UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Explotació d'una base de dades de grafs per a l'anàlisi d'impacte en l'enginyeria de requisits

Projecte Final de Carrera

Realitzat per **Guillem Medina Martínez**

Dirigit per **Patricio Orlando Letelier Torres**

València, Juliol de 2011

Agraïments

Vull expressar els meus sincers agraïments als responsables de l'empresa ADD INFORMATICA sense els quals no s'hauria portat a terme aquest projecte. A Patricio, el meu director de pfc, que m'ha donat aquesta oportunitat i que ha confiat en mi durant tot este temps. Als becaris del grup d'investigació que han aconseguit que inclús les tasques més dures foren senzilles amb el seu suport i companyerisme, en especial a Maria amb la que he treballat conjuntament durant tot este temps i que m'ha ajudat amb la seua experiència i consells.

També voldria dedicar este projecte als meus pares i *abuelos* que amb el seu amor incondicional i el seu gran esforç, m'han anat guiant en la vida i han aconseguit que estiga avui entregant esta memòria. Moltes gràcies, no seria qui soc sense vosaltres.

Als meus amics i amigues, i a la meua xiqueta que m'han animat i recolzat quan les forces més flaquejaven i m'han ajudat a desconectar en els moments que més ho necessitava. I en especial als amics de la universitat que m'han acompanyat durant tots estos anys de carrera, i amb qui he compartit grans dificultats però tambe rises màximes. Vosaltres sou els que sabeu exactament el que hem passat, i sense els quals la meua experiència universitària no haguera sigut el mateix.

Per últim voldria fer una menció especial a les mosques i mosquits que m'han visitat estos mesos i que m'impedien mantindre la concentració necessària per poder escriure esta memòria.

Índex general

1 Introducció

1.1 Motivació

Segons la definició de termes de la norma ISO 8402 [1] el concepte de traçabilitat es defineix com *l'aptitud per rastrejar la història, l'aplicació o la localització d'una entitat mitjançant indicacions registrades*. En l'àmbit del desenvolupament software aquest concepte està directament relacionat amb la traçabilitat dels requisits, documentant la seua cronologia i les seues relacions amb altres entitats.

La traçabilitat afecta nombroses activitats dins l'àmbit de l'enginyeria del software; des de la selecció dels test de regressió passant per la validació dels requeriments, fins l'anàlisi d'impacte de canvis. És precisament aquesta última una de les principals motivacions per mantindre traçabilitat en un producte software.

L'anàlisi d'impacte de canvis de software es defineix com *la determinació de potencials efectes d'un sistema concret com a resultat d'un canvi en el software* [2], és a dir que donada una suposada modificació en la definició d'un sistema i abans que el canvi es produïsca, es pot preveure el seu impacte, directe i indirecte, sobre els diferents elements que conformen el software proporcionant més informació en el procés de desenvolupament i simplificant la tasca posterior de manteniment.

Per tal que l'anàlisi no siga escassa i poc satisfactòria s'ha de recollir un gran volum d'informació i mantindre les relacions de traçabilitat actualitzades. Açò pot suposar un important esforç que moltes vegades no es veu rendibilitzat a causa de mecanismes d'explotació molt rudimentaris. Tot i això, els avantatges que aquesta tècnica ens aporta son clars i si aconseguirem superar els impediments inicials, ens aportaria una eina molt útil per integrar en les nostres metodologies de treball.

En aquest projecte es presenta una nova tècnica de detecció automàtica de les relacions d'interdependència entre requisits on s'intenten solucionar els majors reptes detectats, és a dir, el recull i el manteniment d'aquestes relacions, integrant-ho dins la metodologia i ferramenta de desenvolupament TUNE-UP, on es registraran de manera

automàtica i seran posteriorment explotades indicant a l'equip de desenvolupament quins elements es poden veure afectats pel canvi proposat. La infraestructura elegida per emmagatzemar les relacions ha sigut una base de dades de grafs, ja que ens dóna una gran flexibilitat i expressivitat per modelar el nostre entorn de treball.

Encara que la nostra visió solament inclou elements de tipus requisit o proves d'acceptació, aquesta tècnica és extrapolable a qualsevol procés on puga ser útil conèixer l'impacte dels canvis sobre un determinat producte.

Aquest projecte final de carrera s'ha realitzat en el context d'una beca de col·laboració, pertanyent a un conveni universitat-empresa, amb una PIME de desenvolupament software que comercialitza un ERP per al sector sociosanitari. L'objectiu del treball era aportar un mòdul per a l'anàlisi d'impacte dels requisits i integrar-ho dins l'actual ferramenta de suport a la metodologia TUNE-UP ja implantada. El treball realitzat ha estat implementat seguint les restriccions i necessitats específiques de l'empresa per tal de facilitar la tasca als seus analistes, programadors i testers aportant-los la màxima informació addicional que els permet optimitzar els processos, reduir errors i disminuir costos.

1.2 Objectius

Els objectius específics d'aquest projecte han estat:

- Estudi del paradigma de bases de dades orientades a grafs.
- Disseny d'una base de dades de grafs per representar el model en el qual es basa la nostra ferramenta d'anàlisi d'impacte.
- Creació de la infraestructura necessària per tal de suportar a l'anàlisi d'impacte sobre proves d'acceptació així com un element de comunicació externa.
- Definició d'una interfície per interaccionar i visualitzar els resultats de la ferramenta.
- Implementació d'un gestor d'entitats del model.
- Integració del mòdul en un entorn de gestió de requeriments basat en proves d'acceptació per al desenvolupament software amb metodologies àgils (TUNE-UP).
- Validació de l'anàlisi en projectes reals.

1.3 Estructura del treball

En esta secció es presenta de forma resumida l'estructuració del treball en capítols:

El capítol 1 inclou una introducció del projecte on es presenta la motivació, els principals objectius i un resum de la seua estructura.

El capítol 2 esta centrat en la metodologia TUNE-UP i en general en els processos de desenvolupament Test-Driven Requirements Engineering (TDRE), que es tracta d'una proposta de gestió de requisits sobre el marc Test-Driven Development (TDD) basat en proves d'acceptació. El projecte s'introdueix dins d'un gestor de requisits de TUNE-UP i aporta una nova ferramenta addicional per a seguir aquesta metodologia.

El capítol 3 explica el funcionament bàsic de les bases de dades orientades a grafs (BDOG) que són la infraestructura elegida per donar suport al projecte, els seus principals avantatges i inconvenients, i una xicoteta comparativa amb les bases de dades relacionals. Es farà un breu resum de les propostes actuals de BDOGs centrant-nos en Neo4j que ha sigut l'elegida.

El capítol 4 explica la infraestructura creada per tal de donar suport al recull, manteniment i gestió dels termes de domini que s'utilitzaran posteriorment per a l'anàlisi d'impacte.

En el capítol 5 es mostra el mòdul d'explotació de l'anàlisi d'impacte i s'il·lustra amb un exemple que engloba tots el passos del procés d'anàlisi.

El capítol 6 conclou el projecte amb les conclusions extretes a partir de la nostra proposta, les dificultats abordades i el treball futur que ha sorgit en realitzar-lo.

2 Metodologia TUNE-UP i TDRE

2.1 Introducció a l'anàlisi d'impacte de canvis

El principal propòsit de l'anàlisi d'impacte de canvis és previndre els efectes que un possible canvi tindrà sobre el producte, i aconseguint informació sobre les parts afectades i de quina forma es veuen afectades abans de produir-se la modificació.

Aquests canvis poden anar des d'una lleu modificació d'un tipus de dades d'un camp d'una interfície d'usuari (que veurà afectades altres parts de la IU que continguen aquest mateix camp, així com el camp associat de la base de dades y la lògica interna de la capa de dades); fins a canvis en la terminologia del model d'entitats de l'aplicació que supose grans canvis en totes aquelles parts del projecte relacionades amb aquesta entitat, per exemple un software de gestió o ERP (Enterprise Resource Planning) comercial es pot veure afectat globalment si el concepte *factura* es veu modificat afegint alguna informació addicional obligatòria.

Per tal de poder identificar aquestes associacions, necessitarem informació sobre les relacions entre els artefactes del producte, tot definint el concepte artefacte com qualsevol element involucrat en el producte: terme del model d'entitats, especificació del producte, parts del codi font o elements de la base de dades.

L'anàlisi d'impacte és especialment útil durant les fases de manteniment però pot ser de gran ajuda durant totes les fases del procés de desenvolupament.[4] Els resultats aportats per l'anàlisi d'impacte recolzen decisions en diferents fases i àmbits.

En fases de planificació, ajuda a entendre la complexitat del canvi a realitzar i per tant facilita la tasca d'estimar l'esforç i l'avaluació de la viabilitat o conveniència d'aquest. Durant les fases d'anàlisi i disseny ens aporta informació addicional per especificar el canvi d'una manera completa i coherent amb les especificacions ja definides que es voran afectades. Per últim per als àmbits d'implementació i testeig ens proporciona una visió primerenca que resulta útil per tindre una aproximació dels canvis en el codi o en les possibles modificacions de les proves de regressió a aplicar.

Com podem vore en la Figura 1, les tècniques d'anàlisi d'impacte es poden classificar en tres tipus: *Traçabilitat, Dependència i Experimental.*[5]

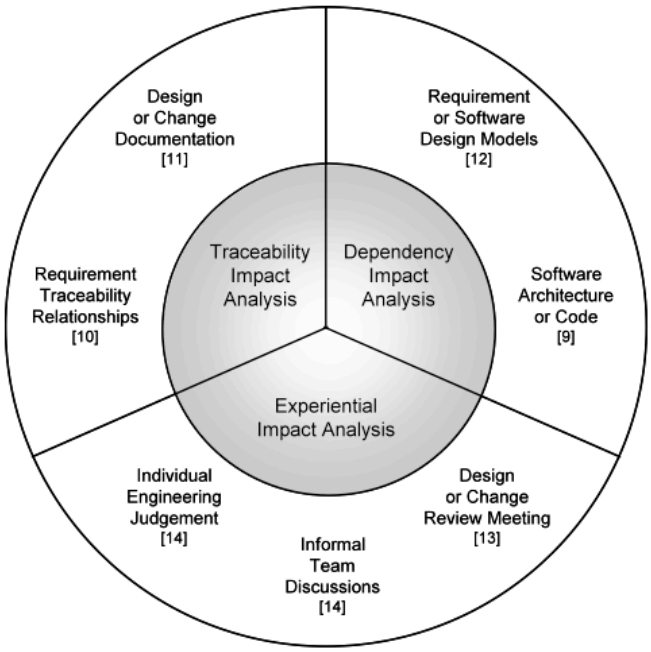

FIGURA 1: Tipus de tècniques d'anàlisi d'impactes [6]

Seguint les definicions i principals característiques d'aquestes tècniques donades per articles acadèmics: [6] [7] [8] [9]

> *L'anàlisi d'impacte de traçabilitat* utilitza l'assignació de requisits del producte amb les seues respectives especificacions i dissenys. Aquestes associacions s'anomenen relacions de traçabilitat, i es produeixen entre els artefactes que especifiquen el producte. Donat un canvi en els requisits del producte, les relacions de traçabilitat emmagatzemades ens mostraran la propagació d'aquesta modificació envers la resta d'artefactes que conformen el nostre sistema.
>
> *L'anàlisi d'impacte de dependència* representa una altra manera d'investigar les conseqüències d'un canvi però centrant-se en relacions entre elements del codi: variables, lògica, mòduls, etc. Aquesta tècnica ens aporta anàlisis molt detallades davant els canvis, que poden ser quasi automàtics però que es focalitzen més a baix nivell que a modificacions de caire general.

L'anàlisi d'impacte experimental es tracta d'un estil complementari als anteriors on l'element principal per trobar l'impacte dels canvis és l'experiència individual o col·lectiva de l'equip de desenvolupament basada en coneixements que no han sigut capturats en cap model o base de dades.

Per tal d'aconseguir una ferramenta sistemàtica i global no podem basar-nos en les anàlisis de dependència ja que aquests solament son realment útils per a petits canvis a nivell de codi i no ens aportarien cap millora en altres àmbits con el de planificació o anàlisi. A més, per la seua pròpia naturalesa no sistemàtica, no podem dependre de les anàlisis experimentals ja que estan basades en l'experiència humana en grups de desenvolupament cada vegada més canviants. Per tant les anàlisis d'impacte basades en traçabilitat son les úniques que ens aporten totes les característiques que necessitem.

El principal problema de les tècniques actuals basades en traçabilitat és l'alt esforç invertit en l'especificació i manteniment de les seues relacions, creant grans obstacles per tal d'aconseguir una ràtio positiva entre l'esforç invertit i l'aprofitament de la informació extreta.[3] Les tècniques tradicionals d'explotació com ara les matrius de traçabilitat es basen en una representació gràfica on es creuen els diferents artefactes amb possibles relacions amb altres artefactes, obtenint informació massa general que necessita treball manual addicional per tal de ser realment útil.

Per a minimitzar aquest esforç i fer l'anàlisi viable i profitosa es va decidir integrar el procés dins la metodologia de desenvolupament TUNE-UP basada en proves d'acceptació que ens ofereix el suport necessari per identificar i mantindre les relacions de manera automatitzada, seguint el curs de la pròpia metodologia. Ha de quedar clar que la nostra motivació no era definir una anàlisi d'impacte global per a qualsevol tipus d'artefacte, si no determinar l'efecte que un canvi en un requisit podia tindre sobre la resta. Es per açò que aquesta integració amb una metodologia enfocada en la gestió de requisits ens evita detectar en fases tardanes del desenvolupament, especificacions incompletes o inconsistents.

2.2 TUNE-UP

TUNE-UP és una metodologia i ferramenta per a la gestió àgil de projectes de desenvolupament i manteniment software. TUNE-UP ha estat refinada en el treball diari d'una PIME de l'àmbit del desenvolupament software des de fa més de 6 anys intentant aconseguir una mescla entre distints aspectes de les metodologies àgils i tradicionals oferint una alternativa que englobe ambdues. TUNE-UP es caracteritza fonamentalment per la combinació dels següents aspectes: [12]

- **Model iteratiu i incremental** per al desenvolupament i manteniment. El treball es divideix en unitats de treball que son assignades a cada versió del producte. Seguint una de les idees àgils bàsiques, els cicles de desenvolupament son curts, entre 3 i 6 setmanes, depèn de la complexitat del projecte.
- **Workflows flexibles** per a la coordinació del treball associat a cada unitat de treball. Els productes segons les seues característiques tenen disponibles un conjunt de *workflows* per assignar a la unitat de treball corresponent. Estos *workflows* son flexibles ja que poden anar avant, endarrere, en paral·lel,...
- **Procés de desenvolupament dirigit per proves d'acceptació.** La definició d'una unitat de treball és bàsicament la especificació del seus requisits en forma de proves d'acceptació. Seguint el model TDRE que explicaré en el punt següent, aquetes es transformaran en el punt central del desenvolupament que implicarà a tots els agents del procés (analistes, programadors i testers).
- **Planificació i seguiment centrats en la gestió de temps.** Les activitats dels workflows associades a cada unitat de treball tenen un registre de seguiments amb els temps estimats o registrats per cada agent. Aquest conjunt d'informació ens permet aconseguit una millor planificació i previsió de les futures versions del producte.

Tots estos aspectes han sigut inclosos dins la ferramenta de suport de TUNE-UP estructurats en una sèrie de mòduls independents.

Inicialment tenim el **Planificador Personal** (PP) que és el mòdul encarregat de presentar d'una forma ordenada el treball que té cada agent per a una determinada versió. Estos treballs estan agrupats seguint les activitats dels diferents *workflows* i poden ser ordenades seguint criteris d'estimació de temps, prioritats (funcionals,

comercials, de desenvolupament,...) o per l'esforç a priori assignat. Després de decidir quina activitat i unitat de treball es realitzarà accedirem al **Gestor d'Unitats de Treball** (GUT).

El GUT és el mòdul que centra tota la informació relacionada amb una determinada Unitat de Treball. En ella es mostren totes les seues característiques com ara identificadors, software al que pertany, projecte, versió, tipus, descripció, etc. i ens aporta una sèrie d'opcions i ferramentes que podran utilitzar els diferents agents implicats en el desenvolupament de la unitat de treball. Entre estes opcions podem destacar la seua gestió del temps i de les activitats que son l'eix fonamental de la gestió de les unitats de treball, i defineixen el seu desenvolupament com un conjunt de salts entre les activitats del *workflow,* associant seguiments de temps a cada una d'aquestes. A més també cal destacar ferramentes específiques per a programadors i testers , un sistema de intercomunicació d'agents en forma de peticions associades a una unitat de treball o a una prova d'acceptació i opcions de planificació.

El **Planificador de Versions** (PV) és el mòdul encarregat de gestionar els productes, les seues versions i els *workflows* disponibles per a cada producte, així com els agents per defecte en cada activitat d'un *workflow* o la realització del seguiment de la versió actual.

Per últim tenim el **mòdul de Gestió de Requisits** (GR). Com s'ha dit anteriorment la metodologia TUNE-UP esta dirigida per proves d'acceptació que defineixen els requisits establerts per a cada unitat de treball, i es precisament este apartat de la ferramenta l'encarregat de gestionar-les.

2.3 TDRE i el mòdul de gestió de requisits

El desenvolupament software seguint la metodologia TUNE-UP inclou un innovador sistema de gestió de requisits anomenat TDRE (Test-Driven Requirement Engineering), basat en l'especificació de requisits mitjançant proves d'acceptació (PAs). Aquest sistema és una extrapolació dels conceptes bàsics del TDD (Test Driven Development) i més concretament de la idea que tot el desenvolupament software ha de girar al voltant de les proves , és a dir, en TDRE la estratègia es implantar una "cultura" de

proves basada en l'aprofitament d'aquestes com a fil conductor del procés de desenvolupament.[10]

Tornant a la idea fonamental dels requisits especificats mitjançant PAs, en TUNE-UP el client amb l'ajuda de la figura de l'analista defineix les unitats de treball en termes de PAs. Posteriorment el programador escriu el codi necessari per tal de satisfer les proves, i per últim el tester estableix les combinacions de dades necessàries per tal de comprovar que el producte compleix l'especificació descrita i acordada en les proves.

En TUNE-UP l'estructura de requisits es representa amb un graf acíclic dirigit, on els nodes representen contenidors de PAs per tal de tindre organitzat i jerarquitzat el conjunt de requisits establerts. Un canvi en el comportament del producte ve definit per una unitat de treball que afecta a un o més nodes de l'estructura de requisits del producte, afegint, modificant o eliminant PAs.[10]

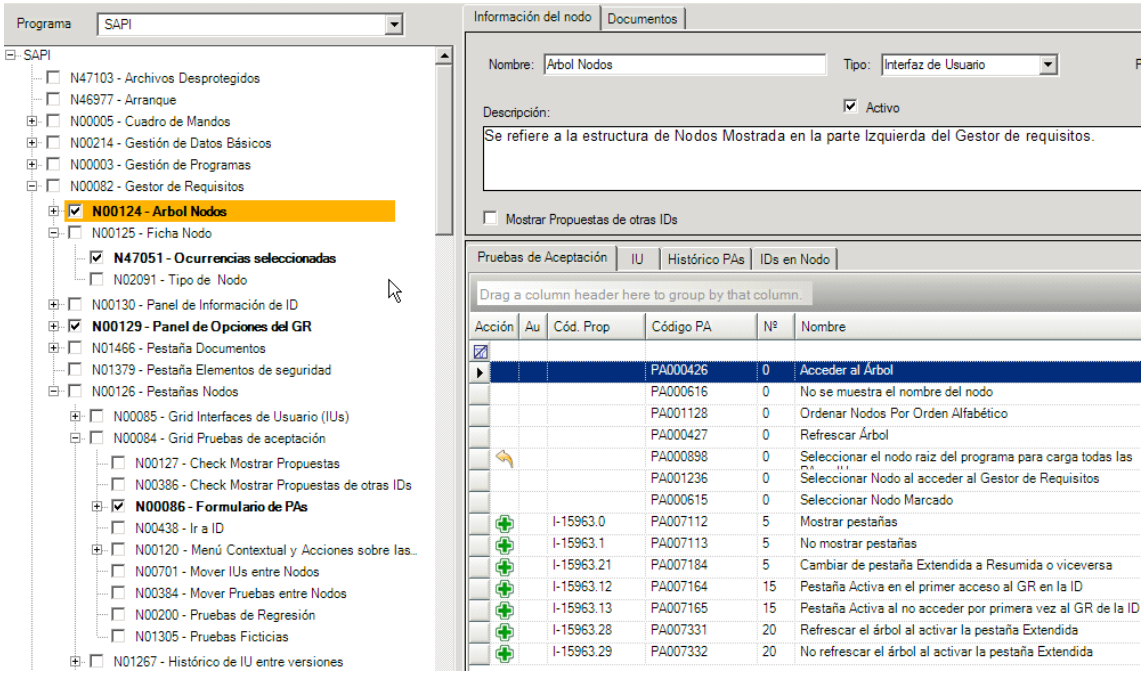

FIGURA 2: Gestor de Requisits d'una ferramenta de suport TUNE-UP

Com es pot vore en la imatge superior, en la ferramenta de suport a la metodologia l'estructura de requisits està definida per un treeview (esquerra) que en aquest cas està associat a una unitat de treball, on apareixen marcats amb *checks* el nodes contenidors de requisits que s'han vist afectats pels canvis produïts en aquest context. En seleccionar un dels nodes afectats, el grid de la dreta ens mostra totes les PAs

contingudes en el node i es reflecteixen els canvis produïts en la unitat de treball amb una sèrie d'icones associades a accions com nova prova, modificació d'una prova, prova de regressió o eliminació d'una prova.

