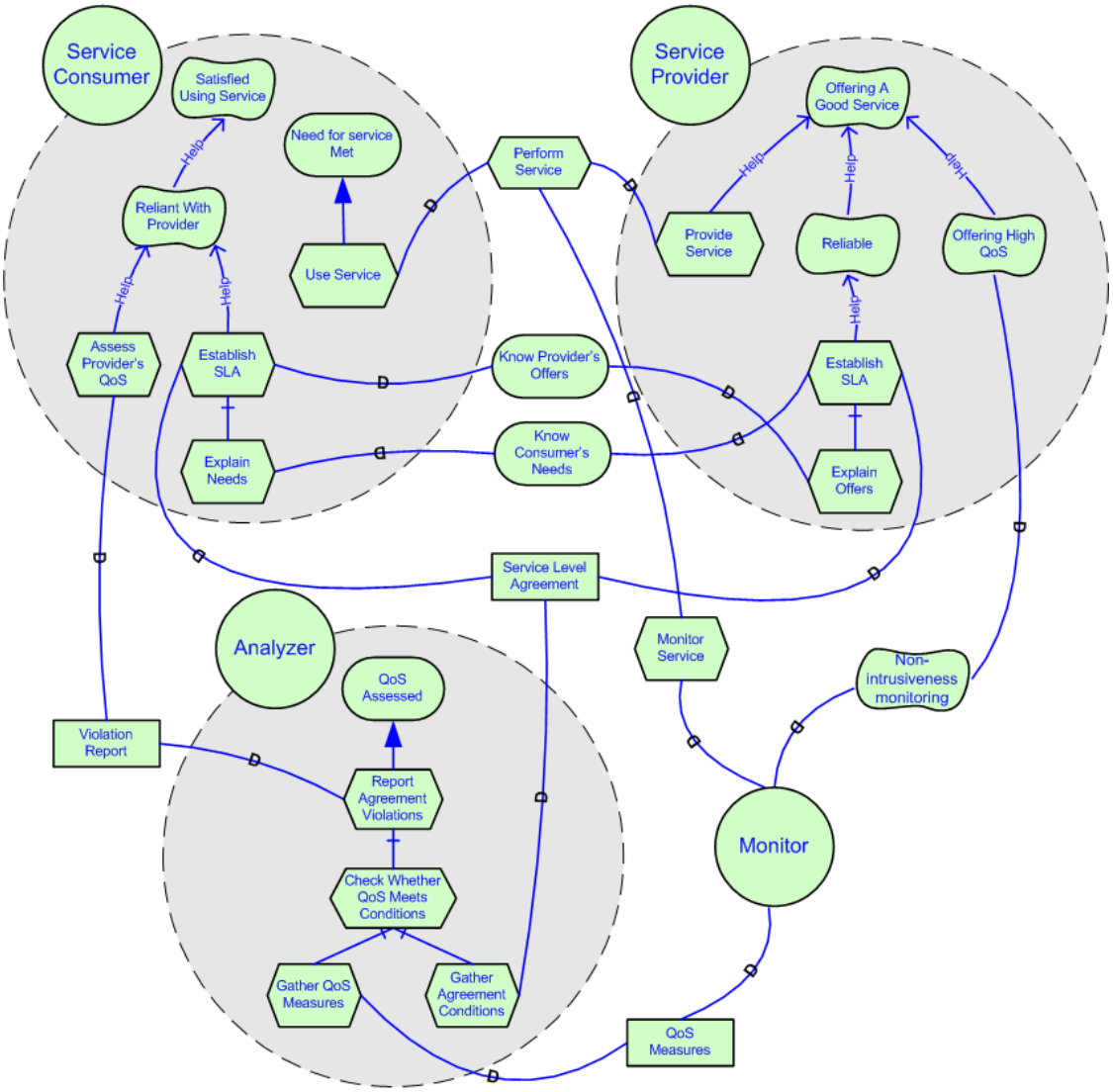

Figura 27: Strategic Rationale Model

Cal notar que l'*Analyzer* i el *Monitor* són dos rols que poden assumir tant el consumidor com el proveïdor concrets. L'únic objectiu del Monitor és monitorar, així que s'ha obviat el contingut del rol en el model.

El problema que es pretén resoldre és automatitzar els actors d'*Analyzer* i *Monitor* (vegeu la Figura 27). Com es pot observar, aquests dos rols depenen de l'entorn en dues ocasions. L'*Analyzer* depèn del SLA per adquirir les condicions de l'acord i poder comprovar si la qualitat respecta el contracte. Per altra banda, el *Monitor* depèn de la realització del servei entre consumidor i proveïdor per monitorar-lo i poder proporcionar les mesures a l'*Analyzer*. A més, el consumidor del servei depèn de l'informe de violacions de l'*Analyzer* per poder tenir garanties vers el proveïdor (tot i que de vegades pot ser el mateix proveïdor).

6.2. Model de casos d'ús

Com que hem de convertir l'*Analyzer* i el *Monitor* en el sistema software, podem observar que les seves dues dependències són en realitat entrades al sistema, i la dependència del consumidor vers el sistema és una sortida que el sistema ha de proporcionar.

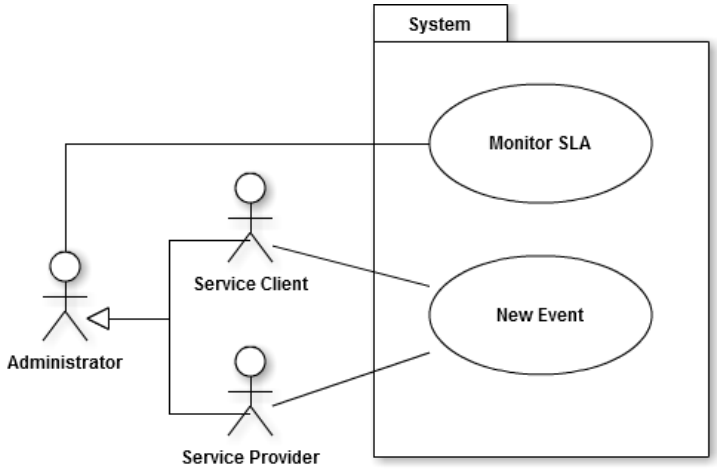

Figura 28: Model de casos d'ús.

Així, veiem que tenim dos casos d'ús (vegeu la Figura 28). D'acord amb els requisits funcionals 2.1 a 2.4 i els requisits no funcionals 1.1 a 1.6 sobre interfícies externes, s'estableix el comportament dels casos d'ús com segueix:

Cas d'ús 1: Monitorar SLA	
Descripció:	
Estableix un document SLA per tal que el sistema monitori el compliment de les clàusules.	
Precondició:	

Actors: Administrador	
Curs típic d'esdeveniments	
Actor	**Sistema**
1. Per a cada mètrica que apareix a les clàusules del SLA, l'usuari introdueix al sistema una nova mètrica amb les seves dades (tipus de dada, unitat i direcció del servei web que la calcula).	
	2. El sistema desa cada nova mètrica.
3. L'usuari introdueix l'SLA a monitorar en el sistema.	
	4. El sistema es prepara per monitorar l'SLA i mostra una confirmació.
Cursos alternatius	
2a. Ja existeix una mètrica amb el mateix nom en el sistema.	
	1. El sistema mostra un missatge d'error amb informació sobre la mètrica existent.
2a. L'usuari determina que és la mètrica que necessita, i omet el pas.	
2b. L'usuari determina que és una mètrica diferent, i afegeix la nova amb un nom diferent.	
3a. L'SLA conté errors i no es pot començar a monitorar.	
	1. El sistema mostra un missatge d'error indicant una o més possibles causes de l'error.

Cas d'ús 2: Nou Esdeveniment	
Descripció:	
Detecta un esdeveniment que el sistema està configurat per monitorar. En calcula la QoS, comprova si viola cap condició de cap SLA, i en cas afirmatiu notifica a les parts interessades.	
Precondició:	
S'ha configurat un SLA (Cas d'ús 1). Un o més SLA monitorats contenen clàusules que involucren l'esdeveniment detectat. Aquestes clàusules depenen de mètriques que han estat configurades i funcionen.	
Actors: Consumidor del servei i Proveïdor del servei.	
Curs típic d'esdeveniments	
Actor	**Sistema**
1. L'usuari proveïdor del servei publica el servei.	
2. L'usuari consumidor del servei en fa ús a través del sistema.	
	3. El sistema detecta un esdeveniment, i en calcula la QoS.
	4. El sistema comprova que la QoS no viola cap SLA.
Cursos alternatius	
4a. La QoS viola una de les condicions d'un SLA.	
	1. El sistema mostra una advertència amb informació sobre la violació.

6.3. Model conceptual

Cal tenir en compte que certes restriccions de disseny del projecte afecten a la forma del model conceptual. En concret, el requisit 5.5 obliga que el sistema faci servir els components *ADA* i *SALMon*. El component *ADA* només reconeix documents SLA sobre serveis web SOAP[1], i el sistema està acoblat a *ADA* mitjançant el document MMD que descriu el requisit 6.2. No obstant, el sistema ha de ser extensible per altres protocols a part de SOAP com descriu el requisit 6.1, i com que fer *ADA* extensible a altres protocols queda fora de l'abast del projecte[2], això es fa mitjançant components canviables d'una arquitectura orientada a serveis tal com descriu el requisit 5.1.

Això fa que el model conceptual tingui dues parts ben diferenciades: Una part acoblada al model d'*ADA* i per tant al protocol SOAP, i una altra de més abstracta que és extensible. Com veurem més endavant[3], aquestes parts corresponen als components *Analyzer* i *Monitor* respectivament.

6.3.1. Model de l'*Analyzer*

Pel que fa a la part acoblada a l'especificació SOAP, el model que abasta el projecte és una representació simple de la complexitat del protocol. Com es pot veure a la Figura 29, el projecte considera el document SLA com una caixa negra de la qual s'ocupa *ADA*.

[1] Vegeu el capítol 3.1.3. ADA (pàgina 20).
[2] Vegeu el capítol 4.4.2. Què NO permet el sistema (pàgina 41).
[3] Vegeu el capítol 7.1.4. Model de desplegament (página 57).

A més, es considera que un SLA estableix condicions sobre un i només un servei web SOAP, que es composa d'informació sobre el servei i les seves diferents operacions. En altres estàndards, això no té perquè ser així.

Un document MMD estableix que s'ha de monitorar un servei web. Conté les dades necessàries per a monitorar-lo, així com la qualitat del servei monitorada fins al moment en forma de mesures de mètriques. A més, es marca el període de temps durant el qual s'ha de monitorar el servei, i un període més concret per cada mètrica concreta.

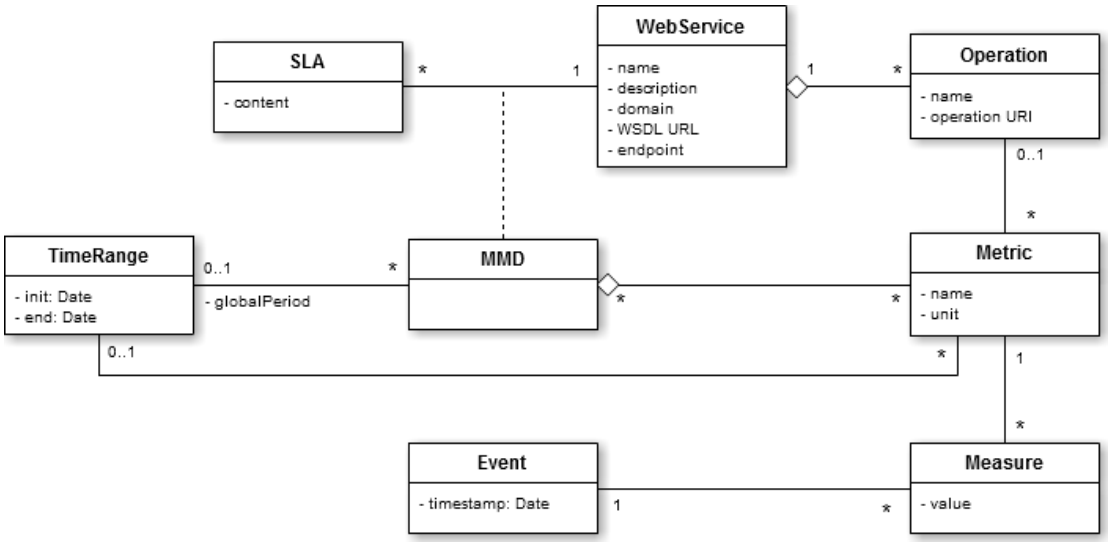

Figura 29: Model conceptual de l'Analyzer

Restriccions textuals:

- **RT1:** Totes les *Metrics* d'un *MMD* que monitoren una *Operation* concreta són d'una *Operation* del seu *WebService*.
- **RT2:** Un *MMD* que no té *TimeRange* es considera que s'ha de monitorar sempre.
- **RT3:** Totes les *Metrics* d'un *MMD* que té *TimeRange* tenen un *TimeRange* comprès dins del rang del *MMD*. Això vol dir que la seva data d'inici ha de ser igual o posterior, i la data de fi igual o anterior a les del *TimeRange* del *MMD*.
- **RT4:** L'instant de temps en el que succeeix la *Measure* d'una *Metric* (el "*timestamp*" del seu *Event*) ha d'estar dins el rang temporal de la mètrica, només si en té.

6.3.2. Model del *Monitor*

El model conceptual del *Monitor* resulta de la modificació del model conceptual original de SALMon (vegeu la Figura 30) per tal que el *Monitor* sigui extensible per a nous protocols (requisit 6.1) i es puguin afegir noves mètriques en temps d'execució (requisit 1.4).

El model original té diverses limitacions que cal resoldre:

- El concepte de mètrica és rígid: Una instància defineix la obligació de monitorar aquella mètrica per un servei i operació concrets, mentre que el concepte de mètrica que aquest projecte contempla queda definit per la subclasse. Afegir una nova mètrica requereix afegir una nova subclasse, amb tots els canvis d'implementació que això comporta. No és possible afegir mètriques en temps d'execució.
- El model diferencia entre mètriques amb abast de servei (Availability, RoundTripTime) i mètriques amb abast d'operació (ResponseTime). Aquesta distinció, útil en el moment en que el model original fou concebut, no és real, ja que no és possible monitorar cap mètrica d'un servei web SOAP sense cridar una operació[1], i el mètode que es feia servir per calcular el RoundTripTime (comprovar el temps de resposta del servidor on s'allotja el servei) és en realitat una mètrica sobre un servei REST[2].
- Les mètriques derivades són subclasse de les mètriques bàsiques. Això impedeix calcular mètriques derivades respecte més d'un servei web o operació, així com calcular mètriques més complexes que depenguin de més d'una mètrica bàsica. En general, dificulta el filtratge de valors de mètriques respecte qualsevol paràmetre rellevant (com, per exemple, la franja horària, la localització geogràfica del client, etc.).
- Assumeix que un servei web només pertany a un sistema SOA. No obstant, un mateix servei pot ser usat en diversos sistemes a la vegada, i pot interessar tant el rendiment del servei en un sistema concret (per a un contracte vers un únic client) com la seva qualitat de servei general.
- El model combina les dades de monitoratge amb les dades sobre com fer Testing dels serveis, acoblant ambdues facetes. No obstant, el Testing és un camp altament dependent del protocol i que per tant no pot ser generalitzat per a tots.
- Es té en compte una funcionalitat que ja és obsoleta: La simulació de mesures ("*Behaviour*").

[1] No és possible amb sondes interceptores, que és l'únic tipus de sonda de què disposa SALMon originalment.
[2] Vegeu el capítol 3.1.4. SALMon (pàgina 20), i en concret la nota al peu de la pàgina 22.

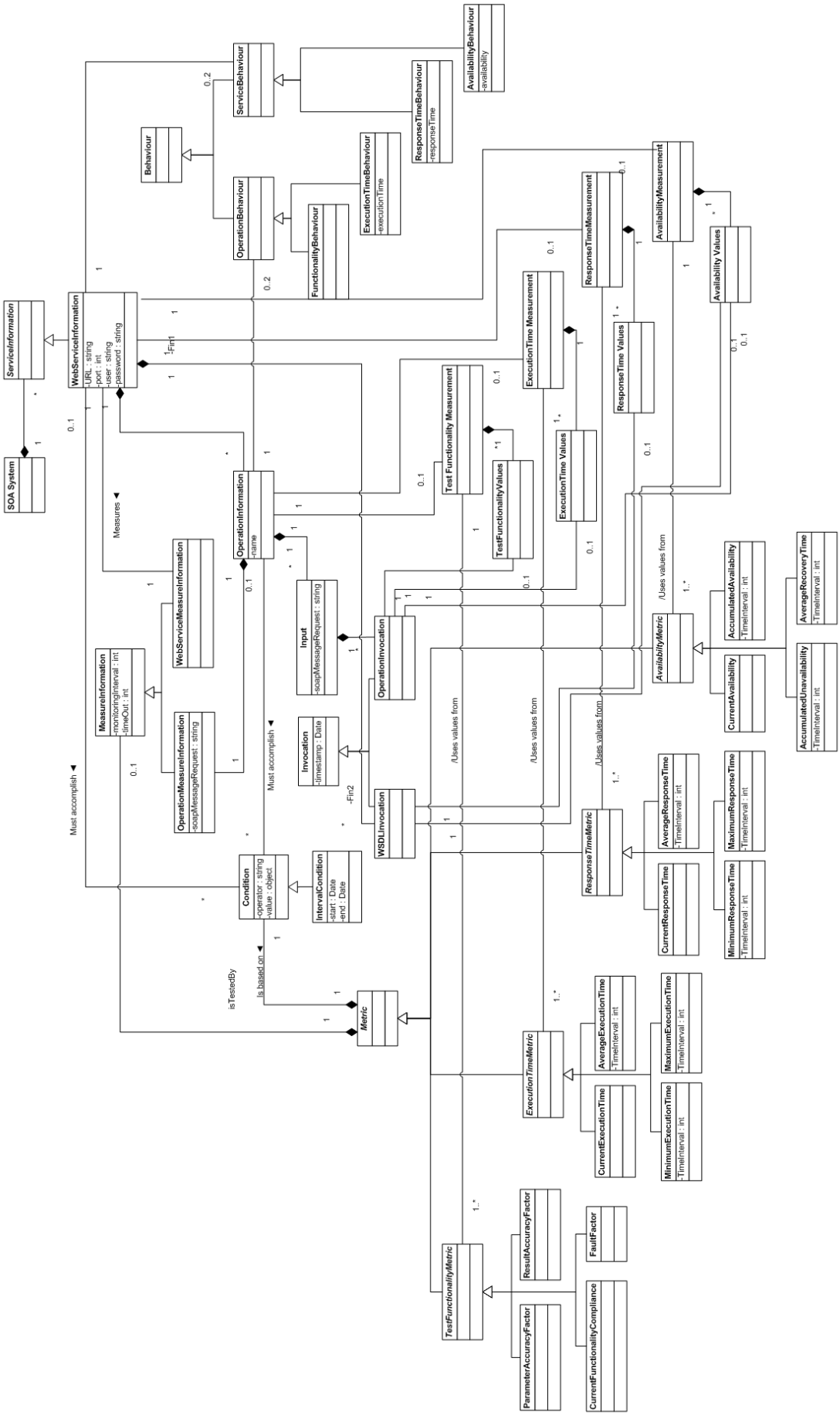

Figura 30: Model conceptual de SALMon original.

El model conceptual del *Monitor* modificat (vegeu la Figura 31) permet tot el que l'original sense aquestes limitacions, a canvi de delegar el testing del servei a sistemes externs (que poden fer servir el *Monitor*), ja que es troba fora de l'abast del projecte[1].

Aquest model es basa en la relació de quatre conceptes importants:

- Un **Context** és un àmbit o característica de la qual pot interessar obtenir dades de qualitat de servei. Entren dins d'aquesta definició tant les entitats a les quals es pot atribuir la qualitat (el sistema o servei i les seves parts) com les característiques de l'esdeveniment monitorat. Així, un servei web i una operació d'aquest servei són contexts, però també ho són un client del servei, una resposta donada pel servei i la localització geogràfica del client, per posar alguns exemples.
- Un **Esdeveniment** és un succés que pot ésser monitorat. Els esdeveniments es defineixen pel moment en el que succeeixen ("timestamp") i pels seus atributs, que són contexts. Per exemple, un esdeveniment de tipus "SOAPResponse" pot tenir com a atributs el client, el resultat, la operació cridada, el servei, el sistema SOA en el context del qual s'ha realitzat la crida, etc.
- Un **Instrument de Mesura** és un programa que, donades com a mínim les dades i context d'un esdeveniment, en calcula el valor d'una mètrica. S'ha modelat com un servei web SOAP ja que el disseny permetrà afegir nous instruments de mesura com a serveis web SOAP.
- Una **Mesura** és qualsevol dada d'un esdeveniment mesurable per un instrument de mesura del monitor. No és el mateix que un context. Conceptualment, un context és una mesura d'una relació de l'esdeveniment amb una entitat del món real, mentre que una mesura és una dada computada que no té perquè reflectir cap entitat real.

Cadascun d'aquests conceptes té tipus en els quals es pot classificar. Aquests tipus ens interessen per diversos motius, entre ells tenir aquest nivell d'abstracció addicional que ens permet afegir mètriques en temps d'execució, però també protocols a monitorar. Aquests conceptes abstractes són:

- El **Tipus de Context**, per exemple, "servei SOAP" o "client". No confondre amb un context específic, com seria "el servei SOAP a http://...Monitor" o "la operació NotifyEvent".
- El **Tipus d'Esdeveniment**, per exemple, "SOAP Request", "REST HTTP POST" o "SVN Commit". No confondre amb un esdeveniment concret, com seria "el missatge SOAP Request ocorregut el moment X amb atributs Y". Un tipus d'esdeveniment, a més, pot ser subtipus d'un altre, cas en el qual es considera que hereta els seus atributs. Això és útil en certs casos, com per exemple el cas que "SOAP Request" i "SOAP Response" són subtipus de "SOAP Call", i comparteixen molts dels seus atributs, com ara el servei, la operació i el missatge SOAP.
- El **Tipus d'Instrument de Mesura**, és a dir, la **Mètrica**[2] que calcula.

---

[1] Vegeu el capítol 4.4.2. Què NO permet el sistema (pàgina 41).

[2] Per a una definició del concepte de mètrica en el model, vegeu el capítol 2.2.2. Les mètriques de qualitat (pàgina 11).

- El **Tipus de Mesura**, per exemple "Response Time d'un SOAP Response". No confondre amb una mesura concreta, com seria "Response Time de l'esdeveniment X mesurat per Y = 150ms".

Noti's la diferència entre el Tipus de Mesura i la Mètrica. Un concepte que es té en compte és el fet que no totes les mètriques tenen sentit per a tots els tipus d'esdeveniment. Per exemple, no té sentit calcular el temps de resposta d'un esdeveniment que no és una crida amb resposta. El model, per tant, defineix aquest Tipus de Mesura com a "Mètrica computable per a un tipus d'esdeveniment" ("**ComputableMetricForEventType**"). Hi pot haver, per tant, mètriques que encara no es consideren computables per a cap tipus d'esdeveniment monitorat, i que per tant no poden ser monitorades encara.

La necessitat d'aquests conceptes pot ser traçada com segueix:

- El concepte de tipus de context és útil per respectar alhora els requisits 4.3 i 6.1, ja que permet afegir nous protocols al sistema en temps d'execució. A més, però, ferho menys flexible (subclasses per cada protocol i tipus d'esdeveniment) no és possible, ja que es necessita poder filtrar les mètriques per servei, operació, i en cas d'altres protocols, filtrar per altres contexts desconeguts en temps de disseny.
- El concepte de mètrica és necessari per respectar el requisit 1.4.
- Els conceptes de tipus d'esdeveniment i de tipus de mesura són necessaris per respectar el requisit 6.1 i permetre nous protocols, perquè no totes les mètriques funcionen per a tots els tipus d'esdeveniments.

Afegint nous tipus de contexts i esdeveniments es poden monitorar altres protocols diferents a SOAP, i fins i tot qualsevol sistema, software o no, monitorable mitjançant software. Això permet que es pugui monitorar pràcticament totes les mètriques de qualitat, com es pretenia[1], i fins i tot mètriques que no són de qualitat.

El model s'ha realitzat en base al patró d'anàlisi *Accountability Knowledge Level* del catàleg *Analysis Patterns: Reusable Object Models* de Martin Fowler [FowlerAP:24-27]. Aquest patró separa els dos nivells d'abstracció del model en dues capes:

- La capa de **coneixement** conté aquelles dades que són poc propenses a canviar en el temps i són normalment introduïdes per l'usuari administrador (en el cas de SALMon, l'únic usuari).
- La capa **operacional** conté aquelles dades sobre casos concrets, amb els quals el sistema tracta en temps real, i solen ser introduïdes pels usuaris normals del sistema (en el cas de SALMon, introduïdes automàticament per les sondes).

Ambdós nivells conserven certa simetria natural en les classes, i això pot ser observat en les restriccions textuals. L'addició de mètriques en temps d'execució en sí afegeix dos nivells d'abstracció en el model, la mètrica i la mesura, que són conceptes imprescindibles. A més, la necessitat de suportar diversos protocols (requisit 6.1) i permetre filtrar mesures per condici-

[1] Vegeu la Figura 5 (pàgina 17).

ons desconegudes[1] (requisit 4.3), fa que tant els esdeveniments com el concepte de serveis web i operacions hagin de pujar un nivell d'abstracció més amunt. A més, es contempla el patró d'anàlisi *Quantity* del mateix catàleg [FowlerAP:36-38], que permet manipular mesures de diferents unitats sense por a cometre errors per conversió d'unitats.

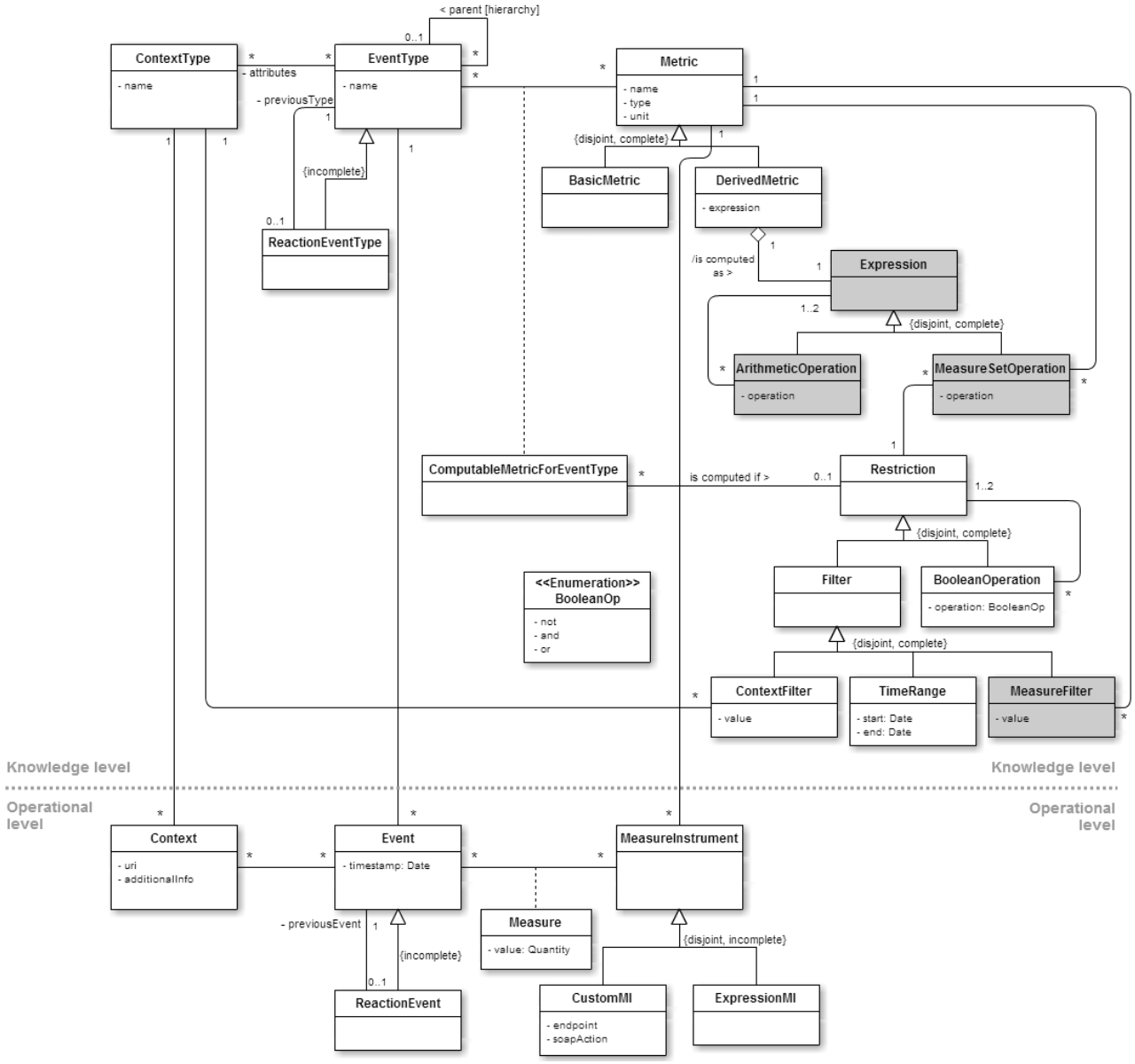

Figura 31: Model conceptual del Monitor. L'ombrejat de certes classes és només una ajuda visual referent al disseny final, i vol dir que les seves instàncies són calculades, no emmagatzemades.