2.3.1 Proves d'acceptació i model d'entitats

Una PA te com a propòsit general demostrar al client el compliment parcial o total d'un requisit del producte. Una PA descriu un escenari d'execució o d'ús del sistema des de la perspectiva del client i per tant podem dir que el conjunt de PAs conforma el comportament esperat del software en forma de requisits funcionals i no funcionals. A continuació es presenta un exemple de la definició de la PA "Afegir un usuari existent com a amic ", este exemple esta dins del context d'una xarxa social com Facebook, Twitter, aNobii, Lastfm, etc. on el principal element per interconnectar persones és per enllaços d'amistat amb altres amb usuaris registrats.

Condició

Ser un usuari registrat

Iniciar sessió amb el compte (*username/password*)

Accedir a un perfil d'usuari existent que no estiga entre els amics.

Passos

Sol·licitar amistat

Resultat esperat

La pàgina informa de que la sol·licitud esta processant-se

S'envia una petició d'amistat al usuari.

En TUNE-UP la definició d'una PA es composa de tres apartats principals. Les *condicions* representen l'estat actual de la base de dades abans de produir-se l'acció així com la navegació necessària inicial, els *passos* denoten pas a pas les accions que s'han de fer per reproduir el comportament definit en la PA i els *resultats esperats* son precisament allò que avalua el correcte funcionament del software. Esta definició s'escriu en llenguatge natural dins de la ferramenta de suport referenciant entitats del model de domini del nostre producte. Aquesta característica principal és la que ens

facilita la inserció de l'anàlisi d'impacte de les Pas establint relacions automàticament entre Pas que referencien el mateix element del domini.

Una entitat de domini pot contindre propietats, atributs i relacions amb altres entitats, en el nostre petit exemple podríem definir un model de domini amb les següents entitats: Un usuari registrat es podria representar amb l'entitat compte, esta té una sèrie de propietats com ara nom i cognoms. Per altre costat esta entitat té associats dos atributs *username* i *password* amb les seues propietats característiques. Finalment tenim la entitat o concepte d'amistat entre usuaris i les seues propietats naturals, com per exemple en ser una relació entre dos usuaris i ser una relació bidireccional i la entitat petició relacionada amb la d'amistat però amb la gran diferència de ser unidireccional.

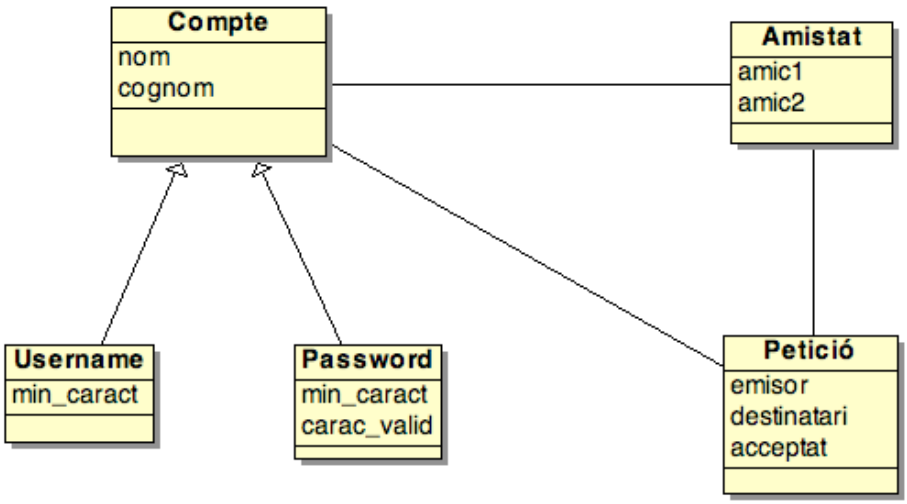

FIGURA 3: Model del domini de l'exemple

En seccions posteriors d'aquesta memòria es retornarà a l'exemple exposat i es completarà i s'especificarà amb més detall per tal de mostrar el complet procés intern de la anàlisis d'impactes dels requisits.

3 Bases de dades orientades a grafs

A continuació es presenta el concepte de base de dades orientades a graf (BDOG), mostrant els seus avantatges i inconvenients, la seua base teòrica, les principals diferències respecte a les bases de dades relacionals, així com les diferents propostes existents en el mercat, centrant-se en Neo4j que es l'opció finalment elegida per a la implementació del projecte.

3.1 Introducció a BDOG

Quan parlem de bases de dades ens referim a conjunts de dades organitzades i estructurades típicament de manera digital, a les que es pot accedir mitjançant un sistema de gestió de bases de dades que facilita l'obtenció, actualització i esborrat d'informació. Les bases de dades segueixen models de dades que son representacions abstractes de la informació en allò referent al seu emmagatzemament, organització i manipulació.

En l'actualitat la major part de bases de dades segueixen el model relacional basat en taules d'esquema fix formades per tuples d'informació, que es relacionen amb altres taules. Les consultes sobre la informació es fan mitjançant el llenguatge SQL i per tal de relacionar les dades de diferents taules s'utilitzen les operacions Join que tendeixen a ser molt costoses amb grans quantitats de dades. No obstant això, existeixen altres models que tot i no tindre sistemes de gestió tan avançats i complets com el relacional, poden ser més adequats per a determinats àmbits.

En el projecte s'ha decidit utilitzar un model de base de dades orientat a grafs ja que es tracta de la manera més natural i intuïtiva de representar el sistema de relacions de traçabilitat. Les BDOG emmagatzemen la informació utilitzant una de les estructures de dades més comuns i estudiades del món de la informàtica i les matemàtiques: els grafs.

Si seguim la seua definició matemàtica tenim que un graf **G** es una parella ordenada **G:= (V,E)**, on **V** és un conjunt de vèrtex o nodes i **E** és a un conjunt d'arestes o línies que representen relacions binaries entre dos elements de **V**. Aquesta definició tan

bàsica ens dóna una idea d'una representació basada en punts relacionats amb altres punts, per exemple *a* i *b*, mitjançant arcs *R* que representen les relacions d'adjacència entre estos elements, que podem definir de manera més compacta com *aRb*. Direm que dos vèrtex son adjacents si existeix la relació *aRb* en el conjunt *E*.

Depenent de les característiques del graf aquest pot ser dirigit o no dirigit depenent si les seues arestes tenen sentit. A més, pot ser cíclic o acíclic depenent de l'existència de camins auto conclusius o no, definint un camí de llargada *n* com a un conjunt de vèrtex $v_0,v_1,v_2,...v_n$ on existeix una aresta $v_{i-1}Rv_i$ per cada $i=1,2,3...n$. Altres propietats a tindre en compte de grafs és que poden ser ponderats, cada node del graf pot contindre un valor que represente un pes, cost, llargada,... poden ser etiquetats de manera que cada node i fins i tot cada aresta del graf siguen distingibles i puguen tindre bucles, es defineix un bucle com una aresta on els seus dos extrems són el mateix node. Si un graf té bucles s'anomena multigraf en cas contrari graf simple.

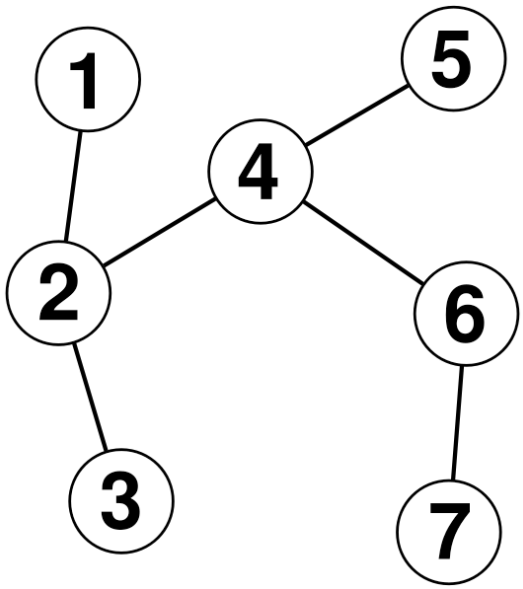

FIGURA 4: Graf etiquetat, simple, acíclic i no dirigit

La Figura 2 mostra la representació gràfica del graf següent:

V:= {1,2,3,4,5,6,7}

E:= {{1,2},{2,3},{2,4},{4,5},{4,6},{6,7}}

Com ja he comentat anteriorment aquesta estructura és una de les més estudiades dins el món de la matemàtica i la computació des del 1736, quan Leonhard Euler va publicar el seu article científic sobre el problema dels ponts de Königsberg on utilitzant una abstracció en forma de graf per representar les diferents parts de la ciutat i els ponts que les interconnectaven, va concloure que no era possible recórrer els 7 ponts existents sense recórrer-ne algun més d'una vegada. Aquest article va establir l'inici de l'estudi sobre les propietats i operacions dels grafs que actualment es coneix com la teoria de grafs. Parlar de la teoria de grafs no és l'objecte d'este projecte final de carrera, però s'ha d'entendre la seua importància i maduresa per assimilar la solidesa teòrica que hi ha darrere de les BDOG.

Els grafs son una potent ferramenta que ens permeten modelar una gran quantitat de sistemes, la seua utilització i estudi al llarg dels últims tres segles ens han permés estudiar problemes matemàtics que actualment trobem representats en molts àmbits. Buscar el camí més curt entre dos nodes, trobar el cicle hamiltonià o eulerià d'un graf, el problema del viatger o qualsevol qüestió relacionada amb fluxos son solament una xicoteta part dels problemes clàssics que ens han aportat una gran quantitat d'algorismes utilitzats en aplicacions actuals. Es per açò que no es estrany vore implementacions basades en Kruskal, Dijkstra, Ford-Fulkenson, algorismes A* o qualsevol algorisme de recorregut en aplicacions actuals de GPS, control de flux de xarxes, distribució de circuits electrònics o xarxes socials.

Les BDOG utilitzen tota aquesta experiència prèvia per crear un model de bases de dades robust i flexible gracies a les propietats innates dels grafs on, depenent de les diferents implementacions, s'han anat afegint mecanismes utilitzats actualment en les bases de dades relacionals com el control de les transaccions i la concurrència, així com mecanismes de persistència, índex, recuperació de dades o fins i tot llenguatges de tipus *query*. A més, per tal d'estendre la informació que contenen, s'han modelat els nodes com a un conjunt de parells *{key, value}* anomenats propietats on emmagatzemar les característiques del node.

A continuació exposaré d'una manera més exhaustiva els principals avantatges que aquest tipus de bases de dades ofereix respecte altres, mostrant en quins àmbits

destaca la seua potència i fent una xicoteta comparativa amb el model més utilitzat, les bases de dades relacionals.

3.2 Comparativa i principals avantatges

Com ja s'ha exposat en el punt anterior, les BDOG ens aporten un nou model de dades completament diferent al model relacional predominant en el món de la informàtica. Les seues diferències ens poden resoldre problemes i reduir el cost computacional en determinants àmbits, destacant especialment en aquells on l'explotació de les relacions siga predominant. Tot i que una comparativa directa entre models no ens pot mostrar de manera objectiva quin es més òptim, sí que podem fer una comparativa general entre les seues propietats per vore quines s'adeqüen millor al nostre sistema. Les principals característiques que ens aporten les BDOG son:

- **Flexibilitat a canvis en l'estructura de dades**. Els canvis en el model produeixen modificacions en el conjunt de nodes, relacions o propietats que es corresponen amb les seues operacions bàsiques i permeten una ràpida evolució. En les Bases de Dades Relacionals (BDR) les modificacions sobre el model es corresponen en modificacions de les estructures fixes de les taules, les claus alienes corresponents , així com l'actualització de les dades actuals i de les restriccions i possibles problemes que estes puguen aportar.
- **Conjunt de dades semi-estructurades**. La definició dels models no és prefixada com ho son les taules de les BDR, en canvi, els nodes i relacions contenen pocs camps obligatoris (identificadors), als que s'afegeixen propietats (parells *{key, value}*) en funció de les necessitats del model i no per la seua rigidesa. Aquesta característica a més de donar-nos una gran flexibilitat alhora de representar el nostre sistema ens millora el cost espacial de la informació a emmagatzemar. També cal destacar que les propietats son de longitud variable, evitant haver de definir el seu volum i que poden combinar valors complexos i plurivalents.
- **Intuïtiu amb models Orientats a Objectes (OO).** Així com en les BDR es necessari definir un *mapping* entre els objectes de la capa de domini i les

taules del model de relacional de persistència, no existeix un desajust inherent entre el model d'objectes i l'orientat a graf, ja els primers poden ser representats directament com a grafs. A més en les BDOG la representació de les dades, els esquemes i les seues consultes estan unificats en la figura del graf.

- **Consultes optimitzades per a l'explotació de les relacions.** Les consultes realitzades en les BDOGs son àmplies, no estan emmarcades per taules com les de les BDRs, evitant així les costoses operacions join, i estan optimitzades específicament per a sistemes fortament interconnectats, recorrent les relacions entre els nodes seguint uns criteris de cerca i donant com a resultat un subgraf amb els nodes o relacions que els hagen satisfet. A més ens aporten un nou nivell d'expressivitat ja que podem recórrer les dades de manera jeràrquica i ens faciliten la navegació a traves dels distints nivells de profunditat de les relacions.

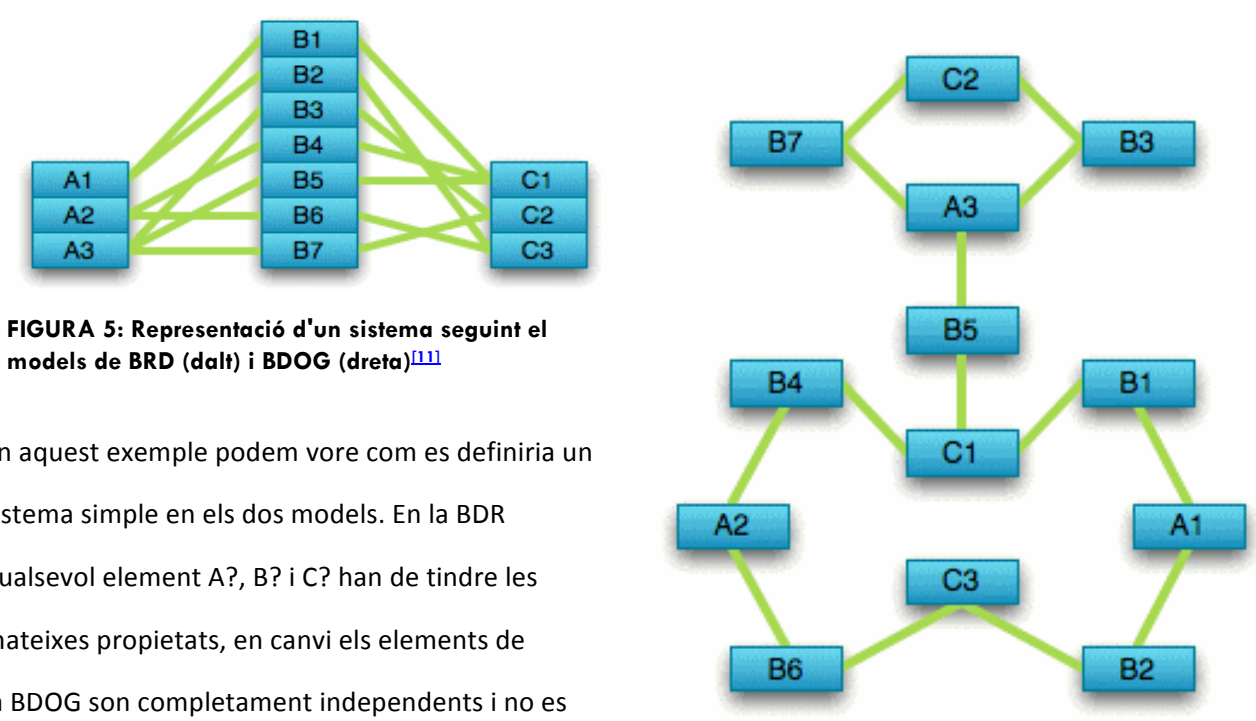

FIGURA 5: Representació d'un sistema seguint el models de BRD (dalt) i BDOG (dreta)[11]

En aquest exemple podem vore com es definiria un sistema simple en els dos models. En la BDR qualsevol element A?, B? i C? han de tindre les mateixes propietats, en canvi els elements de la BDOG son completament independents i no es classifiquen seguint cap tipus.

Les BDR per altre costat poden ser més eficient que les BDOG en els casos on els elements del sistema estiguen dèbilment relacionats, així com en totes aquelles

consultes que impliquen poques taules amb independència del nombre de registres que continguen. D'altra banda, també cal destacar la maduresa i estabilitat dels sistemes de gestió de bases de dades relacionals que aporten ferramentes addicionals al propi model que milloren la usabilitat de l'usuari. Índex, triggers, mecanismes de modelatge o generació automàtica de consultes, per exemple, son eines que hi trobem de manera habitual i que normalment encara no estan implementades en els sistemes de gestió de BDOGs. Un dels principals problemes que he enfrontat al realitzar el projecte està directament lligat a aquest punt, ja que es tracta d'una tecnologia relativament nova que encara no esta àmpliament estesa.

Com hem dit amb anterioritat les BDR no son ni millor ni pitjor que les BDOG però cal saber elegir correctament quina serà la més apropiada per a modelar el nostre sistema. En el nostre cas hem elegit un model de bases de dades seguint els següents criteris:

- El nostre sistema es pot modelar de manera natural, elegant i simple com un graf on els nodes son PAs i Elements de Domini (EDs) interconnectades entre elles mitjançant diferents tipus de relacions a explotar posteriorment per a l'anàlisi d'impacte.
- Modificació incremental del domini. Els principals objectes a considerar en l'anàlisi d'impacte en la versió inicial son les PAs i els EDs, però s'espera que l'anàlisi es puga estendre per incloure altres elements con nodes, clients o inclús classes del codi. Com hem dit anteriorment les BDOG ens aporten gran flexibilitat envers canvis de l'estructura.
- Heterogeneïtat dels diferents elements del domini. Els EDs poden ser molt diversos i cada un serà representat per un conjunt distint de propietats. Per exemple l'element *factura* no tindrà les mateixes característiques que l'ED *client*. A diferència de les BDR on no hi ha una manera simple de representar aquestes diferències (a no ser que es malgaste espai amb camps buits), en les BDOG els nodes no estan tipats i poden tindre diverses conjunts de propietats de manera natural
- Importància de les relacions en el nostre sistema. El nostre sistema es vol dissenyar principalment per tal d'explotar les relacions de traçabilitat entre els

artefactes, les BDOG estan especialment orientades a la navegació d'aquestes relacions.

Una vegada escollit el model de bases de dades, el següent pas tracta d'elegir concretament quina de les bases de dades existents en el mercat s'adequa millor a les nostres necessitats i restriccions.

3.3 BDOGs existents

Tot i ser un enfocament relativament nou en l'àmbit de les bases de dades (almenys a la pràctica), en els darrers anys han eixit molts projectes relacionats amb aquest model. Per estudiar quina era la BDOG a utilitzar es va definir una sèrie de criteris basats en les restriccions tecnològiques de l'empresa i la maduresa del projecte de bd.

En l'empresa s'utilitza .Net de Microsoft per desenvolupar software, per tant era important utilitzar alguna BDOG que ens aportara una api de comunicació en C# o almenys algun element de comunicació mitjançant servicis. Un altre aspecte a considerar era la maduresa del projecte, tant d'estabilitat com de completesa, la documentació associada i en general el volum d'usuaris que conformava la comunitat. Per últim també calia centrar-se en el tipus de llicència del producte, preferiblement alguna llicència opensource que ens permetrà utilitzar-la sense cap cost addicional per a l'empresa. Seguint aquests criteris, vàrem estudiar un conjunt d'opcions buscant informació per internet i posant-nos en contacte amb els desenvolupadors. Cal tindre en compte que les BDOG estudiades actualment poden tindre característiques diferents de quan es va prendre la decisió:

3.3.1 DEX

Es tracta d'un projecte d'investigació dut a terme pel DAta MAnagement group de la universitat politècnica de Catalunya (DAMA-UPC) que es comercialitza per l'empresa Sparsity-Technologies. Dex es una llibreria de gestió de grafs escrita en C++ i Java que aporta una API pública en aquest últim llenguatge per a la seua comunicació i gestió directa, les seues transaccions són parcialment *ACID* ja que solament suporten consistència i aïllament. El seu model de dades consisteix en grafs amb múltiples relacions per node, direccionals, amb nodes i relacions tipades. Té una llicència comercial en funció del volum de la base de dades.

Aquesta opció va ser descartada ja que no oferia cap eina de comunicació alternativa a la API de Java, la documentació era escassa, i a més la llicència suposaria un cost extra per a l'empresa. També cal destacar que no existia una

comunitat real al darrere fent ús d'aquesta que donara suport o que ens mostrara un ús real de la base de dades.

3.3.2 HiperGraph

Aquest és un projecte desenvolupat per l'empresa Kobrix Software i distribuït utilitzant una llicència LGPL. El model de dades utilitzat es un poc diferent de la resta, ja que es tracta d'un model basat en hipergrafs, és a dir que cada aresta pot connectar més de dos nodes. En HiperGraph tots els elements son *àtoms* tipats amb propietats preestablertes que poden estar relacionats amb 0..n elements. El seu sistema de transaccions es parcialment *ACID*, ja que no ens assegura la durabilitat. Esta base de dades et permet treballar amb un entorn local o utilitzant un sistema distribuït basat en protocols P2P. La comunicació amb la BD es fa mitjançant la seua API de Java.

Tot i ser un projecte amb una llicència poc restrictiva per al nostre us amb una comunitat d'usuaris al darrere i basar-se en un model orientat a objectes, va ser descartat per estar en una fase inicial de desenvolupament i oferir exclusivament una API de Java com a element comunicador.

3.3.3 Infogrid

Desenvolupada inicialment per l'empresa NetMesh, actualment es tracta d'un projecte de codi obert distribuït amb una llicència AGPLv3. Infogrid es tracta d'una base de dades de grafs (amb múltiples relacions per node, direccional on cada node i relació pot contindre propietats) implementada en Java, especialment pensada per al desenvolupament d'aplicacions web basades en servicis REST, ja que cada objecte de la base de dades té una url associada seguint els estàndards REST. La gestió i navegació de la base de dades es fa mitjançant la seua API nativa de Java i ens ofereix posteriorment un accés als elements mitjançant servicis web.

Infogrid era un projecte a considerar ja que oferia totes les propietats que necessitàvem del graf i a més es tractava d'un projecte de codi obert, però va ser descartat ja que no vàrem trobar documentació associada respecte a la gestió de les transaccions ni tampoc hi havia una alternativa a la API nativa de Java

respecte a les operacions sobre el graf, tot i tindre un accés als elements mitjançant servicis REST la comunicació pel que fa a les accions era confusa.

3.3.4 Neo4j

Aquesta és la opció finalment elegida per donar suport a la anàlisis d'impacte del requisits. Aquesta BDOG ens aporta totes les propietats que hem enumerat anteriorment i gràcies a ser un projecte de codi obert amb una llicència flexible com és la AGPLv3 suposa l'opció més adequada a les nostres necessitats. En el punt següent es dóna una explicació més detallada de Neo4j amb totes les seues característiques, millores al pas del temps i parlarem de la API que estem utilitzant.

3.3.5 InfiniteGraph

Desenvolupada per una companyia amb el mateix nom, InfiniteGraph es tracta d'una BDOG desenvolupada en Java que ofereix una API molt simple de comunicació en el mateix llenguatge i una capa de persistència especialment optimitzada per al seu escalat i distribució. Es tracta d'un dels projectes més madurs entre les propostes estudiades i ens aporta moltes de les característiques necessàries per a la implementació del nostre mòdul. El seu model de dades esta basat en grafs direccionals on nodes i relacions poden tindre propietats i cada node pot estar relacionat amb 0...n nodes. InfiniteGraph suporta concurrència i aporta un sistema transaccional plenament ACID de sessions, a més conté una col·lecció d'algorismes de cerca de camins, un sistema complex d'indexació i cerca i un complet *framework* de creació de traversals configurable per recórrer el graf.

Tot i ser una de les millors opcions analitzades fins al moment, es va descartar el seu ús en un primer moment, ja que encara que hi havia anunciat un binding per a c# (a dia de avui encara no l'han implementat ni esta programada la seua versió de publicació), en el moment no oferia cap element de comunicació que no fora la API de Java, a més la seua llicència comercial suposaria un cost extra addicional.

3.3.6 AllegroGraph

Aquesta base de dades desenvolupada per l'empresa Franz Inc. utilitza una arquitectura completament diferent a la de la resta. La persistència de la base de dades utilitza l'estàndard RFD de la W3C per modelar el coneixement en *triples* formades per afirmacions de la forma {subjecte, predicat, objecte, graf, triple-id} i utilitza un complex sistema d'índex per tal d'aconseguir eficientment aquestes triples utilitzant cerques basades en *pattern matching*. La base de dades ofereix nativament APIs en SPARQL, Prolog i RDFS++ per codificar les *queries* i a més, com utilitza una implementació de RFD basada en *Sesame 2.0,* ens aporta una interfície REST amb clients ja implementats per a diversos llenguatges on s'inclou el C#.

El principal problema que ens va fer descartar aquesta BDOG es el fet que funciona exclusivament en màquines amb el sistema operatiu Linux ja que actualment tots els servidors de la empresa treballen amb entorns Windows i açò suposaria haver de tindre un servidor dedicat que juntament amb els costos de la llicència suposarien un cost massa elevat per al projecte. A més la seua arquitectura basada en predicats feia que la corba d'aprenentatge s'accentuara allargant el temps de desenvolupament inicial.

3.3.7 GraphDB

GraphDB és una base de dades desenvolupada per Sones que a diferència de la resta esta implementada amb tecnologies .Net i que ofereix una API en llenguatge C# per a la seua gestió. Es tracta d'una BDOG amb característiques similars a HiperGraph que a més aporta un llenguatge propi de *queries* anomenat GQL basat en SQL amb les modificacions necessàries per treballar amb elements d'un graf. Utilitza elements (nodes i relacions) tipats que cal instanciar per representar el nostre model. GraphDB és un projecte open source que utilitza una llicència comercial per la seua distribució i suport.

Aquesta va ser la nostra primera opció ja que ens permetia comunicar-nos amb la base de dades de manera nativa sense necessitat de cap tipus de pont tecnològic com els servicis web. Finalment va ser descartada degut a la poca

documentació associada que hi havia per a la versió open source i la inexistència d'una comunitat d'usuaris al darrere, però sobretot perquè la versió lliure solament permet emmagatzemar la informació en memòria i si es vol un sistema persistent basat en disc cal comprar una llicència comercial d'alt cost.

3.4 Neo4j

Neo4j és una base de dades de grafs desenvolupada en Java per la companyia sueca NeoTechnology, és probablement la base de dades de grafs amb una major comunitat de desenvolupadors al darrere i una major estabilitat i robustesa.

Com altres BDOG ja definides, el seu model de dades esta basat en grafs on tota informació esta representada per nodes, relacions i propietats. Una relació s'encarrega de connectar dos nodes, aquesta pot tindre propietats, direcció i obligatòriament tindrà un tipus que s'utilitzarà posteriorment per a la navegació. Els nodes en canvi no estan tapats i tenen un conjunt de propietats associat exclusivament a la instància del node. En definitiva el model de dades de Neo4j representa la informació com una xarxa fortament interconnectada.

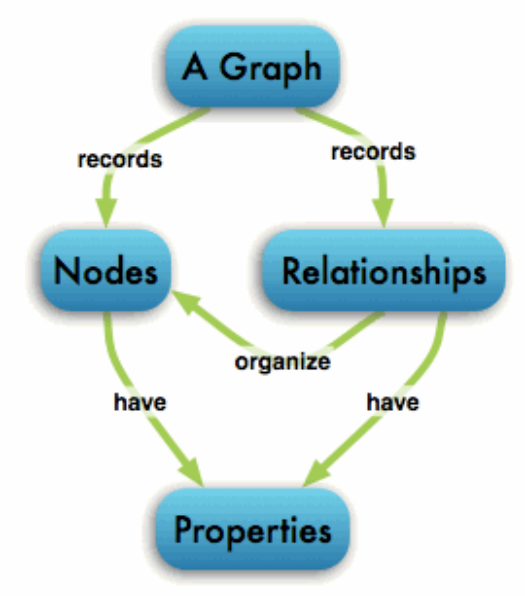

FIGURA 6: Representació del model de dades de Neo4j [11]

3.4.1 Característiques

A continuació es van a enumerar les característiques principals de Neo4j remarcant aquelles que ens han fet decidir-nos finalment pel seu ús:

- Representació intuïtiva. Com ja s'ha explicat anteriorment el model de dades representa de manera natural sistemes on les relacions tenen un paper important. En el nostre cas volem representar les relacions de traçabilitat entre els elements del domini i els requisits (en forma de nodes i proves d'acceptació).
- Emmagatzemament persistent basat en disc, completament optimitzat tant en rendiment com en escalabilitat per guardar estructures basades en grafs. En les últimes versions s'ha millorat l'emmagatzemament d'*strings* de gran volum
- per disminuir el seu impacte espacial.
- Escalabilitat. Actualment neo4j pot suportar diversos bilions de nodes, relacions i propietats en una única màquina, a més pot ser fragmentada per tal d'escalar-la a través de múltiples servidors. Les últimes versions de la base de dades ens aporten un mode d'alta disponibilitat, que permet replicar la informació en diferents màquines seguint un model master/slaves per tal de tindre una arquitectura tolerant a fallades i suportar una major càrrega d'operacions de llegida sobre la base de dades.
- Flexibilitat del modelat. Com ja s'ha comentat, un dels principals avantatges de les BDOG és la gran flexibilitat que aquestes ens donen en modelar el nostre sistema de manera incremental gràcies a la simplicitat de la seua estructura. A més d'açò, Neo4j ens aporta un altre nivell de flexibilitat que no tenen la resta d'implementacions i es tracta de la inexistència de nodes tipats. En neo4j cada node té una col·lecció de propietats que son independents de la resta de nodes, aquesta propietat ens serà de gran ajuda perquè cada element del model d'entitats té unes característiques completament diferenciades.
- Transaccions ACID. Neo4j té un sistema de transaccions que suporta completament les propietats ACID, és a dir, Atomicitat, Consistència, aïllament (Isolation) i Durabilitat. Totes les operacions d'escriptura sobre la base de dades

estan incloses dins una transacció de la forma {començar transacció, operar, marcar com a satisfactòria o no, finalitzar} on en cas de una transacció no exitosa es produïra un *rollback* per mantindre la consistència.

Per defecte el nivell d'aïllament es un *read_commited*, és a dir, que les operacions de lectura es duran a terme sobre la informació guardada satisfactòriament després de l'últim *commit* i no es tancarà el lock de lectura donant opció a lectures concurrents. Neo4j té capacitat per utilitzar nivells d'aïllament més estrictes amb la gestió manual del locks d'escriptura i lectura. Per defecte els *locks* d'escriptura són bloquejats per a nodes o relacions quan s'està creant, eliminant o afegint propietats sobre l'element i es desbloquejaran en acabar la transacció. Neo4j detecta els possibles *deadlocks* abans que es produisquen i marca la transacció com a *rollback*. Totes aquestes propietats ens aporten una seguretat sobre la integritat de les dades i el seu correcte emmagatzemament.

- API d'accés simple. Si utilitzem Neo4j en mode integrat podem modificar directament la base de dades utilitzant la API pública que ens ofereix. Es tracta d'una API orientada a objectes escrita en Java, encara que també ens proporcionen dos *bindings* per a Ruby i Python. En el nostre cas no farem un ús directe d'aquesta funcionalitat, ja que nosaltres accedirem a la base de dades mitjançant servicis web.
- StandAlone Server. Neo4j permet utilitzar la base de dades en mode servidor en comptes d'estar integrat dins l'aplicació. Per tal de comunicar-se amb la base de dades ens ofereix uns sistemes de configuració de ports i una API REST basada en crides http. És el mecanisme que utilitzem en el projecte per comunicar-nos i evitar la incompatibilitat tecnològica. En parlarem amb més detall en un altre capítol.
- Potent mecanisme de recorregut. Com hem vist anteriorment, en la majoria d'implementacions de BDOG els elements més utilitzats per recórrer el graf d'una manera natural son els *Traversals* que s'encarreguen de explorar les dades utilitzant cerques en nivell o profunditat on es defineixen altres criteris de cerca com el tipus de relacions a recórrer, la seua direcció, la profunditat des

del node inicial, el valor d'un cert atribut,... i retornen el subgraf resultant del recorregut guiat. Neo4j ens ofereix un mecanisme de creació de traversals tant en la seua API nativa com en el servici REST. Aquesta última serà la que s'utilitzarà en el projecte per tal d'explotar les relacions de traçabilitat i aconseguir resultats en l'anàlisi d'impacte.

- Complet sistema d'indexació. Per tal d'accedir de manera més eficient als elements (nodes i relacions) seguint criteris de cerca semblant als emprats en les base de dades relacionals (per exemple cerques per string), Neo4j inclou la possibilitat d'utilitzar *Apache Lucene* que ens permet crear i organitzar diversos índex de nodes o relacions. Esta és una poderosa eina que encara no utilitzem en el projecte, ja que de la versió 1.2 a la 1.3 de Neo4j es va produir un gran canvi en la implementació i el sistema de fitxers i es va deixar de costat fins que no es normalitzara el seu funcionament.
- Ferramentes addicionals. Una de les característiques més interesants de Neo4j és la quantitat de ferramentes que tenim a la nostra disposició per fer un bon ús de la base de dades. Sense tindre en compte les ja exposades com el sistema d'índex, el servidors o els *traversals*, esta BDOG ens aporta també altres eines com un sistema incremental de backups (en l'edició *Enterprise*), ferramentes de monitorització i algorismes ja implementats de cerca de camins més curts disponibles des de les dos API.

  Però les eines més destacades son aquelles relacionades a l'extensió del servidor i la seua API de comunicació. Des de la versió 1.3, Neo4j possibilita la incorporació de plugins definits pels usuaris per tal d'estendre la funcionalitat de l'API actual. Seguint esta ferramenta, s'han integrat en les últimes publicacions els plugins de *Gremlin* i *Cypher* que es tracta de dos llenguatges de tipus *query* i *pattern matching* respectivament per tindre una major expressivitat en les consultes. Ací es pot veure un xicotet exemple per fer-se una idea general de la seua sintaxis:

  Gremlin: $e:= g.v(0).outE ==> e[2][0-related_to->1]

  Cypher: start n = (0) match (n) - [:related_to] -> (rel) return rel

A més d'açò en l'última versió s'ha afegit una nova funcionalitat en el servici REST per tal d'executar operacions dins de transaccions complexes. Finalment cal destacar el administrador Web que ens ofereix el servidor que ens permet monitoritzar i visualitzar les dades de la base en tot moment fent ús exclusivament d'un navegador. Com es pot veure en la figura inferior la informació se'ns presenta en distintes representacions i inclou una visualització en forma de graf.

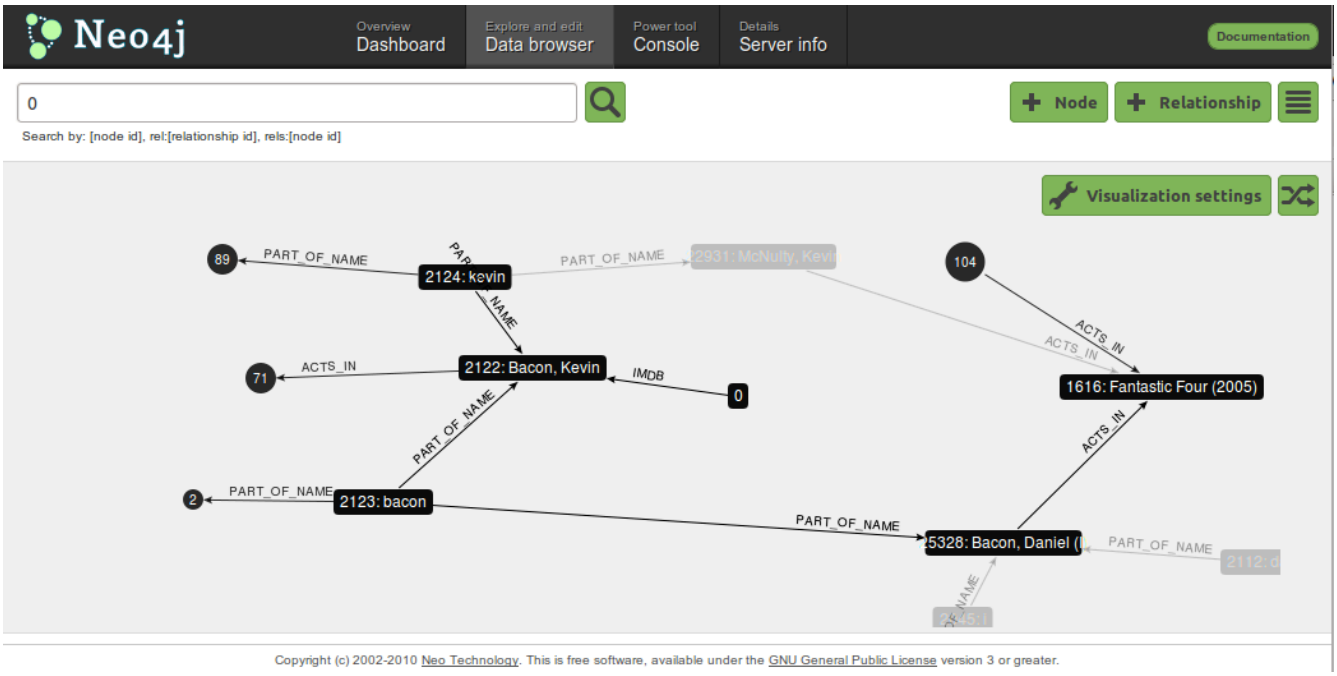

FIGURA 7: Webadmin ofert pel servidor Neo4j

- Llicència dual. Es tracta sense dubte d'una de les característiques més importants en l'elecció de Neo4j per suportar la infraestructura del projecte. Com hem dit anteriorment esta BDOG és un projecte de codi obert que ofereix dos tipus de llicències completament diferenciades.

  Per un costat tenim una sèrie d'edicions no comercials; l'edició *Community* sota una llicència GPL que ens permet utilitzar Neo4j sense cap tipus de restricció, amb l'única excepció dels casos on es vol distribuir l'aplicació que utilitza la base de dades amb codi tancat, on caldria adquirir una llicència comercial, i les edicions Advanced i Enterprise sota una llicència AGPL que aporten ferramentes de monitorització i alta disponibilitat a Neo4j. D'altra banda tenim

una llicència de pagament OEM que permet a les empreses utilitzar i distribuir software que utilitza Neo4j inclús tractant-se d'aplicacions de codi tancat.

Aquesta llicència tant flexible és perfecta per al nostre projecte, ja que encara que anem a dissenyar un mòdul de codi tancat, la ferramenta serà exclusivament d'ús intern de la companyia i per tant no es distribuirà i ens donant la possibilitat d'utilitzar l'edició *Community* que no suposarà cap cost addicional per a l'empresa.

- Comunitat i documentació. Un dels principals avantatges de Neo4j és precisament l'extensa documentació que aporten en la seua web, amb un manual molt complet i diversos exemples d'implementacions reals. A més, gràcies a ser un projecte de codi obert amb una extensa comunitat, existeix un wiki amb informació addicional sobre Neo4j i les seues ferramentes, així com altres projectes derivats i una llista de correus molt activa on expressar els teus dubtes.
- Cicle de desenvolupament. Neo4j és una base de dades que basa els seu desenvolupament en cicles curts d'aproximadament 6 mesos, on cada mes s'allibera una versió *milestone* per a que la comunitat vaja provant el rendiment i les millores, aconseguint així *feedback* que s'utilitza per a les futures implementacions. En cada release es millora el rendiment de les operacions actuals, s'ofereixen noves ferramentes i es completen les APIs actuals.

  Aquesta manera de treballar, acosta els desenvolupadors de Neo4j a la comunitat i guien la seua creació en allò que els és més demandat. Per exemple en les últimes versions alliberades s'ha afegit l'opció de crear plugins per al servidor, així com un sistema de transaccions complexes mitjançant la interfície REST o el llenguatge Cypher basat en pattern matching per a la cerca de node.

3.4.2 Mode Servidor i REST API

Abans d'aprofundir en l'especificació de l'API que Neo4j posa al nostre abast, començaré fent una introducció d'alguns conceptes o explicacions que s'hi relacionen.

Primer de tot cal remarcar l'arquitectura modular en la que es basa Neo4j. Consisteix en un nucli central format al mateix temps per altres mòduls més atòmics com ara els

mecanismes de persistència, control de transaccions i concurrència o la gestió d'índex lucene, envoltat per una capa externa que defineix la interfície o API pública de les accions que es poden fer sobre la base de dades. El nucli central o *kernel* ja pot ser utilitzat com a base de dades integrada dins de l'aplicació (mode *embedded*).