Cal notar que es divideixen les mètriques en bàsiques i derivades, igual que a la versió anterior. Es defineix una mètrica bàsica com aquella que pot ser calculada en base a la informació que detecten les sondes i qualsevol informació addicional externa, i el seu càlcul es pot descriure amb un algorisme. Les mètriques derivades són aquelles que es calculen sobre els valors

[1] Se sap que per SOAP pot interessar filtrar per operació, per servei, o altres variables, però a priori no es pot saber ni classificar de cap manera més concreta les variables que altres estàndards requeriran, com per exemple per recurs o per petició HTTP en serveis web REST.

d'altres mètriques, siguin bàsiques i derivades, i el seu càlcul es pot descriure, per tant, en una expressió matemàtica.

Aquest model funciona com un marc de monitoratge genèric d'esdeveniments, i permet un processament d'esdeveniments complex[1] [Michelson] restringit a "arbres" d'esdeveniments on cada esdeveniment només té un predecessor. Això es fa mitjançant els esdeveniments de reacció.

Un **Esdeveniment de reacció** és un esdeveniment que no pot ser processat fins que no s'ha detectat el seu esdeveniment predecessor. Això és necessari per tipus d'esdeveniments de petició i resposta, doncs cal que el sistema sàpiga a quina petició correspon cada resposta.

Restriccions textuals[2]:

- **RT5:** Un *Context* no pot tenir *Events* amb un *EventType* que no es trobi entre els *EventTypes* del seu *ContextType*.
- **RT6:** Un *Event* no pot tenir cap *Context* que tingui un *ContextType* que no estigui entre els *ContextTypes* del seu *EventType*, o dels seus *ContextTypes* heredats. Es defineix recursivament el conjunt de *ContextTypes* heredats d'un *EventType* com els *ContextTypes* del seu parent unió els *ContextTypes* heredats d'aquest parent. Si no té parent, el conjunt és buit.
- **RT7:** Un *Event* no pot tenir *MeasureInstruments* amb una *Metric* que no es trobi entre les *Metrics* del seu *EventType*.
- **RT8:** Un *MeasureInstrument* no pot tenir *Events* amb un *EventType* que no es trobi entre els *EventTypes* de la seva *Metric*.
- **RT9:** L'*EventType* d'un *ReactionEvent* és de tipus *ReactionEventType*.
- **RT10:** El *previousEvent* d'un *ReactionEvent E* té el mateix *EventType* que el *previousType* del *ReactionEventType* de *E*.
- **RT11:** La relació parent entre *EventTypes* no conté cicles (modela una herència).
- **RT12:** Una *Measure* té una *ComputableMetricForEventType*. És a dir, la *Metric* del seu *MeasureInstrument* té una *ComputableMetricForEventType* per a l'*EventType* del seu *Event*, i viceversa.
- **RT13:** No hi ha dos *Contexts* amb el mateix *uri*.
- **RT14:** No hi ha dues *Metrics* amb el mateix *name*.
- **RT15:** Un *BooleanOperator* amb *operation* "*not*" només té una *Restriction*.
- **RT16:** Un *BooleanOperator* amb *operation* diferent de "*not*" té sempre dues *Restrictions*.
- **RT17:** Una *ComputableMetricForEventType* no pot tenir *Measures* tals que la *Restriction* avaluï a fals[3]. La *Restriction* de la *ComputableMetricForEventType* d'una *Measure* és una expressió booleana que és certa només quan un *Event* ha de ser

[1] Vegeu el capítol 7.1.1. Arquitectura dirigida pels esdeveniments (pàgina 63).

[2] La numeració de les restriccions continua les del model de l'Analyzer, per així tenir una referència única cadascuna.

[3] Un exemple per pair la definició: Configurem el sistema perquè monitori el ResponseTime per al tipus d'esdeveniments SOAPResponse. Afegim una restricció a la *ComputableMetricForEventType*, però, de la forma "webservice='CurrencyConverter' or webservice='ExchangeRatesService'". Això farà que, quan es detecti un esdeveniment de SOAPResponse que no sigui de cap d'aquests serveis web, no es monitorarà el ResponseTime.

monitorat amb aquella *Metric*. Es diu que una *Restriction* avalua a fals per a una *Measure* quan l'expressió booleana que formen els seus *Filters* i *BooleanOperators* avalua a fals per aquella *Measure*:

- Un *BooleanOperator* avalua a fals de la forma tradicional (si la seva *operation* és "*or*", avalua a fals només si les seves dues *Restriction* avaluen a fals per aquella *Measure*, etc.).
- Un *TimeRange* avalua a fals per una *Measure* només quan el *timestamp* del *Event* de la *Measure* no és posterior o equivalent al seu *start* i anterior o equivalent al seu *end* a la vegada (es troba dins aquell rang temporal).
- Un *ContextFilter* avalua a fals per una *Measure* només quan l'*Event* de la *Measure* no té cap *Context* amb un *uri* i *ContextType* iguals al *value* i *ContextType* del *ContextFilter* (l'*Event* té aquest *Context*).
- Un *MeasureFilter* avalua a fals per una *Measure* només quan l'*Event* de la *Measure* no té cap *Measure* amb un *value* i una *Metric* de la seva *ComputableMetricForEventType* iguals al *value* i *Metric* del *MeasureFilter* (l'*Event* té aquest valor d'aquesta mètrica[1]).

- **RT18:** La *Restriction* d'una *ComputableMetricForEventType* no pot contenir *Filters* de tipus *MeasureFilter* (totes les *Measures* d'una *ComputableMetricForEventType* són de la mateixa *Metric*; no té sentit restringir-les, perquè s'obtindria el conjunt idèntic o el conjunt buit).
- **RT19:** La *Metric* d'un *MeasureInstrument* que no sigui de tipus *ExpressionMI* és de tipus *BasicMetric*.
- **RT20:** La *Metric* d'un *ExpressionMI* és de tipus *DerivedMetric*.
- **RT21:** L'*Expression* d'una *DerivedMetric* és resultat de parsejar la seva *expression*.
- **RT22:** El *value* de tota *Measure* d'una *DerivedMetric* ha de ser, en el moment que es comenci a calcular, resultat directe de la seva *expression*[2]. El resultat d'una *expression* es calcula com el resultat d'una expressió matemàtica composta per operacions aritmètiques ("*ArithmeticOperations*") i valors ("*MeasureSetOperations*"). El resultat d'una *MeasureSetOperation* es calcula com una operació d'agregació (sumatori, màxim, mínim, mitjana, nombre, etc.) sobre un conjunt de valors de la seva *Metric*. Aquest conjunt de valors s'obté d'entre els *values* de *Measures* d'aquesta *Metric* tals que la *Restriction* de la *MeasureSetOperation* avalua a cert per elles (vegeu la restricció RT17, més amunt).

[1] Útil per filtrar, per exemple, les mesures de totes aquelles SOAPResponse que no contenien errors.

[2] Noti's que en qualsevol moment després de començar a calcular el *value*, nous *values* poden aparèixer al sistema que alterin el resultat de la *expression*. Fins i tot, nous *values* d'*Events* amb un *timestamp* anterior al de l'*Event* de la *DerivedMetric*. Retards en la detecció d'esdeveniments poden causar això, i el model fa explícit que no garanteix la consistència d'expressió i resultat al llarg del temps.

7. Disseny

En aquest capítol es descriu com s'ha dissenyat el sistema per implementar l'especificació descrita. Primerament s'explica com és l'arquitectura a tots els nivells i després s'entra en el disseny particular de cadascun dels mòduls implementats.

7.1. Arquitectura software

L'arquitectura del sistema està bastant restringida a una opció concreta, ja que diversos factors obliguen a que sigui així.

Per una part, l'especificació orienta el sistema a la detecció i monitorat d'esdeveniments. Com ja s'ha explicat[1], el monitoratge és una tasca lligada als esdeveniments, i fer el sistema centrat en la detecció d'esdeveniments és, si no l'única, la manera més senzilla de respectar els requisits 4.1, 4.2 i 4.3.

Per altra banda, el sistema ha d'integrar ADA i SALMon (requisit 5.5), els quals tenen ambdós una interfície SOAP. El sistema ha de ser orientat a serveis (requisit 5.1). A més, tots els components (Analyzer, ADA, SALMon, i els que apareguin) han de ser mòduls separats intercanviables (requisit 6.2). Això fa que la millor opció sigui fer que tots ells siguin serveis web.

Així doncs, el sistema és en part un sistema SOA i en part una arquitectura dirigida pels esdeveniments o *Event-Driven Architecture* (EDA). A continuació s'expliquen ambdues facetes en detall.

7.1.1. Arquitectura dirigida pels esdeveniments

Es pot dir que una arquitectura dirigida pels esdeveniments o EDA és un patró arquitectònic basat en la producció, detecció, consumpció i reacció a esdeveniments. La seqüència de processament d'esdeveniments (vegeu la Figura 32) pot ser dividida en quatre capes lògiques segons [Michelson]:

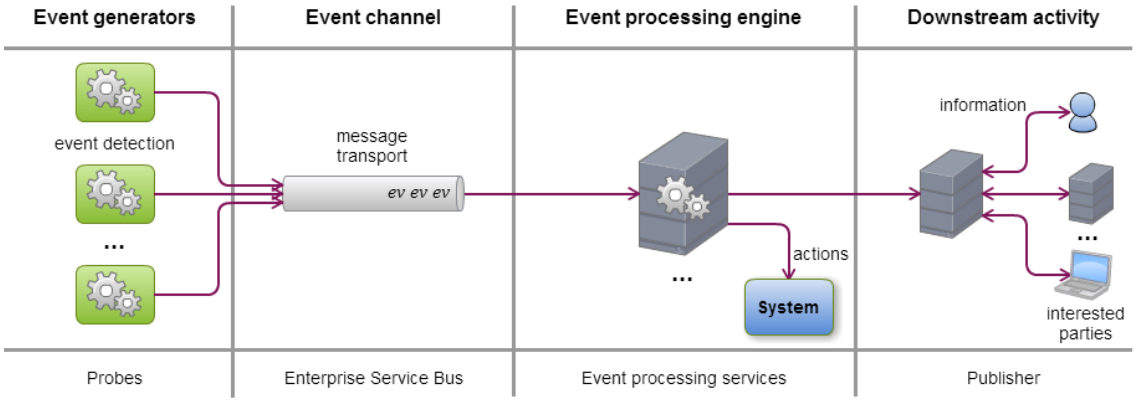

Figura 32: Seqüència de processament d'esdeveniments típica. A baix, components del projecte responsables de cada capa.

- **Generació d'esdeveniments:** Un o més components detecten certs esdeveniments succeïts, generen una estructura de dades i l'envien al canal d'esdeveniments.

[1] Vegeu el capítol 2.2.4. Monitoratge de mètriques (pàgina 13).

Aquesta estructura, anomenada també "esdeveniment" o *Event*, conté la informació bàsica de l'esdeveniment (nom, tipus, moment del succés) i tota aquella informació addicional que pugui resultar útil.

- **Canal d'esdeveniments:** El canal és la infraestructura de transport d'aquestes estructures de dades fins al motor de processament.
- **Processament d'esdeveniments:** Un o més motors processadors d'esdeveniments analitzen els esdeveniments detectats i els processen d'acord amb les regles preestablertes, efectuant les accions necessàries sobre el sistema (incloent generar nous esdeveniments). Si el motor analitza correlacions entre diferents esdeveniments abans de tractar-los, es parla de processament d'esdeveniments complex.
- **Activitats derivades:** Ja sigui per publicació automàtica o per consulta externa, el sistema inicia noves activitats i seqüències de negoci necessàries per a l'esdeveniment processat, posant-se en contacte amb persones, serveis, o altres sistemes.

Una arquitectura EDA pot ser fàcilment adaptada a una arquitectura SOA, formant el que s'anomena *Event-driven SOA*. La seva principal característica és que està composta per serveis, i normalment un Enterprise Service Bus s'ocupa de gestionar el canal d'esdeveniments. Tot i que n'hi ha d'altres, aquestes són les dues úniques característiques que aquest projecte aprofita d'aquesta arquitectura combinada.

En el cas de SALMonADA, doncs (vegeu els títols sota la Figura 32), les sondes són els generadors d'esdeveniments, mentre que el *Monitor* és el motor de processament d'esdeveniments. Cal, a més, notificar a l'*Analyzer* que s'ha monitorat un esdeveniment, iniciant així una nova seqüència de negoci per a comprovar si cap dels nous valors monitorats viola cap SLA. Això ho farà un nou component, que anomenarem *Publisher*, per mantenir la lògica de l'*Analyzer* completament desacoblada del *Monitor*.

7.1.2. Protocol de detecció d'esdeveniments

L'Arquitectura de SALMon permet que qualsevol programa pugui actuar com a sonda, detectant un esdeveniment, generant-ne la descripció lògica i enviant-lo al *Monitor* perquè el monitori. Malgrat tot, el conjunt d'esdeveniments inicial que interessa que el sistema monitori són missatges SOAP a serveis web monitorats, i el sistema ha d'incorporar una sonda capaç d'interceptar-los.

S'estableix com a precondició[1] per al client del servei web monitorat que els missatges han de ser enviats a través d'un *Enterprise Service Bus*[2] designat a tal fi. Aquest *ESB* s'encarrega d'enviar cada missatge al proveïdor del servei real i, paral·lelament, notificar al *Monitor* d'aquests esdeveniments.

El funcionament de l'*ESB* com a sonda es divideix en quatre fases (vegeu la Figura 33):

[1] Vegeu el capítol 4.4.2. Què NO permet el sistema (página 41).

[2] Cal tenir en compte que el sistema actua com a intermediari i, per tant, no respecta la privacitat dels missatges. En cas que un client vulgui preservar aquesta privacitat, caldrà que els protegeixi encriptant-los adequadament. Això no és diferent de qualsevol ús sense intermediaris, però el sistema no es fa responsable de la manca de seguretat, ja que no és un requisit del projecte.

1. El client envia el missatge SOAP a l'*ESB*.
2. L'*ESB* envia el missatge SOAP al proveïdor. Després, i només després d'haver enviat el missatge al proveïdor, construeix un esdeveniment de tipus SOAPRequest amb la informació del missatge (moment de la recepció, contingut del missatge, procedència, destí...) i l'envia al *Monitor*.
3. El proveïdor pot respondre en algun moment posterior a rebre el missatge, enviant el missatge al seu client, l'*ESB*.
4. L'*ESB* envia el missatge SOAP al client. Després, i només després d'haver enviat el missatge al client, construeix un esdeveniment de tipus SOAPResponse amb la informació del missatge i l'envia al *Monitor*.

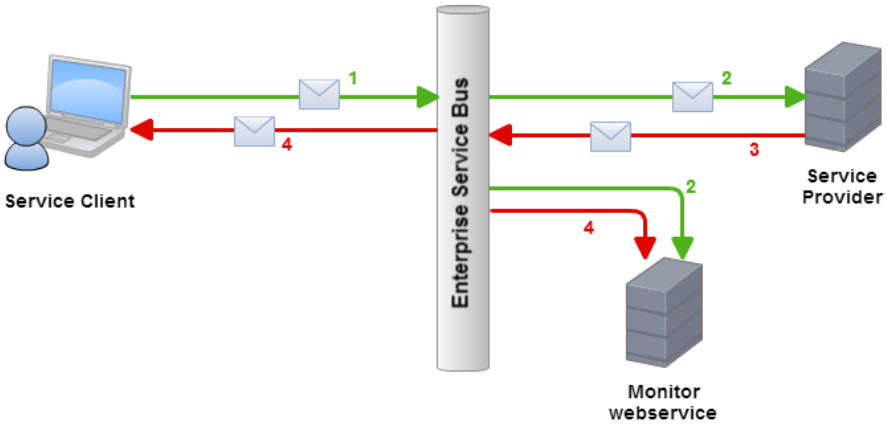

Figura 33: Protocol d'intercepció de missatges SOAP i generació d'esdeveniments.

El protocol garanteix que la presència de SALMonADA en el sistema interfereix el mínim en el temps de resposta del servei (requisit 3.1). No garanteix, però, que ambdós esdeveniments seran rebuts en ordre pel *Monitor*. Això dependrà del temps de resposta del proveïdor i del temps de processament de l'*ESB*.

L'*ESB* té la tasca complexa de construir les notificacions a partir del missatge SOAP rebut. Després d'un anàlisi de les possibilitats, s'ha seleccionat l'ESB Apache Synapse perquè donava facilitats a aquesta tasca.

7.1.3. Arquitectura orientada a serveis

El disseny de SALMonADA planteja la seva arquitectura com una coreografia de serveis, per tant un conjunt de serveis que es fan servir els uns als altres però, a diferència d'una orquestració[1], cap d'ells gestiona el procés de negoci sencer.

A SALMonADA, un servei inicia el procés de negoci, serveix d'interfície per a l'usuari i està acoblat a la vegada amb SALMon i ADA. Aquest és l'*Analyzer*. Té dues funcions: La primera és rebre peticions de monitorat d'un SLA i gestionar el procés de configuració del sistema; la segona és notificar a les parts interessades quan ocorre una violació d'un dels SLAs monitorats. Encara que pugui semblar un servei orquestrador, no ho és del tot.

[1] Vegeu el capítol 2.1. El context: Arquitectures orientades a serveis (pàgina 9).

L'*Analyzer* és el servei que inicia i gestiona el cas d'ús 1 Monitorar SLA. El cas d'ús 2 Nou Esdeveniment, però, ha de ser iniciat per un altre component. Aquest component és l'*ESB*.

Per descriure com es comporta aquesta coreografia s'ha usat un model de coreografia mitjançant una *xarxa Petri d'interaccions* o *Interaction Petri Net*, un llenguatge formal que estén el llenguatge de xarxes de Petri tradicional i pot ser usat per descriure coreografies de serveis tal com descriu [Decker].

La semàntica és intuïtiva: Una transició A > B de tipus C es dispara quan A envia a B un missatge de tipus C. Com a la resta de xarxes de Petri, una transició només pot ser disparada si té un o més *tokens* (punts negres) a cada *place* (cercle) d'entrada. En ser disparada, consumeix un token de cada place d'entrada i afegeix un token a cada place de sortida. A continuació es pot observar aquest model (vegeu la Figura 34).

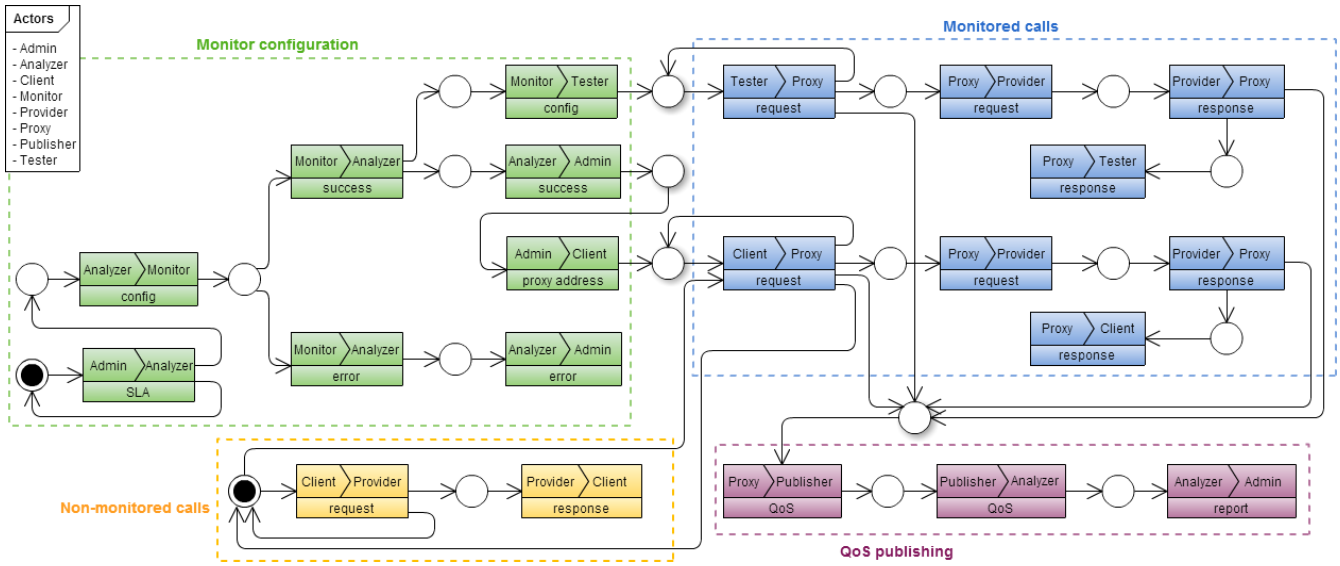

Figura 34: Model de coreografia de SALMonADA.

Cal notar que l'actor "Admin" del model fa referència a l'usuari que introdueix el SLA a monitorar; l'actor "Tester" no existirà en el sistema, però pot ser afegit; l'actor "Analyzer" inclou també ADA, ja que sempre que es crida l'*Analyzer*, aquest crida a *ADA*; i que l'actor Proxy és la combinació de l'*ESB* i del servei *Proxy*.

Fins ara, s'ha parlat indistintament de SALMon com a *Monitor* del projecte, però cal observar que el disseny de SALMon està format per dos components: El *Monitor* i el *Proxy*. El servei *Monitor* té una sola funció: Rebre un MMD i preparar-se per monitorar aquell servei. El servei *Proxy*[1] té com a funció rebre l'avís de la sonda d'un nou esdeveniment, monitorar-lo si escau, i notificar a l'*Analyzer* que comprovi si cap SLA ha estat violat. Al model, l'actor *Proxy* inclou a l'*ESB*, ja que sempre que l'*ESB* actua, aquest crida al *Proxy*.

[1] En el codi, el Proxy rep el nom de "SALMonEtE" per motius històrics. Ara, l'acrònim vol dir "SALMon Event Engine".

El model mostra que les crides no poden ser interceptades pel servei *Proxy* fins que l'*Analyzer* ha configurat degudament el *Monitor*. A més, una vegada s'ha monitorat una crida, la QoS pot ser notificada a l'*Analyzer* per mitjà del component *Publisher*[1].

S'ha dissenyat el sistema perquè es puguin afegir noves mètriques en temps d'execució tal com diu el requisit 1.4. Per fer-ho, s'ha dissenyat el motor de processament d'esdeveniments de forma que envia la informació de l'esdeveniment a tots els *Instruments de Mesura* de les mètriques involucrades. Aquests *Instruments de Mesura* poden ser classes internes predefinides o serveis web SOAP externs amb una interfície predefinida.

Pel fet de ser una arquitectura orientada a serveis, el disseny incorpora els patrons de Remote Facade i Data Transfer Object (DTO) [FowlerEAA] per a totes les crides a un servei.

7.1.4. Model de desplegament

Com que el projecte tracta amb una arquitectura distribuïda i escalable, resulta útil tenir un model de desplegament per explicar com funciona el sistema en conjunt. A continuació es pot observar un model de desplegament més o menys informal per a tal propòsit (vegeu la Figura 35). El model ha estat dividit en les tres capes lògiques clàssiques descrites al catàleg *Patterns of Enterprise Application Architecture* [FowlerEAA], tot i que s'han usat noms diferents.

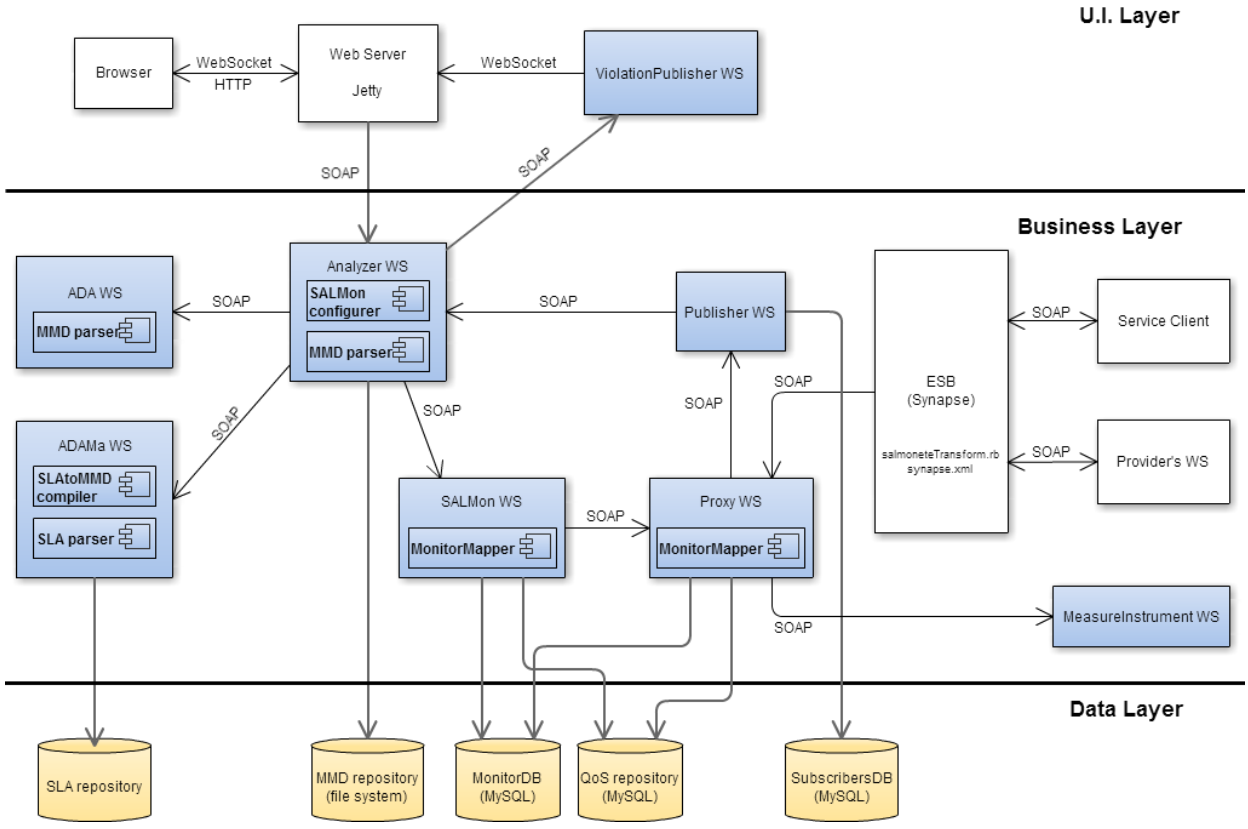

Figura 35: Model de desplegament de SALMonADA

[1] Vegeu el capítol 7.1.1. Arquitectura dirigida pels esdeveniments (página 65).

A la implementació, la BD de QoS (*QoS repository*) i la BD de informació de monitorat (*MonitorDB*) són la mateixa. Cal observar que alguns *instruments de mesura* són serveis web externs que es poden afegir al sistema com a extensions en temps d'execució.

Molts d'aquests components es poden replicar, cosa que fa el sistema molt escalable, però aquesta escalabilitat respon a certes restriccions. Quines són aquestes restriccions?

7.1.5. Escalabilitat del sistema

Les restriccions d'escalabilitat són les següents:

- Un *Monitor* pot tenir diverses *bases de dades de QoS*, però totes les dades d'un servei han d'estar a la mateixa *BD*.
- Un *Proxy* pot tenir diverses *bases de dades de QoS*, però totes les dades d'un servei aniran a la mateixa *BD*.
- Una instància de l'*ESB Synapse* pot informar dels esdeveniments detectats a diversos *Proxys*, però totes les crides al mateix servei interceptades han d'anar al mateix *Proxy*.
- Un *Publisher* només pot tenir una *base de dades de QoS*.
- Per a poder monitorar un servei, la informació de monitorat d'aquest servei ha d'estar en una *BD* accedida per un *Monitor*, un *Proxy* i un *Publisher*.
- Per a poder monitorar un servei, les seves dades de QoS han d'estar en una *BD* accedida per els mateixos *Proxy* i *Publisher* que poden accedir a la seva informació de monitorat.

Aquestes restriccions es deuen a que tenir les dades de QoS d'un mateix servei distribuïdes en diverses BDs complica el procés de consulta de dades a l'hora de comprovar violacions. És el cas típic de les BDs distribuïdes, i preparar aquest sistema a tal fi no forma part dels requisits.

El coll d'ampolla del sistema és per tant la *BD de QoS*. El sistema pot ser escalat degudament tal que (vegeu la Figura 36):

- Si hi ha un alt nombre de crides al mateix servei, es poden replicar les *sondes interceptores* i dirigir uns clients a una i uns a una altra.
- Si a més el *Proxy* processa massa poc a poc, o hi ha un alt nombre de mètriques, es poden replicar els *Proxys* per al mateix servei. Tots ells usen la mateixa *BD*.
- Si hi ha massa clients i el *Publisher* va massa lent, es poden replicar els *Publishers* per al mateix servei. Tots ells usen la mateixa *BD*.
- Si hi ha massa canvis en la configuració del *Monitor* (cas atípic), es poden replicar els *Monitors*.
- Si hi ha massa serveis, es pot replicar el sistema SALMon (format per *Monitor*, *sondes*, *Proxy*, *Publisher*, i les seves *BDs*) sencer.
- Si hi ha massa crides i l'*Analyzer* va massa lent comprovant si hi ha violacions, es pot replicar el sistema *ADA*, però totes les notificacions que afecten al mateix SLA han de ser notificades al sistema *ADA* que gestiona aquell SLA. A més, l'administrador pot decidir particionar un SLA a nivell de condicions independents.