D'altra banda en aquest *kernel* se li poden acoblar altres mòduls que estenen les capacitats o possibilitats de la base de dades, entre elles podem trobar el mòdul de back-up o alguns altres de monitorització o visualització. Però el que ens interessa en aquest punt és aquell que converteix Neo4j en un servidor, actuant com a *wrapper* sobre el nucli, i afegint tota la infraestructura associada a la xarxa com ara protocols de transferència o el propi administrador web. A més de tota aquesta infraestructura, Neo4j ens aporta uns servicis de comunicació independents de la tecnologia utilitzada que fan servir crides al protocol *http* per tal d'accedir als recursos de la base de dades. Aquests servicis estan basats en l'arquitectura REST.

REST és un acrònim per a *Representational State Transfer* i és tracta d'una arquitectura software per a recursos distribuïts que es fonamenta en unes idees simples, aquestes son les més destacades:

- Totes les comunicacions al servici es duran a terme mitjançant crides del protocol HTTP.
- Tots els recursos que volem oferir en el nostre servici han de tindre una URL associada a la qual poder accedir amb l'ús de crides GET.
- La manipulació dels recursos i operacions disponibles es farà mitjançant les crides POST (creació), PUT(modificació) i DELETE (eliminació). A aquestes comunicacions se'ls pot aportar informació addicional per al servici que sol estar emmagatzemada en XML o JSON.

Neo4j aporta un conjunt de servicis REST per tal de comunicar-se amb el servidor definint una interfície d'operacions disponibles. A continuació enumeraré les operacions i mostraré algun exemple amb la informació adjunta a alguna crida o al seu resultat:

- Obtenció de recursos: es fa una crida GET sobre la URL del recurs, actualment podem accedir a nodes, propietats del nodes, relacions dels nodes, tipus de relacions, propietats de les relacions. Exemple de l'obtenció de les propietats del node 1:

  Crida : GET $servidor/node/1/properties

  Resultat 200:OK { "nom": "Guillem", "cognoms": "Medina Martínez"}

- Eliminació de recursos: es fa un GET sobre la URL de l'element a eliminar. El servidor tornarà un codi 200 o 40x en funció de si ja pogut eliminar o no el recurs.
- Creació i modificació de recursos: es fa una crida POST/PUT en funció de si estem creant o modificant sobre la URL de l'element amb el que volem interactuar, com en el cas anterior estos poden ser nodes, relacions o propietats associades a algun dels altres dos elements. Exemple de la creació del node 2 amb la propietat nom:

  Crida: POST $servidor/node { "nom": "Pep"}

  Resultat 201: Ok node creat http://$servidor/node/_idNouNode
- Gestió dels Índex: Utilitzant les crides ja exposades podem crear índex, eliminar-los, llistar índex de nodes o relacions, indexar i desindexar elements i buscar elements dins de l'índex per coincidència exacta de cadenes o utilitzant el llenguatge de *queries* suportat per Lucene. No entrarem en més detall en estes crides perquè encara no han sigut utilitzades en el projecte.
- Cerca de camins entre dos nodes: la interfície REST de Neo4j ens dóna l'oportunitat d'utilitzar alguns dels algorismes ja codificats que ofereixen, actualment es dóna suport a *shortestPath, allPaths, allSimplePaths* i *Dijkstra.* Tampoc entrarem en detall en els elements d'este punt.
- Traversals: per tal de recórrer el graf, l'API ens permet fer un post sobre un node inicial, travessar-lo seguint uns criteris i retornar un conjunt d'elements del tipus que nosaltres definim. Este és un exemple de traversal:

  Crida: POST /node/1/traverse/node

```
{
        "order": "depth_first",
        "relationships": [
                { "type": "KNOWS", "direction": "out" },
                { "type": "RELATED" }
        ],
        "prune_evaluator": {
                "language": "javascript",
                "body": "position.endNode().getProperty('date')>1234567;"
        },
        "return_filter": {
                "language": "builtin",
                "name": "all"
        },
        "max_depth": 2
}
```

En la crida podem vore que començant des del node 1 volem travessar el graf i que ens torne el conjunt de nodes que complisquen els criteris demarcats en el fitxer JSON adjunt. En aquest cas indicant que es vol recórrer el graf per profunditat, solament travessant les relacions KNOWS en direcció d'eixida i RELATED en ambdues direccions, amb una profunditat màxima de dos salts on es retornaran tots els nodes pels que s'haja passat però esporgant aquelles branques del graf on el node tinga una propietat *date* amb un valor superior a 1234567.

Resultat: 200 OK i com hem demanat que ens torne un conjunt de nodes,

```
[
        { "self": "http://localhost:7474/db/data/node/64",
          "data": { "name": "Guillem" },
          ...
        },
        { "self": "http://localhost:7474/db/data/node/635",
          "data": { "name": "Pep" },
          ...
        }
]
```

Esta és la interfície que utilitzarem per definir l'anàlisi d'impacte sobre les PAs i on definirem els criteris que volem utilitzar per fer-ho.

4 Infraestructura per a l'AI

En esta secció s'exposarà el conjunt d'elements creats per a l'anàlisi d'impacte dels requisits. Començarem explicant el model de dades utilitzat mostrant la seua evolució des de les primeres versions fins l'actual, a continuació definirem el gestor de termes creat per tal de organitzar el model d'entitats del domini de l'aplicació i per últim s'explicaran les modificacions sobre l'API Rest per a C# de Neo4j que s'han fet per a aquest projecte.

4.1 Model de dades

4.1.1 Consideracions prèvies

Quan parlem del model de dades s'ha de recordar que estem utilitzant el paradigma de BDOG per tal de definir i organitzar la informació. Aquesta decisió canvia la representació del coneixement i per tant les estratègies utilitzades per tal de dissenyar el model de dades, és així com les definicions mitjançant un model entitat-relació, que són tant utilitzades en les bases de dades relacionals per la seua senzilla correspondència, no tenen sentit en el nostre context. Per tal de definir el nostre model de dades hem de trobar una representació clara, intuïtiva i que ens mostre tota la informació rellevant sobre l'estructura de les dades.

En la literatura acadèmica referent als models de dades de grafs [13] se'ns mostra una gran varietat de possibles definicions, tot i això no existeix un acord o estàndard generalitzat sobre el qual basar-nos, així que açò ens dona una llibertat a l'hora de representar la nostra base de dades. Analitzant les propostes existents (GOOD, LDM, GROOVY, HML, Simatic-XT, GDM, GOAL, RDF, ...) s'ha arribat a la conclusió que ninguna d'elles satisfeia d'una manera simple les necessitats del projecte en ser, depenent de la situació, massa complexes, poc intuïtives i clares o en estar basades en una implementació del paradigma de grafs diferent a la nostra, com per exemple els hipergrafs.

Finalment s'ha optat per utilitzar una definició pròpia amb diverses característiques de les propostes anteriors. Inicialment definirem l'esquema de dades en dos nivells de profunditat, en el primer nivell mostrarem una representació simplificada amb els tres

dominis de nodes que tractarem i en el segon especificarem tots els tipus de nodes i relacions definides, així com les seues possibles restriccions i propietats obligatòries. D'altra banda definirem una representació d'una instància de la base de dades basada en els esquemes definits. Per últim enumerarem les modificacions que s'han de dur a terme per a la seua implementació en Neo4j i les possibles millores d'optimització.

Un altre aspecte a considerar a l'hora de dissenyar un model de dades de grafs és la seua flexibilitat. A diferència de les bases de dades relacionals basades en rígides taules i camps tipats, en les BDOG el model evoluciona d'una manera natural afegint nous *tipus* de nodes (que no nodes tipats, ja que solament es tracta d'una definició abstracta) i nous tipus de relacions entre aquests. Son precisament estos tipus, i en especial els de les relacions, els que cal definir amb més deteniment ja que seran els utilitzats per explotar la informació de la base de dades mitjançant els criteris de recorregut dels traversals.

En aquest projecte es mostrarà solament l'última versió utilitzada del model de dades, remarcant en alguns casos perquè s'ha optat per aquesta opció i definint algunes possibles modificacions futures.

4.1.2 Definició del model

Com ja hem definit en el punt anterior, el model de dades vindrà representat per una sèrie de esquemes propis que anirem explicant detingudament. Inicialment es mostra una representació abstracta que podríem caracteritzar com a nivell 0 de l'esquema general del nostre model de dades que expressa d'una manera global l'arquitectura de la base de dades i la seua organització, aquest primer esquema es interesant per tal d'introduir conceptes en els que es profunditzarà posteriorment

Com es pot vore en la imatge inferior, el model esta dividit en dos seccions completament diferenciades i , encara que estiguen interconnectades, representen dos conjunts d'elements distints des d'un punt de vista conceptual.

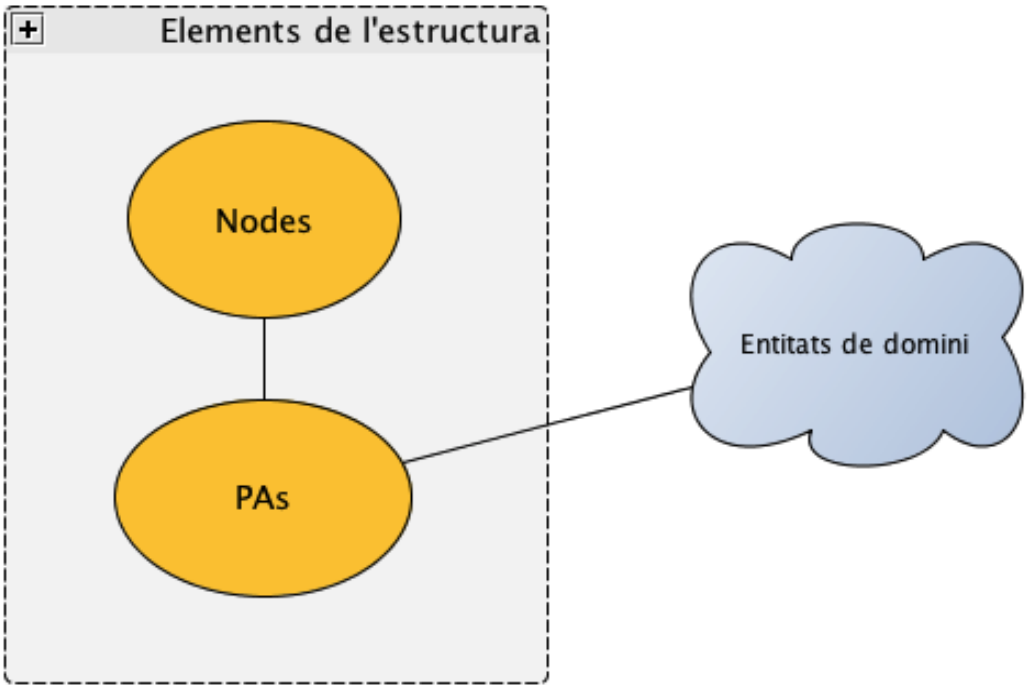

FIGURA 8: Esquema de nivell 0 del model de dades

Per un costat hem de destacar els components etiquetats com "Elements de l'estructura"; aquesta terminologia es refereix als contenidors naturals de la ferramenta de suport per a la metodologia TUNE-UP en la que estem basant la nostra anàlisi d'impacte. En esta secció englobem les proves d'acceptació que son l'element bàsic sobre les que definirem les relacions de traçabilitat i per altre costat els nodes, que seguint la seua pròpia definició TUNE-UP, son contenidors de proves d'acceptació utilitzats per organitzar-les en nivells, creant una jerarquia de requisits per facilitar l'accés i la navegació entre elles.

Per tant, seguint estes idees, els nodes de la BDOG que formaran els conjunts etiquetats com a "Nodes" i "PAs", son representacions d'elements ja existents en la base de dades relacional. Com la quantitat existent d'aquests elements és molt elevada i el nombre de proves que estem utilitzant actualment per a l'anàlisi d'impacte és considerablement reduït, s'ha optat per afegir incrementalment aquests elements estructurals en crear una relació de traçabilitat sobre una de les proves, ja siga una de les anteriorment definides o de les recentment creades.

Per tal de no caure en equivocacions terminològiques, a partir d'aquest moment ens referirem a *Nodes* quan voldrem fer referència a un element de l'estructura jeràrquica de requisits i a *nodes* (en minúscula) quan, en canvi, fem referència a l'element bàsic de les bases de dades orientades a grafs que utilitzarem per a modelar tots els conceptes que volem emmagatzemar.

D'altra banda, i tornant a la figura superior, podem vore com els elements pertanyents al conjunt de les proves d'acceptació estan relacionats amb una entitat que hem dibuixat amb un núvol. Esta representació ve influenciada per la idea que aquestes entitats o termes del domini son un conglomerat de conceptes desiguals que únicament tenen en comú el fet que poden ser utilitzats com a enllaç, entre les diferents proves d'acceptació gràcies a les relacions de traçabilitat.

No existeix una definició formal de què és un terme dins aquesta amalgama de nocions. En un principi el núvol s'ha constituït amb elements d'allò que es pot conéixer com entitats del domini d'acció de l'empresa en la qual s'ha dut a terme aquest projecte i que, en aquest cas , tracta de terminologia associada al món sociosanitari i de gestió de residents. Seguint aquesta idea, en el conjunt actual podem trobar termes relacionats amb treballadors específics d'aquestes entitats, temes relacionats amb els comptes d'usuaris/residents i amb la seua informació, conceptes referents a la facturació de les diverses entitats i fins i tot aspectes legals lligats a les aplicacions de la companyia com ara la llei de protecció de dades.

Aquesta seria una primera versió del concepte termes del domini que englobaria únicament a les entitats implicades explícitament en el context de l'empresa, en futures modificacions i a petició dels responsables del projecte s'espera ampliar la definició d'aquest concepte per tal que incloga altres aspectes que també poden ser útils com a elements d'enllaç de l'anàlisi com referències a altres PAs, Nodes o altres entitats com ara clients de l'empresa o el conjunt de programes desenvolupats.

Una vegada definit l'esquema general del model de dades es convenient aprofundir en la implementació real de l'esquema en la nostra BDOG, detallant amb més exactitud el conjunt de nodes, relacions i propietats fonamentals que el defineixen, indicant les seues característiques més importants.

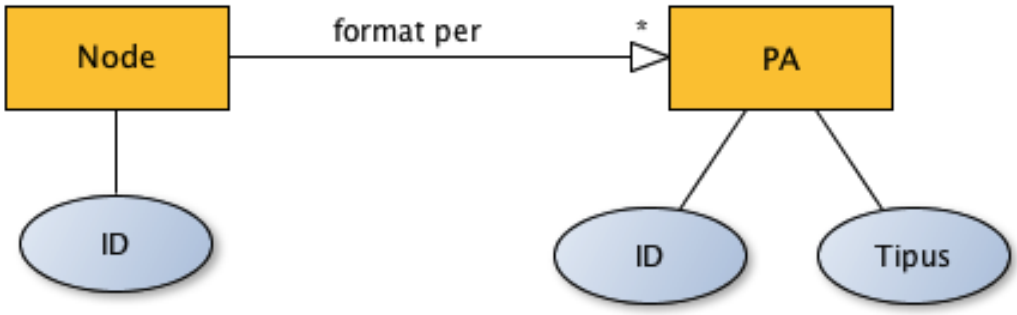

FIGURA 9: Esquema 1 de la part estructural del model de dades

Centrant-nos únicament en el subconjunt format per les parts estructurals del model de dades, definim l'estructura com un conjunt de nodes de tipus Node connectats mitjançant relacions de tipus "format per" com es pot vore en la imatge superior. Encara que en l'estructura real de la ferramenta els Nodes també estiguen relacionats amb altres Nodes, aquesta informació no es rellevant per a l'anàlisi d'impacte i per tal d'evitar la reiteració d'informació, solament es veurà reflectida la idea general que un node es un contenidor *format per* PAs, en la definició de la imatge també s'apunta al fet que un node pot contindre moltes PAs, però en canvi una PA solament pot estar relacionada amb un únic node.

Encara que la relació "format per" es mostra en la imatge com un vincle direccional que va des d'un Node fins a una PA, s'ha de comentar que el fet de representar el nexe amb una punta blanca, denota que esta direccionalitat no té una implicació real en la definició de la base de dades, però en canvi si que la té des d'un punt de vista conceptual.

Finalment destacar que encara que els nodes de la base de dades Neo4j no estan tipats i per tant no estan lligats a cap estructura prefixada, per als elements de l'estructura que tenen una relació directa amb els seus homònims en la base de dades relacional, hem definit un conjunt simple per representar la informació bàsica a explotar. Per tant, els Nodes únicament estaran definits per la propietat "ID" que es correspon amb el identificador de la taula *GRNodos* de la ferramenta i les PAs es voran definides per un ID i un tipus. Aquest binomi es deu al fet que en l'arquitectura del

gestor de requisits, existeixen dos tipus de proves d'acceptació (propostes i proves actuals) associades a taules diferents i per tant l'identificador associat al node de la PA ha de tindre marcat el context. Cal destacar que en cas d'afegir informació addicional sobre els nodes de les estructures, no es vora utilitzada en l'explotació de les relacions de traçabilitat.

Per altra banda, si ens centrem en el nucli del model de dades centrat en els termes del domini, podem trobar una definició més detallada com la que es veu en la imatge inferior.

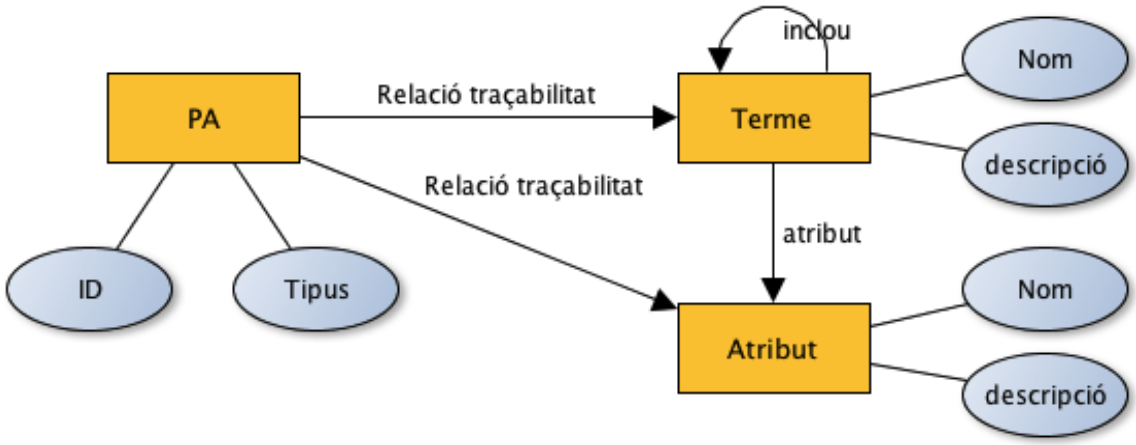

FIGURA 10: Esquema de nivell 1 de la part del domini del model de dades

En la part esquerra de la imatge podem vore la secció relacionada amb les proves d'acceptació i que es correspon amb la part dreta de la imatge anterior. Aquesta esta relacionada amb dos *tipus* de nodes anomenats termes i atributs per un conjunt de relacions de traçabilitat que són la base de l'anàlisi d'impacte.

Aquestes relacions estan tipades i categoritzades en funció de la secció en que s'han definit dins de la descripció de la prova, aleshores seguint les opcions actuals trobem 5 tipus de nexes que poden ser explotats de manera diferent, creant algoritmes d'anàlisi més complexos i que s'ajusten més a les necessitats de l'empresa. Els tipus de connexions que tenim definides actualment són: relacions de les condicions prèvies a la prova, relacions dels passos a seguir en la seua definició, relacions en els resultats esperats, relacions en les observacions addicionals dels analistes i per últim relacions en les motivacions de cada prova d'acceptació.

Com ja hem definit, aquestes relacions estan connectades amb el que podem anomenar com núvol del domini. Aquest esta format per *termes* i *atributs*, que són conceptes jerarquitzats que defineixen el context de les relacions de l'anàlisi d'impacte. Sobre aquests elements parlarem més detalladament en la secció següent, tot i així, cal destacar algunes de les seues propietats representades en l'esquema.

D'una banda definir que un *terme* pot estar vinculat a altres *termes* seguint una escala de nivells, en canvi, els *atributs* han d'estar associats necessàriament a un únic *terme.* En relació a les propietats de nom i descripció representades, destacar que aquestes són les característiques requerides que han definit els analistes per tal de poder identificar les entitats del domini. A banda d'aquestes 2 propietats, actualment s'estan utilitzant altres com el concepte *tags* o etiquetes relacionades amb el terme o altres camps relacionats amb distints idiomes.

Actualment s'estan avaluant modificacions respecte a esta part que aporten més facilitats a la tasca dels analistes, en concret, s'està estudiant la possible creació de tipus de dades associats a un terme o atribut concret. La categorització dels termes servirà principalment per a definir un conjunt de propietats preestablertes associades a les entitats de cada classe; per exemple, un terme associat al tipus numèric, pot tindre associat per defecte propietats com ara la unitat de valor, el valor per defecte, el número d'enters i decimals, saber si es negatiu o no, o si existeixen límits superiors i inferiors. De portar a terme esta representació tan exhaustiva del tipus de dades integrada en la pròpia ferramenta de gestió dels requisits, es podrien definir fins i tot petites ferramentes de generació automàtica del codi associat als requisits, aportant una nova eina d'ajuda per als programadors software, de moment però, aquest camí de desenvolupament està encara en els seus moments inicials i no entrarem en més detalls.

Una altra modificació a implementar en el model de dades, és l'emmagatzemament de les rutes d'accés a les entitats. Degut a la naturalesa del domini i a la seua interconnectivitat, poden existir diverses rutes d'accés a un mateix element i aquest camí també es pot utilitzar de criteri per tal de fer l'anàlisi de les relacions. Per exemple al terme "Factura" es pot accedir directament o mitjançant el camí "Client ->

Factura", i la seua definició pot vindre simplement associada al context de la prova on s'ha definit la relació. Actualment aquestes rutes no s'estan explotant i s'està seguint un criteri basat en la ruta més llarga possible, però la modificació ens permetria crear un nou filtre que podem aplicar en determinades situacions on poguera ser útil.

Per acabar aquesta secció del model de dades presentem tot seguit una última imatge relacionada amb una possible instància de l'esquema de dades definit prèviament, molt simplificada per tal de mostrar els elements més significatius. A banda d'objectes reals, s'hi mostraran també nodes auxiliars que utilitzem com a accessos o filtres dins de la base de dades.

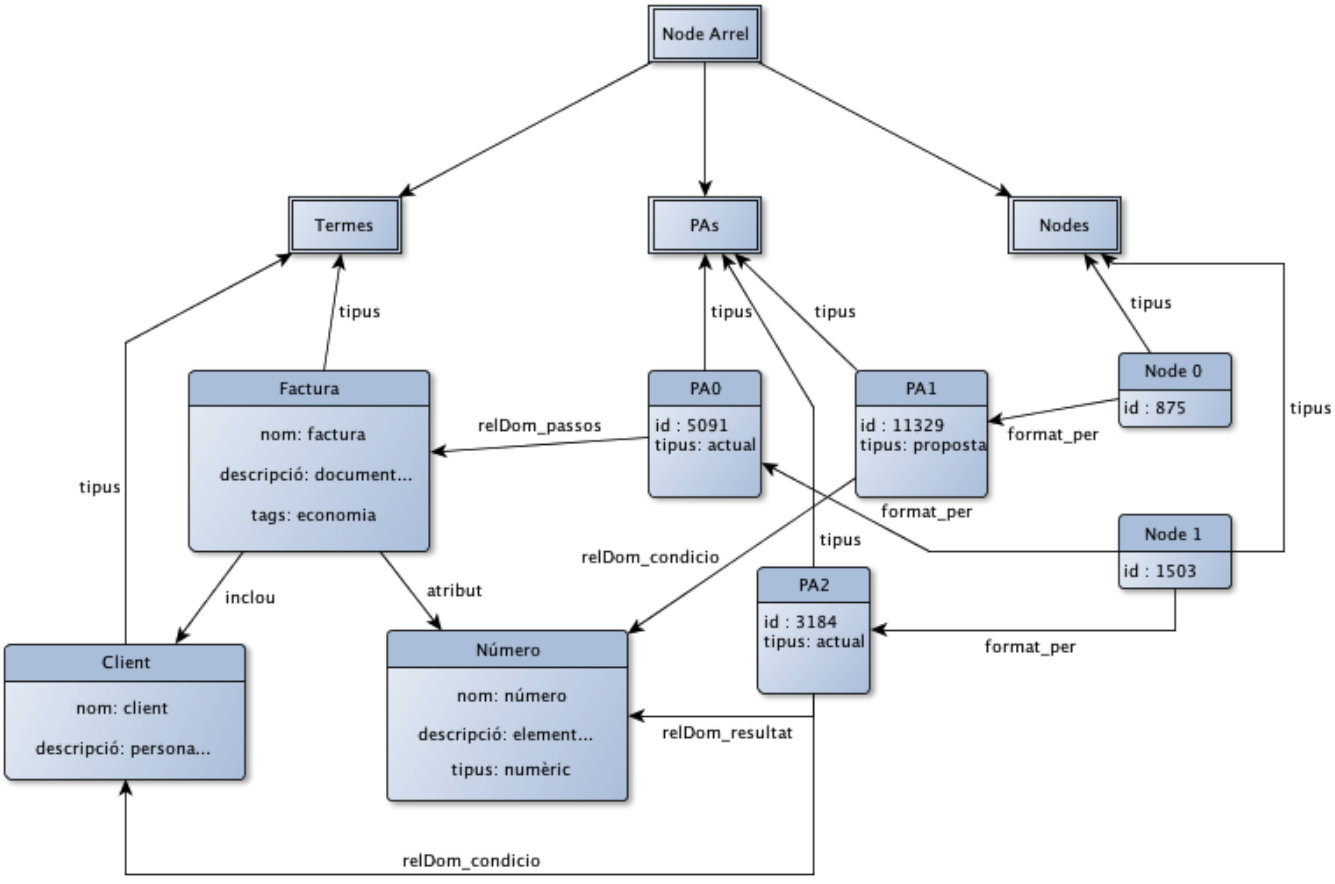

FIGURA 11: Instància de la nostra base de dades neo4j

Per començar l'explicació referent a la imatge de la instància de la nostra BDOG, cal centrar-se en els quatre nodes superiors representats com rectangles amb doble delimitació. Aquests no estan associats als esquemes prèviament definits ni, en general, al model de dades del mòdul, però son necessaris des d'un punt de vista

logístic, ja que per al funcionament de les base de dades basades en els grafs es necessiten punts d'accés a la informació.

En els grafs, la navegació a través dels seus components ve caracteritzada pel recorregut de les relacions entre nodes, però per poder començar-la cal tindre un punt de partida i aquest és precisament el paper del node arrel que defineix Neo4j de manera automàtica en inicialitzar la base de dades.

A banda, podem trobar altres 3 nodes fent tasques auxiliars similars a la ja expressada, els nodes etiquetats com *Termes*, *PAs* i *Nodes* són mecanismes interns que hem afegit per tal de facilitar-nos l'accés a les dades i filtrar, des de la pròpia estructura, el conjunt de nodes sobre el que hem d'executar les nostres consultes. En un futur, si ampliem el concepte de termes per incloure uns altres elements externs a les entitats de domini, podem afegir uns altres nodes auxiliars que actuen com a filtre o índex entre els diferents tipus de termes.

Centrant-nos ara en la part inferior de la imatge, podem vore representats en forma de nodes un conjunt simplificat de *termes, PAs* i *Nodes.* En aquest conglomerat cal destacar el primer nivell de relacions entre els nodes auxiliars i alguns nodes del model de dades que ens ajudaran a definir el tipus del node al qual estem accedint. D'altra banda veiem com les PAs definides sempre tenen associades un únic node, i que aquests en canvi son un contenidor il·limitat, per exemple, el Node 1 esta format per les PAs 0 i 2.

Per últim, si ens fixem en les entitats de domini i en les seues relacions amb les PAs, trobem un conjunt jeràrquic amb dos *termes* relacionats (Factura i Client) i un *atribut* (Número) associat a un *terme* pare (Client). En les relacions de traçabilitat (designades amb el prefix relDom_*), se'ns mostra la connexió entre diversos elements mitjançant l'ús de relacions tipades (marcades en el nom de la relació). En l'exemple podem vore proves amb més d'una relació de traçabilitat, i també, *termes* amb més d'un vincle associat. És precisament aquesta última situació la que servirà de base per al nostre anàlisi d'impacte dels requisits.

4.2 Gestor de termes

Com ja s'ha definit en altres parts del projecte, l'anàlisi d'impacte que es durà a terme està basat en relacions de traçabilitat entre proves d'acceptació i termes del model d'entitats. Les proves d'acceptació es van definint incrementalment en funció de les necessitats dels clients i es defineixen en el formulari corresponent dins del mòdul del gestor de requisits de la ferramenta TUNE-UP. En canvi el model de dominis de la ferramenta es un conjunt abstracte de termes relacionats amb el domini de l'aplicació desenvolupada, que els analistes han de definir per tal de trobar interconnexions entre els diferents elements d'una especificació de requisits.

Al començament de la definició del nostre projecte, aquest model d'entitats estava pobrament definit com a un conjunt no ordenat de termes (amb alguna informació addicional) dins un document de *microsoft word* compartit mitjançant un servidor *sharepoint,* per tal de donar accés a tots els usuaris involucrats en el desenvolupament de la ferramenta. Aquesta organització de la informació no permetia una extracció directa de la informació ja que no estava integrada dins de la ferramenta TUNE-UP i complicava la tasca dels analistes i, a més, no existia una estandardització de la informació per la qual cosa la seua explotació per a l'anàlisi d'impacte es feia molt complicada.

Per tal de gestionar les entitats de domini es va demanar un nou mòdul integrat que els permetera definir i organitzar els termes amb facilitat i tingueren accés des del propi formulari on es defineixen les PAs, és a dir un gestor encarregat de la part de les entitats de la base de dades orientada a grafs. Per determinar l'estructura del nou gestor així com la seua navegació i funcionalitat cal concretar primer una organització dels termes del domini, entre les seues relacions, nivells i propietats. Seguint criteris relacionats amb l'ús que es donarà a aquests termes, s'han definit dos nivells d'entitats de domini: el primer nivell son els *termes*; entitats independents amb propietats i relacions amb altres elements del mateix nivell. D'altra banda tenim els *atributs* que són entitats de segon nivell amb propietats que només tenen sentit en el context d'un *terme* i per tant estaran relacionades amb un terme i únicament amb un.

Tots els termes i atributs poden ser utilitzats en les relacions de traçabilitat necessàries per a l'anàlisi d'impacte de requisits i aquestes estaran tipades en funció del camp corresponent dins del formulari de proves d'acceptació (condició, passos, resultats, motivació,...). En canvi, les propietats només mostren les descripcions o restriccions de cada element que podem associar a la PA, per tal que els analistes, els programadors i els testers tinguen una completa informació a l'hora de prendre les seues decisions basades en l'anàlisi.

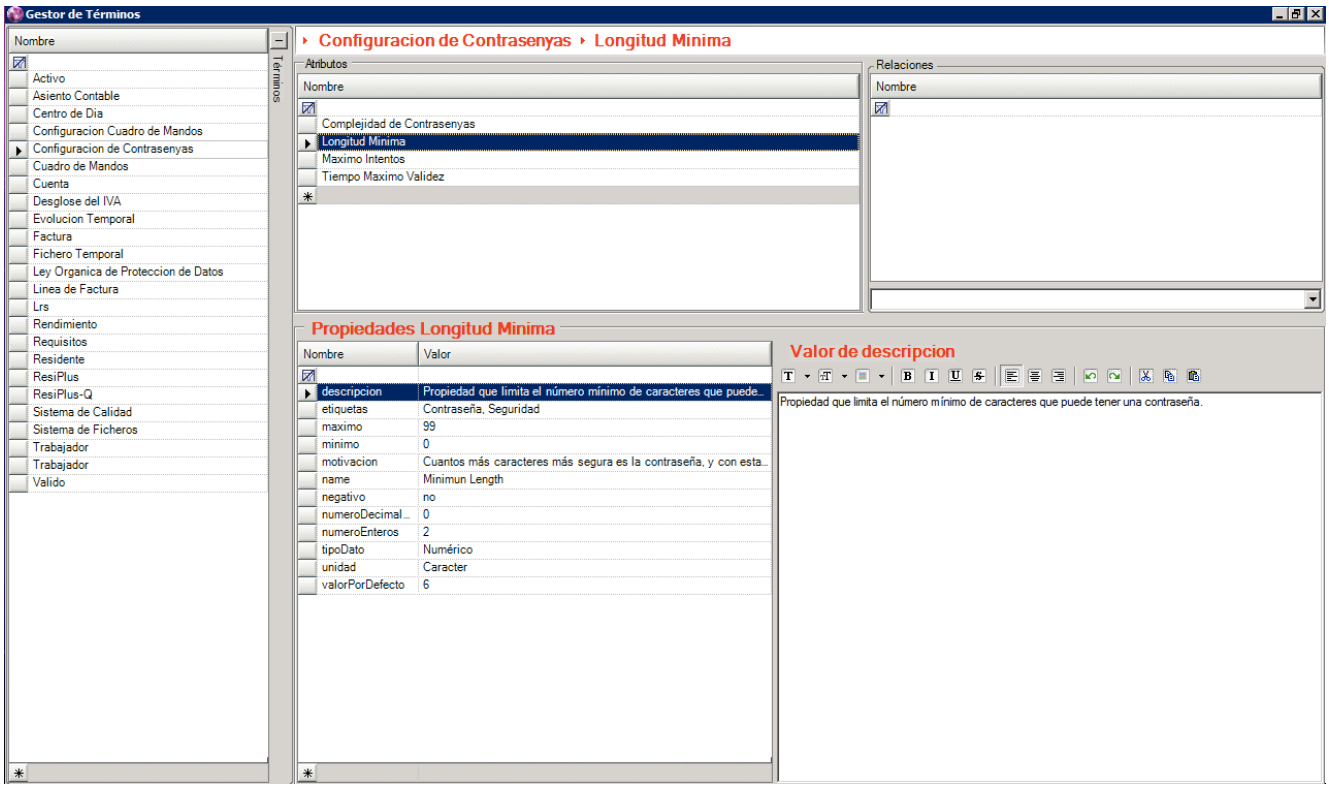

FIGURA 12: Gestor de termes

Després de nombroses propostes i reunions amb els analistes i seguint les pautes exposades anteriorment, es va arribar a definir el gestor de la imatge superior. El formulari esta separat en quatre seccions ben diferenciades.

Per començar, en la part esquerra del gestor hi ha un panell ocultable amb un grid que mostra tots els *termes* que hi ha actualment en la base de dades de grafs, amb una interfície d'introducció de nous elements. La informació mostrada en la resta de seccions ve determinada per la fila seleccionada del grid de *termes*, és a dir, en seleccionar una fila, les dades de la resta de grids s'actualitzaran amb les

característiques de l'element. En la part superior tenim dos seccions relacionades directament amb aquest element: les àrees d'*atributs* i *relacions*. En la primera es mostren en un grid els *atributs* o elements de segon nivell relacionats amb un *terme* i ens permet afegir-ne de nous. En la segona trobem un grid amb les relacions del *terme* amb altres elements del mateix nivell i un combobox amb opcions de cerca per tal d'afegir-ne de noves.

Finalment en la part inferior es troba la secció de les propietats que mostra mitjançant un grid totes aquelles propietats i els seus valors de l'element seleccionat, així com un RichTextBoxExtended associat a la fila seleccionada que permet enriquir el text relacionat amb la característica. Les dades mostrades en esta secció poden estar relacionades tant a *termes* com a *atributs* en funció de quin element ha estat seleccionat (per seleccionar un atribut cal fer un clic sobre la seua fila en la secció d'atributs).

Per tal de minimitzar la possible confusió produïda en reutilitzar la mateixa secció per propietats d'elements de diferent nivell, s'han afegit indicadors visuals on cal destacar el navegador superior. Aquest es un *breadcrumb* que ens indica en tot moment quin és l'element contextual del formulari i que a més ens aporta una nova manera de navegar pels diferents *termes* i *atributs* del model de domini, mitjançant una carrega gradual de la seua estructura en funció dels elements visitats.

Actualment existeixen restriccions d'eliminació i modificació sobre els elements del gestor de termes. Els noms dels *termes* i *atributs* no poden ser modificats ni eliminats i tampoc es permet eliminar les seues relacions si algun dels elements involucrats en l'operació ja ha estat utilitzat en les relacions de traçabilitat de l'anàlisi d'impacte. Aquesta limitació temporal ha estat implementada i acceptada per tots els agents a causa de les inconsistències que es produirien en els textos de les PAs, ja que les relacions per a l'anàlisi queden reflectides internament dins un arxiu RTF de les PAs amb decoracions per tal d'identificar-les i per tant, en tractar-se de text simple, no es voria actualitzat en fer modificacions. En futures versions del mòdul s'espera implementar un mecanisme automàtic d'actualització que es durà a terme o bé en fer la modificació des del gestor de termes o bé periòdicament com a ferramenta externa.

Per finalitzar esta secció, només comentar que per fer la interfície gràfica s'han utilitzat les llibreries de *windows forms* de la companyia *Infragistics* (utilitzades en tota la ferramenta de suport de TUNE-UP) i internament el navegador utilitza una interfície simple de connexió amb la base de dades orientada a grafs mitjançant l'API que es descriu en el punt següent.

4.3 Neo4j API

Un dels principals inconvenients que vàrem trobar a l'hora d'interactuar amb la base de dades orientada a grafs és la incompatibilitat tecnològica que ja es va comentar al capítol 3. Neo4j esta implementada en Java i ofereix una API pública de comunicació en aquest llenguatge de programació, mentre que la ferramenta de suport a la metodologia TUNE-UP on volem integrar el nostre mòdul, esta desenvolupat amb tecnologies .NET, més concretament codificada amb el llenguatge C#.

Encara que *NeoTechonologies* no ofereix cap *wrapper* o embolcall tecnològic directe per a una comunicació senzilla amb el *framework* de *Microsoft*, si que ofereix la possibilitat d'executar la base de dades com a servidor i comunicar-se amb ella mitjançant crides HTTP, gràcies al servici REST implementats. Inicialment es varen plantejar altres solucions, com la creació pròpia d'una capa d'interconnexió entre ambdues tecnologies o bé crear un client extern en Java que caldria instal·lar posteriorment en totes les màquines per tal que tots els agents pogueren comunicar-se amb la base de dades, però en determinar la senzillesa de la interfície REST aquestes idees es van deixar de banda.

Una vegada decidit que aquest seria l'element de comunicació emprat en el nostre projecte calia determinar si Neo4j oferia de manera oficial una implementació en .NET de la seua API REST. En accedir als recursos de la seua pàgina web es podia vore que solament existien APIs per als llenguatges de programació Python, Ruby i Java, però segons alguns emails escrits pels responsables del projecte, la implementació de C# estava programada en el seu *product backlog* per a futures versions.

Actualment, després d'un any de prendre aquestes decisions, les prioritats del grup de desenvolupament de Neo4j han canviat notablement i es centren en optimitzacions temporals i espacials i en general noves característiques i ferramentes, i no s'espera

aquesta funcionalitat en ninguna de les properes actualitzacions de software. A més, en créixer la comunitat de desenvolupadors al voltant del projecte, s'espera que tot allò relacionat amb APIs o plugins per al servidor siguen aportacions externes per part d'aquesta comunitat. Després de diverses consideracions, i en no saber una data concreta de publicació d'aquesta, es va optar per crear-ne una des de cero, basant-nos en les especificacions i el disseny de les APIs ja implementades.

4.3.1 Creació d'una API REST amb tecnologia .NET

La llibreria que es va definir s'havia d'encarregar principalment de 3 punts claus: en primer lloc serialitzar i deserialitzar objectes de C# a format JSON, a continuació crear un mecanisme per fer les corresponents crides HTTP al servici REST de la base de dades i finalment oferir una interfície per a utilitzar en la nostra capa d'accés a dades.[14]

Per començar la implementació de l'API ens vàrem decidir per crear un nucli de comunicació HTTP utilitzant la llibreria de HttpWebRequest i el seu pare en l'esquema d'herència, WebRequest. Es va construir un esquema en dos capes on en la més profunda es creaven crides bàsiques encarregades d'enviar peticions i extraure les respostes de les contestacions del servidor, i s'envoltava d'una capa que cridava internament a la prèviament definida, indicant el tipus de crida HTTP (get, post, put o delete) que s'anava a utilitzar i retornant la resposta ja extreta.

A continuació, calia definir els elements que serien visibles des de la nostra capa d'accés a dades i al mateix temps crear les classes necessàries per fer-ho possible. Aquestes classes serien també utilitzades per serialitzar-les en format json utilitzant alguna de les llibreries existents. Veient la implementació d'altres APIs i en general, coneixent el domini de les bases de dades de grafs i havent treballat anteriorment en la creació de clients REST, s'arriba a la conclusió que els dos elements inicials que cal definir en forma de classes son els nodes i les relacions.

Els *nodes* són els elements bàsics de les bases de dades i s'identifiquen amb un número únic (identificador o clau primària) i una URL associada al recurs corresponent en la BDOG, a més, els nodes tenen un conjunt de propietats que es poden representar com una *array* de tuples del tipus (clau, valor) i una *array* de relacions (entrant i eixint) que

representen les connexions amb altres nodes. Per altre costat els nodes tenen una sèrie d'operacions naturals que podem representar mitjançant nodes com ara la modificació (creació/alteració/eliminació/obtenció) d'una propietat o les seues relacions.