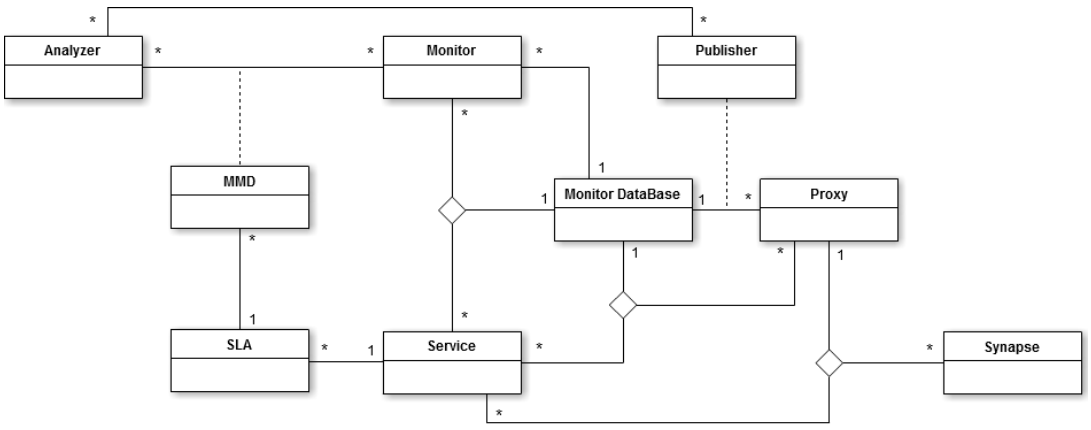

Figura 36: Model d'escalabilitat de SALMon.

7.2. Disseny dels components

Molts dels components del sistema són serveis web, i molts d'ells accedeixen a bases de dades relacionals, de manera que comparteixen característiques de disseny. Aquestes característiques són:

- **Arquitectura de dues capes:** Tots els serveis web del sistema tenen una arquitectura interna de dues capes; domini i dades. La capa de presentació és implementada pel middleware Apache Axis2, que proporciona una interfície SOAP.
- **Façana remota:** Tots els serveis web del sistema tenen una classe *Controller* que implementa el patró façana remota com a punt d'entrada al servei i controlador de la capa de domini.
- **Classe resultat:** Tots els serveis web del sistema fan servir una classe estàndard per retornar resultats de les operacions del servei. Aquesta és la classe *Result*, que pot contenir l'identificador de l'objecte modificat per la crida, i conté informació sobre el resultat de l'operació (èxit, avís o error) amb un missatge informatiu.
- **Table Data Gateway:** Totes les capes de dades del sistema implementen el patró Table Data Gateway. Tots els Gateways hereden de la classe *Mapper*[1].
- **Fitxer de configuració de desplegament:** Tots els paràmetres d'execució dels serveis web (endpoints de serveis i credencials i direccions de bases de dades, entre d'altres) es llegeixen d'un fitxer Java .properties i es refresquen en temps d'execució, sense que faci falta reiniciar els serveis.
- **Logs:** Per complir amb el requisit 2.5, tots els serveis web del sistema disposen d'un sistema de registre d'incidències concurrent mitjançant la classe *Logger*.

La Figura 37 descriu el model de classes d'aquestes característiques.

[1] El nom indueix a pensar que es tracta del patró Data Mapper, però no és així: Es conserva per motius històrics.

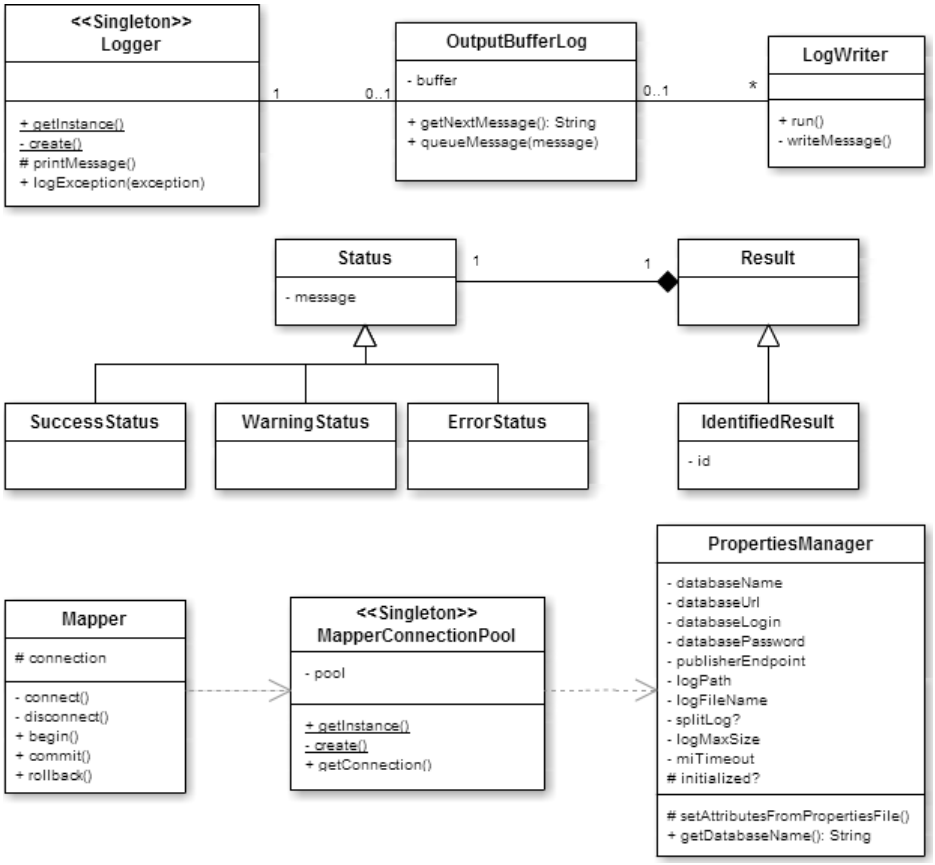

Figura 37: Diagrama de classes de les característiques ubiqües al sistema.

El comportament d'aquestes classes és descrit en els següents diagrames de seqüència (vegeu la Figura 38, la Figura 39 i la Figura 40), que il·lustren el comportament de les classes *Mapper* i *Logger*, mentre que la classe *Result* és el tipus retornat en totes les crides a serveis web.

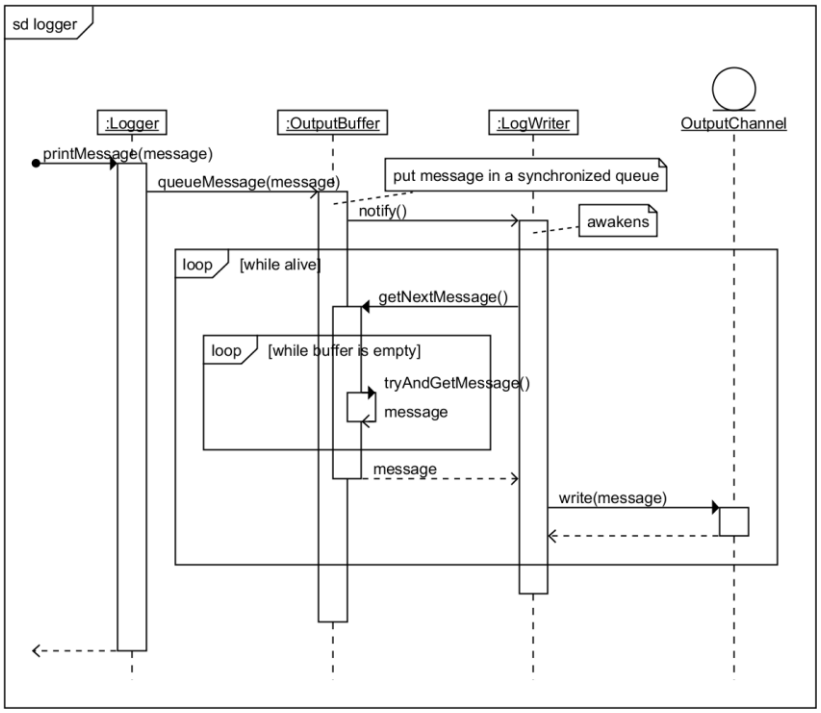

Figura 38: Diagrama de seqüència del sistema de logs, sense mostrar la concurrència.

Cal notar que el diagrama del sistema de logs (vegeu la Figura 38) no és una representació exacta del procés concurrent, sinó que només il·lustra l'ordre d'interacció de les classes quan es produeix una incidència.

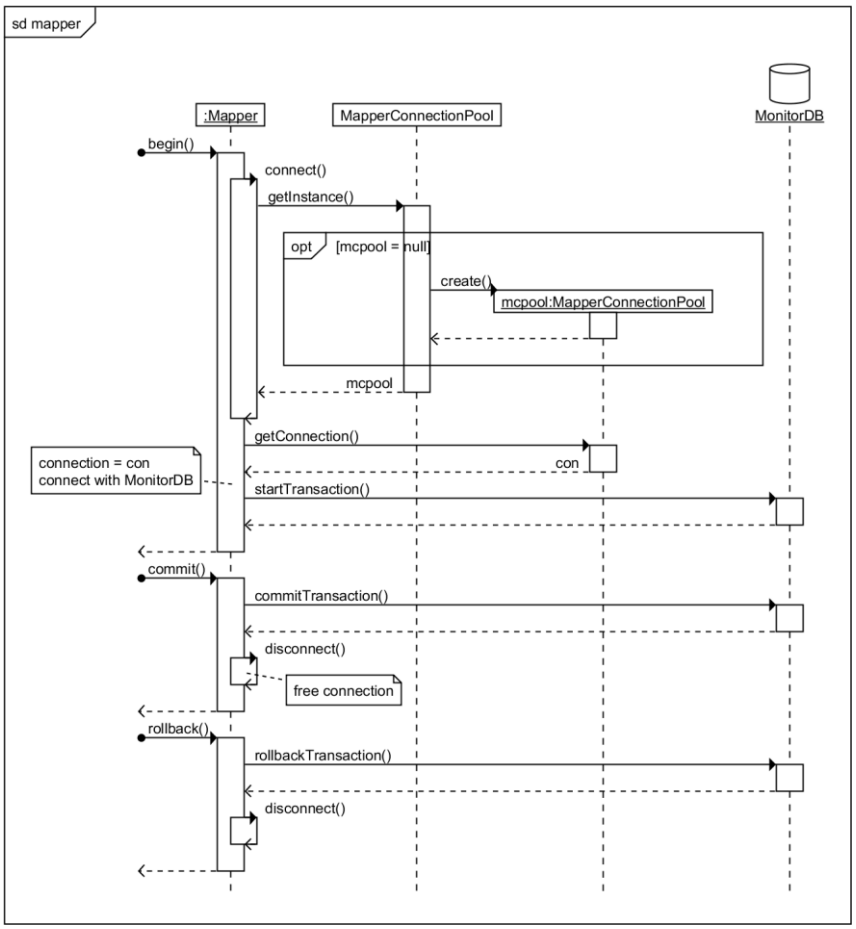

Figura 39: Diagrama de seqüència de la classe Mapper.

El diagrama de seqüència del *Mapper* (vegeu la Figura 39 i la Figura 40) mostra el funcionament del patró *Unit of Work* [FowlerEAA] que s'ha implementat per encapsular transaccions SQL. En acabar totes les operacions de modificació a la base de dades, en cas d'error, s'efectua una crida a l'operació *rollback()* per revertir els canvis indesitjats. No obstant, en els diagrames de seqüència de tots els components només es descriu el curs típic d'esdeveniments, deixant el tractament d'excepcions com a responsabilitat del codi del projecte.

Per gestionar la concurrència en l'accés a les bases de dades i garantir així el compliment del requisit 3.2 s'ha decidit usar un *pool*[1] de connexions a les bases de dades. El funcionament d'un pool de connexions es basa en centralitzar la gestió de les connexions en una classe preparada per a l'accés concurrent. Cada nova transacció demana una connexió lliure al pool, en fa ús, i finalment l'allibera per què una altra la pugui usar. El pool s'encarrega de mantenir les connexions obertes o tancar-les per alliberar recursos de forma transparent, i assegura que dues transaccions no s'interferiran de cap manera.

[1] Vegeu el capítol 3.3. Eines, plataformes i estàndards (pàgina 23).

Després d'una exhaustiva comparació entre les dues biblioteques Java més populars de pools de connexions, C3P0 i DBCP, es va decidir usar el paquet DBCP (Database Connection Pools) d'Apache Commons perquè l'ajuda de la comunitat d'usuaris és més abundant, mentre que les dues llibreries aparentment han deixat de ser mantingudes i ambdues ofereixen les capacitats requerides pel projecte.

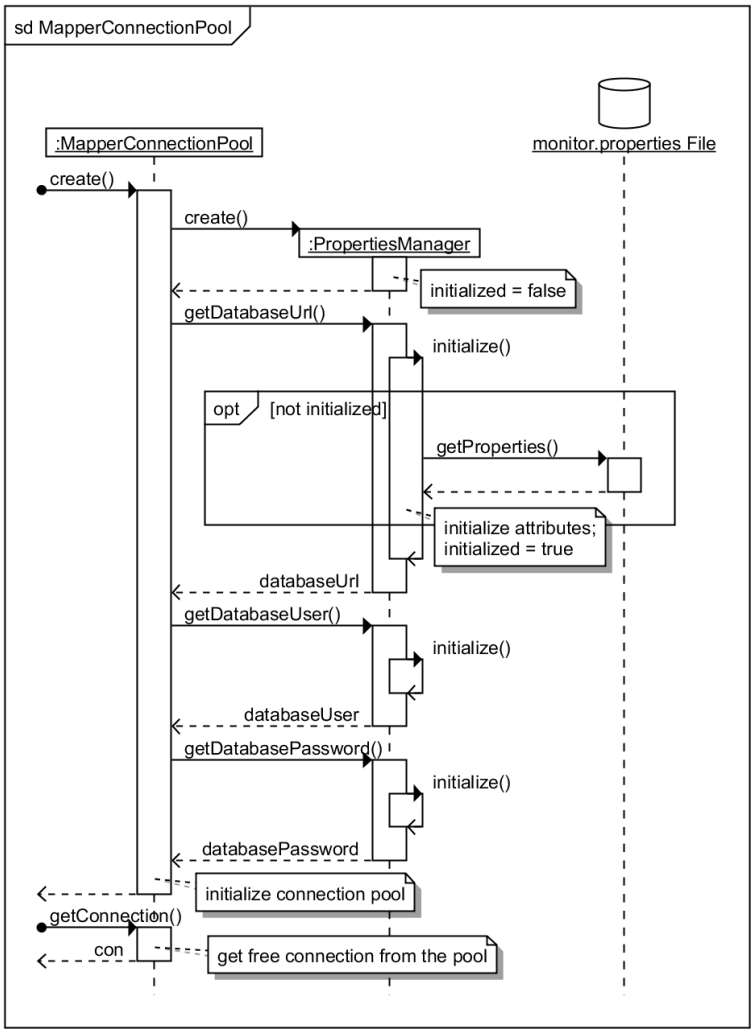

Figura 40: Diagrama de seqüència del MapperConnectionPool.

7.2.1. Disseny del *Monitor*

La funció del *Monitor* és la de configurar el sistema amb les dades d'un document MMD per tal que quan el *Proxy* detecti un esdeveniment corresponent a aquell MMD, sàpiga que l'ha de monitorar i tingui les dades sobre les mètriques que cal calcular i el context complet de l'esdeveniment. Per això, *Monitor* i *Proxy* comparteixen base de dades.

Aquesta configuració es fa en passos. El *Monitor* no accepta documents MMD talment, ja que el format MMD va ser enginyat després que el *Monitor*, que és anterior a aquest projecte. Accepta, però, un seguit d'operacions que permeten el mateix (vegeu el model a la Figura 41):

- **SetClient** afegeix al sistema un nou client a qui notificar els valors de QoS monitorats. Això permet a l'*Analyzer* registrar-se a sí mateix com a client en el sistema i poder comprovar en temps real la satisfacció dels contractes SLA.

- **SetBasicMetric** i **SetDerivedMetric** permeten afegir noves mètriques personalitzades i noves mètriques derivades al sistema.
- **SetService** afegeix un nou context de tipus servei web, amb la seva direcció com a identificador i informació sobre el servei.
- **SetOperation** afegeix un nou context de tipus operació de servei web. Aquesta operació pertany a un sol servei web, i té un identificador únic.
- **MonitorMetricForOperation** estableix que una mètrica ha de ser monitorada per a una operació concreta prèviament introduïdes al sistema.

Executades en aquest ordre i repetida cadascuna els cops necessaris, les operacions permeten configurar el sistema per monitorar un MMD sense cap restricció. El component *Analyzer* té, doncs, un component intern que tradueix un MMD a crides al *Monitor*.

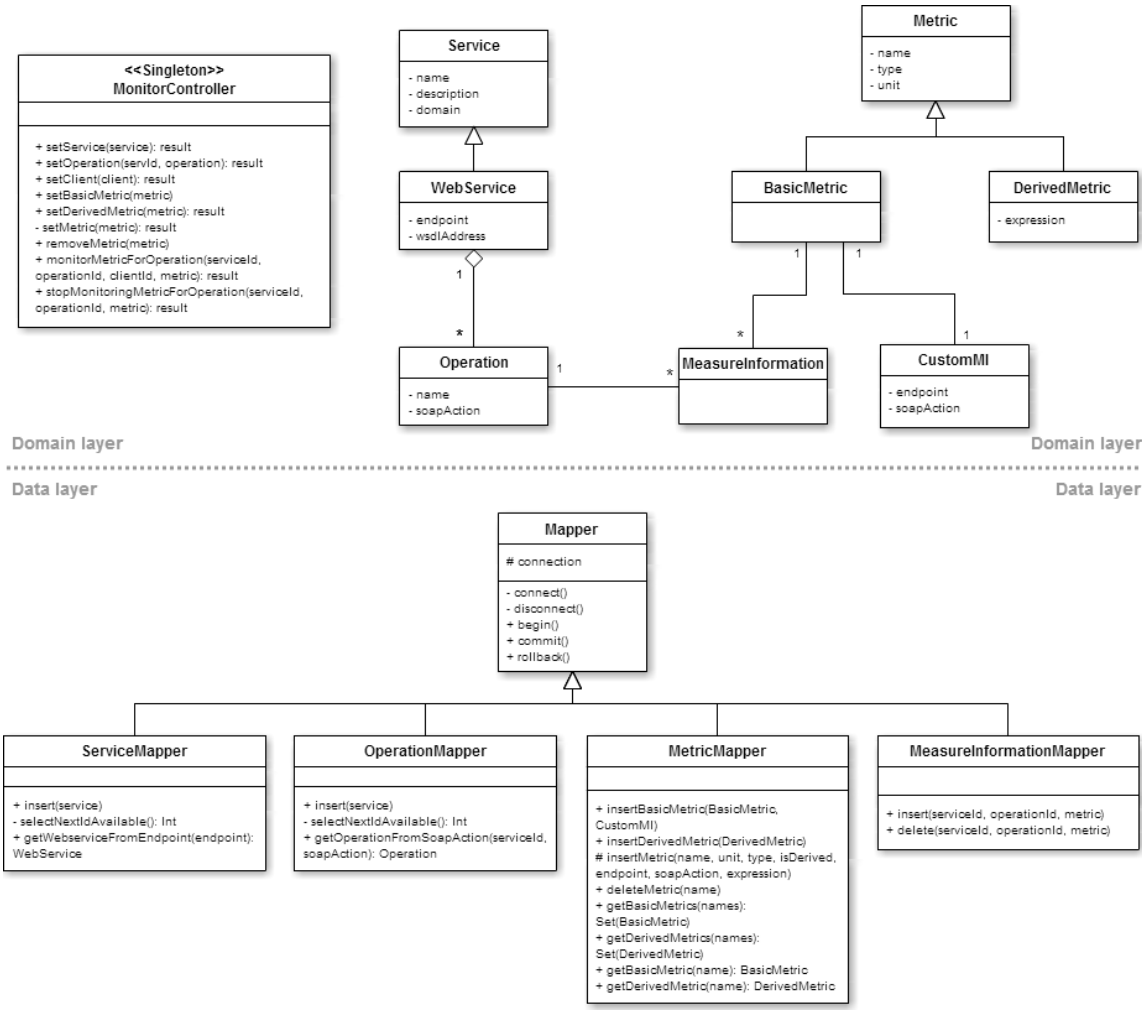

Figura 41: Diagrama de classes del Monitor, sense el compilador d'expressions.

En aquesta memòria només es mostra en detall el funcionament de la part implementada per a aquest projecte, en particular la operació SetClient, que és nova, i les operacions SetBasicMetric, SetDerivedMetric i MonitorMetricForOperation que, si bé ja existien anteriorment, s'han hagut de modificar per adaptar-se al nou model, on una mètrica és un concepte més extens.

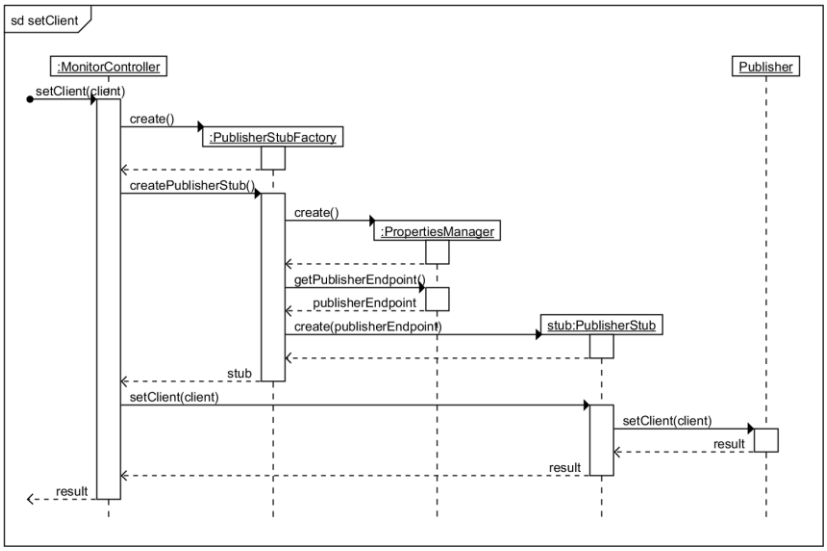

Figura 42: Diagrama de seqüència de la operació *setClient*.

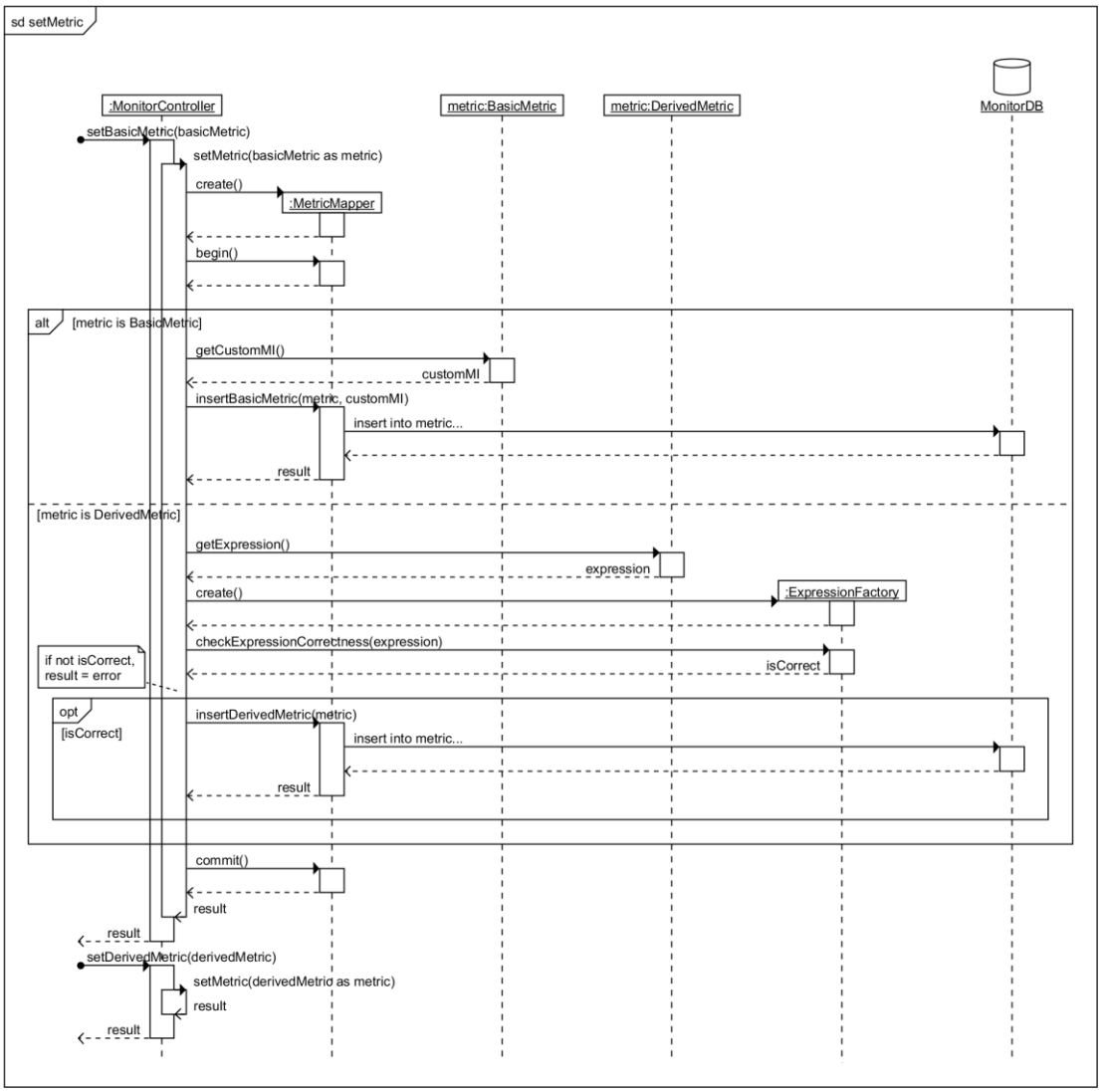

Figura 43: Diagrama de seqüència de les operacions *setBasicMetric* i *setDerivedMetric*.

La operació *setClient* del *Monitor* serveix com a interfície per al servei *Publisher*, el qual no té perquè estar accessible al públic, mentre que el *Monitor* sí. L'únic que fa el *Monitor*, doncs,

és enviar la petició al *Publisher* que hi ha configurat, o mostrar error si no n'hi ha cap (vegeu la Figura 42).

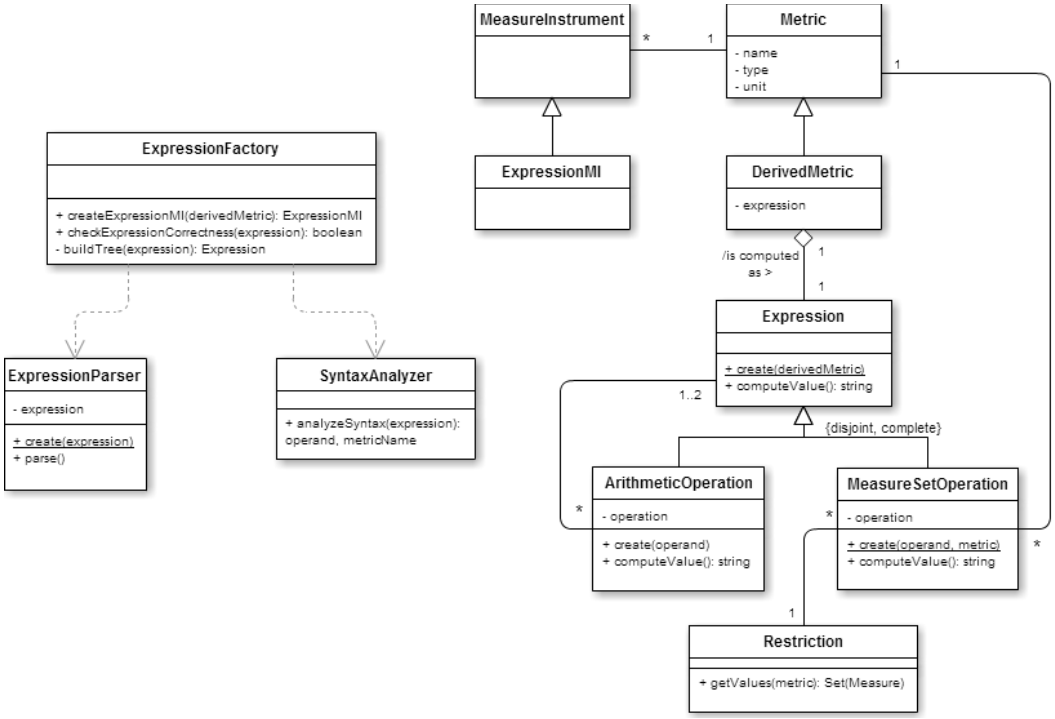

Figura 44: Diagrama de classes del compilador d'expressions de mètriques derivades, tant per al *Monitor* com per al *Proxy*.

El requisit 1.4 ofereix tres possibilitats d'afegir mètriques al sistema: Indicant la direcció del programa que la calcula, descrivint l'expressió matemàtica i descrivint l'algorisme. Com que un programa extern pot implementar qualsevol algorisme que descrigui com mesurar la mètrica, i fer que SALMon pugui compilar algorismes requereix un gran esforç que se'n surt de l'abast del projecte, s'ha optat per dissenyar la primera i la segona funcionalitats, que corresponen a afegir mètriques bàsiques i mètriques derivades, perquè així s'han definit[1].

La operació d'afegir noves mètriques al sistema (vegeu la Figura 43) funciona diferent segons s'afegeix una mètrica bàsica o una derivada. En cas d'una mètrica bàsica, s'afegeixen les dades de comunicació del servei web que fa d'*Instrument de Mesura*, però no es comprova el seu funcionament. Aquest Instrument de Mesura ha d'implementar l'algorisme de càlcul de la mètrica, i ha d'estar desplegat i accessible un cop la mètrica es comença a monitorar. En cas d'una mètrica derivada, sí que es comprova que l'expressió matemàtica de la mètrica derivada sigui correcta i reconeguda pel sistema, i en cas de no ser-ho, es llença un error.

Per comprovar la correctesa de l'expressió matemàtica es fa servir la mateixa classe que calcula els resultats de les expressions, la classe *ExpressionFactory*. Aquesta i les classes que usa apareixen replicades als components *Monitor* i *Proxy*. Referides en conjunt, són el compilador d'expressions de mètriques derivades (vegeu el model a la Figura 44).

[1] Vegeu el capítol 6.3.2. Model del *Monitor* (pàgina 60).

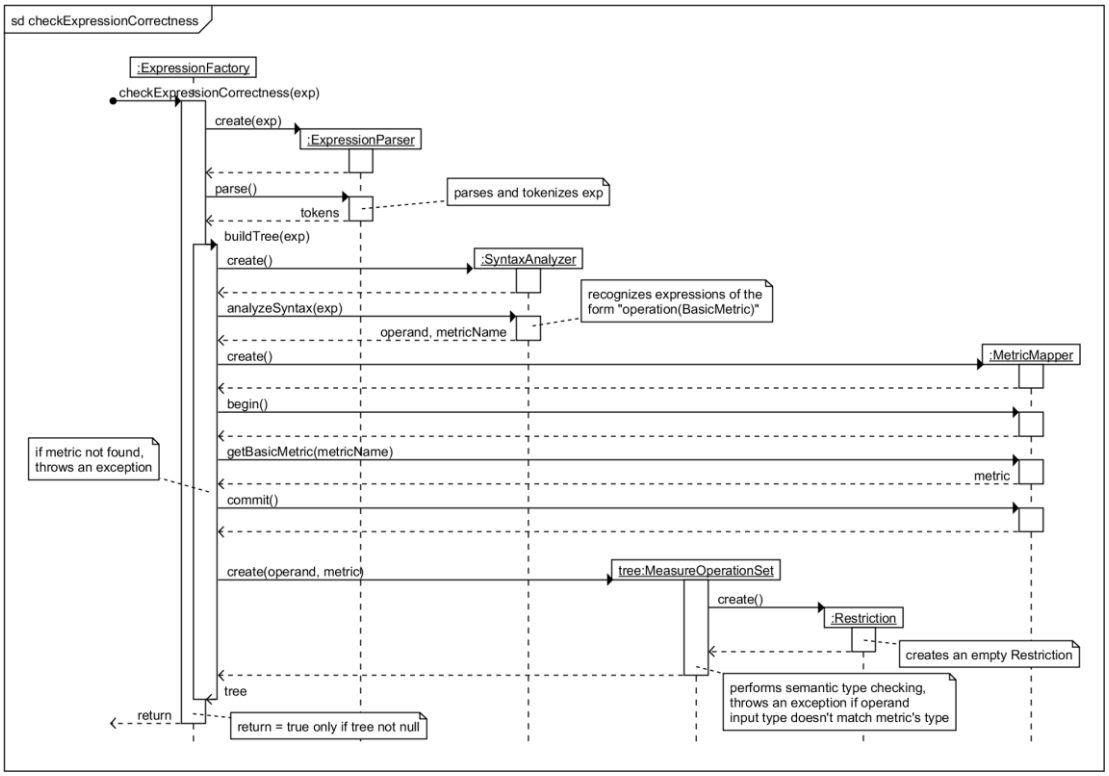

Figura 45: Diagrama de seqüència de la comprovació de correctesa de l'expressió.

S'ha dissenyat SALMon per a tenir un compilador d'expressions matemàtiques simple però completament extensible que pugui calcular les mètriques que demana el requisit 2.8. Aquest compilador reconeix una gramàtica molt petita, de la forma:

$$S \leftarrow O(M)$$
$$O \leftarrow min|max|avg|uptimepc|downtimepc|avgdowntime$$
$$M \leftarrow m, \forall\, m\ \in BasicMetric$$

O és un àtom que pot ser una expressió sobre valors en un conjunt (en particular, "màxim històric", "mínim històric" i "mitjana absoluta" sobre conjunts de valors numèrics, i "percentatge de temps cert", "percentatge de temps fals" i "temps mig de recuperació a cert" sobre conjunts de valors booleans). M és un àtom que pot ser el nom d'una mètrica bàsica prèviament introduïda al sistema i, com s'aprecia, ha d'estar entre parèntesis.

$$G \leftarrow select\ A\ from\ T\ R$$
$$G' \leftarrow select\ A\ from\ T'\ R$$
$$\ldots$$
$$T \leftarrow M, T|\lambda$$
$$T' \leftarrow M', T'|\lambda$$
$$M' \leftarrow m, \forall\, m\ \in Metric$$

Fer que M hagi de ser una mètrica bàsica ha estat una decisió de disseny, però res impedeix usar una M' que reconegui qualsevol mètrica. Cal notar que, amb una gramàtica amb total expressivitat[1], G i G' reconeixen el mateix llenguatge, ja que una mètrica derivada d'una altra

[1] Una gramàtica que es pugui reduir a la gramàtica de SQL, és a dir, sigui igual de poderosa que SQL.

mètrica derivada es pot expressar com a mètrica derivada de les operacions sobre mètriques bàsiques. No obstant, aquesta modificació afegiria comoditat d'ús al sistema, ja que permet assignar un alies a una sub-expressió.

Finalment, la operació *MonitorMetricForOperation* (vegeu la Figura 46) estableix quines mètriques han de ser monitorades per a quines operacions, i prepara el sistema per notificar les mesures d'aquestes mitjançant el *Publisher*. Aquesta relació es pot cancel·lar mitjançant l'operació complementària *StopMonitoringMetricForOperation*. Aquesta fa que SALMon deixi de monitorar la mètrica per a aquesta operació, però no cancel·la la subscripció al *Publisher*, ja que si no es monitora, no hi haurà cap mesura nova que notificar.

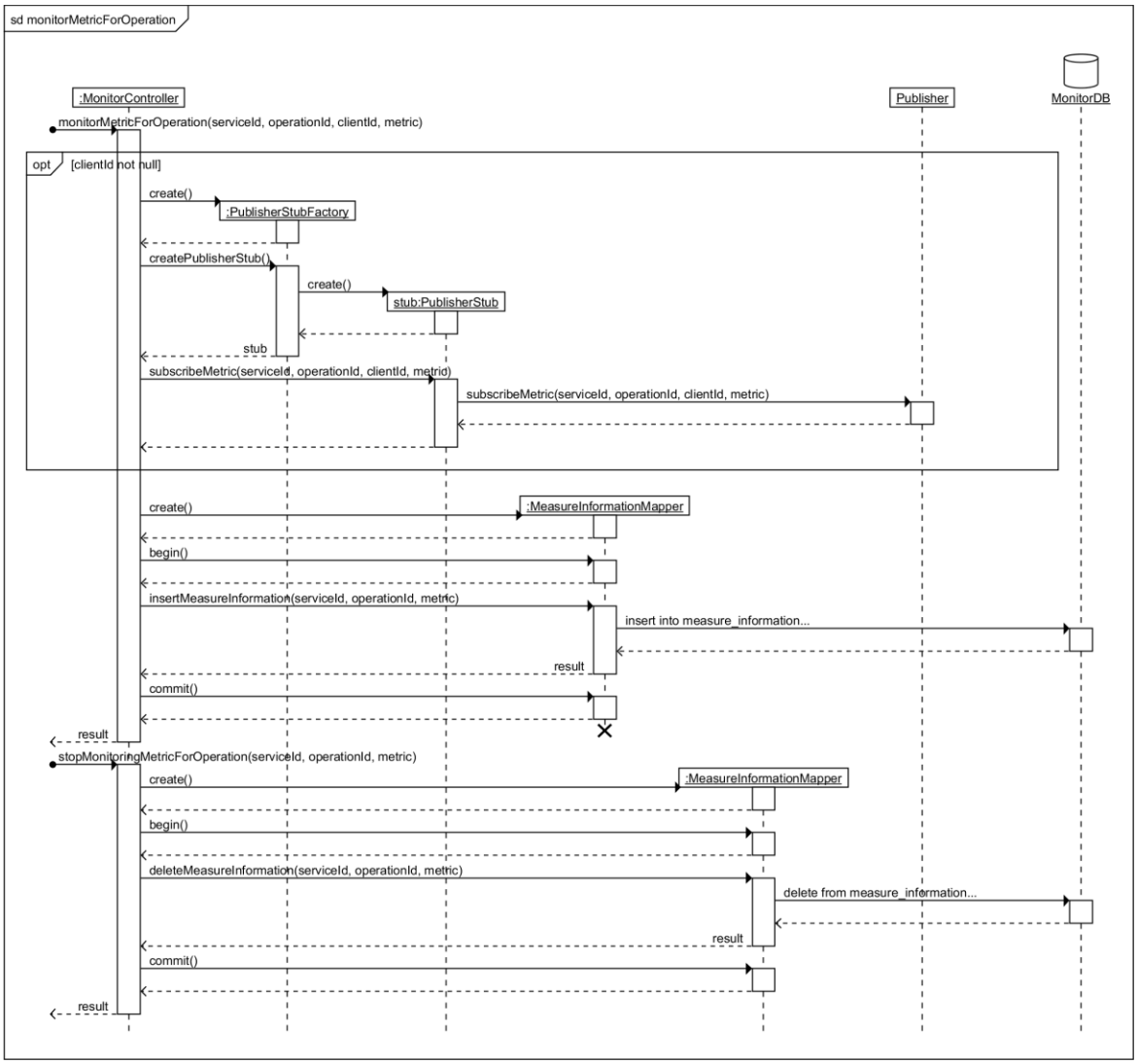

Figura 46: Diagrama de seqüència de la operació *monitorMetricForOperation*.

Per altra banda, la base de dades del *Monitor* (vegeu la Figura 47) desa les dades de configuració i les mesures realitzades de cada mètrica. El model de dades es desprèn del model conceptual amb certs canvis:

- La classe *ContextType* desapareix, substituïda per unes poques instàncies seves materialitzades en les classes *Service*, *Webservice*, *Operation*, i *Input*. Aquestes classes són necessàries i suficients per monitorar serveis web SOAP.

- L'herència de les classes *BasicMetric* i *DerivedMetric* es transforma mitjançant el patró *Single Table Inheritance* [FowlerEAA] en una sola taula. El booleà *is_derived* basta per indicar el tipus, doncs és una herència de dues classes completa i disjunta.
- Tot i que el model conceptual permet múltiples *MeasureInstruments* per a una mateixa mètrica, aquest disseny considera un sol *MeasureInstrument* per mètrica suficient. Aquesta relació 1-1 es transforma en atributs a la taula *Metric* que han de ser nuls en cas de no ser una mètrica bàsica.
- La classe *EventType* desapareix; la base de dades només contempla un sol tipus d'esdeveniment: SOAPCall. Es materialitza en la classe *OperationInvocation*. Es manté la classe *Event* per a major extensibilitat.
- La taula de la classe *ComputableMetricForEventType* s'anomena *MeasureInformation*.

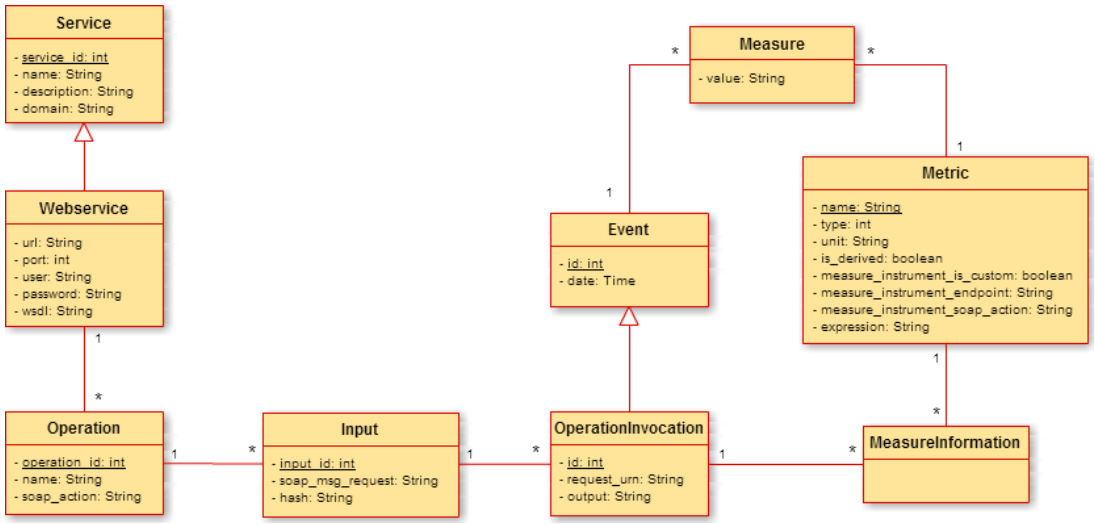

Figura 47: Model de dades del *Monitor*.

7.2.2. Disseny del *Proxy*

La funció del *Proxy* és la de ser notificat de l'ocurrència d'esdeveniments i cridar als Instruments de Mesura per què els monitorin. Això es fa mitjançant una única operació: *NotifyEvent*. Aquesta operació rep les dades d'una sonda sobre un esdeveniment i les posa a una cua per ser monitorades.

El monitoratge de serveis web SOAP es pot classificar en dos tipus d'esdeveniments: SOAPRequest i SOAPResponse, corresponents a l'enviament d'un missatge de petició al servei i l'enviament de la resposta al client. Ambdós comparteixen els atributs servei, operació i missatge, ja que són missatges SOAP. Per això, s'han fet heretar d'un supertipus abstracte anomenat SOAPCall. A més, el SOAPResponse té un atribut per indicar si hi ha hagut error en la resposta.

Aquests tipus d'esdeveniments (instàncies de la classe *EventType*) es poden veure en el model d'objectes de la Figura 49. Cal notar que, tot i ser instàncies d'una classe, aquesta classe té un nivell d'abstracció superior, i per tant aquests objectes esdevindran classes en el diagrama final (vegeu la Figura 48).

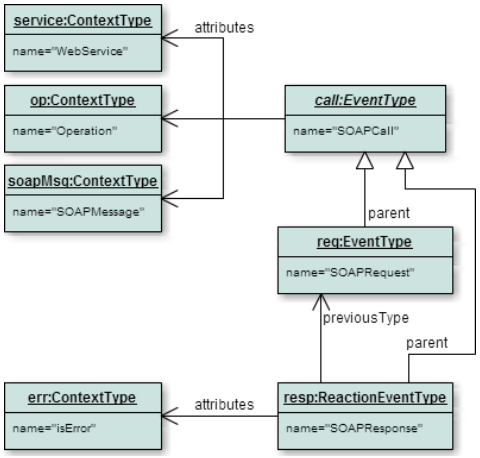

Figura 49: Model d'objectes de la classe *EventType*.

Per a crear els esdeveniments del tipus adequat s'usa un patró *Factory Method* [GoF] combinat amb una classe que escull la *Factory* adequada al tipus. Aquesta classe és l'única que està acoblada a tots els tipus d'esdeveniment, reduint així l'acoblament al mínim.

El diagrama, a més, mostra nous subtipus de la classe *MeasureInstrument*, nominalment *ResponseTimeMeasureInstrument* i *AvailabilityMeasureInstrument*.

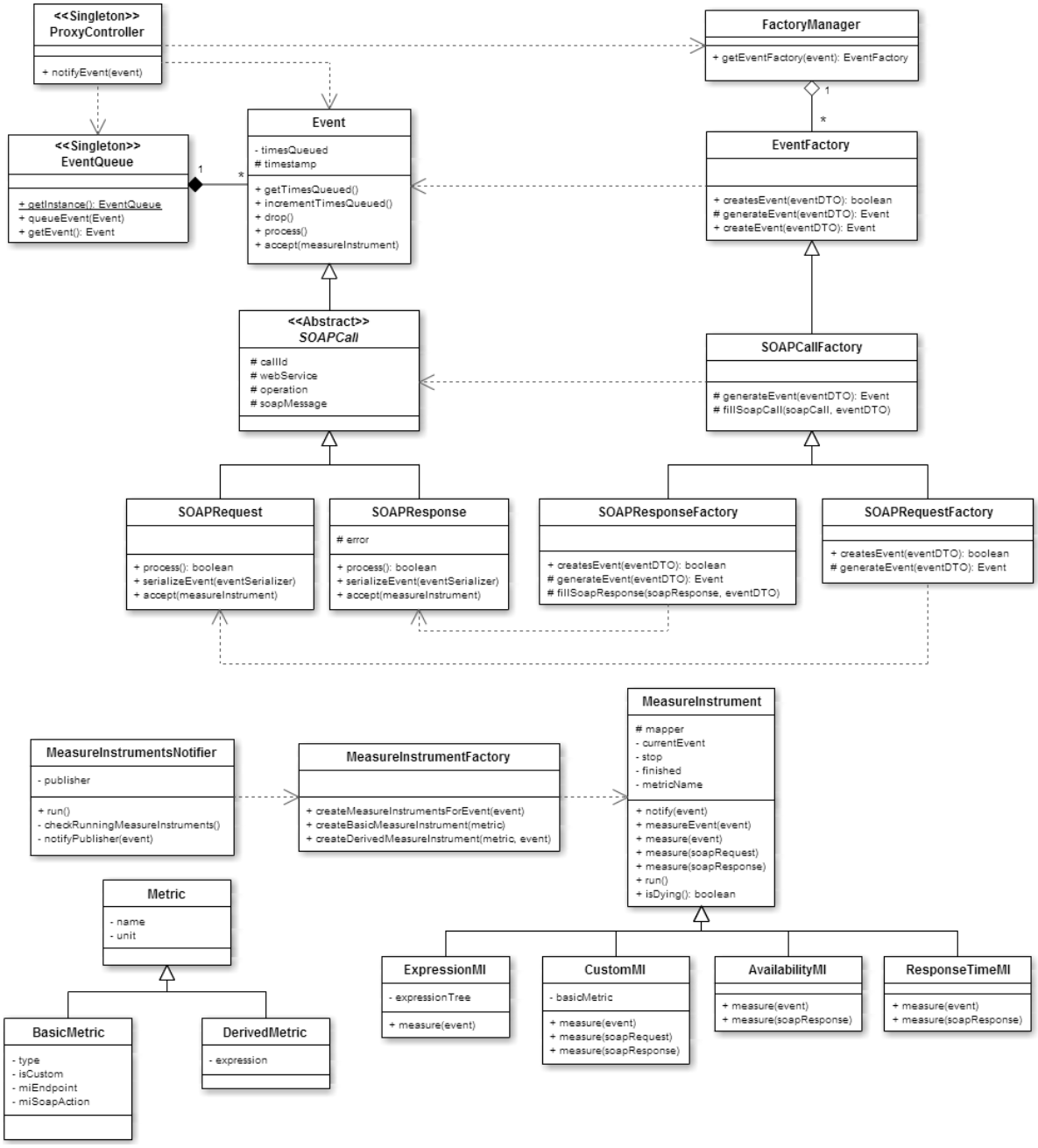

Figura 48: Diagrama de classes de domini del Proxy, sense el compilador d'expressions.

Aquests subtipus són les mètriques bàsiques predeterminades. Degut a que el requisit 2.8 demana aquestes mètriques explícitament, i són mètriques relativament usuals de monitorar, resulta més adient que formin part del sistema en comptes de ser serveis web auxiliars, reduint així el temps de càlcul.

El *Proxy* comparteix gran part de la capa de dades amb el *Monitor*. La part addicional es pot observar a la Figura 50.

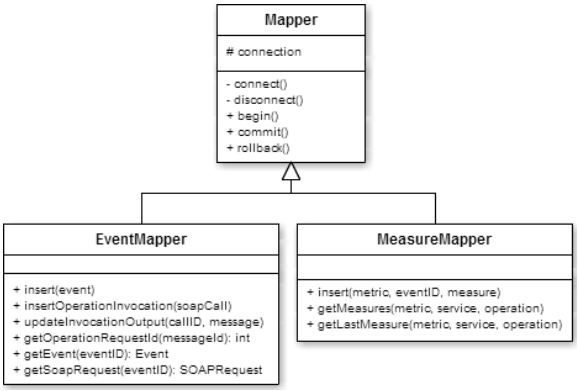

Figura 50: Diagrama de classes de dades del Proxy.

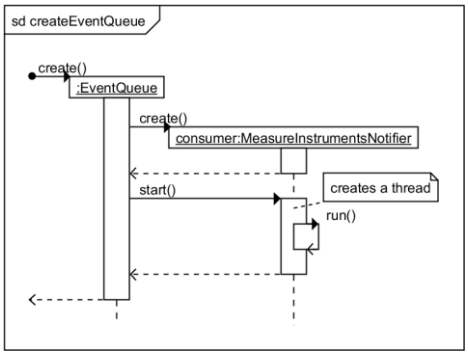

Figura 51: Diagrama de seqüència de la construcció de la cua d'esdeveniments.

Quan s'invoca la operació *notifyEvent* (vegeu la Figura 53), s'usa el patró *Factory* per a transformar l'objecte *eventDTO* en un objecte *Event* del tipus adequat. La classe *FactoryManager* és la que selecciona la classe Factory adequada al tipus. Els objectes *Event* són després afegits a la cua. La operació pot ser cridada simultàniament per múltiples sondes, així que la cua funciona amb un patró productor-consumidor amb múltiples productors (les sondes) i un sol consumidor (el motor de processament d'esdeveniments).

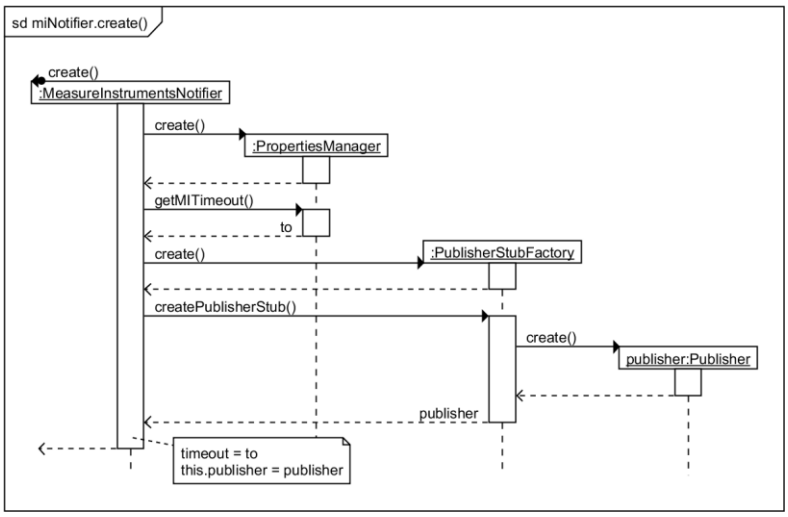

Figura 52: Diagrama de seqüència de la construcció de la classe *MeasureInstrumentsNotifier*.

El motor d'esdeveniments pròpiament dit és implementat per la classe *MeasureInstrumentsNotifier*, que recull esdeveniments de la cua. Es poden veure els detalls de la inicialització de la cua i el motor a les figures Figura 51 i Figura 52. Quan recull un esdeveniment (vegeu la Figura 55) el processa, afegint la informació a la base de dades (vegeu la Figura 54).

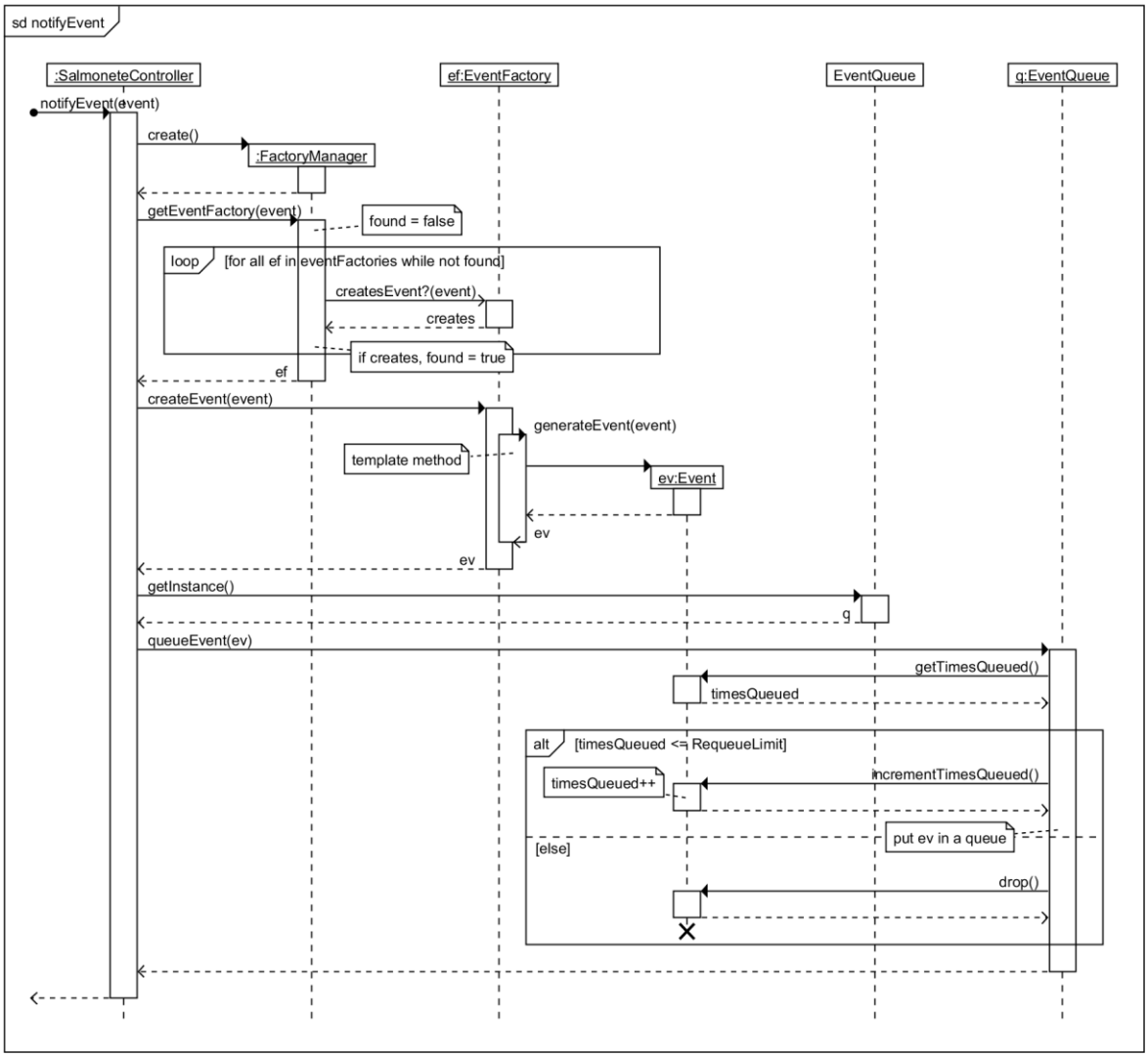

Figura 53: Diagrama de seqüència de la operació *notifyEvent*.

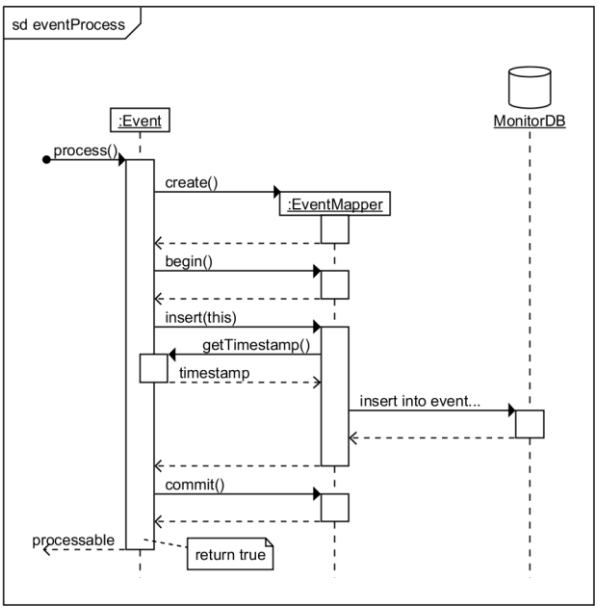

Figura 54: Diagrama de seqüència del processament previ dels esdeveniments.

Si l'esdeveniment no està llest per ser processat encara (per exemple, perquè és de tipus *SOAPResponse* i depèn d'un esdeveniment previ que encara no ha estat monitorat[1]), aleshores es torna a afegir al principi de la cua, tenint en compte que en condicions normals de funcionament és molt possible que el seu esdeveniment previ hagi arribat poc després. Per assegurar que això no omple la cua d'esdeveniments que no seran processats mai, s'afegeix una data de caducitat a l'esdeveniment, un nombre màxim de vegades que es pot tornar a afegir a la cua. Després d'aquest nombre de vegades, l'esdeveniment serà retirat de la cua i descartat (vegeu la operació *queueEvent* a la Figura 53).