D'altra banda, les *relacions* denoten vincles entre nodes i venen identificades al mateix temps per una URL i un número únic. Les relacions a més tenen un tipus obligatòriament i un node inicial i final, també caldria denotar si es tracta d'una relació unidireccional o bidireccional (*boolea*). Com també hem definit en els nodes, les relacions poden tindre propietats que es vorien representades pel mateix vector de tuples i un conjunt d'operacions naturals relacionades amb les seues propietats, el tipus o els nodes inicials que podríem modelar com a mètodes públics. A banda d'aquests dos elements fonamentals, es podria considerar la necessitat d'altres abstraccions referents a les propietats o als índex, però es poden anar definint posteriorment seguint l'esquema que es va crear en funció de les necessitats futures.

Després de codificar les classes anteriors, caldria implementar els seus mètodes utilitzant d'una banda, el nucli de comunicació HTTP ja creat, creant les URLs en funció de l'objecte i l'operació a realitzar i d'altra banda caldria utilitzar alguna de les llibreries de *json* de la plataforma .NET com ara JSON.NET, JSON to .NET, fastJSON o JSONFx per tal de serialitzar els objectes C# en text en format vàlid per a poder enllaçar en la crida HTTP.

A més de tota aquesta feina per tal de crear aquest mecanisme elemental, cal considerar que cada crida/operació que ens permet utilitzar el servici REST de Neo4j utilitza elements molt diferents i per tant cal fer modificacions sobre el model inicial. A més, posteriorment s'ha de crear una eina de deserialització que permeta utilitzar les respostes de la base de dades i per altre costat cal definir, dissenyar i crear un mecanisme independent que s'encarregue de l'ús dels *traversals* necessari per a l'anàlisi d'impacte.

En definir la quantitat de treball a realitzar per tal de crear aquesta API, es va vore que la quantitat d'hores necessàries per a la seua implementació superava de llarg la quantitat d'hores disponibles. Cal considerar que aquest projecte s'ha implementat en

un conveni empresa-universitat i que el temps dedicat a l'empresa no s'inverteix completament en l'elaboració del nostre projecte, si no que la major part del temps, es dedica a la modificació del software de suport a la metodologia TUNE-UP que utilitza la companyia. Tenint en compte esta situació, es va decidir buscar alternatives a la nostra llibreria dins de la comunitat de desenvolupadors de Neo4j.

Encara que la llibreria finalment no es va acabar d'implementar, els esforços i temps emprat no s'han de considerar perduts, ja que els coneixements apressos en esta experiència van servir com a aprenentatge en l'ús de la base de dades i en la resolució de problemes futurs relacionats.

4.3.2 Alternatives a la API iniciada

Una vegada decidit que havíem d'utilitzar fonts externes per comunicar-se amb la base de dades, vàrem contactar amb la comunitat mitjançant la llista de correus de Neo4j preguntant l'existència de projectes oberts de clients REST en .NET. La contestació per part dels desenvolupadors no es va fer esperar (la llista de correus és molt activa, amb una mitja de 20 correus diaris) i ens encaminaren a tres llocs webs amb projectes a considerar.

El primer d'ells [14] tractava més d'un conjunt de consells respecte al correcte desenvolupament del nostre propi client, que d'un client en si. En aquest s'hi tractaven temes lligats amb l'arquitectura de la nostra llibreria i una xicoteta comparativa respecte a les llibreries externes a utilitzar per tal de treballar amb el format JSON.

Aquesta alternativa oferia un arxiu comprimit amb un seguit de fitxers de codi en el llenguatge de programació C# amb un xicotet exemple d'implementació d'un client. El principal problema d'aquesta opció es troba precisament en la seua codificació, ja que es tracta d'un exemple molt puntual, poc intuïtiu i còmode per treballar com a API per utilitzar des de la capa d'accés a dades. A més, per tal de poder utilitzar-lo en un projecte real cal estendre la seua funcionalitat amb un cost temporal tant alt com el que ens va fer abandonar la idea de crear el nostre propi client.

La següent alternativa [16] tracta d'un projecte software de codi obert allotjat en els servidors de la plataforma *codeplex*, es molt similar a l'anterior precisament perquè esta basat en els consells i el codi inicial que l'article anterior oferia. Tot i ser en un

principi l'alternativa més recomanada, el seu desenvolupament es va vore aturat abruptament sense cap publicació i deixant la llibreria sense funcionalitats tant bàsiques com la modificació i obtenció de propietats per als nodes i les relacions.

L'última alternativa a analitzar[15] es tracta d'un projecte acadèmic suec que, com la resta, també havia estat recentment abandonat quan vàrem decidir treballar amb la base de dades Neo4j. Aquesta llibreria tot i tampoc estar completament acabada, si que contenia una implementació de totes les crides bàsiques relacionades amb la creació, modificació, eliminació i obtenció de nodes, relacions i propietats, fet que va decantar la nostra decisió envers l'ús d'esta opció.

Una vegada decidit que començaríem amb la utilització d'aquesta API, es varen fer una sèrie de proves simples que ens permeteren comprovar si la implementació funcionava correctament i per tant era apta per a la utilització en el nostre mòdul per a l'anàlisi d'impacte. En aquests tests mitjançant les crides definides, es va poder crear satisfactòriament una estructura simple de nodes i relacions amb propietats, i juntament amb la senzillesa i usabilitat de la seua interfície de comunicació, va reforçar la nostra decisió i la dels responsables de l'empresa.

Amb el projecte amb llum verda i amb la definició del que seria la primera versió del model de dades, vàrem crear un prototip de base de dades seguint el nostre esquema, i vàrem intentar carregar-la amb informació que emulara el que nosaltres considerarem un comportament natural del mòdul d'anàlisi, però en aquest punt és on començaren els problemes. En intentar introduir una quantitat superior a 500 o 600 nodes prototips, el flux d'informació quedava completament aturat i la informació no s'emmagatzemava correctament. Els errors encara eren més greus ja que, en estar treballant amb software relativament nou, cal considerar que Neo4j acabava de publicar la seua primera versió release, i sense existir documentació associada al seu ús real en projectes, començarem a dubtar de l'estabilitat de la pròpia BDOG.

Després de moltes proves per trobar l'origen del problema, es va arribar a la conclusió que l'error estava en el codi utilitzat, i descartàrem així el possible dèficit de rendiment de la base de dades. Finalment trobàrem l'origen del *bug* en el nucli de comunicació de l'API. Com que la definició i arquitectura del nucli era molt semblant a la que nosaltres

havíem definit per a la nostra versió de la llibreria, vàrem poder utilitzar els coneixements i el codi previ per solucionar l'error. A partir d'aquest moment no hem tingut més problemes de rendiment respecte a la base de dades.

A banda del problema esmentat, s'han anat detectant altres errors de menor mesura en l'API utilitzada, referents a crides concretes de la seua interfície. Per exemple, entre les opcions d'obtenció de les relacions d'un node, hi ha una crida que permet passar com a paràmetre un enumerat amb el conjunt de tipus de relacions que volem considerar per al filtrat d'aquesta consulta, però desafortunadament aquesta funció no ens torna el resultat esperat.

En no haver utilitzat totes les funcions definides per la llibreria i en no existir una bateria de proves completa i correcta, ja que l'existent consisteix en un conjunt de proves unitàries molt bàsiques que no detectaven ni tan sols el nostre error inicial de rendiment, es molt complicat saber exactament l'estat i la usabilitat actual de l'API, i s'han trobat errors com aquest en anar utilitzant les possibilitats que ens ofereix. Tot i així hem decidit seguir utilitzant aquesta interfície que ens aporta uns altres valor afegits, i en cas de trobar-nos en altres errors relacionats amb alguna crida concreta, els anirem solucionant en funció de les necessitats i el temps de què disposem.

4.3.3 Característiques de la llibreria utilitzada

Les principals característiques que ens han fet decidir-nos finalment per aquesta llibreria de comunicació estan basades en la senzillesa i claredat de la seua interfície pública. Aquesta segueix una estructura general similar a la que ens vàrem plantejar en l'etapa de creació de la nostra pròpia API, basant totes les crides en la figura del node i la relació. Sobre aquestes es poden executar comandes de modificació amb ells mateixos o sobre propietats relacionades, açò permet que els elements de d'intercanvi d'informació siguen molt intuïtius des d'un punt de vista orientat a objectes com el que es treballa en projectes .NET, i més concretament C#.

Un altre aspecte a considerar són els missatges de tornada de la base de dades i la facilitat amb la que podem explotar la informació consultada. La llibreria utilitzada ens torna d'una manera simple, mitjançant una variable *booleana,* el resultat satisfactori o advers de l'operació realitzada en el cas de tractar-se d'una inserció, modificació o

eliminació. D'altra banda, en el cas de tractar-se d'una consulta, aquesta API de Neo4j ens aporta un mètode de consulta dels resultats utilitzant el Language-Integrated Query (LINQ) de *Microsoft*.

LINQ segons la definició que podem trobar en la web del projecte, és un conjunt d'extensions per al Framework .NET que engloba consultes integrades en el llenguatge, conjunts i operacions de transformació. És una extensió de C# i Visual Basic amb una sintaxis nativa per a consultes, i aporta llibreries de classes per aprofitar els seus avantatges.

En definitiva, la integració d'aquest framework amb l'API utilitzada de Neo4j mitjançant la llibreria JSON.net, ens facilita la manipulació de les dades retornades per la base de dades i aporta un conjunt de ferramentes addicionals que ens permeten utilitzar nous filtres o sistemes d'ordenació des de la part del client, açò es especialment útil en relació a operacions no natives des d'un punt de vista orientat a grafs que tendeixen a ser molt costoses, però que son molt reduïts seguint els patrons de LINQ. Per posar un exemple concret, la cerca exacta per text no es una operació nativa de Neo4j que tinga associada una crida en http. Tot i ser possible la seua definició mitjançant la codificació de filtres en *Javascript* dins el JSON associat a un *traversal,* pot ser més simple i òptim en determinades situacions definir amb Neo4j el conjunt de dades sobre el que es vol aplicar el filtre de text i posteriorment amb la clàusula *Where* de LINQ seleccionar aquells termes que acompleixen els criteris definits.

Si es ben cert que aquesta llibreria te entre els seus punt forts la facilitat d'ús i el complet conjunt d'operacions disponibles, també cal remarcar alguna de les seues deficiències, com ara el fet que les crides d'entravessament o *traversals* no funcionen correctament o ho facen de manera molt limitada, tot i ser una de les ferramentes claus en l'apartat de consultes i explotació de la informació.

Per tal d'aportar solucions respecte al recorregut del graf, en les primeres versions del software es va implementar un algorisme iteratiu molt poc eficient, basat en la crida node.*getRelations*(...) dels nodes implicats en l'anàlisi d'impacte. Posteriorment s'ha intentat estendre la funcionalitat de l'API internament per suportar tot allò referent a

les crides *traversal*, però la manca de temps i la rigidesa de l'estructura de la llibreria no han permés moltes vegades dur-ho a terme.

Cal destacar que en començar el projecte les prioritats dels responsables estaven centrades en l'anàlisi d'impacte pròpiament dit, però mentre passava el temps, el criteri s'ha vist modificat fins l'actualitat, on els majors esforços estan centrats en el desenvolupament del gestor de termes i en general en la definició i l'ús dels elements de domini per part de tots els agents de la companyia.

Aquest canvi en els criteris ha vist minvat el temps dedicat a l'anàlisi d'impacte i els seus criteris i per tant, per tal de poder dur a terme les consultes utilitzant els *traversals*, s'ha decidit utilitzar únicament el nucli de l'API per a la seua realització. Per tal de portar-lo endavant i utilitzar les possibilitats que ens aporta l'arquitectura de la llibreria, s'ha decidit definir la URL i el text en format JSON associat a la crida a la base de dades, i crear manualment la petició HTTP que s'enviarà al servidor i filtrar posteriorment el resultat retornat amb la mateixa llibreria JSON utilitzada per la llibreria.

Per últim ens agradaria remarcar que la importància real del projecte resideix en la infraestructura creada i la seua integració amb la ferramenta de suport, més que en el propi algorisme d'anàlisi d'impacte que tot i ser l'objectiu final del nostre projecte, es pot redefinir i evolucionar en el temps modificant criteris com ara nivell de profunditat d'impacte o camp de l'impacte en la definició de les proves d'acceptació, simplement canviant la consulta o *traversal* associat.

5 Mòdul de l'anàlisi d'impacte

Tot allò desenvolupat i explicat en el punt anterior defineix la infraestructura del nostre projecte creada per tal de dur a terme el nostre objectiu principal: l'anàlisi d'impacte dels requisits.

Fent-ne un resum general, podem dir que per tal de reduir costos de desenvolupament i manteniment, volem tindre una ferramenta d'anàlisi que ens permetrà, en les etapes inicials dels nostres projectes, tindre informació que ens facilitarà la pressa de decisions i el disseny enfront de canvis o de millores en les especificacions del nostre producte.

La metodologia utilitzada en l'empresa, amb la que s'ha treballat per implementar el mòdul, defineix aquesta especificació mitjançant requisits software en forma de proves d'acceptació, i es caracteritza per tindre'n una gestió mitjançant un esquema de contenidors (nodes) jerarquitzats seguint una estructura de graf acíclic dirigit, d'altra banda les proves d'acceptació segueixen un flux basat en estats (nou, modificació, regressió, eliminació o actual) que defineix els seu cicle de vida. El nostre nou mòdul d'anàlisi d'impacte ve integrat dins aquest context de definició dels requisits i esta especialment dissenyat pensant en el sistema de treball de l'empresa.

Com hem pogut observar en el punt anterior, s'han creat diversos elements per sostindre el nostre mòdul d'anàlisi, però per tal de visualitzar d'una manera simple com interactuen entre ells per dur a terme el nostre objectiu prioritari, ens basarem en un parell d'exemples que ens conduiran per l'activitat diària dels agents del grup de desenvolupament i mostren com interactuar amb el nou mòdul d'anàlisi d'impacte.

Així mateix, definirem els criteris i el comportament actual de l'anàlisi, puntualitzant possibles alternatives al model i les seues conseqüències. A més, s'explicaran els actuals nivells d'anàlisi d'impacte i els elements involucrats en el procés.

5.1 Visualització de l'AI

Per tal de mostrar l'ús del nostre mòdul de l'anàlisi d'impacte, utilitzarem com a base el domini d'exemple que vàrem definir en el punt 2.3.1 d'aquest treball. Hi vàrem situar un domini relacionat amb elements bàsics d'una xarxa social, com poden ser: un *compte d'usuari,* una *petició d'amistat,* o el propi concepte d'amistat. Per tal de recordar la seua definició tornem a mostrar la imatge ja que ens acompanyarà per tota la definició dels exemples utilitzats.

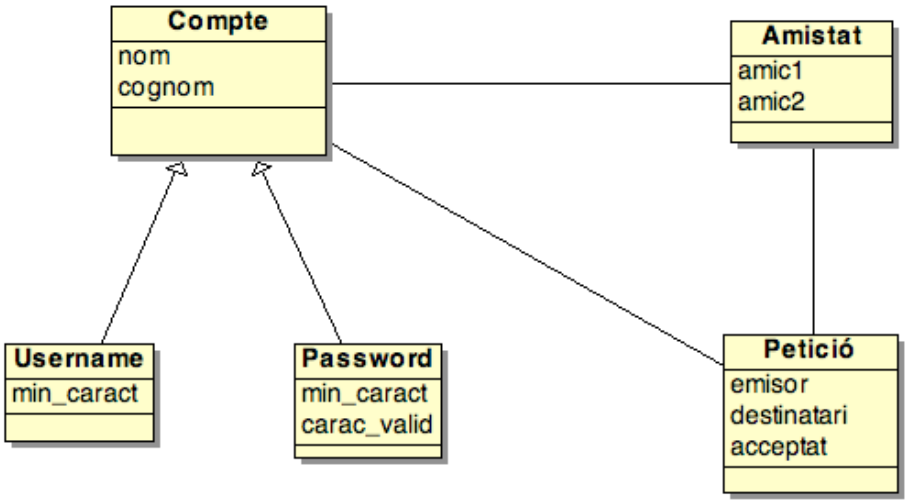

FIGURA 3: Model del domini

Comencem posant-nos en la pell d'un dels analistes de la xarxa social que estem desenvolupant. El client en un intent per millorar la seguretat de la tota la informació relacionada amb els usuaris, ha decidit enfortir els criteris de validesa de les claus d'accés. Després d'una reunió entre els dos agents, s'ha decidit que la contrasenya d'inici de sessió podrà ser un conjunt alfanumèric amb una longitud mínima de 12 caràcters. Aquesta modificació s'ha d'implementar per exemple en la creació d'un usuari dins la xarxa social.

Abans de començar a definir la prova o proves d'accés relacionades amb el nou requisit hem d'anar a la seua actual estructura de requisits per seleccionar els nodes que es voran afectats amb la nostra modificació i vore si actualment ja existeix alguna prova d'acceptació relacionada amb allò que volem definir o si, per contra, hem de definir una proposta nova.

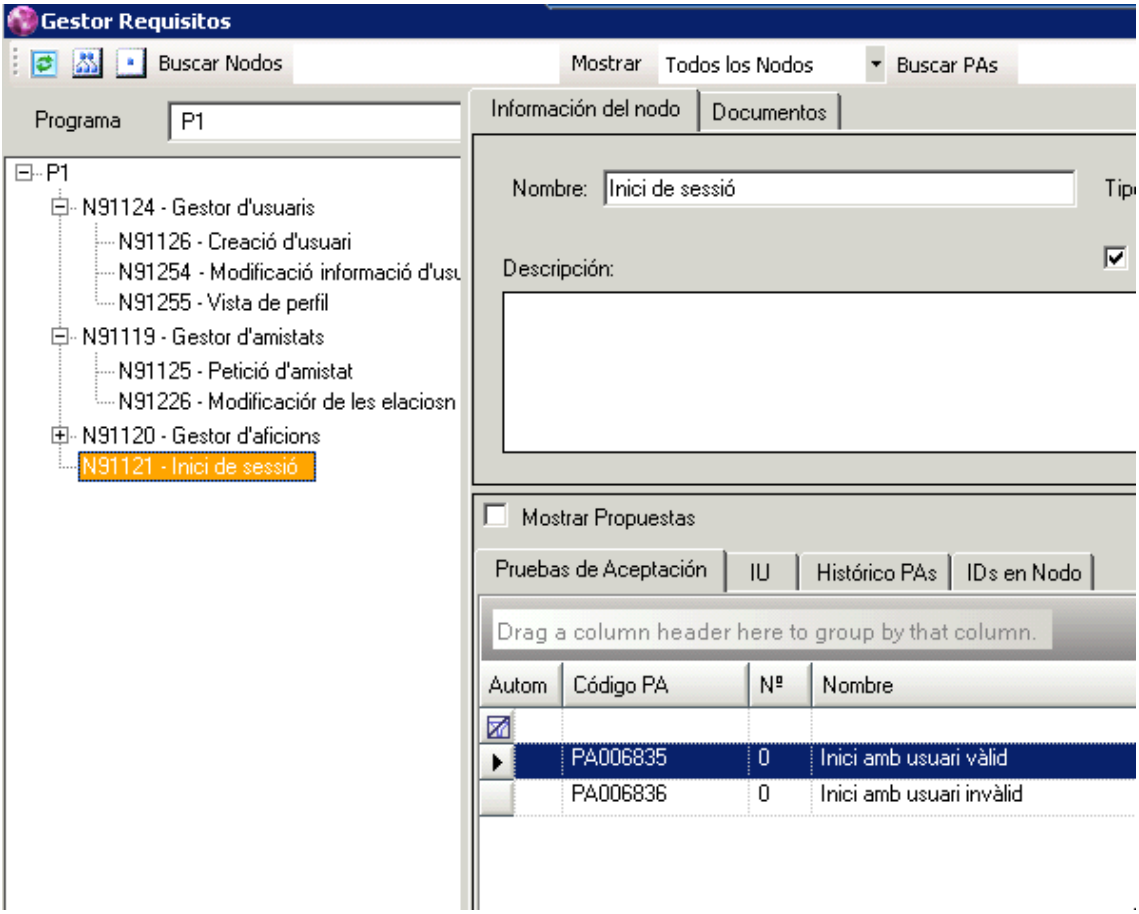

FIGURA 13: Vista del gestor de requisits per al nostre exemple

Com podem vore en la imatge superior, en la part dreta tenim una representació en forma d'arbre de la nostra jerarquia de requisits. En aquesta es poden vore els contenidors de PAs relacionats amb el domini de les xarxes socials. A la part dreta, tenim la informació del node contenidor i el conjunt de proves i interfícies gràfiques associades a aquest.