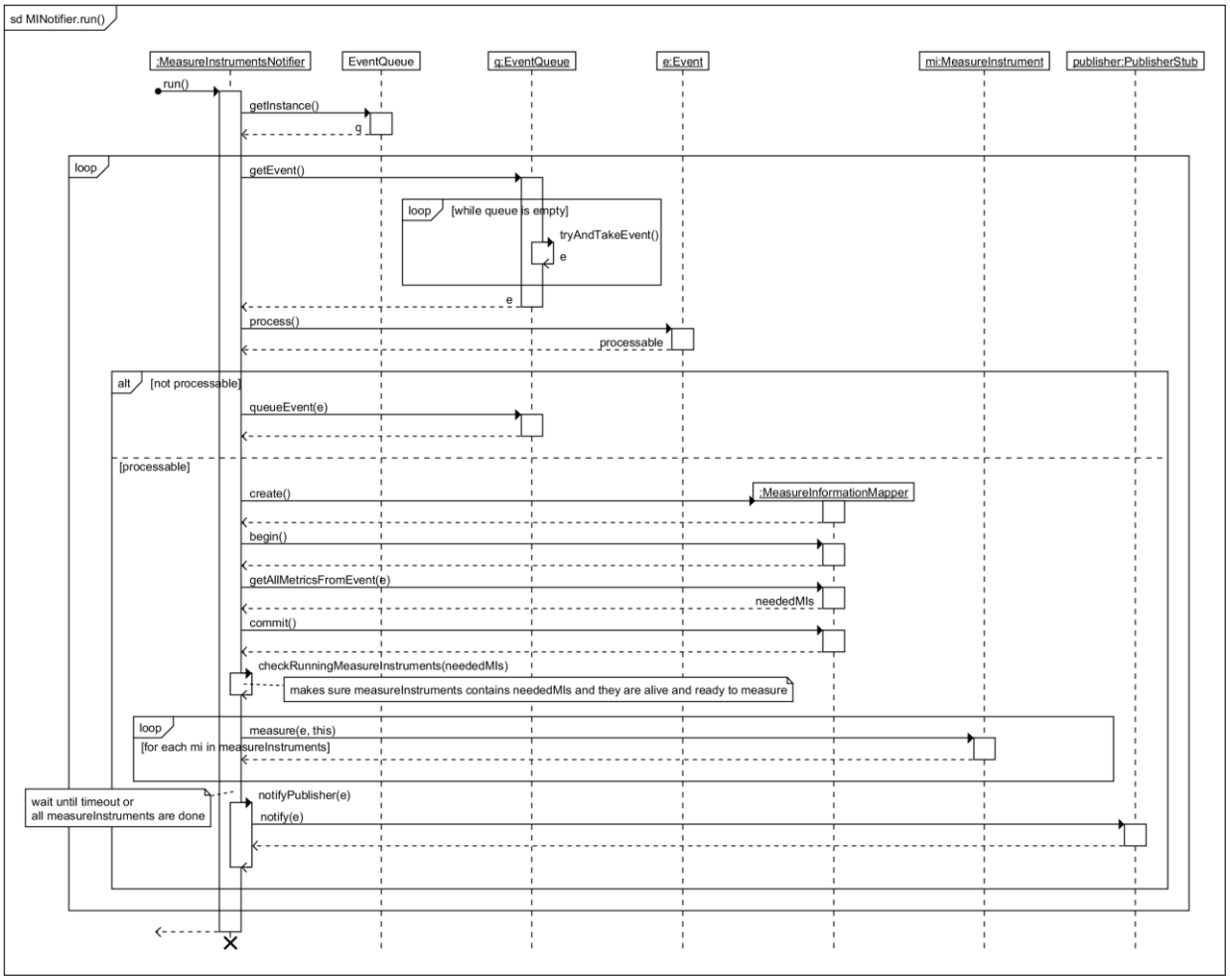

Figura 55: Diagrama de seqüència del motor d'esdeveniments implementat a *MeasureInstrumentsNotifier*.

Una vegada l'esdeveniment ha estat processat, el motor comprova la base de dades per saber quines mètriques n'ha de monitorar. La classe *MeasureInstrumentsNotifier* manté referències a tots els *MeasureInstruments* actius. Cada *MeasureInstrument* té un fil d'execució propi que permet que els càlculs de mesura es facin en paral·lel. El *MeasureInstrumentsNotifier* comprova que tots els fils dels instruments requerits estan vius, torna a crear els que no ho estan i crea els que encara no han estat creats (vegeu la Figura 56).

[1] Això pot ser degut a retards en les sondes i la comunicació, i no es pot garantir que aquest ordre es respecti. Respectar-lo és responsabilitat del motor d'esdeveniments.

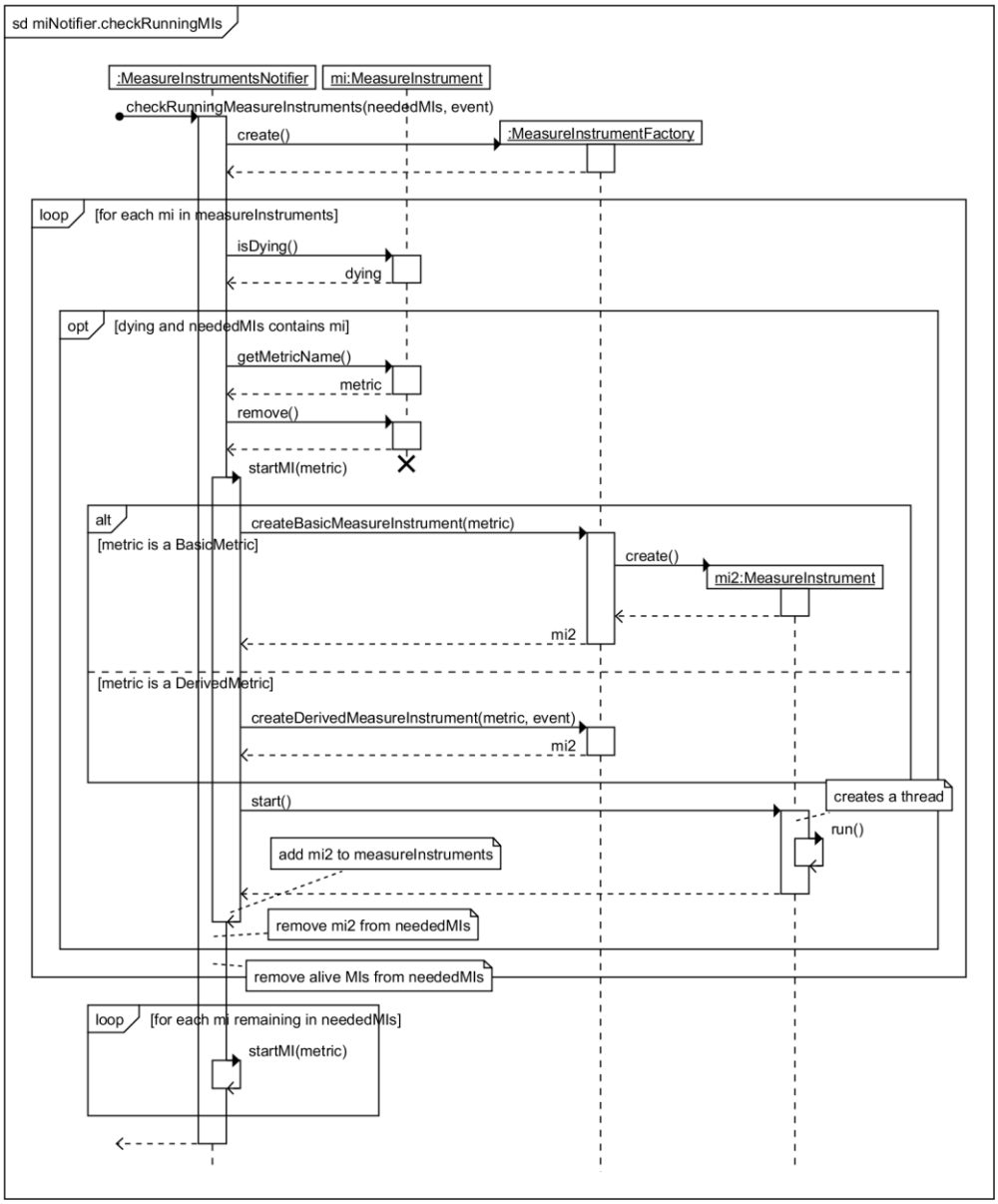

Figura 56: Diagrama de seqüència de l'operació *checkRunningMeasureInstruments* del *MeasureInstrumentsNotifier*.

Tan bon punt tots els *MeasureInstruments* requerits estan preparats, el *MeasureInstrumentsNotifier* sincronitza amb ells un semàfor per detectar quan han acabat de processar, i a continuació els envia simultàniament l'esdeveniment a monitorar. Un esdeveniment no es pot monitorar fins que els seus predecessors han estat monitorats, ja que les mètriques podrien dependre de valors anteriors, així que el *MeasureInstrumentsNotifier* s'espera a que tots els *MeasureInstruments* acabin per seguir amb el consum. En cas que un *MeasureInstrument* excedeixi el temps d'espera màxim ("timeout" en anglès), es cancel·la la mesura d'aquella mètrica i el seu fil mor per ser reiniciat la pròxima vegada que es necessiti.

D'aquesta manera, un timeout massa curt pot fer que el sistema no monitori mai mètriques de càlcul costós. Si això passa, s'ha d'augmentar el timeout per permetre a aquestes mètriques monitorar. Aquest timeout es pot modificar, però, mitjançant el fitxer de configuració *.properties*.

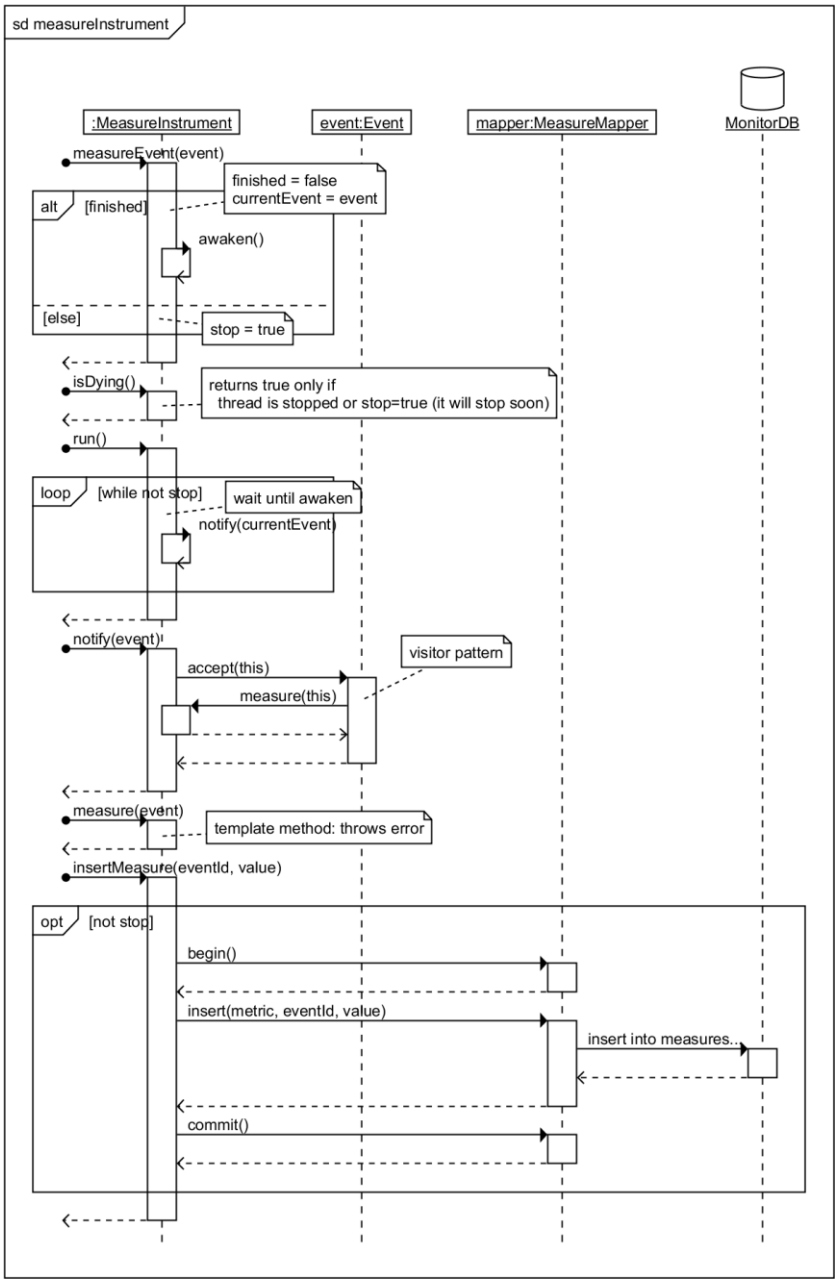

Figura 57: Diagrama de seqüència del *MeasureInstrument*.

Cada *MeasureInstrument* té un fil d'execució propi, i fan servir un protocol pel qual se sincronitzen amb el *MeasureInstrumentsNotifier* (vegeu la Figura 57). Un *MeasureInstrument* és una màquina d'estats amb 3 estats: A l'espera d'un nou esdeveniment, mesurant un esdeveniment, i morint. Quan el motor intenta assignar una nova tasca a un *MeasureInstrument* que encara està treballant en l'anterior, és perquè aquest ha excedit el timeout i es considera que no acabarà la seva execució en un temps acceptable. Aleshores, es programa el *MeasureInstrument* per morir posant un booleà *stop* a cert.

Quan un *MeasureInstrument* està morint, el motor el considera ja mort, i en crea un de nou per substituir-lo. Eventualment, el fil antic s'aturarà quan arribi al final de la iteració, o bé serà interromput per una excepció de timeout i morirà. A més, qualsevol càlcul que finalitzi el *MeasureInstrument* mentre mor no s'escriurà a la base de dades, ja que canviaria la veracitat de les

mètriques derivades que l'haurien d'haver tingut en compte, i això aniria en contra del requisit 4.2.

Per la seva banda, el *MeasureInstrumentsNotifier* s'espera a que tots els *MeasureInstruments* acabin la seva tasca i tornin a l'estat d'esperar un nou esdeveniment per seguir amb el procés.

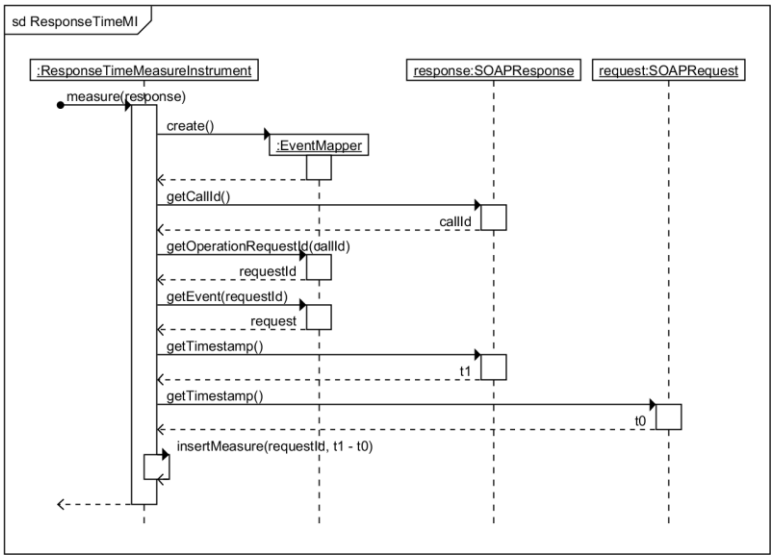

Figura 58: Diagrama de seqüència del *ResponseTimeMeasureInstrument*.

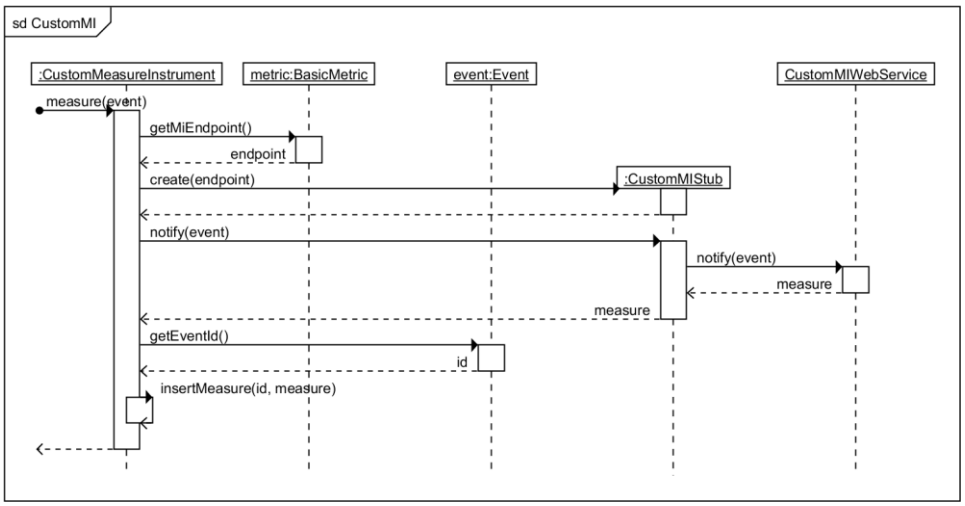

Figura 59: Diagrama de seqüència del *CustomMeasureInstrument*.

Per la seva banda, els instruments de mesura fan servir el patró *Visitor Pattern* per determinar el tipus d'esdeveniment a mesurar en temps d'execució. Mitjançant el patró, l'instrument crida la funció *accept* de la classe *Event*, adequadament sobreescrita en cada subclasse, que a la vegada invoca la funció *measure* del *MeasureInstrument*, que s'encarrega de fer els càlculs oportuns i desar el valor de la mesura a la base de dades.

A la Figura 58 es pot observar el funcionament d'això per al cas del *ResponseTimeMeasureInstrument*. Tot i que no es mostra, el cas del *AvailabilityMeasureInstrument* és anàleg, i el càlcul es fa en funció de l'atribut *isError* de la resposta.

Mentre que els instruments de mesura *CustomMeasureInstrument* (Figura 59) i *ExpressionMeasureInstrument* (Figura 60) serveixen per monitorar totes les mètriques noves que es

puguin introduir al sistema, bàsiques i derivades respectivament, aquest disseny permet afegir mètriques predeterminades al codi del sistema sempre que es preservi el funcionament del patró *Visitor*.

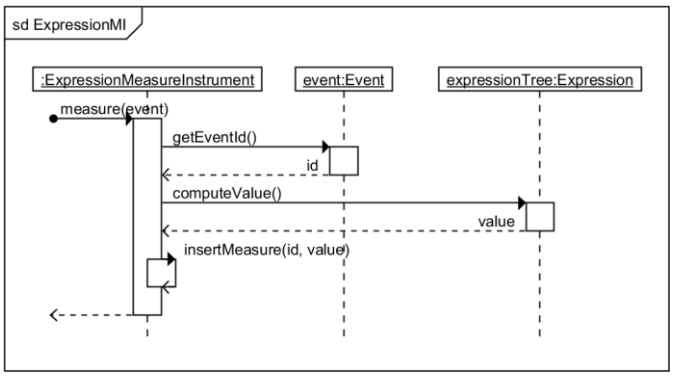

Figura 60: Diagrama de seqüència del *ExpressionMeasureInstrument*.

Per acabar de parlar del disseny del *Proxy*, a les figures Figura 61, Figura 62 i Figura 63 es pot veure la construcció dels diferents tipus d'esdeveniment mitjançant les classes Factory. Cada Factory construeix el seu tipus d'esdeveniment i la part corresponent als atributs del seu tipus per a les subclasses mitjançant un patró *Template Pattern* [GoF].

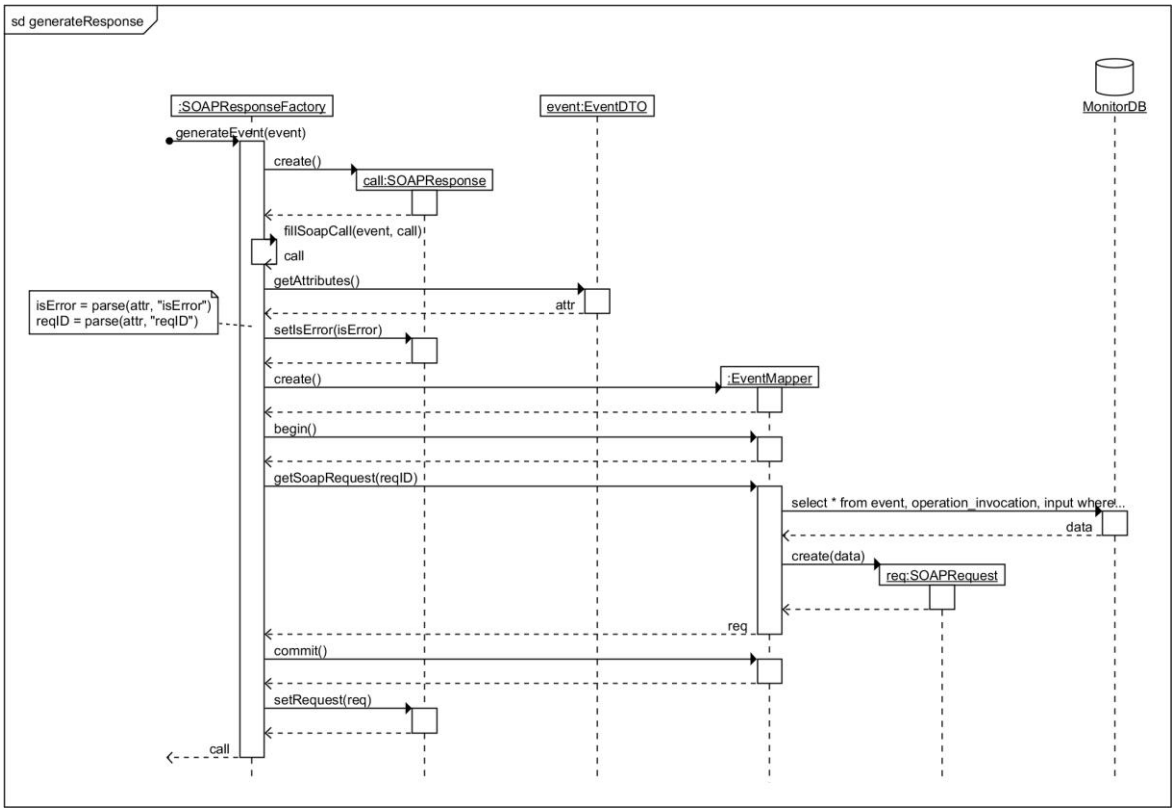

Figura 61: Diagrama de seqüència de la classe *SOAPResponseFactory*.

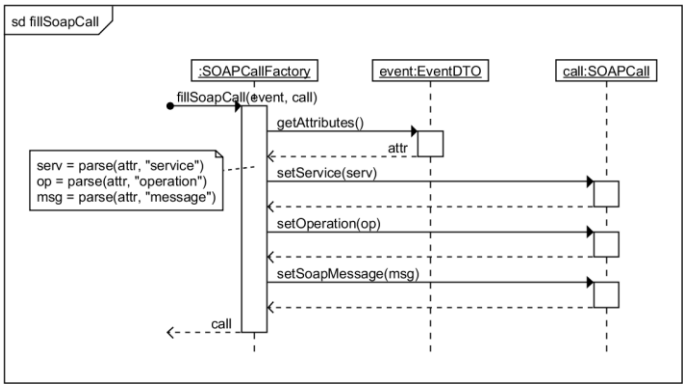

Figura 62: Diagrama de seqüència de la classe *SOAPCallFactory*.

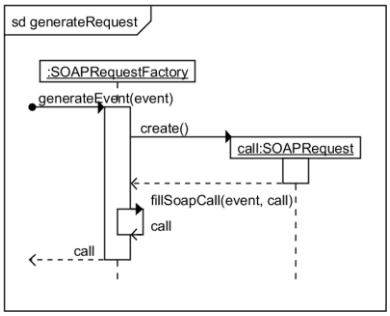

Figura 63: Diagrama de seqüència de la classe *SOAPRequestFactory*.

7.2.3. Disseny del *Publisher*

El *Publisher* és el component encarregat de notificar als clients de SALMon la mesura de nous valors de mètriques. Aquests clients, però, no són els mateixos que els clients de SALMonADA, interessats en rebre informació sobre violacions. En el sistema complet, el client de SALMon és el component *Analyzer*. És el que rebrà les notificacions sobre els nous valors i comprovarà que no violen cap condició dels SLAs.

És el publicador al final de l'arquitectura conduïda per esdeveniments, i implementa el patró *Observer* [GoF]. Els clients es registren i se subscriuen a mètriques a través del *Monitor*. Quan el *Proxy* processa un nou esdeveniment i els instruments de mesura calculen nous valors, envia un avís al *Publisher*, i aquest notifica a tots els clients interessats en aquelles mètriques.

En el cas de SALMonADA, només hi ha un sol client, però SALMon podria formar part de diversos sistemes a la vegada, de manera que el *Publisher* ha de poder distingir quins clients estan subscrits a què.

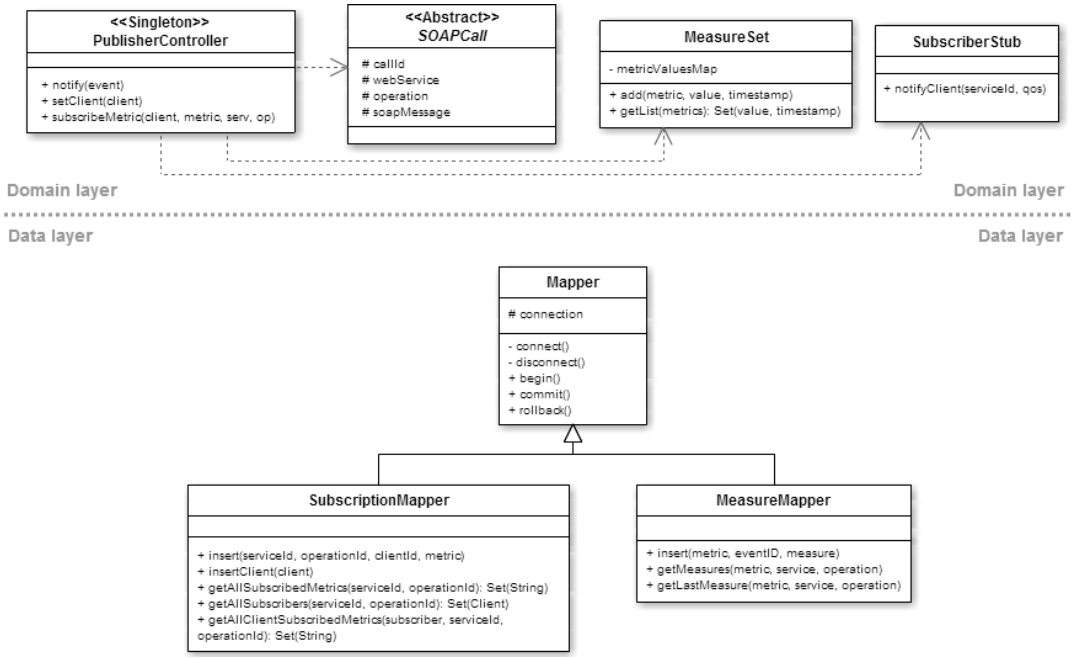

Figura 64: Diagrama de classes del *Publisher*.

Per fer-ho possible, primer s'ha d'afegir un client a la base de dades mitjançant la operació *setClient*. El client ha de ser un servei web SOAP que implementi una interfície de subscriptor concreta que el sistema posa disponible públicament. Amb l'operació *setClient*, el *Publisher* desa la informació de contacte del subscriptor, podent enviar-li d'ara endavant les notificacions de nous valors de mètriques (vegeu la Figura 66).

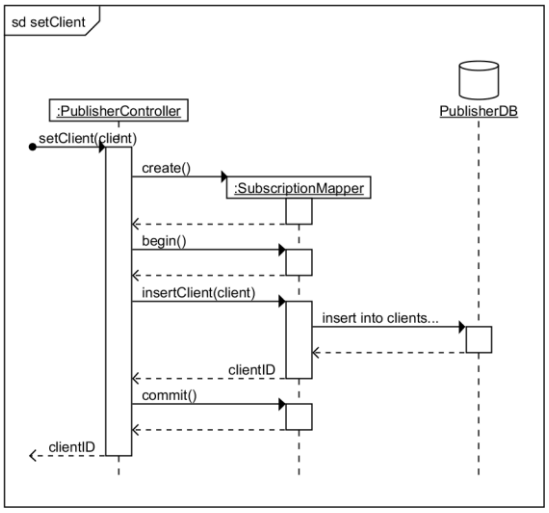

Figura 66: Diagrama de seqüència de l'operació *setClient*.

Una vegada s'ha registrat un client, aquest es pot subscriure a les notificacions per a mètriques i contexts concrets mitjançant l'operació *subscribeMetric*. Aquesta operació és cridada pel *Monitor* durant l'execució de l'operació *monitorMetricForOperation*.

L'operació comprova que tots els contexts requerits hagin estat inserits a la base de dades, i en cas negatiu els afegeix. El client, però, ha d'haver estat registrat prèviament, ja que es necessiten dades addicionals a l'identificador, així que la operació dona error si el client no existeix. Si tot és correcte, subscriu el client a la mètrica (vegeu la Figura 65).

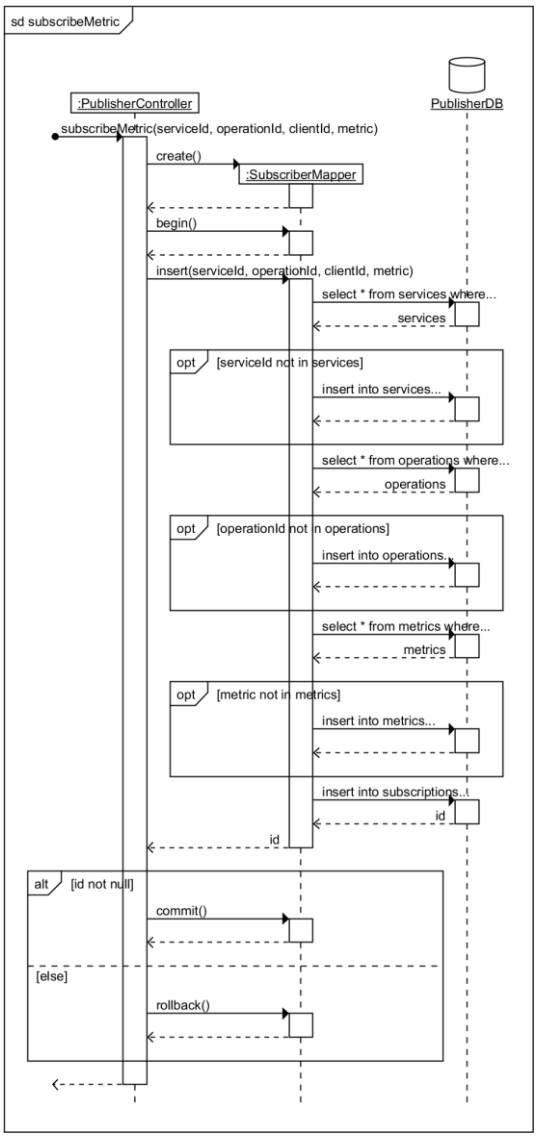

Figura 65: Diagrama de seqüència de l'operació *subscribeMetric*.

A partir d'aquest punt, tota notificació de noves mesures en aquesta mètrica per a aquests contexts provocarà que el *Publisher* notifiqui aquest client. Aquestes notificacions es fan mitjançant l'operació *notify*.

Aquesta operació es connecta a ambdues bases de dades, la del *Monitor* i la del *Publisher*, per a determinar els subscriptors que han de ser notificats i consultar els valors de les mètriques que han canviat. Finalment notifica a cada subscriptor només les mètriques a les quals està subscrit (vegeu la Figura 67).

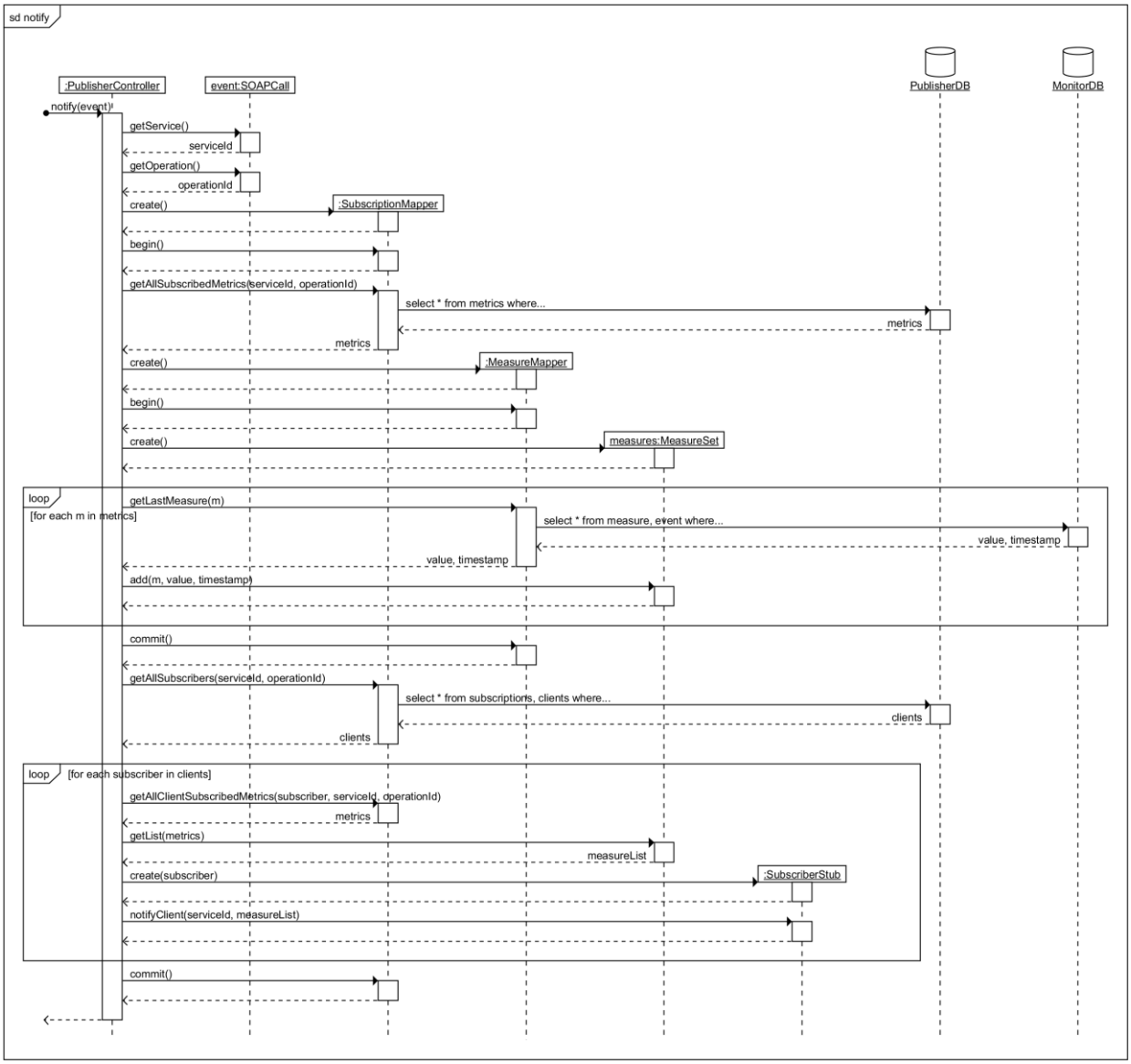

Figura 67: Diagrama de seqüència de l'operació *notify*.

El *Publisher*, a més, disposa de la seva pròpia base de dades per desar la informació de les subscripcions. És un model simple que es pot observar a la Figura 68.

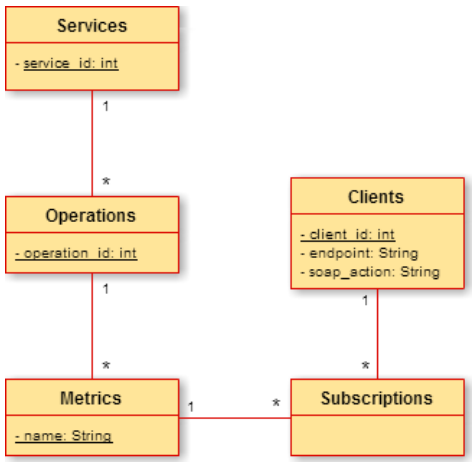

Figura 68: Model de dades del *Publisher*.

7.2.4. Disseny de l'*Analyzer*

L'*Analyzer* és el component que integra els dos sistemes del projecte: ADA i SALMon. Depèn de les dues interfícies d'aquest sistema, i també ha d'implementar la interfície de client del *Publisher* i dependre de la interfície d'un component que s'ocupi d'enviar les notificacions de les violacions als usuaris del sistema: Aquest component és el *Violation Publisher*.

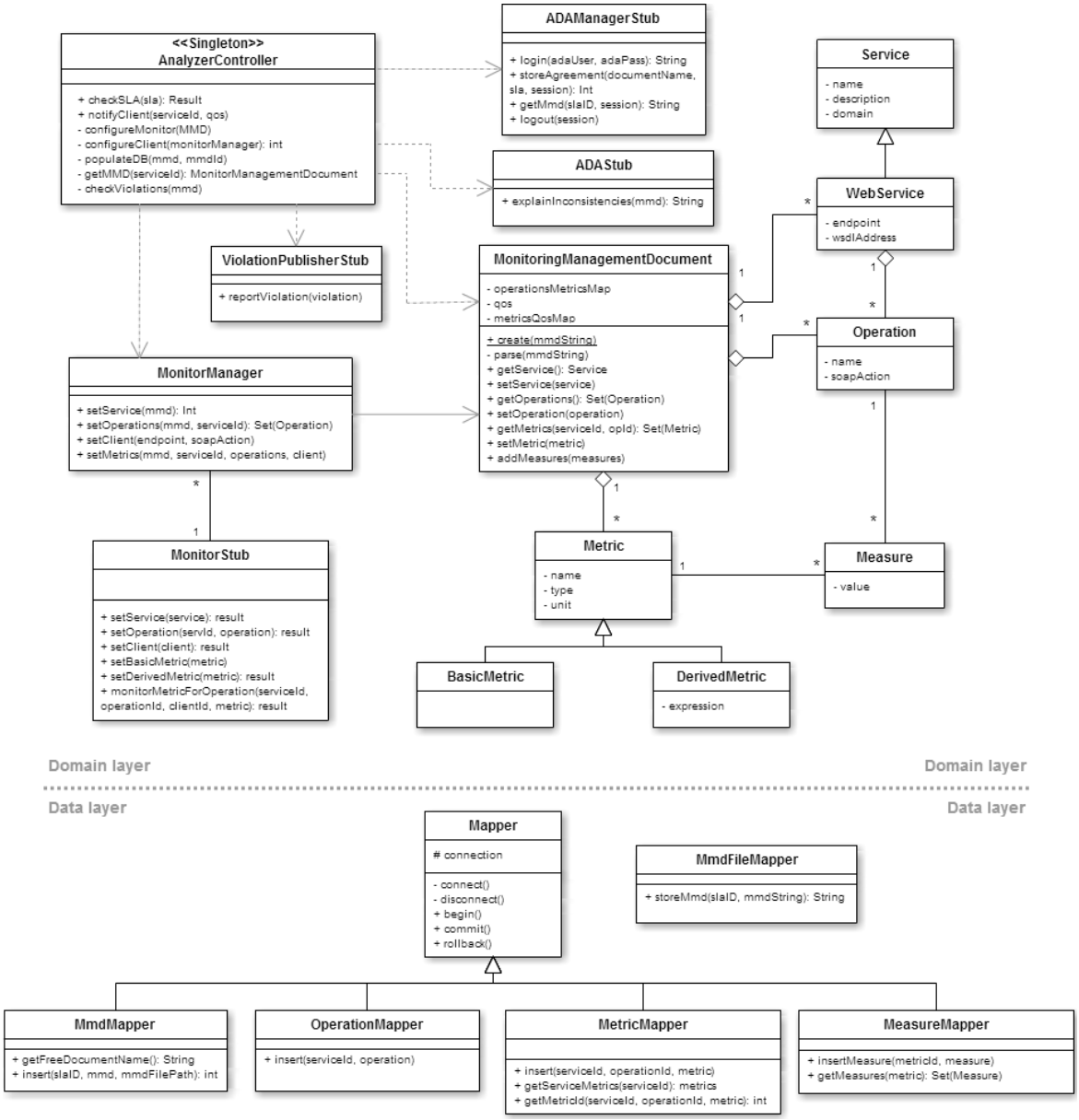

Figura 69: Diagrama de classes de l'*Analyzer*.

Cal notar, també, que ADA consta de dos components amb interfícies independents: *ADAManager*, que s'ocupa del registre de nous SLAs i la compilació a MMDs, i *ADA*, que s'ocupa de verificar que un MMD amb dades de QoS compleix les condicions d'un SLA.

Internament, l'Analyzer consta amb dos components importants: El *Parser MMD* i el *Configurador del Monitor*. El parser ha estat dissenyat i implementat per l'equip ISA de Sevilla, ja que era necessari a ADA també. El configurador l'implementa la classe *MonitorManager*, que és la classe que queda realment acoblada amb la interfície del *Monitor*, simplificant les crides a aquest (vegeu la Figura 69).

La interfície de l'*Analyzer* té només dues operacions:

- *CheckSLA* afegeix un document SLA al sistema per a ser monitorat, configurant el *Monitor* pròpiament.
- *NotifyClient* implementa la interfície de subscriptor, comprovant que les noves mesures de QoS rebudes no violin cap condició i, en cas de violar-ne, notificant al *Violation Publisher*.

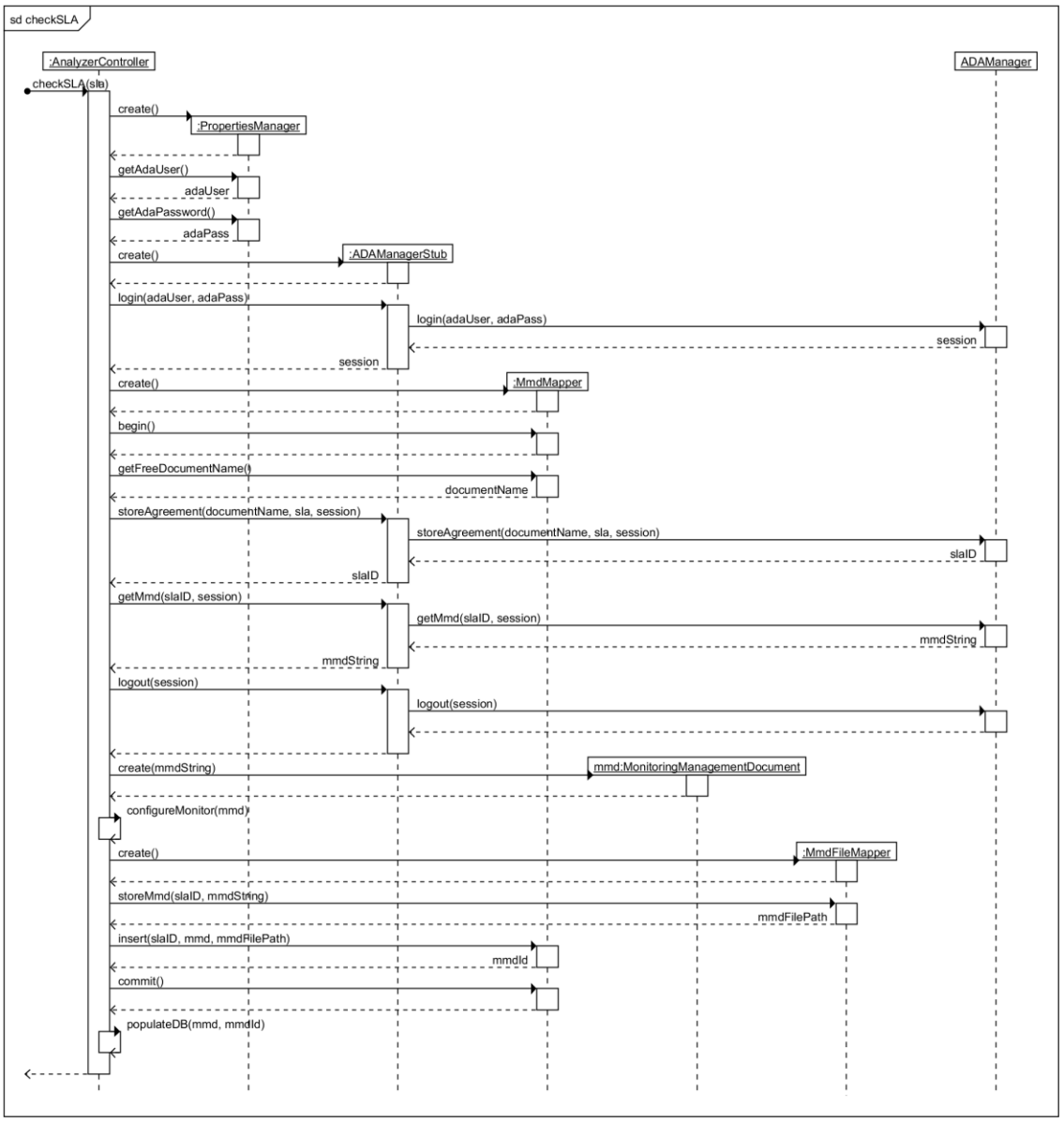

Figura 70: Diagrama de seqüència de l'operació *checkSLA*.

L'operació *checkSLA* comença registrant l'SLA al component *ADAManager*, que s'accedeix mitjançant un patró *Session State* [FowlerEAA]. *ADAManager* retorna un document MMD, que és parsejat i desat en un fitxer a disc per l'*Analyzer* (vegeu la Figura 70).

Tot seguit, l'*Analyzer* configura el *Monitor* a partir del MMD mitjançant el *MonitorManager*. Mentre s'executa el procés, el sistema desa l'estructura d'operacions i mètriques del servei en memòria, relacionant-los amb els identificadors que el *Monitor* els atribueix (vegeu la Figura 71).

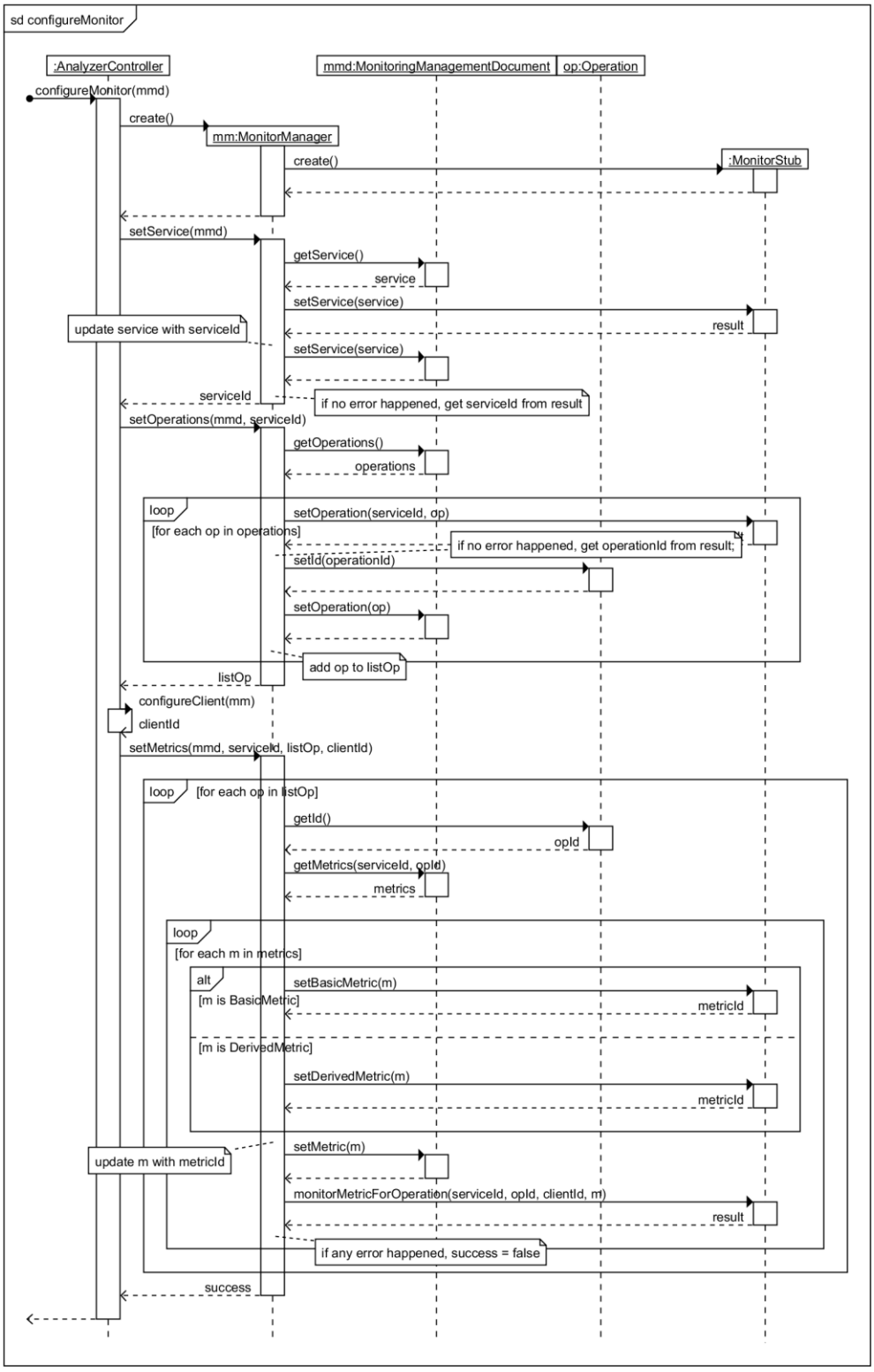

Figura 71: Diagrama de seqüència de la configuració del *Monitor* a través del *MonitorManager*.

L'Analyzer s'afegeix a sí mateix com a client del Monitor i se subscriu a totes les mètriques que cal monitorar. Per a fer-ho (vegeu la Figura 72), cal que l'*Analyzer* conegui la seva pròpia direcció de desplegament. Per evitar maldecaps aconseguint aquesta informació del sistema operatiu, aquestes dades es desen en un fitxer *.properties* junt amb la resta de la configuració del servei.

Després de configurar el *Monitor*, l'*Analyzer* mapa el MMD i l'estructura del seu contingut a la base de dades (vegeu la Figura 76), per facilitar l'emmagatzemament de les mesures posteriorment (vegeu la Figura 73).

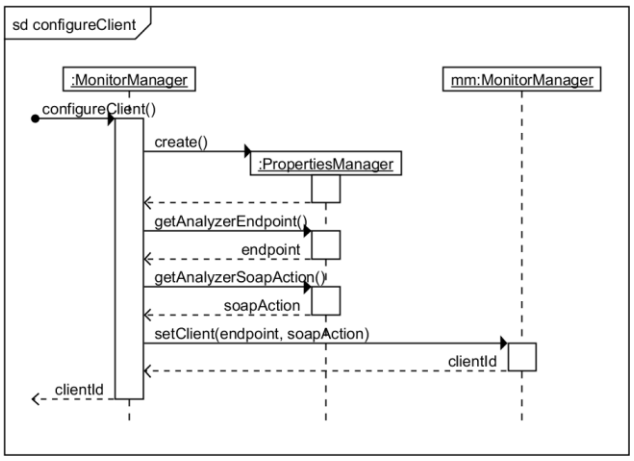

Figura 72: Diagrama de seqüència de la configuració del client.

Aquestes mesures són emmagatzemades a la base de dades, i l'actualització del document MMD amb la qualitat del servei es fa sota demanda, en comptes de fer-se de forma incremental, degut a restriccions d'implementació sobre l'escriptura de fitxers.

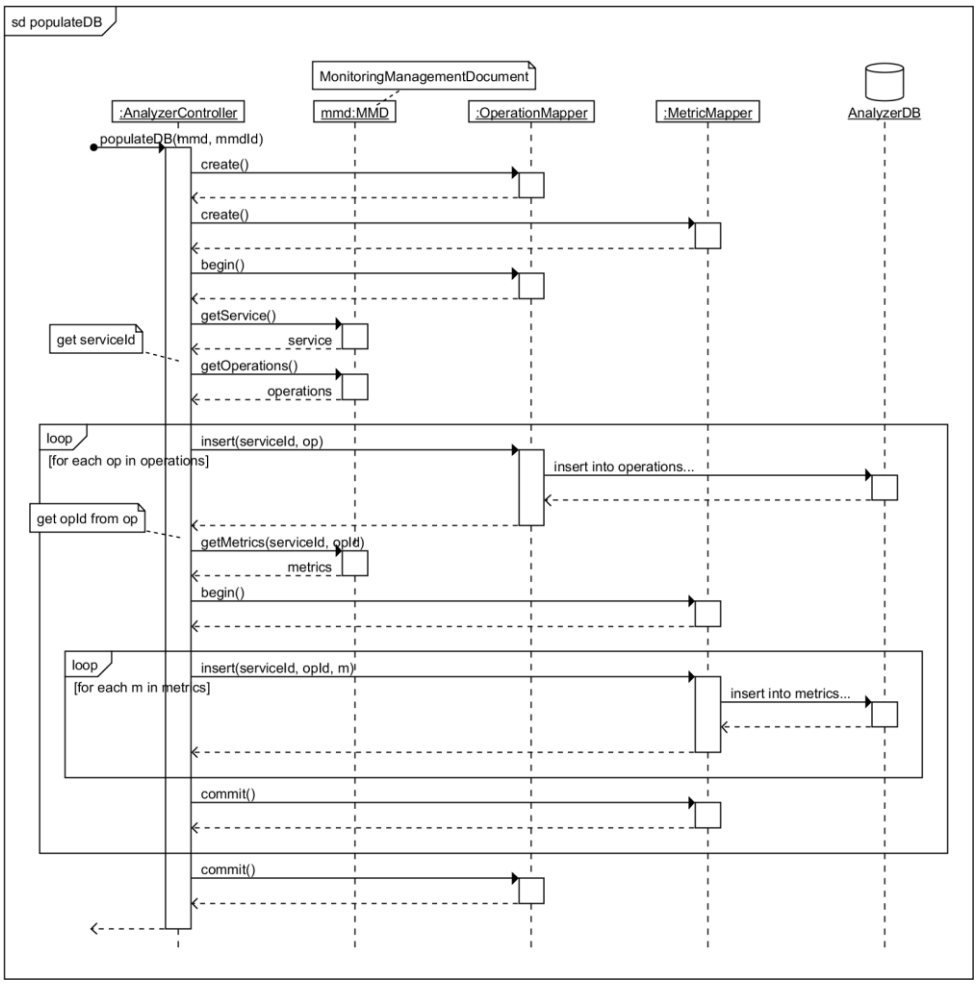

Figura 73: Diagrama de seqüència del mapatge del MMD a la base de dades.

L'operació *notifyClient* s'executa quan el Publisher envia un nou avís com a resultat d'un esdeveniment monitorat, i inclou la qualitat del servei monitorada. L'Analyzer incorpora aquestes mesures a la base de dades (vegeu la Figura 74).

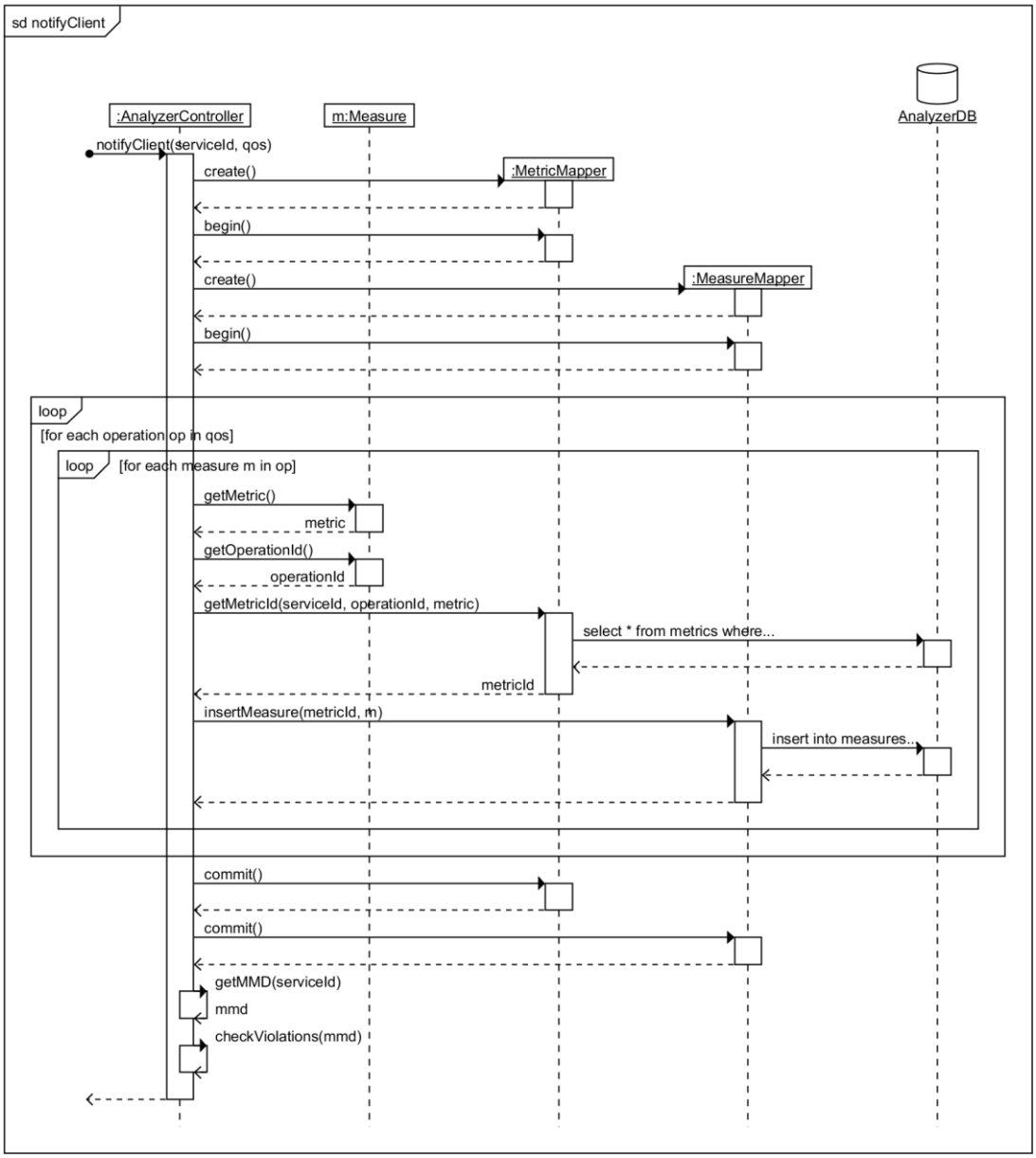

Figura 74: Diagrama de seqüència de l'operació *notifyClient*.

Tot seguit, recupera el document MMD corresponent al servei que ha estat monitorat. Per fer-ho, consulta la ruta del fitxer a la base de dades, i després llegeix el fitxer i el parseja (vegeu la Figura 75). El document és actualitzat amb totes les mesures necessàries, i després és enviat al servei ADA (vegeu la Figura 77), que s'ocupa de verificar de forma creuada cada condició amb la qualitat del servei del document, i retorna l'explicació de les violacions ocorregudes en cas que n'hi hagi cap. Aquesta explicació té el format que requereix el requisit 1.6.