Per al nostre cas particular, com a analistes, definiríem que el nou requisit esta relacionat amb el de creació d'un usuari com ja havíem decidit anteriorment i ens n'adonem que el node d'inici de sessió també esta relacionat, ja que es tracta d'una activitat que precisa de l'aportació d'una clau d'accés. Una vegada seleccionem el node de Creació d'usuari, trobem que ja existeixen dos proves d'acceptació conformant el node i poden contindre la nova informació que ens ha aportat el client. En obrir la PA anomenada *Creació d'usuari amb informació vàlida* trobem la definició amb els criteris de validesa d'un nou compte.

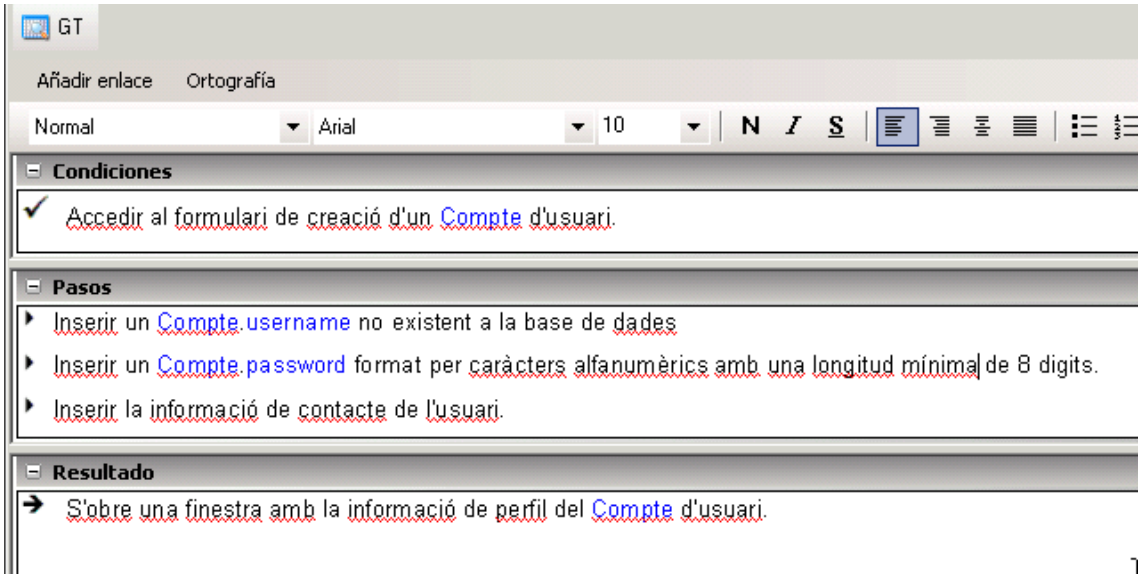

FIGURA 14: Descripció de la PA 'creació d'usuari amb informació vàlida'

Com podem vore a la imatge superior, en esta PA tenim definida una condició inicial de navegació per tal d'accedir directament al formulari on estem reflectint la funcionalitat establerta. A continuació es mostren un conjunt de passos relacionats amb la introducció de la informació requerida per dur a terme l'acció i finalment se'ns mostra el resultat esperat, que en aquest cas ve relacionat amb la representació de la informació de l'usuari que s'està creant.

Entre la informació que s'ha d'introduir per tal de crear el compte, també apareixen criteris de viabilitat dels camps obligatoris, i entre ells la prova te definit que el caràcter mínim de les claus d'accés o passwords es de 8 dígits, diferent al criteri actual. D'altra banda, com es pot apreciar en la mateixa frase, el terme *Compte.password* apareix remarcat en blau, aquesta decoració ens indica que aquesta PA té una relació de traçabilitat amb aquest terme o atribut. En exemples posteriors mostrarem com crear les relacions.

Per tal de definir el canvi pactat, modificarem la descripció de la PA per tal de reflectir el nou criteri de 12 dígits, a més, en aquest cas la variació també esta afectant a la definició del propi terme de domini i per tant caldria comprovar les seues característiques des del gestor de termes. Per fer-ho accedirem a ell mitjançant el botó etiquetat amb "GT", situat a la part superior del formulari de proves d'acceptació.

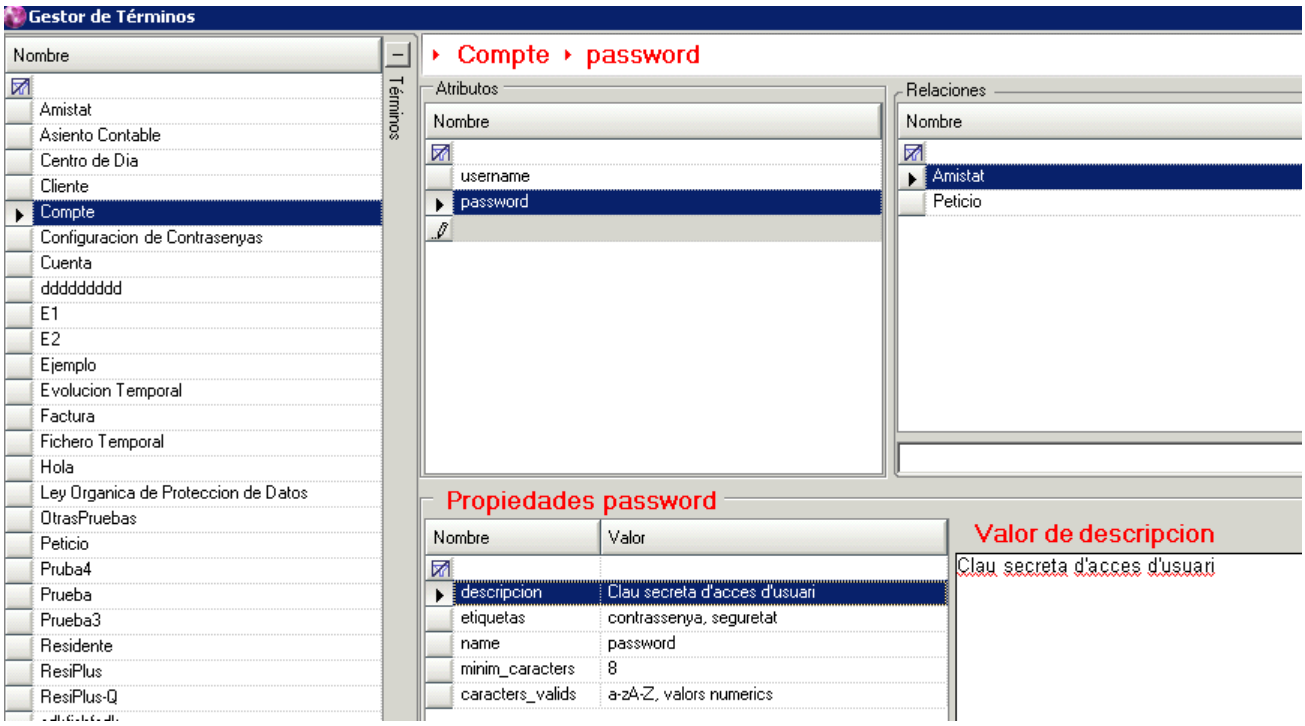

FIGURA 15: Gestor de termes amb les característiques de l'atribut password

Com es veu en la imatge superior el canvi sol·licitat també afecta al terme de domini i per tant caldrà reflectir-ho també ací. En fer aquesta modificació sobre el gestor, com el conjunt d'elements del domini està centralitzat en un servidor extern, es voran reflectits els canvis per a tots els agents del grup de desenvolupament i per tant, aquesta informació facilitarà la correcta implementació del software ja que tots els agents tenen accés a la informació.

Tornant al punt de vista de l'analista, un vegada definida la primera prova d'acceptació del nou requisit podríem començar a utilitzat la nostra anàlisi d'impactes sobre la PA que acabem d'escriure per tal de trobar altres proves que es poden vore afectades pel canvi. Per fer-ho, seleccionarem l'opció inclosa en el propi formulari de proves com es pot vore en el cantó en la dreta de la imatge inferior.

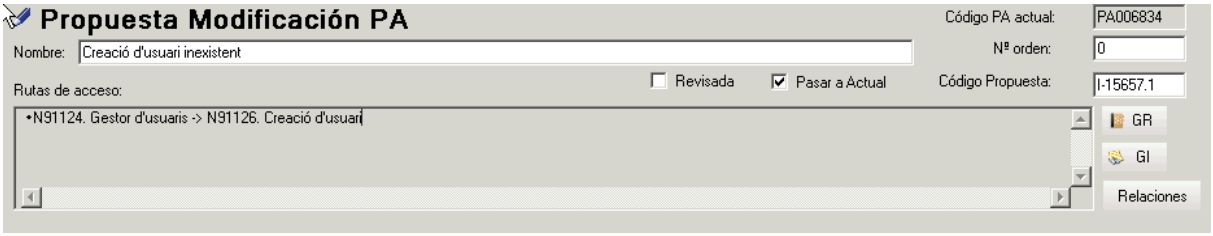

FIGURA 16: Capçalera del formulari de PAs

En seleccionar l'anàlisi de les relacions de la PA el mòdul busca totes aquelles relacions de traçabilitat definides en la descripció i busca per a cada una de elles, tota la resta de proves que també contenen referència al terme afectat.

En el nostre exemple anterior relacionat amb la prova "Creació d'usuari amb informació vàlida" podem vore en la seua descripció, com hi ha relacions de traçabilitat amb 3 sentits del domini: Compte, Compte.username i Compte.password. De totes maneres, com en el nostre cas concret la modificació afecta sobretot al terme *Compte.password*, hem decidit mostrar un exemple un poc més reduït.

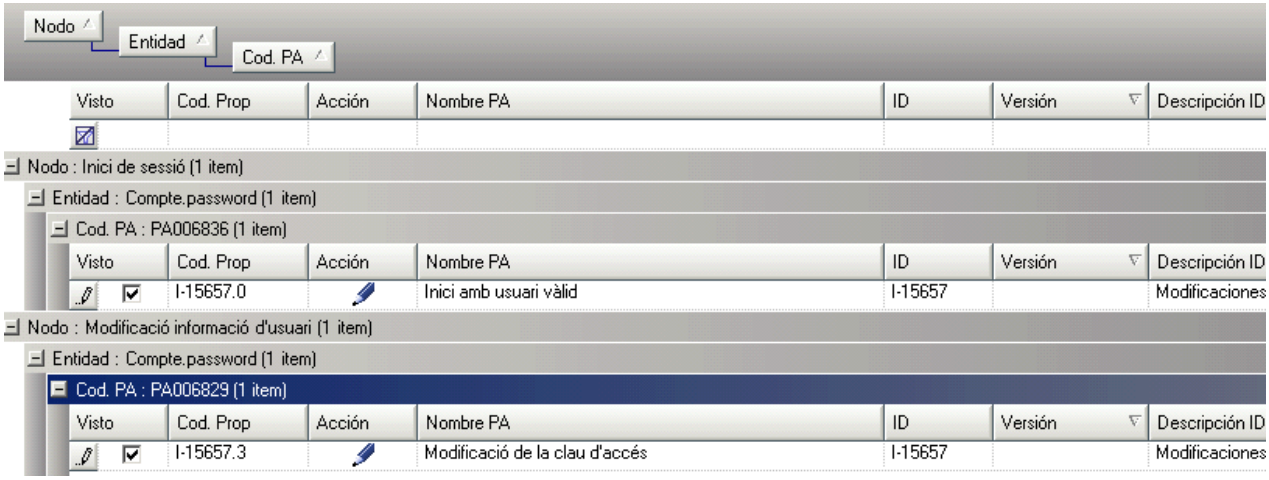

Figura 17: Anàlisi d'impacte sobre una PA relacionada amb el terme Compte.password

Com es pot vore en la imatge superior, s'ha decidit representar la informació inicialment basant-nos en la figura del *grit* multinivell, ja que es tracta d'una ferramenta senzilla de gestionar i a més el personal de la companyia esta acostumat a treballar amb aquest element gràfic. Inicialment el *grid* apareix agrupat en 3 nivells: primer *nodes*, després *entitats* i finalment *PAs*, per tal de tindre una visió més simple i organitzada de les relacions.

Seguint la imatge podem definir que la nostra PA simplificada, esta relacionada amb el node "Inici de Sessió" mitjançant l'entitat "Compte.password" que està definit en la prova I-15657.0; es tracta d'una proposta de modificació sobre la PA006836 definida en la unitat de treball I-15657 i que s'anomena "Inici amb usuari vàlid", i a més està relacionada amb el node "Modificació informació d'usuari" mitjançant el mateix terme

però relacionat amb la prova I-15657.3, proposta de modificació sobre la PA006829 definida en la mateixa unitat de treball i anomenada "Modificació de la clau d'accés".

Encara que en un primer moment tota aquesta informació pot semblar inintel·ligible i caòtica, tota aquesta terminologia es comú dins de la metodologia TUNE-UP i està forçament estesa dins dels desenvolupadors de l'empresa.

Aquesta informació es pot simplificar dient que la prova que acabem de definir està relacionada mitjançant el terme "Compte.password" amb les propostes "Inici amb usuari vàlid" i "Modificació de clau d'accés". La resta d'informació relacionada amb els codis de les proves, les propostes, els nodes o la unitat de treball associada pot ser ignorada, ja que el mòdul ofereix una integració completa amb la resta del software de suport TUNE-UP i proporciona accessos directes als diferents formularis relacionats amb els elements anteriorment esmentats.

Amb aquesta simplificació sí que podem observar el valor real de la nostra anàlisi d'impacte, ja que davant un canvi en els nostres requisits i després de definir únicament una prova d'acceptació, tenim informació real sobre unes altres parts dels requisits que es voran afectades per aquesta decisió. Després de vore l'exemple, pot semblar molt senzill fer una relació entre tots aquests elements, però cal tindre en compte que davant de grans sistemes complexos com és el principal software que produeix la companyia per a la que s'ha implementat, hi ha milers de proves a considerar i les connexions no son tan obvies.

Cal destacar també que existeix un altre nivell d'anàlisi que en compte de centrar els esforços sobre una única prova d'acceptació ho fa respecte a un node que, si recordem, es defineix com un contenidor de PAs. Aquesta altra opció pot ser molt interesant davant dels grans canvis estructurals que centren els màxims esforços en un únic node o en un conjunt reduït de nodes, i per tant és en aquest nivell abstracte on té més sentit aplicar el nostre estudi per obtindre una imatge més global de les conseqüències dels nostres possibles canvis.

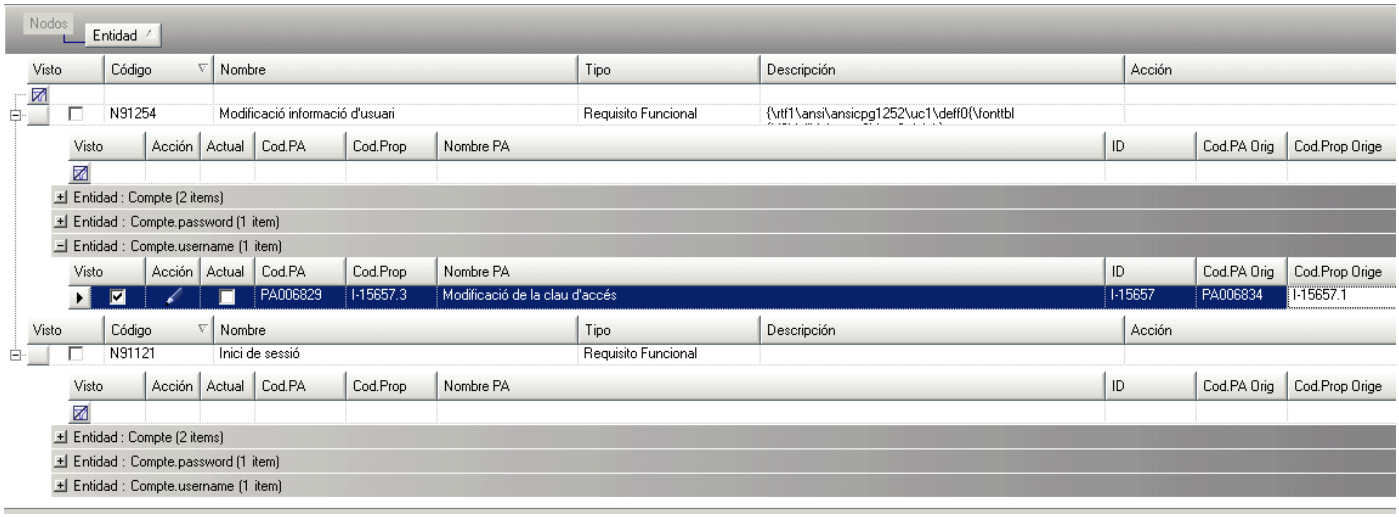

FIGURA 18: Formulari d'anàlisi d'impacte sobre un node

En la imatge superior podem vore el resultat que ens oferiria l'anàlisi d'impacte sobre el node que conté la PA dels nostres exemples i que d'una manera semblant a la anterior, però sense entrar en massa detalls ens mostra, seguint el format basat en *grid,* un conjunt de nodes afectats a causa de les proves internes i els termes de domini que hi contenen. Per últim destacar que en aquesta anàlisi es tenen en compte tant les PAs origen afectades (les que formen part del node) com aquelles PAs externes afectades (contingudes en altres nodes).

A continuació, i tornant al punt de vista d'un dels analistes de la companyia, anem a definir un altre exemple d'ús del nostre mòdul per acabar de mostrar la funcionalitat que ens aporta. Començarem definint el nou requisit que el possible client de la xarxa social que estem desenvolupant ens ha demanat: les peticions d'amistat entre usuaris registrats han de poder contindre (opcionalment) un text de sol·licitud per tal de facilitar la seua acceptació i en general el reconeixement de la persona implicada en la petició.

Si recordem la imatge relacionada amb el model de domini, un dels termes descrits era el concepte de petició, però en la seua representació inicial només es mostren propietats relacionades amb els dos usuaris involucrats i amb la seua acceptació.

Per tal d'introduir el nou requisit de text dins de la petició, podríem modelar-ho directament com una propietat interna del terme, enunciant que les peticions han de tindre associat un text de sol·licitud, però aquest disseny ens limita molt la seua

funcionalitat, ja que no tenim cap manera simple de definir que l'element és opcional o quines son les seues característiques, ni tampoc podrem crear relacions amb el concepte de text . Una solució millor per tant es basaria en definir el text de la petició com un atribut associat al propi terme *Petició*, amb una sèrie de propietats associades al text com ara la seua dimensió màxima, definint així les seues característiques i donant opció a crear relacions amb esta nova entitat.

Des del punt de vista de l'analista, en rebre la descripció del nou requisit per part del client, començaria pensant sobre quin node o nodes de l'arbre de requisits es vorien centrats aquests canvis, que en este cas concret es tractaria del node "Petició d'amistat". En cas de no existir es crearia i es continuaria amb la creació d'una prova nova que definira el funcionament d'aquesta opció.

Durant el procés de definició de la nova descripció, en voler relacionar la prova amb el text de la petició, se n'adonaríem de la inexistència d'aquest element gràcies a la ferramenta que explicarem a continuació i per tant mitjançant el gestor de termes que ja hem explicat anteriorment definiríem el nou concepte, com podem vore en la imatge inferior. Una vegada definit començaríem a escriure les *Condicions*, els *Passos* i els *Resultats* que descriurien la nova funcionalitat al mateix temps que crearíem les relacions de traçabilitat amb les entitats corresponents.

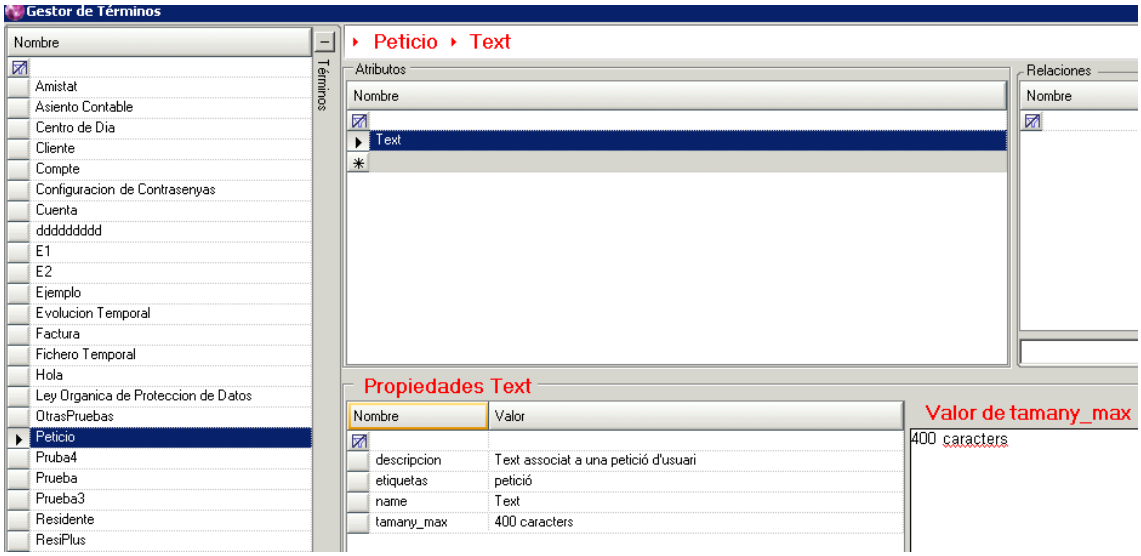

Figura 19: Gestor de termes mostrant les característiques de l'atribut 'Text' associat al terme 'Petició'

Per tal de crear aquestes relacions de traçabilitat s'ha optat per desenvolupar una utilitat pròpia que, inclosa en el control d'usuari encarregat de la definició textual de la prova d'acceptació, actue com una funció d'autocompletar o com s'anomena normalment *IntelliSense,* que es tracta d'una implementació que s'utilitza en programes de desenvolupament integrat com Eclipse o Visual Studio. Aquesta elecció ve fonamentada entre altres raons, per la seua gran senzillesa i usabilitat així com pel fet que tots els desenvolupadors del grup de treball siguen conscients del seu ús gràcies a la ferramenta de Microsoft esmentada anteriorment.