Posteriorment, aquest missatge és enviat al *Violation Publisher* perquè el notifiqui a la interfície d'usuari del sistema.

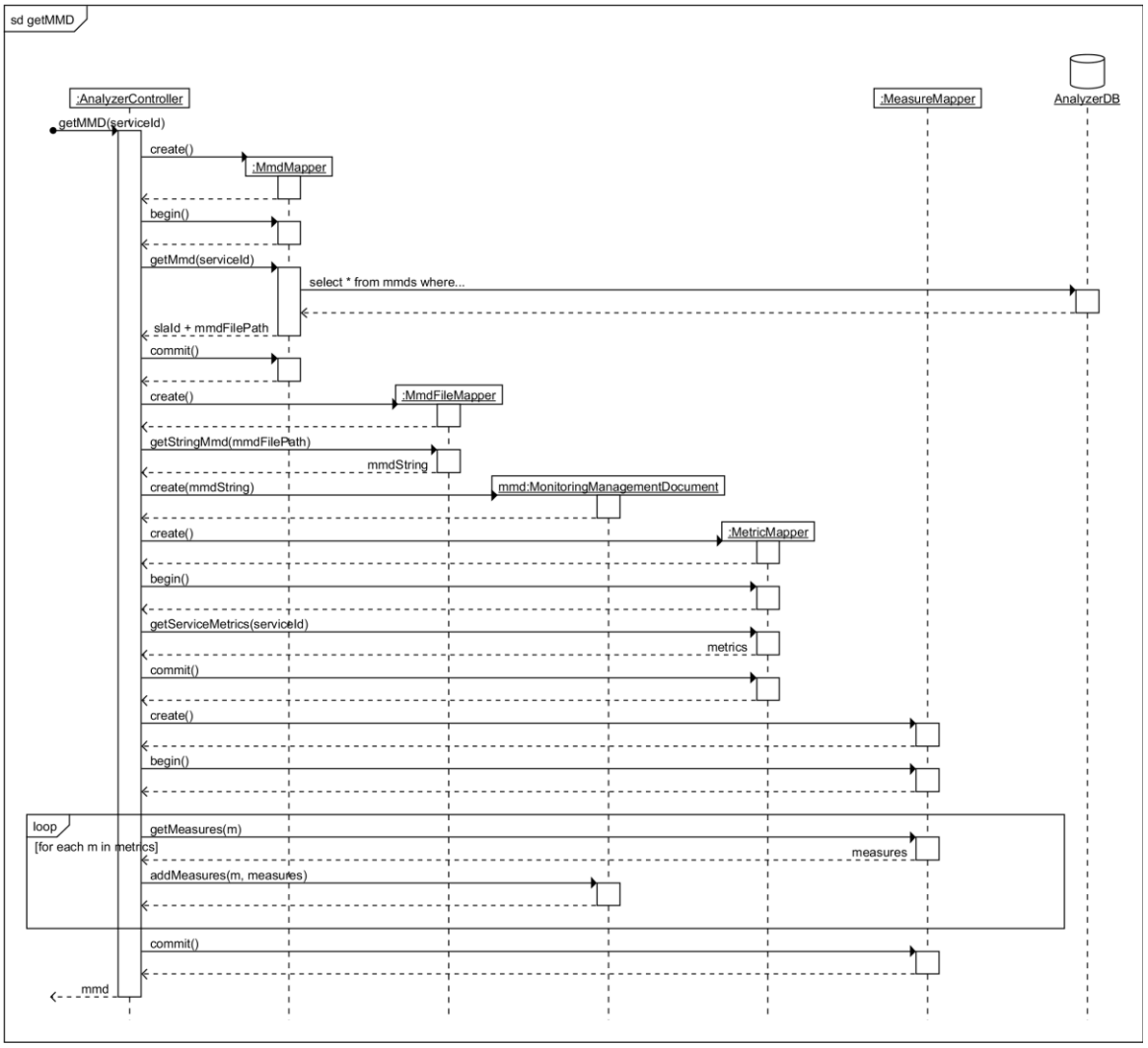

Figura 75: Diagrama de seqüència de l'obtenció del document MMD amb valors de QoS.

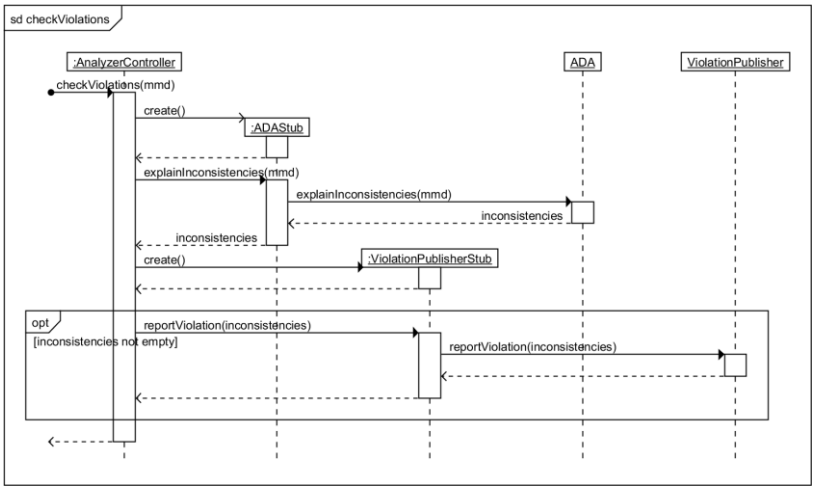

Figura 77: Diagrama de seqüència de la comprovació de violacions.

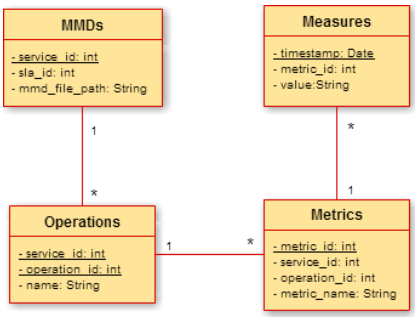

Figura 76: Model de dades de l'*Analyzer*.

7.2.5. Disseny del Violation Publisher

El *Violation Publisher* és un component molt simple la funció del qual és desacoblar l'*Analyzer* dels mètodes de notificació de violacions. En aquest projecte les violacions són notificades a través d'una interfície web, però si en comptes d'això es volguessin notificar per correu electrònic o mitjançant una xarxa social, només faria falta canviar el Violation Publisher, sense haver de modificar l'*Analyzer*.

El seu funcionament és trivial a nivell de disseny: Rebre un missatge a través de l'operació *reportViolation*, connectar-se a la *interfície web* i enviar el missatge.

A nivell d'implementació, però, requereix estar acoblat amb la tecnologia de recepció de missatges de la interfície web. Vegeu el capítol 7.2.6. Disseny de la interfície web a continuació per saber-ne més.

7.2.6. Disseny de la interfície web

La *interfície web* és el punt d'entrada del sistema, doncs actua d'interfície gràfica d'usuari. Les seves dues funcions són permetre pujar documents SLA al sistema en forma de fitxers WS-Agreement, i mostrar les violacions ocorregudes.

Mentre que la pujada de fitxers és una funció relativament freqüent en aplicacions web que no té excessiva complicació, la notificació de violacions en temps real necessita de la tècnica Server Push per aconseguir enviar les notificacions al client web. Això és un requisit que ha fet que el disseny es vegi altament influït per la implementació, ja que és una tècnica complicada que requereix tecnologies específiques cadascuna amb protocols diferents.

El disseny, doncs, s'ha fet mitjançant el prototipat de la funció de Server Push amb la tecnologia de WebSockets. Una vegada s'ha comprovat que el funcionament complia els requisits per al projecte, s'ha dissenyat la resta de la interfície. Per això, aquest model (vegeu la Figura 78) mostra un disseny conscient de la implementació.

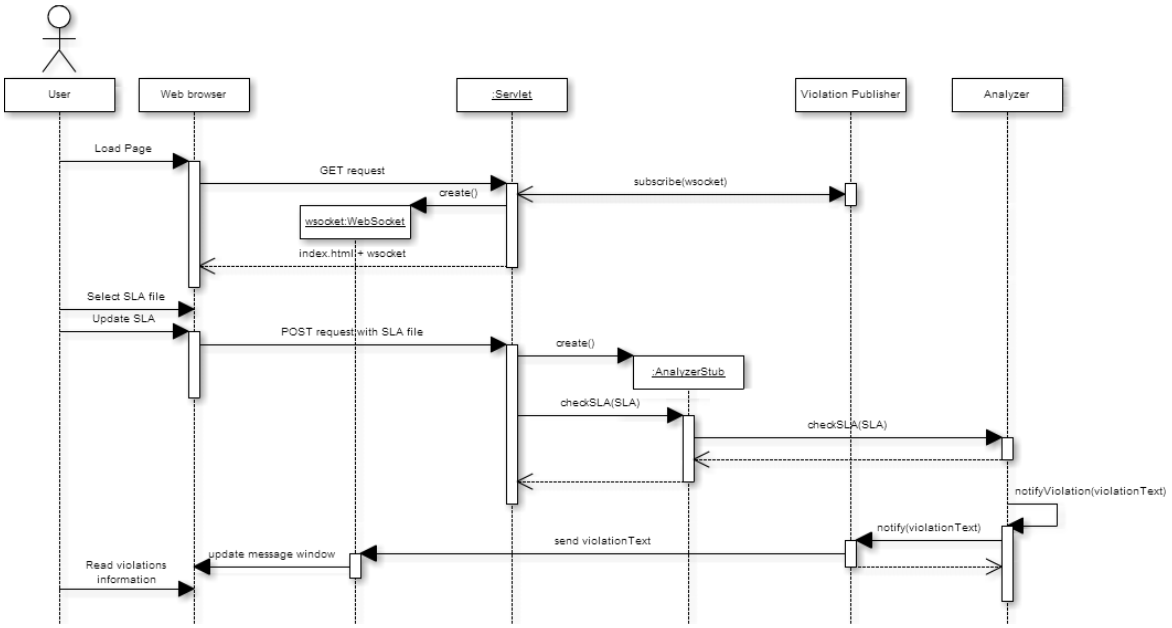

Figura 78: Diagrama de seqüència de la *interfície web*.

8. Implementació

En aquest capítol són discutides qüestions concretes de la implementació del sistema. Com que el sistema és massa gran per entrar en detall en totes les seves capacitats i serveis, només són analitzades parts concretes de la implementació.

Les parts explicades a continuació s'han seleccionat bé perquè van portar a problemes no trivials la solució dels quals és interessant, bé perquè no han estat tractades en prou detall al capítol de disseny.

8.1. Intercepció de missatges

La intercepció dels missatges SOAP mitjançant el ESB Apache Synapse ha estat un dels reptes del projecte.

Synapse és configurat a través d'un fitxer XML on s'especifiquen les regles que ha de seguir per filtrar, redirigir, duplicar i transformar missatges. Malgrat que és molt flexible, hi ha coses que són impossibles de fer, com per exemple la manipulació dels missatges SOAP com si fossin text pla.

Aquesta manipulació és necessària per aconseguir el cos del missatge i enviar-lo a SALMon. Mentre que es poden consultar i manipular dades de dins les etiquetes XML, no és possible accedir a les etiquetes XML sense conèixer quines són, i el sistema no té manera de saber com estan construïts els missatges de tots els serveis web que poden ser monitorats.

Per fer-ho, però, Synapse dona una facilitat addicional: La possibilitat d'executar scripts quan s'especifiqui. Aquests scripts sí que poden accedir i manipular el missatge de totes les maneres possibles, i poden ser escrits en determinats llenguatges: Python, Ruby, i Bash. Aquests scripts són executats per una màquina virtual Java, i tenen accés a una API Java de Synapse. Per al projecte es va fer servir Ruby (en realitat, JRuby) perquè se'n tenien coneixements previs. Com que és un llenguatge dèbilment tipat executat sobre una màquina virtual dins un ESB, resulta un entorn difícil de depurar.

Al començament del projecte, Synapse estava configurat com un *man-in-the-middle*[1]. Synapse rebia els missatges del client, els encapsulava dins una crida al *Proxy* i el *Proxy* els reenviava al proveïdor, monitorant el temps de resposta directament.

Això comporta certs problemes: Augmenta el temps de resposta percebut pel client, i compromet la comunicació, posant-la en mans del *Proxy*. Aquests problemes comprometen els requisits 3.1, 5.6 i 6.3.

Per evitar aquests problemes es va decidir fer que Synapse enviés el missatge original al proveïdor i una còpia al *Proxy*, tal com descriu el protocol al capítol 7.1.2. Protocol de detecció d'esdeveniments (pàgina 68). Això es fa clonant el missatge a enviar i forçant que s'enviïn en seqüència, enviant primer el missatge al proveïdor, i minimitzant així l'impacte en el temps de

[1] Tècnica de comunicació en xarxa, normalment fraudulenta, que funciona posant una tercera part en una comunicació entre dos punts, interceptant els missatges i enviant-los al destinatari com a ell li interessi i fent la seva presència transparent.

resposta percebut. A més, s'elimina la possibilitat que el codi de SALMon pugui modificar la informació del missatge, protegint el compliment del requisit 6.3.

Quan s'envia el missatge, Synapse executa el script. Aquest calcula el moment exacte de l'enviament, construeix l'esdeveniment amb les dades detectades i l'envia al *Proxy*. En el moment de la recepció, i després d'haver enviat el missatge de tornada al client, fa el mateix amb l'esdeveniment de resposta. D'aquesta manera, es pot considerar que Synapse, actuant com a sonda, és qui calcula el temps de resposta, però qui fa el càlcul és l'instrument de mesura. Aquesta diferència només és apreciable, però, en mètriques de càlcul més complex que una simple resta.

El *Proxy*, per la seva banda, relaciona les crides entre elles gràcies a un paràmetre *messageID* que és afegit per Synapse a les capçaleres dels missatges SOAP, i que és idèntic en la petició i la resposta de la mateixa crida.

8.2. Problemes trobats i solucions

Certs problemes que van sorgir durant la implementació van obligar a dissenyar una solució durant aquesta. A continuació es parla dels més interessants pel que fa al coneixement reutilitzable a altres projectes.

8.2.1. Concurrència d'accés a dades

Problema: Durant els tests unitaris dels instruments de mesura concurrents es detecten errors a la base de dades de connexions tancades abans d'hora. Es determina que les transaccions estan usant la mateixa connexió per culpa del disseny antic i es solapen entre elles.

Possibles solucions: Canviar el disseny del Mapper, fent que sigui creat i destruït cada vegada, obrint una nova connexió en ser creat; fer que el Mapper usi una connexió d'un pool de connexions.

Solució i raonament: Canviar el disseny del Mapper s'estima pràcticament tan costós en temps com l'altra solució. Per altra banda, obrir una nova connexió cada vegada disminueix en gran mesura l'eficiència del codi. S'opta per tant per una solució híbrida: Es canvia el disseny del Mapper fent que deixi de ser Singleton (no tan costós), i fer que usi una connexió d'un pool de connexions. Es cerquen llibreries de pools de connexions, i els resultats es redueixen a dues: DBCP i C3P0. Es comparen tot dissenyant el prototip de pool de connexions, i es determina que per l'ús relativament simple que se'n vol fer, DBCP és més senzill. Es completa el prototip de pool de connexions i s'integra amb el Mapper.

DBCP utilitza la classe *javax.sql.DataSource* de la biblioteca JDBC per construir les instàncies de la classe *Connection* que han de poblar el pool de connexions. La classe es configura amb unes credencials d'accés a la base de dades en ser creada, i un cop el pool ha estat iniciat, no hi ha manera de canviar-les. No obstant, el sistema consta de serveis que han d'estar en execució constant en condicions normals, mentre que la base de dades pot canviar d'ubicació. Per fer que les noves connexions carreguin sempre les credencials del fitxer de configuració *.properties*, s'ha estès la classe *DataSource* fent una classe *UpToDateDataSource* que llegeix el fitxer cada vegada que ha de crear una nova connexió, crea un *BasicDataSource* auxiliar a partir de les credencials i el fa crear una connexió.

8.2.2. Recepció d'esdeveniments

Problema: Els tests unitaris del Proxy amb la cua d'esdeveniments i els instruments de mesura concurrents ja implementats mostren errors estranys; insercions a base de dades fallides i, després de depuració intensiva, temps de resposta negatius. Es determina que l'ordre en la recepció d'esdeveniments no és sempre el mateix, de vegades el SOAPResponse arriba abans que el SOAPRequest de la mateixa crida. Concretament, la primera crida després de desplegar el Synapse.

Possibles solucions: Fer que la inserció dels esdeveniments a la cua sigui bloquejant fins que l'esdeveniment predecessor hagi estat inserit; moure al principi de la cua els esdeveniments que es comencen a processar abans que el seu predecessor, i afegir-los un nombre màxim de reinsercions permeses.

Solució i raonament: Cal assumir que és possible que un esdeveniment predecessor no arribi mai. En aquest cas, fer la inserció bloquejant té el risc d'omplir la memòria del sistema amb fils adormits que no despertaran mai. La implementació, a més, és més costosa. Per altra banda, tornar a posar a la cua els esdeveniments pendents posa en perill el requisit 1.6, en cas que la cua sigui molt llarga, perquè perd la detecció en temps real. S'opta per la inserció bloquejant, però al sistema s'implementa de forma provisional l'altra solució.

8.2.3. Instruments de mesura

Problema: Certs instruments de mesura són serveis externs, amb temps de resposta impredictible. Al principi del desenvolupament, els instruments són executats de forma seqüencial, i cal paral·lelitzar la crida a tots ells. Les mètriques d'un esdeveniment, però, han de ser calculades després que totes les mètriques disponibles dels esdeveniments anteriors hagin estat calculades.

Possibles solucions: Implementar cada instrument de mesura com un fil amb la seva pròpia cua d'esdeveniments pendents; sincronitzar els instruments de mesura amb el motor de processament d'esdeveniments.

Solució i raonament: Una cua d'esdeveniments a cada instrument soluciona el problema per a les mètriques bàsiques, però no per a les mètriques derivades. A més, replicar tots els esdeveniments a cada cua fa que la memòria usada incrementi exponencialment. Sincronitzar els instruments, però, augmenta el temps de procés de cada esdeveniment al temps de procés de l'instrument més lent, i aquest temps és impredictible (pot ser infinit). S'opta per sincronitzar els instruments amb el motor i afegir un timeout fixat per l'administrador del sistema en el fitxer *.properties*. Les mesures que superin el timeout no seran tingudes en compte. S'implementa la sincronització amb la biblioteca *java.concurrency* en comptes de fer servir els mètodes sincronitzats clàssics de Java, els monitors i els semàfors. S'usen objectes *Latch* per fer que els instruments esperin a la notificació del motor per començar a processar, i per fer que el motor esperi a que els instruments o el timeout acabin. Aquests objectes bloquegen el motor fins que tots els fils dels instruments de mesura han acabat. Per fer això, el motor compta primer el nombre d'instruments de mesura involucrats, i després els passa l'objecte *Latch*.

8.2.4. Compilador d'expressions

Problema: Les expressions que defineixen les mètriques derivades s'han de compilar, i el llenguatge que les descriu és molt ampli.

Possibles solucions: Usar una biblioteca de compilació; programar el compilador manualment; i programar-lo manualment usant llibreries d'ajuda.

Solució i raonament: Reconèixer mètriques derivades més enllà de les requerides pel requisit 2.8 és fora de l'abast del projecte, i les possibilitats del llenguatge són innumerables, des de limitacions temporals complexes ("temps de resposta mitjà del servei a primera hora dels dilluns no festius d'hivern") fins a càlculs condicionals ("temps de resposta mitjà de les respostes sense errors"). Programar el compilador manualment és costós i es pot titllar de "reinventar la roda". Usar una biblioteca de compilació és un procés massa costós per quelcom fora de l'abast del projecte. S'opta per limitar la gramàtica acceptada perquè compleixi amb el requisit 2.8. Es programa manualment un compilador simple usant la biblioteca *java.io.StreamTokenizer*. Es prepara el compilador per poder ser estès amb una biblioteca de compilació completa que compili un llenguatge més poderós.

9. Proves del sistema

En aquest capítol es detallen algunes de les proves a què s'ha sotmès el sistema.

Les proves del sistema s'han dividit en proves unitàries a components concrets a l'entorn de desenvolupament, i proves generals del sistema en entorns de desplegament. A més, el sistema ha estat desplegat i obert a revisió en la versió del llançament C1.3 per als revisors dels articles [SALMonADA_PESOS] i [AKertesz], i en la versió del llançament C2.1 per als revisors de l'article [SALMonADA_JCR].

De totes les proves que s'han fet, a continuació es tracten les més crítiques.

9.1. Proves de requisits funcionals

Verifica correcció dels SLA (requisit 2.1)

1. L'usuari introdueix un document WS-Agreement incorrecte (que conté un o més errors d'acord a l'especificació de WS-Agreement) a un prototipus de la interfície web.
2. L'usuari fa clic a monitorar SLA.

I també:

1. L'usuari introdueix un document WS-Agreement inadequat al sistema (que conté un o més errors d'acord a l'especificació concreta dels SLAs permesos per ADA).
2. L'usuari fa clic a monitorar SLA.

Condicions de satisfacció:

- El sistema mostra un missatge d'error.

Monitora missatges (requisits 2.2 i 2.6)

1. L'usuari introdueix un contracte SLA sobre un servei S amb una condició sobre la mètrica AverageResponseTime.
2. L'usuari fa clic a monitorar SLA.
3. Un client fa una crida al servei S a través de Synapse.
4. El servei S respon a la crida.
5. Els punts 3 i 4 són repetits entre 5 i 10 vegades.

Condicions de satisfacció:

- Apareix informació sobre l'esdeveniment a la base de dades, amb el timestamp adequat, i els missatges de petició i resposta, per cada crida realitzada.
- Apareix una nova mesura de les mètriques CurrentResponseTime i AverageResponseTime a la base de dades, ambdues lligades a l'esdeveniment corresponent, per cada crida.
- Les mesures tenen un valor mitjà en mil·lisegons aparentment correcte. Els valors molt inferiors a 100 mil·lisegons i superiors a 2000 s'ha observat que són escassos i, per tant, sospitosos. S'ha d'assumir la presència d'outliers (valors molt allunyats de

la mitjana o dels valors esperats), sobretot en temps més llargs en les primeres crides després del desplegament.

Notifica violacions (requisits 2.3 i 2.4)

1. L'usuari introdueix un contracte SLA sobre un servei S amb una condició sobre la mètrica CurrentResponseTime que exigeix un valor absurd (per exemple, inferior a 10 mil·lisegons).
2. L'usuari fa clic a monitorar SLA.
3. Un client fa una crida al servei S a través de Synapse.
4. El servei S respon.

Condicions de satisfacció:

- Apareix a la interfície un missatge conforme ha ocorregut una violació del SLA per culpa de la mètrica CurrentResponseTime.

Registre de logs[1] (requisits 2.5 i 2.6)

1. L'usuari configura el *Monitor* per monitorar una mètrica bàsica que usa un instrument de mesura extern per a un servei S.
2. L'usuari comprova que no hi ha cap instrument de mesura desplegat a l'endpoint indicat.
3. L'usuari configura el fitxer *monitor.properties* amb una ruta d'un fitxer de logs apropiada.
4. Un client fa una crida al servei S.
5. El servei S respon.

Condicions de satisfacció:

- Apareix una entrada al fitxer de logs conforme no s'ha pogut monitorar la mètrica perquè l'instrument de mesura no està disponible.

Presumpció de no violació (requisit 2.7)

1. L'usuari introdueix un contracte SLA sobre un servei S amb una condició sobre la mètrica bàsica MyResponseTime que exigeix un valor absurd, i una altra condició igual sobre la mètrica CurrentResponseTime.
2. L'usuari comprova que no hi ha cap instrument de mesura desplegat a l'endpoint indicat.
3. Un client fa una crida al servei S.
4. El servei S respon.

Condicions de satisfacció:

- Apareix a la base de dades informació sobre l'esdeveniment.

[1] Àmpliament testejat durant la depuració de tots els casos de test.

- La interfície notifica una violació d'una clàusula per un CurrentResponseTime massa gran.
- La interfície no notifica cap violació de clàusula per un MyResponseTime massa gran.

Monitora mètriques de SALMon (requisit 2.8)

1. L'usuari configura el *Monitor* per monitorar les mètriques CurrentResponseTime, AverageResponseTime, MinimumResponseTime, MaximumResponseTime, CurrentAvailability, AccumulatedAvailability, AccumulatedUnavailability i AverageRecoveryTime d'un servei S.
2. Un client fa una crida al servei S.
3. El servei S respon.

Condicions de satisfacció:

- Apareixen a la base de dades mesures de totes les mètriques i informació d'un sol esdeveniment.

9.2. Proves unitàries

CustomMeasureInstruments

1. S'implementa un instrument de mesura per a una nova mètrica, *Efficiency*, definida com la mida del missatge de petició en bytes dividida entre el temps de resposta en mil·lisegons, com a un nou servei web SOAP, i es desplega.
2. Es configura el *Monitor* per a monitorar la mètrica Efficiency per a un servei S.
3. Un client fa una crida al servei S.
4. El servei S respon.

Condicions de satisfacció:

- Apareix a la base de dades una mesura de la mètrica Efficiency.

Resultat de la prova: Satisfactori.

Concurrència en instruments de mesura

1. S'implementa un prototip de *MeasureInstrumentsNotifier* on diversos instruments de mesura esperen un temps configurable i escriuen per pantalla un missatge informatiu quan monitoren cadascun de dos esdeveniments consecutius.
2. S'executen diferents proves amb els temps d'espera: Tots amb el mateix; uns amb més que uns altres; i un instrument de mesura amb un temps d'espera fora del permès.

Condicions de satisfacció:

- Tots els instruments de mesura amb un temps no excessiu escriuen el missatge per pantalla.
- Cap d'ells monitora un esdeveniment abans que l'últim monitori l'esdeveniment anterior.

Resultat de la prova: Incorrecte.

Causes i solucions dels problemes:

- L'instrument de mesura amb temps excessiu monitora l'esdeveniment quan els altres ja monitoren l'esdeveniment posterior. La funció de Java *Thread.isAlive()* és inadequada perquè pot donar falsos positius i falsos negatius. S'implementa un booleà *stop* i una funció *isDying()* que retorna cert si l'instrument de mesura està aturat o té el booleà stop a cert. Es força l'instrument a finalitzar quan acabi l'execució amb el booleà stop, i s'impedeix que afegeixi mesures a la base de dades quan aquest és cert.

Server Push

1. S'implementa un prototip d'interfície web amb un WebSocket.
2. L'usuari es connecta a la interfície web.
3. L'usuari obre una altra instància de la interfície i fa clic a un botó preparat per a la prova.

Condicions de satisfacció:

- Ambdues instàncies de la interfície reben un missatge quan el botó ha estat premut.
- Ambdues instàncies de la interfície reben un missatge generat pel servidor cada 10 segons.

Resultat de la prova: Incorrecte.

Causes i solucions dels problemes:

- Les interfícies no es poden connectar al WebSocket. La direcció s'ha d'indicar damunt la classe que implementa el WebSocket amb una annotació Java tal que *@WebServlet("/WebSocketTest")*.

10. Execució i valoració econòmica

Una vegada enllestit el projecte, aquest capítol fa una ullada als resultats assolits i la valoració del projecte des d'un punt de vista empresarial.

10.1. Execució del projecte

Com s'explica al capítol 5.2. Pla de treball (pàgina 51), el projecte s'ha executat en diverses iteracions. Moltes de les iteracions han estat per realitzar components o afegir funcionalitats, però cal notar que les dues iteracions més llargues han estat l'actualització dels components *Monitor* i *Proxy*.

Aquestes dues iteracions han transformat la base de dades del *Monitor* de la Figura 30 (pàgina 61) dues vegades, amb tots els canvis que ha comportat en les capes de dades i domini d'ambdós serveis web. La primera iteració va deixar la base de dades tal com s'aprecia a la Figura 79, i la segona iteració tal com s'apreciava al capítol del disseny a la Figura 47 (pàgina 82).

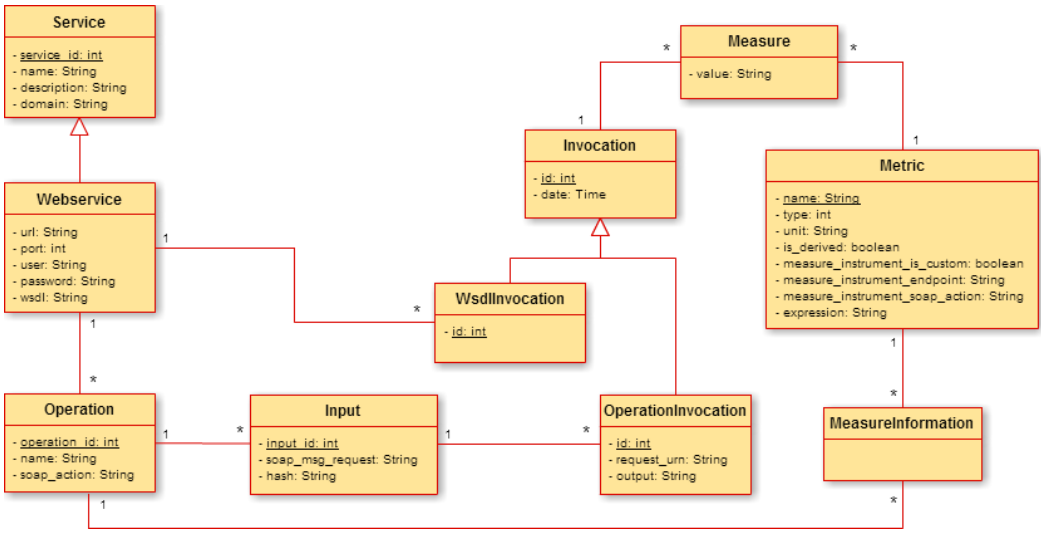

Figura 79: Model de dades del *Monitor* al moment del llançament C2.2.

Els llançaments C1.3 i C2.1 es van usar, a més, per donar suport empíric als articles [SALMonADA_PESOS] i [SALMonADA_JCR], respectivament.

El projecte compleix els objectius *a*, *b*, *c* i *d* establers al capítol 1.3.3. Objectius (pàgina 6), i compleix els requisits del capítol 4.3. Requisits específics (pàgina 30) tal com descriu la Taula 4 i tal com es demostra al capítol 9. Proves del sistema (pàgina 105).

Els riscs trobats s'han limitat al Risc 3 que, malgrat les contingències, ha suposat un fre important. Malgrat haver planificat el projecte amb prediccions pessimistes, el temps requerit ha estat superior, com predeia el Risc 3.

Requisit	Satisfet?	Comentaris
1. Interfícies externes		
1.1. Carrega documents SLA	Sí	
1.2. Reconeix SLA amb info. monitoratge	Sí	
1.3. Reconeix SLA amb info. mètriques	Sí	
1.4. Suporta mètriques noves	Sí	Les expressions de mètriques derivades tenen un llenguatge limitat.
1.5. Intercepta missatges SOAP	Sí	
1.6. Notifica violacions per interfície	Sí	
1.7. Mostra violacions incrementalment	Sí	
2. Requisits funcionals		
2.1. Verifica correcció SLA	Sí	
2.2. Monitora missatges	Sí	
2.3. Comprova compliment SLA	Sí	
2.4. Notifica violacions	Sí	
2.5. Registra errors en fitxers log	Sí	
2.6. Desa un valor per mètrica i missatge	Sí	
2.7. Pressuposa no violació del SLA	Sí	
2.8. Monitora les mètriques de SALMon	Sí	
3. Requisits de rendiment		
3.1. Intercepció no intrusiva	Sí	
3.2. Usuaris múltiples	Sí	
3.3. Base de dades gran	Sí	Més d'1GB d'espai verificable. En principi, sense limitacions.
4. Requisits lògics de la base de dades		
4.1. Desa missatges de petició i resposta	Sí	
4.2. Desa dades mètriques	Sí	
4.3. Permet filtrar per moment i operació	Sí	No permet filtrar per qualsevol altre atribut.
4.4. Classifica segons missatge de petició	Sí	
4.5. Tracta tipus de valor de mètriques	Sí	
5. Restriccions de disseny		
5.1. Arquitectura SOA	Sí	
5.2. Adherit a estàndards SOAP i WSDL	Sí	
5.3. Monitora serveis SOAP	Sí	
5.4. Reconeix SLAs en WS-Agreement	Sí	
5.5. Usa ADA i SALMon	Sí	
5.6. No intrusiu al client o proveïdor	Sí	
6. Atributs de qualitat		
6.1. Extensible per altres tipus de serveis	Sí	Permet altres tipus de serveis, però cal modificar el codi. Cal una modificació de la capa de dades per permetre l'extensibilitat total.
6.2. Canviable amb pas de documents	Sí	
6.3. Monitoratge transparent	Sí	
6.4. Càrrega senzilla de SLAs	Sí	

Taula 4: Compliment dels requisits del projecte.

A continuació (vegeu la Figura 80) es pot apreciar un diagrama de Gantt que mostra el desenvolupament dels paquets de treball en relació a la planificació inicial. Com es pot apreciar, els estadis de desenvolupament del codi del *Monitor* van ser els més subestimats en cost temporal.

WBS	Task Name	Predecessors	Team	% Complete	Work	Start	Finish
1	**Plataforma per al monitoratge i l'explicació de violacions de clàusules de documents SLA**			**0%**	**1.020 hrs**	**Tue 22/01/13**	**Thu 23/01/14**
1.1	**SLA**			**0%**	**0 hrs**	**Wed 23/01/13**	**Thu 31/01/13**
1.1.1	Definir WS-Agreement-MD	22	ISA	0%	0 hrs	Wed 23/01/13	Thu 24/01/13
1.1.2	**Parser SLA**		**ISA**	**0%**	**0 hrs**	**Thu 24/01/13**	**Tue 29/01/13**
1.1.3	**Repositori SLA**		**ISA**	**0%**	**0 hrs**	**Fri 25/01/13**	**Wed 30/01/13**
1.1.4	**Compilador SLA a MMD**		**ISA**	**0%**	**0 hrs**	**Fri 25/01/13**	**Wed 30/01/13**
1.1.5	**Verificador MMD**		**ISA**	**0%**	**0 hrs**	**Thu 24/01/13**	**Thu 31/01/13**
1.2	**Analyzer**			**0%**	**229 hrs**	**Tue 22/01/13**	**Mon 13/01/14**
1.2.1	**Parser MMD**		**ISA/GESSI**	**0%**	**32 hrs**	**Tue 22/01/13**	**Mon 13/01/14**
1.2.2	Especificar sistema		GESSI	0%	70 hrs	Tue 22/01/13	Mon 04/02/13
1.2.3	Dissenyar Analyzer	27;3	GESSI	0%	20 hrs	Mon 04/02/13	Wed 06/02/13
1.2.4	**Components Analyzer**		**GESSI**	**0%**	**86 hrs**	**Thu 07/02/13**	**Tue 04/06/13**
1.2.5	Testejar Analyzer	34	GESSI	0%	15 hrs	Tue 01/10/13	Tue 29/10/13
1.2.6	Desplegar Servei Analyzer	35	GESSI	0%	6 hrs	Tue 29/10/13	Fri 01/11/13
1.3	**Violation Publishing**			**0%**	**74 hrs**	**Fri 23/08/13**	**Fri 25/10/13**
1.3.1	**Violation Publisher**		**GESSI**	**0%**	**29 hrs**	**Mon 26/08/13**	**Fri 25/10/13**
1.3.2	**Interfície Web**		**GESSI**	**0%**	**45 hrs**	**Fri 23/08/13**	**Tue 17/09/13**
1.4	**SALMon**			**0%**	**455 hrs**	**Fri 08/02/13**	**Mon 06/01/14**
1.4.1	**Publisher**		**GESSI**	**0%**	**53 hrs**	**Fri 15/02/13**	**Tue 26/02/13**
1.4.2	Especificar Monitor		GESSI	0%	45 hrs	Fri 08/02/13	Mon 06/05/13
1.4.3	Dissenyar Monitor	70	GESSI	0%	42 hrs	Thu 28/02/13	Thu 07/03/13
1.4.4	**Modificar disseny mètriques**		**GESSI**	**0%**	**90 hrs**	**Wed 13/03/13**	**Thu 07/11/13**
1.4.5	**Afegir tractament d'esdeveniments**		**GESSI**	**0%**	**103 hrs**	**Thu 04/04/13**	**Fri 02/08/13**
1.4.6	**Afegir filtratge de valors**		**GESSI**	**0%**	**34 hrs**	**Fri 05/04/13**	**Fri 23/08/13**
1.4.7	Implementar mètriques predefinides	56;65	GESSI	0%	19 hrs	Tue 19/11/13	Tue 03/12/13
1.4.8	Testejar Monitor	58;63	GESSI	0%	28 hrs	Tue 03/12/13	Mon 23/12/13
1.4.9	**Measure Instruments**		**GESSI**	**0%**	**41 hrs**	**Tue 26/02/13**	**Mon 06/01/14**
1.5	**Documents**			**0%**	**262 hrs**	**Tue 22/01/13**	**Thu 23/01/14**
1.5.1	Manual d'usuari		GESSI	0%	8 hrs	Mon 13/01/14	Fri 17/01/14
1.5.2	Manual de desenvolupador		GESSI	0%	9 hrs	Fri 17/01/14	Thu 23/01/14
1.5.3	Memòria		GESSI	0%	217 hrs	Thu 14/03/13	Tue 16/04/13
1.5.4	Informe previ		GESSI	0%	12 hrs	Mon 11/03/13	Tue 12/03/13
1.5.5	Gestió del projecte		GESSI	0%	16 hrs	Tue 22/01/13	Tue 28/05/13

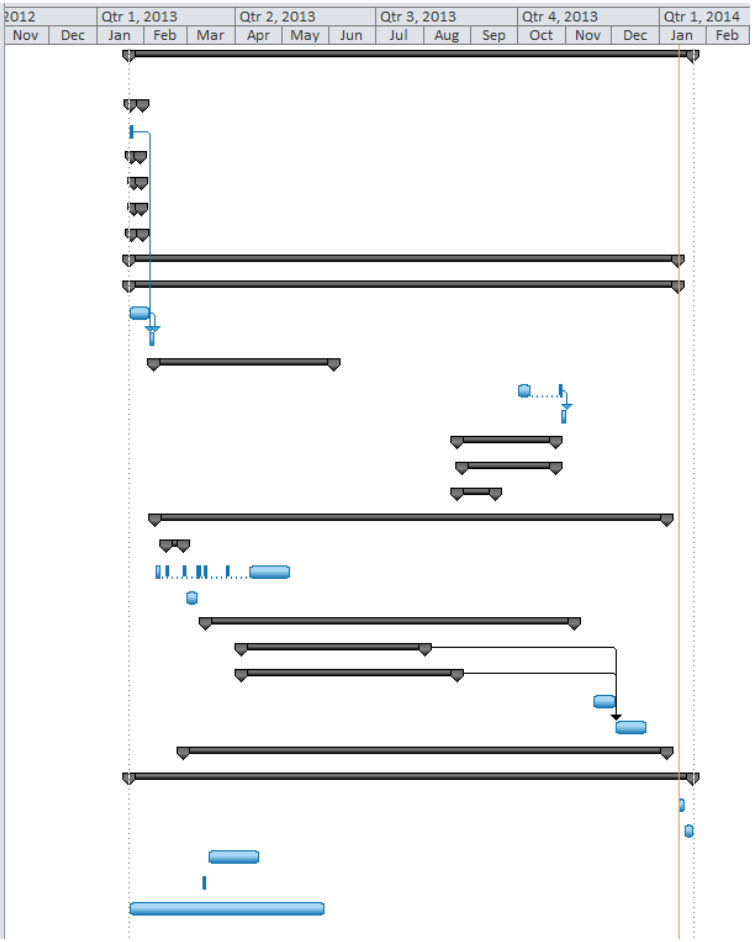

Figura 80: Diagrama de Gantt que mostra l'evolució final dels paquets de treball.

10.2. Anàlisi de costs

Aquest projecte ha estat realitzat en l'àmbit de la recerca. No obstant, cal fer un estudi del cost del projecte.

Pel que fa als recursos humans, en tractar-se d'un projecte de final de carrera hi ha hagut una sola persona desenvolupant el projecte sota la supervisió del director de projecte. En un cas real, el projecte seria desenvolupat per un equip amb diversos rols, cadascun d'ells amb un cost econòmic diferent (vegeu la Taula 5).

Recurs	Cost econòmic
Cap de projecte	60 €/h
Analista	45 €/h
Dissenyador	45 €/h
Programador	25 €/h
Tester	15 €/h

Taula 5: Costs econòmics dels diferents rols del desenvolupament de sistemes software.

Tenint en compte aquestes dades, s'ha calculat el cost d'aquest projecte en funció de les hores treballades, classificant cada tasca en el rol que normalment la realitza. Vegeu la Taula 6 per més detall.

WBS	Task Name	Work	Cost
1.2	**Analyzer**	**229 hrs**	**7.795,00 €**
1.2.1	**Parser MMD**	**32 hrs**	1.020,00 €
1.2.1.1	Definir MMD	0 hrs	0,00 €
1.2.1.2	Implementar Parser MMD	0 hrs	0,00 €
1.2.1.3	Especificar temporalitats MMD	16 hrs	720,00 €
1.2.1.4	Modificar Parser MMD	6 hrs	150,00 €
1.2.1.5	Testejar temporalitats	10 hrs	150,00 €
1.2.2	Especificar sistema	70 hrs	3.150,00 €
1.2.3	Dissenyar Analyzer	20 hrs	900,00 €
1.2.4	**Components Analyzer**	**86 hrs**	2.350,00 €
1.2.4.1	Implementar Repositori MMD	12 hrs	300,00 €
1.2.4.2	Especificar Interfície Analyzer	10 hrs	450,00 €
1.2.4.3	Implementar SALMon Configurer	20 hrs	500,00 €
1.2.4.4	Implementar prototip client ADA	10 hrs	250,00 €
1.2.4.5	Implementar Analyzer	34 hrs	850,00 €
1.2.5	Testejar Analyzer	15 hrs	225,00 €
1.2.6	Desplegar Servei Analyzer	6 hrs	150,00 €
1.3	**Violation Publishing**	**74 hrs**	**2.450,00 €**
1.3.1	**Violation Publisher**	**29 hrs**	805,00 €
1.3.1.1	Dissenyar Violation Publisher	4 hrs	180,00 €
1.3.1.2	Implementar Violation Publisher	25 hrs	625,00 €
1.3.2	**Interfície Web**	**45 hrs**	1.645,00 €
1.3.2.1	Recerca frameworks amb Server push	23 hrs	1.035,00 €
1.3.2.2	Especificar Interfície Web	2 hrs	90,00 €
1.3.2.3	Dissenyar Interfície Web	4 hrs	180,00 €

1.3.2.4	Implementar Interfície Web	10 hrs	250,00 €
1.3.2.5	Testejar Interfície Web	6 hrs	90,00 €
1.4	**SALMon**	**455 hrs**	**12.995,00 €**
1.4.1	**Publisher**	**53 hrs**	1.565,00 €
1.4.1.1	Dissenyar Publisher	15 hrs	675,00 €
1.4.1.2	Implementar Publisher	32 hrs	800,00 €
1.4.1.3	Testejar Publisher	6 hrs	90,00 €
1.4.2	Especificar Monitor	45 hrs	2.025,00 €
1.4.3	Dissenyar Monitor	42 hrs	1.890,00 €
1.4.4	**Modificar disseny mètriques**	**90 hrs**	2.250,00 €
1.4.4.1	Pas a mètriques abstractes BD	3 hrs	75,00 €
1.4.4.2	Pas a mètriques abstractes codi	56 hrs	1.400,00 €
1.4.4.3	Pas a mètriques abstractes interfície	31 hrs	775,00 €
1.4.5	**Afegir tractament d'esdeveniments**	**103 hrs**	2.575,00 €
1.4.5.1	Pas a esdeveniments BD	6 hrs	150,00 €
1.4.5.2	Pas a esdeveniments codi	78 hrs	1.950,00 €
1.4.5.3	Pas a esdeveniments interfície	5 hrs	125,00 €
1.4.5.4	Pas a esdeveniments proxy	14 hrs	350,00 €
1.4.6	**Afegir filtratge de valors**	**34 hrs**	850,00 €
1.4.6.1	Pas a filtres BD	3 hrs	75,00 €
1.4.6.2	Pas a filtres codi	26 hrs	650,00 €
1.4.6.3	Pas a filtres interfície	5 hrs	125,00 €
1.4.7	Implementar mètriques predefinides	19 hrs	475,00 €
1.4.8	Testejar Monitor	28 hrs	420,00 €
1.4.9	**Measure Instruments**	**41 hrs**	945,00 €
1.4.9.1	Especificar Interfície Measure Instrument	6 hrs	270,00 €
1.4.9.2	Implementar prototip Measure Instrument	15 hrs	375,00 €
1.4.9.3	Testejar prototip Measure Instrument	20 hrs	300,00 €
1.5	**Documents**	**262 hrs**	**15.465,00 €**
1.5.1	Manual d'usuari	8 hrs	360,00 €
1.5.2	Manual de desenvolupador	9 hrs	405,00 €
1.5.3	Memòria	217 hrs	13.020,00 €
1.5.4	Informe previ	12 hrs	720,00 €
1.5.5	Gestió del projecte	16 hrs	960,00 €
	TOTAL	**1020 hrs**	**38.705,00 €**

Taula 6: Desglossament de costs per tasca.

Pel que fa a recursos materials, el cost de l'equip local va ser de 500 €, mentre que no es disposa de dades suficients per calcular el cost de la infraestructura del Laboratori de Càlcul de la UGDSI.

Així doncs, el cost total del projecte ascendeix a **39.205,00 €**.

11. Treball futur

El fet de desenvolupar aquest projecte, sobretot a la fase d'especificació, ha tret a la llum moltes característiques desitjables d'un monitor de documents SLA com aquest que aquest projecte no requeria. Aquí se'n comenten unes quantes.

11.1. Catàleg de mètriques

En el sistema, les mètriques es reconeixen pel seu nom. Mentre que això resulta molt pràctic, també planteja un problema: Què passa si dos documents SLA volen usar el mateix nom per a dues mètriques que es defineixen diferent, o dos noms diferents per a la mateixa mètrica?

A tall d'exemple, no és estrany pensar que algú pot definir la mètrica eficiència amb el seu instrument de mesura particular que la calculi com a mida de la petició entre el temps de resposta, mentre que algú altre pot calcular-la tenint en compte la dificultat del càlcul que el servei ha hagut de realitzar, amb coneixements de com aquest servei està implementat.

En el sistema actual, però, aquest problema se soluciona retornant un error quan s'introdueix una mètrica nova amb un nom ocupat. L'usuari ha de pensar aleshores un altre nom per a la mètrica, i això a la llarga porta a que cada SLA defineixi els noms de les mètriques amb un identificador propi. El nombre de mètriques es multiplica absurdament, i això pot fer alentir el sistema.

Una possible solució al problema seria fer un catàleg de mètriques, mitjançant una descripció humana, paraules clau o, en el cas de les derivades, la pròpia expressió, que en cas de voler inserir una mètrica que ja es troba present en el sistema (bé pel nom o bé per l'expressió) proposi a l'usuari usar alguna de les mètriques ja existents abans d'afegir-ne una de nova.

11.2. Tester

Una de les funcionalitats que ha sacrificat el sistema per a poder evolucionar degudament és el Tester, el component que actua com un client artificial del servei web i permet monitorar-lo sense clients reals i, per tant, en un entorn de desenvolupament.

Abans, el Tester podia formar part del component *Proxy* perquè la crida al servei final es feia des del *Proxy* mateix, que actuava com a *man-in-the-middle*. Ara, però, les crides són redirigides directament des de l'ESB. El Tester ha de ser un component extern que actuï com un client qualsevol.

Fer un Tester, per tant, és tan senzill com fer un client del servei web que es desitja monitorar, però el que és desitjable és tenir un Tester genèric que serveixi per a qualsevol servei SOAP, o fins i tot per a qualsevol context d'un protocol suportat. Aquest Tester podria ser programat per fer crides o provocar esdeveniments en certs moments i en ordres particulars, adaptable a casos de prova.

11.3. Parser de missatges SOAP

Tots els instruments de mesura auxiliars que han de tractar amb propietats del missatge SOAP, ja sigui la petició o la resposta, es veuen obligats a implementar el seu propi parser de missatges SOAP. El sistema podria proporcionar als instruments de mesura el missatge classificat per

atributs i valors mitjançant un objecte genèric que pogués contenir un o més objectes genèrics i un o més parells d'atribut i valor.

11.4. Compilador de mètriques derivades

Mentre que el sistema incorpora un compilador molt simple d'expressions de mètriques derivades, aquest compilador pot ser estès per reconèixer totes les expressions que permet el model conceptual.

Això inclou filtres sobre el conjunt de valors de cada mètrica que es desitja agafar per un càlcul. Aquests filtres poden ser per temps, pels diferents contexts de les mesures, o fins i tot condicionals sobre els valors de les mesures.

11.5. Consulta de QoS sota demanda

El sistema permet detectar violacions dels contractes SLA, però no ofereix cap manera de consultar la qualitat del servei si aquest no viola el contracte. Resulta interessant, a l'hora de comparar diferents ofertes de servei entre elles, saber quina ofereix la millor qualitat.

El sistema podria oferir una operació per consultar una mètrica o un conjunt de valors mitjançant una expressió com si d'una mètrica derivada es tractés, calculant-la en el moment i oferint les dades al moment.

Cal notar que això permetria que els instruments de mesura auxiliars utilitzessin aquesta operació per calcular les mesures, amb la qual cosa s'altera la definició de mètrica bàsica.

11.6. Altres contexts

El sistema suporta SOAP perfectament, però altres protocols i contexts monitorables poden ser afegits al sistema.

Es poden desenvolupar sondes d'altres tipus d'esdeveniments i programar el sistema per què els reconegui degudament.

11.7. Processament complex d'esdeveniments

Un altre possible punt d'extensió és augmentar el poder del motor d'esdeveniments. Això pot resultar útil per aplicacions usuals dels sistemes monitors com ara l'avaluació retrospectiva de la qualitat de servei de serveis que ofereixen prediccions.

A nivell conceptual, n'hi ha prou amb permetre que un esdeveniment tingui més d'un esdeveniment predecessor. D'aquesta manera, es pot programar el sistema per què reconegui les prediccions com a esdeveniments predecessors (igual que les peticions en SOAP), i les comprovacions del esdeveniment predit com a esdeveniments successors (igual que les respostes en SOAP). D'aquesta manera, la qualitat del servei de les prediccions no és mesurada fins que no es pot comprovar si s'ha complert la predicció.

12. Conclusions

S'ha desenvolupat un sistema software que, en relació als objectius inicials del projecte, permet:

a) Configurar el monitorat d'un servei web SOAP partint únicament de les dades d'un document SLA de tipus WS-Agreement a través d'una interfície web.
b) A partir d'aquest moment, monitorar la qualitat del servei d'aquest per a totes les mètriques definides en el document SLA.
c) A cada nova mesura, comprovar que la qualitat del servei acumulada fins al moment no violi cap de les condicions del document SLA.
d) En cas de violar una condició, mostrar automàticament una notificació a l'usuari a través de la interfície web, explicant les clàusules violades i els motius de les violacions.

El sistema és una arquitectura SOA distribuïda que pot ser usada tant pels clients com pels proveïdors de serveis web per a comprovar la qualitat del servei en temps real. Consta de dos subsistemes: ADA, que s'ocupa d'interpretar els documents SLA i comprovar si la qualitat de servei en viola cap condició, i SALMon, que s'ocupa de monitorar els serveis.

S'afegeix un nou component, Analyzer, que serveix de pont entre els dos sistemes i la interfície, i es modifica el sistema SALMon per permetre l'addició de noves mètriques en temps d'execució tal com demanen els requisits. Aquestes mètriques són calculades per serveis web SOAP que poden ser afegits al sistema a voluntat de l'usuari, representant l'única excepció a l'objectiu *a*, ja que la implementació d'aquests serveis web no es troba dins el document SLA, però afegeix al sistema poder de monitorat.

Es dissenya el SALMon per a ser una arquitectura conduïda per esdeveniments, amb un motor de processament d'esdeveniments que executa els instruments de mesura i modifica la base de dades. Es dissenya el monitor per què intercepti missatges SOAP a través d'un ESB Apache Synapse que actua de sonda del sistema. Aquest detecta esdeveniments d'enviament i resposta dels missatges SOAP que siguin enviats a través de l'ESB i els envia al monitor per què els monitori, si escau. El monitor, alhora, manté un registre de què cal monitorar i què no, i quines mètriques en cada cas.

S'implementa el monitor amb una cua d'esdeveniments, un motor d'esdeveniments amb instruments de mesura concurrents sincronitzats esdeveniment a esdeveniment, i un sistema de publicació-subscripció mitjançant el qual es manté a l'Analyzer al corrent de la qualitat del servei detectada en temps real. A la vegada, es modifica la seva base de dades per permetre l'addició de mesures en temps d'execució i els esdeveniments, i s'implementa un pool de connexions a la base de dades per facilitar la concurrència.

Finalment, s'implementa una interfície web amb Java Servlets que implementa un sistema de Server Push mitjançant WebSockets per a mostrar les violacions del SLA en temps real.

Tot i que el projecte obre moltes portes a la millora del sistema, també assenta les bases d'aquestes millores, proposant un disseny extensible i identificant moltes d'aquestes possibles millores.

El projecte ha seguit una metodologia àgil que itera sobre capacitats del sistema i sobre components, fent múltiples desplegaments i mantenint el sistema usable durant gran part del temps de desenvolupament. Això ha facilitat en gran mesura l'ús del projecte en altres àmbits, com ara articles, però ha incrementat molt el temps de desenvolupament.

Cal notar que el factor més crític en aquest augment de temps ha estat el desplegament de cada component i versió, i no el desenvolupament dividit per etapes. Les etapes del desenvolupament pràcticament no se solapaven, i no ha calgut implementar moltes coses repetides vegades. Malgrat tot, en retrospectiva es pot dir que potser hauria estat més fàcil modificar la base de dades sencera en una sola iteració, i reservar les iteracions a les funcionalitats del codi. Per altra banda, l'addició d'un sistema de logs ha estat indispensable per a la depuració del projecte als entorns no locals, fent d'aquest una recomanació extensible a altres projectes.

En un món de serveis que tendeix a l'automatització, aquest projecte és un pas petit però necessari. No és el primer pas, però tampoc serà l'últim.

13. Bibliografia i referències

[AKeller] Keller A. and Ludwig H. (2003): The WSLA Framework : Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, 11, Issue 1, 57-81.

[AKeller2] Keller A. and Ludwig H. (2002): Defining and monitoring service level agreements for dynamic e-business. In: *Proc. of the 16th USENIX System Administration Conference, no. November*.

[AMichlmayr] Michlmayr A.; Rosenberg F.; Leitner P. et al. (2009): Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In: ACM (Hg.): *Proc. of the 4th International Workshop on Middleware for Service Oriented Computing, MWSOC '09*. S. 1-6.

[AKertesz] Kertesz A.; Kecskemeti G.; Oriol M. et al. (2013): Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach. *Journal of Grid Computing*, 699-720.

[BASchroeder] Schroeder B.A. (1995): On-Line Monitoring: A Tutorial. *Computer*, 28, 6, 72-78.

[CWeis] Weisert C.: Waterfall methodology: there's no such thing! *http://www.idinews.com/waterfall.html*, visitada el 21 de febrer de 2013.

[Decker] Gero Decker M.W. (2007): Local Enforceability in Interaction Petri Nets. *Lecture Notes in Computer Science*, 4714, 305-319.

[FowlerAP] Fowler M. (1997): Analysis Patterns: Reusable Object Models. *Addison-Wesley Professional*.

[FowlerEAA] Fowler M.; Rice D.; Foemmel M. et al. (2002): Patterns of Enterprise Application Architecture. *Addison Wesley.*

[GoF] Gamma E.; Helm R.; Johnson R. et al. (1994): Design Patterns: Elements of Reusable Object-Oriented Software. *Addison-Wesley.*

[IEEE830] 830-1993 A.S. (1993): IEEE Recommended Practice for Software Requirements Specifications.

[ISO8402] ISO I. (1994): ISO 8402:1994 - Quality management and quality assurance - Vocabulary.

[iStarWiki] (2011): i* wiki (iStarWiki.org). *http://istarwiki.org/tiki-index.php?page=iStarQuickGuide*, visitada el 2 de desembre de 2013.

[Ludwig] Ludwig H.; Keller A.; Dan A. et al. (2003): A Service Level Agreement Language for Dynamic Electronic Services. *Electronic Commerce Research*, 3, Issue 1-2, 43-59.

[Michelson] Michelson B.M. (2006): Event-driven architecture overview. *OMG. http://www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf*.

[MPalacios] Palacios M.; Garcia-Fanjul J.; Tuya J. et al. (2010): A Proactive Approach to Test Service Level Agreements. In: *Fifth International Conference on Software Engineering Advances*.. S. 453–458.

[Oriol] Oriol M. (2009): Quality of Service (QoS) in SOA Systems. A Systematic Review.

[PGPSI] Prats D. (2011): Diapositives de Planificació i Gestió de Projectes de Sistemes Informàtics.

[QWang] Wang Q.; Shao J.; Deng F. et al. (2009): An Online Monitoring Approach for Web Service Requirements. *IEEE Transactions on Services Computing*, 2, Issue: 4, 338-351.

[SALMon] Oriol M.; Marco J.; Franch X. and Ameller D. (2008): Monitoring Adaptable SOA-Systems using SALMon. In: *Workshop on Service Monitoring, Adaptation and Beyond (Mona+),* 19-28, June 2008.

[SALMonADA_JCR] Müller C.; Oriol M.; Franch X.; Marco J; Resinas M.; Ruiz-Cortés A.; Rodríguez M. (2012): Comprehensive Explanation of SLA Violations at Runtime. In: *Services Computing, IEEE Transactions on.* IEEE, December 2013.

[SALMonADA_PESOS] Muller C.; Oriol M.; Rodriguez M.; Franch X; Marco J; Resinas M; Ruiz-Cortés A. (2012): SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents. In: *2012 ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS)*. S. 43-49.

[Vol] (2012): Volere: Requirements Specification Template. *http://www.volere.co.uk/template.htm*, visitada el 13 de febrer de 2013.

[WSQM] Open O.: Quality Model for Web Services v2.0. *http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc*, visitada el 5 de desembre de 2012.

[Yu] Yu E. (2011): i* Intentional STrategic Actor Relationships modelling - istar. *http://www.cs.toronto.edu/km/istar/*, visitada el 2 de desembre de 2013.

[Zeng] Zeng L.; Lei H. und Chang H. (2007): Monitoring the QoS for Web Services. In: *ICSOC 2007, Lecture Notes in Computer Science, vol. 4749*.. S. 132-144.