Una vegada elegit el sistema amb el qual els usuaris podran definir les relacions de traçabilitat s'ha de remarcar quina serà l'organització que es s'utilitzarà per a la seua explotació. En el nostre cas s'ha optat per una estructura per nivells, on en el primer es mostraran tots les entitats de domini definides en el gestor de termes i on cada un dels seus subnivells associats oferiran totes aquelles entitats i atributs associats/relacionats amb el terme del primer nivell. En futures modificacions del mòdul, si finalment s'afegeixen els *Nodes* i les *PAs* com a elements per a l'anàlisi, s'influirà un nou nivell inicial que filtrarà els elements de primer nivell en seccions, per tal d'ajudar a la cerca d'allò que volem relacionar.

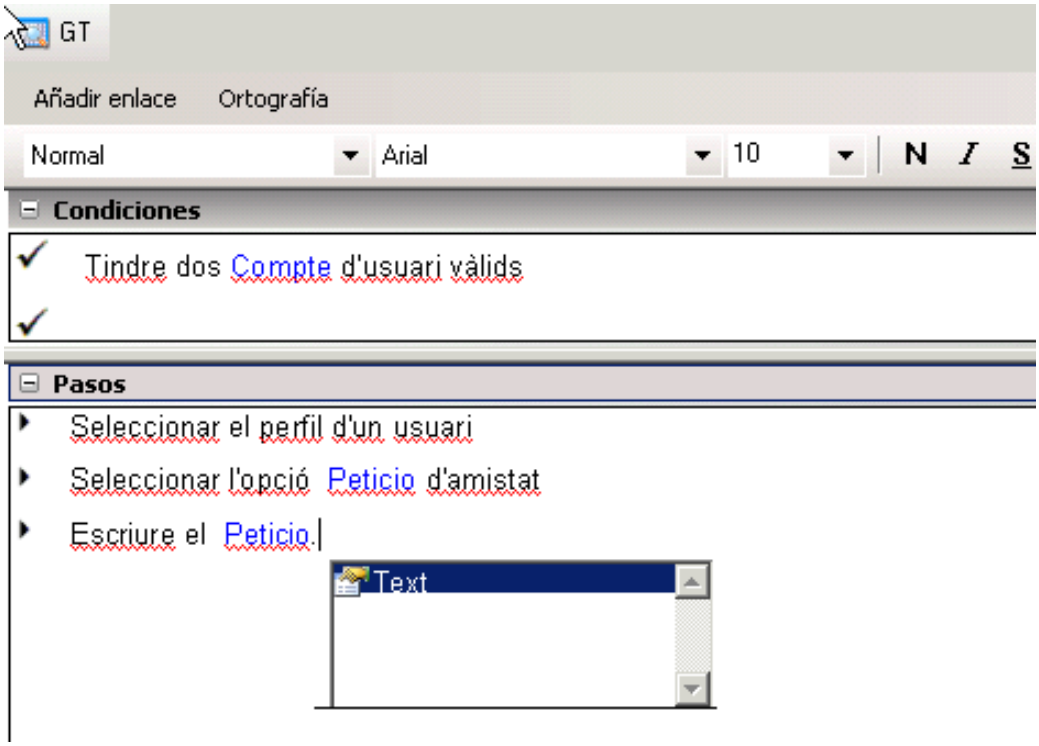

FIGURA 20: Exemple de creació d'una relació de traçabilitat mitjançant intelisense

En la imatge superior podem vore la utilitat explicada, aplicada a l'exemple que hem estat seguint. Després de definir relacions amb els termes *Compte i Petició*, l'usuari esta creant una nou vincle amb l'atribut *Text* que vindrà representat per la cadena "Petició.Text".

Finalment caldria remarcar que a causa de l'arquitectura utilitzada, les relacions per a l'anàlisi que es fan sobre un element poden tindre diferents camins d'accés, per exemple el concepte Text que hem estat utilitzant anteriorment podria tindre referències mitjançant les cadenes "Petició.Text", però també "Compte.Petició.Text" ja que hi ha una relació des de l'entitat Compte fina l'entitat Petició. Actualment no s'està fent cap tipus de distinció entre ambdues terminologies en l'anàlisi d'impacte, ja que en la base de dades la relació de traçabilitat es fa directament entre la PA i l´últim node representat en la cadena i no es té en compte la ruta d'accés. En posteriors versions s'espera aquesta nova funcionalitat emmagatzemant el camí com a propietat de la relació per poder filtrar els resultats amb un nou criteri.

5.2 Criteris d'anàlisi actuals i alternatives

En els exemple anteriors hem vist com, davant els canvis en una PA i tenint una estructura ben definida de relacions de traçabilitat que afecten aquesta prova, podem trobar amb molta facilitat uns altres elements que es voran afectats per aquesta modificació. Per tal de fer possible aquesta anàlisi doncs, és completament necessari que el nostre sistema o la nostra jerarquia de requisits estiga ben informada, és a dir, amb un ampli conjunt de relacions entre proves d'acceptació i entitats del model de domini.

Aquesta és una tasca àrdua i complexa que no es pot implementar en un temps definit, però que amb la seua integració i el seu ús dins de la ferramenta de suport anirà millorant gradualment la riquesa de l'anàlisi d'impacte.

Amb un sistema complex interrelacionat, es pot donar la situació que amb els nostres criteris bàsics d'anàlisi els resultats esperats no siguen satisfactoris. Açò es pot deure a un elevat nombre de connexions entre PAs, que ens aporte una quantitat tan gran de relacions d'impacte que no ens siga realment útil.

Hem definit precisament una primera implementació simple, ja que els algorismes utilitzats per a l'anàlisi, han de ser entitats flexibles que evolucionen amb el temps en funció del conjunt d'informació emmagatzemada o els criteris establerts pels responsables de la companyia. A més, en la fase actual en la que es troba el projecte s'ha decidit donar importància a la part del mòdul encarregada de la inserció de noves relacions de traçabilitat i integració en el software TUNE-UP.

Seguint esta idea bàsica i simple s'ha definit per tant un algoritme d'explotació basat en aquests criteris fonamentals:

- Per a l'anàlisi sobre les relacions d'una PA, es tornaran totes aquelles proves relacionades directament amb els termes inclosos en la definició de la PA analitzada, independentment del camp on s'hagen definit dins de la descripció.

- Per a l'anàlisi sobre les relacions dels Nodes, es tornarà el conjunt de nodes que continguen les proves relacionades amb les PAs contingudes en el node analitzat seguint els mateixos criteris que en l'anàlisi d'una única PA, i remarcant la prova origen (del node analitzat) i destí (nodes afectats) de cada una d'elles.

Aquesta primera versió pot ser d'utilitat en un sistema inicial on la quantitat de interconnexions no siga molt elevada, però amb sistemes més rics aquests criteris ens aportaran un nombre tan gran de relacions que no es podrà traure profit real de la informació i per tant seria contraproduent. Per tant, en un sistema com el nostre una part important és l'evolució dels algorismes d'explotació, que s'han d'anar desenvolupant i madurant al mateix temps que va creixent la base de dades orientada a grafs. Es per açò que la flexibilitat del nostre model i al mateix temps l'estabilitat de la infraestructura creada ens faciliten aquesta tasca, es precisament en aquest punt on resideix la importància del nostre mòdul d'anàlisi d'impacte.

Dins d'aquesta possible evolució algorítmica es poden tindre en compte diversos criteris que, en major o menor mesura, ja s'han anat comentat en aquest treball. Per un costat, una característica important a tindre en compte és el nivell de profunditat

en l'explotació de les relacions. Actualment podríem dir que utilitzem un únic nivell de profunditat marcat per les relacions directes entre PAs i termes, però també es podrien definir altres criteris amb més profunditat on termes molt relacionats entre si donarien com a resultat relacions indirectes de segon nivell.

D'altra banda, un possible filtrat a tindre en consideració podria vindre caracteritzat per la secció on s'ha definit la relació de traçabilitat dins de la descripció de la prova d'acceptació. Encara que en un primer moment siga estrany fer aquesta distinció, pot ser una característica interesant per garbellar la gran quantitat de resultats obtinguts. A més, s'ha de tindre en compte que no es igual una relació basada en els passos de la prova (gran importància) que en algun altre camp, com per exemple les observacions.

A banda com havíem comentat anteriorment també podem utilitzar la ruta d'accés al terme, dins el text de la descripció, per decidir la importància de les relacions. Aquesta afirmació també ens obre la porta a definir la possibilitat de mostrar la informació per nivells de prioritat en funció dels criteris ja esmentats, de tota manera aquestes implementacions s'han d'anar estudiant i discutint amb els responsables de l'empresa dins del context .

Seguint esta última idea de les prioritats, es poden anar definint internament en el gestor de terme associant nivells d'importància a les entitats, ja que no tots els termes utilitzats en la creació de relacions poden tindre una diferent preponderància i fins i tot no ser interesant des del punt de vista de l'explotació. Finalment destacar que com aquestes idees, en podem definir moltes altres que basen la seua implementació en la modificació parcial de l'estructura de les relacions, com ara afegir propietats a les relacions més destacades. Encara que aquestes mesures poden tindre un cost més elevat a nivell d'implementació, també poden aportar-nos una gran complexitat en l'anàlisi. Totes aquestes possibles modificacions s'aniran implementant amb el temps per vore el seu impacte real en un sistema complex com el que esperem.

6 Conclusions i treball futur

En l'actualitat les empreses de desenvolupament software produeixen projectes en un ambient o ecosistema inhòspit, on els clients exigeixen cada vegada aplicacions més completes, estables i que els represente un menor cost, tant temporal com monetari. Aquesta situació obliga a emprar ferramentes avançades de codificació i suport i noves metodologies que permeten reduir les despeses de desenvolupament i manteniment de les nostres creacions.

Dins d'aquesta política la companyia, amb la que hem treballat per implementar aquest projecte, va decidir, anys enrere, treballar conjuntament amb la Universitat Politècnica de València per tal de crear una metodologia i ferramenta àgil que els ajudarà en el seu treball diari. D'aquesta unió es va concebre el que coneguem actualment com TUNE-UP. Després de més de huit anys treballant amb aquesta metodologia i gràcies als bons resultats que ha suposat la seua implementació en la manera de treballar dels seus agents, ha portat a ambdues institucions a seguir apostant per la seua evolució.

Des de les primeres versions s'han anat afegint diversos mòduls, entre els que cal destacar el gestor de requisits que ha passat a ser una part essencial del flux de treball dels agents i és, sense cap mena de dubte, la secció on més esforços s'han destinat durant els últims mesos. En aquest context podem englobar el nostre mòdul d'anàlisi d'impacte dels requisits, que basa la seua arquitectura en l'actual estructura jeràrquica dels requisits per tal d'aportar una nova ferramenta avançada, que permetrà als desenvolupadors dur a terme les seues tasques d'una manera més informada.

Conèixer les conseqüències que les possibles millores o nous requisits tinguen sobre el producte actual abans d'implementar-les, és un gran avantatge que pot marcar una diferència en les fases d'anàlisi, implementació i testeig. Així, els desenvolupadors poden centrar d'una manera més òptima els seus esforços en les àrees del software que es vegen realment afectades pels canvis proposats, i minimitzar el temps que

normalment s'empra en fer una cerca manual i el que és més important, reduir els costos associats a les etapes de manteniment.

El nostre treball s'ha desenvolupat incrementalment seguint les necessitats i les indicacions directes dels responsables de l'empresa. Cal considerar que encara que el nostre mòdul ja siga operatiu i estiga en funcionament, el conjunt del projecte encara es troba en fases inicials i necessitarà de moltes iteracions i una forta utilització per tal de comptar realment amb una eina madura.

El projecte s'ha desenvolupat dins un marc de col·laboració empresa-universitat on s'han invertit, sense comptar les hores emprades en altres projectes relacionats amb la ferramenta de suport de TUNE-UP, una mitja de huit hores setmanals per la seua realització. Es tracta d'un temps molt limitat per dur a terme el global del projecte, si tenim en compte que moltes d'aquestes hores es varen utilitzar per fer un estudi previ de les diferents alternatives i, una vegada feta l'elecció, per posar a punt la base de dades orientada a grafs que es trobava encara en fases inicials. Així i tot s'ha aconseguit implementar una primera versió del mòdul completament funcional que anirà evolucionant a mesura que augmentarà el seu ús i les necessitats específiques dels usuaris.

Per tal d'evitar problemes i situacions no desitjades a causa de la integració de la nova ferramenta en la metodologia de treball diari dels agents de la companyia, es va decidir aplicar una integració gradual dins l'empresa. Seguint aquesta idea, es va involucrar inicialment al departament d'I+D+i per tal d'analitzar el funcionament actual, aportar millores i aconseguir, en definitiva, una ferramenta madura que evitara pèrdues de eficiència en el conjunt de la plantilla pels possibles errors. Durant aquest últim any hem estat treballant conjuntament amb aquest departament a través de reunions periòdiques de seguiment, on es discutien aspectes molt diversos com la usabilitat d'alguns elements gràfics, l'anàlisi d'allò que ja estava en funcionament i el que és més important, la planificació de les accions futures. Després d'aquesta bona experiència col·laborativa, continuarem treballant conjuntament per millorar la ferramenta abans d'obrir-ho a la resta de departaments.

Una de les qüestions a tindre en compte respecte al projecte desenvolupat, és que no es pot comptabilitzar de cap manera simple i acurada la millora real que ens aporta. L'efectivitat del nostre mòdul es podrà començar a notar a mitjà i llarg termini, i tot i així, en tractar-se d'una ferramenta d'informació i ajuda a la pressa de decisions, solament podem utilitzar estadístiques relacionades amb les conseqüències esperades del nostre projecte com ara l'increment o la disminució en el nombre d'errors relacionats amb una versió o en l'eficiència temporal en la implementació de les unitats de treball. De tota manera, qualsevol estudi que férem relacionat amb aquests elements, ens aportaria únicament una lleugera idea no demostrable de l'impacte de la ferramenta, ja que hi han massa factors externs interferint en aquest tipus de característiques.

Tot i això, com que es tracta d'una eina demanada expressament pels propis agents de la companyia i desenvolupada seguint les seues necessitats, els usuaris estan molt satisfets amb els avanços. A més, independentment de la falta de resultats computables, s'espera una major productivitat per part de tots els agents de l'empresa ja que tant els analistes, com els programadors, com els testers podran detectar situacions i modificacions en fases inicials de desenvolupament que optimitzen i baixen costos en fases més tardanes con la de manteniment. A més a més, la ferramenta aporta als programadors i als analistes informació relacionada amb les àrees del codi que s'han de vore modificades agilitzant el procés de implementació.

Una altre avantatge per a la viabilitat del projecte i la satisfacció dels responsables, esta relacionada amb el baix cost que suposa aquest canvi per a la companyia. Com ja havíem especificat en seccions inicials d'aquesta memòria, l'experiència mostra que mantindre les relacions de traçabilitat per poder dur a terme una anàlisi d'impacte útil, moltes vegades suposen un cost tan elevat que la informació aportada no compensa l'esforç realitzat. En canvi, amb la nostra integració amb la ferramenta TUNE-UP, hem aconseguit que aquest manteniment es faça de manera natural seguint la metodologia diària a la que ja estan acostumats els agents, sense introduir-hi grans canvis. A més, la nostra implementació ha aportat noves ferramentes que integren altres recursos (com el registre d'entitats de domini), que ja s'estaven utilitzant dins de l'eina de gestió de

les unitats de treball, simplificant la gestió, optimitzant el temps relacionat amb aquestes tasques i permetent la seua explotació.

A nivell personal, he de destacar que un dels aspectes que més m'ha aportat en desenvolupar i implementar aquest projecte, ha estat el fet de conéixer i treballar amb Neo4j i en general amb les bases de dades orientades a grafs. Anteriorment sempre havia utilitzat emmagatzemament de dades basat en sistemes relacionals representats mitjançant diagrames entitat-relació. Però aquesta experiència m'ha permés conèixer uns altres paradigmes dins de la família coneguda col·loquialment com NOSQL, i m'ha aportat nous punts de vista i coneixements que podré utilitzar en futurs projectes.

En concret, he quedat gratament sorprès amb les prestacions que Neo4j ens aporta. Denota un sistema molt fàcil d'utilitzar i modelar, amb gran flexibilitat i que per a determinades situacions pot ser més òptim que els sistemes tradicionals, tant a nivell de costos computacionals com a nivell de gestió i desenvolupament. Actualment, l'únic punt negatiu comú a totes les bases de dades de grafs, es la falta de maduresa dels seus sistemes de gestió i ferramentes de suport. No obstant, amb l'increment de projectes relacionats amb xarxes socials i altres dominis on la representació natural esta basada en grafs, i amb el creixement de la comunitat de desenvolupadors i de les empreses interessades en el seu creixement, estic segur que en un període de temps relativament curt tindrem ferramentes de gestió que puguen competir amb les d'altres paradigmes més tradicionals.

Tornant amb el mòdul que hem desenvolupat i com ja s'ha comentat anteriorment el projecte actual es únicament la primera etapa d'un sistema complex que ha de mantindre una evolució constant. En seccions prèvies de la memòria ja s'han comentat quines podrien ser algunes de les tasques a realitzar en un futur i aquestes engloben seccions molt diferents del mòdul implementat.

D'una banda, a nivell d'infraestructura, s'ha d'optimitzar l'API de comunicació amb el servidor de base dades, per tal d'utilitzar les noves ferramentes implementades en Neo4j: índex, auto indexació, llenguatges de tipus query, millores en el traversal, etc. A més, cal enriquir el model de dades utilitzat per tal d'aportar més criteris de selecció i incloure, entre altres, els nodes i les PAs com a termes del model de domini.

També cal destacar modificacions relacionades amb la visualització i la usabilitat, on cal destacar una possible representació dels resultats en un format més net, basat en grafs. A més, s'ha de millorar la integració de l'actual mòdul per tal d'aportar més informació als programadors i als testers en el seu treball diari i la sincronització, mitjançant la creació de ferramentes, que permeten tindre actualitzades les descripcions de les proves en relació a les denominacions dels termes de domini. A banda de totes aquestes mesures concretes, s'han de continuar modificant els criteris utilitzats en l'anàlisi d'impacte per tal d'aconseguir l'evolució desitjada.

Finalment destacar que aquesta experiència m'ha motivat per continuar treballant en temes relacionats amb metodologies i eines de suport, i continuaré amb el projecte exposat per tal d'aconseguir una ferramenta madura d'anàlisi d'impacte de requisits que siga, senzilla d'utilitzar i mantindre, i que al mateix temps aporte resultats útils amb una representació clara i concisa. Com hem pogut vore, el camí per aconseguir-ho és llarg, però l'experiència acumulada en aquesta fase inicial del projecte ens ha deixat veure les majors carències del sistema actual i les millores que poden marcar la diferència.

7 Referències

1. ISO-International Organization for Standardization. *ISO 8402, Quality Management and quality assurance - Vocabulary*. 1986/1994. Substituïda i inclosa en la ISO 9000 l'any 2000.

2. Arnold, R. S., and Bohner, S. A., "Impact Analysis - Towards A Framework for Comparison," Proc. of the Conf. on Software Maint., pp. 292-301, Sept. 1993.

3. Company, M. Letelier, P. Marante, M. Suarez, F. Análisis de impacto en requisitos soportado de forma semi-automática y un marco de desarrollo TDD basado en pruebas de aceptación.2011.

4. Göknil, A and Kurtev, I. and van der Berg, K.G. Change Impact Analysis based on Formalization of Trace Relations for Requirements. European Conference on Model Driven Architechture (ECMDA) Traceability Workshop 2008, pp. 59 - 75.

5. Kilpinen, M.S. (2008). The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis. University of Cambridge. Cambridge, UK.

6. Malia S. Kilpinen and P. John Clarkson. Exploiting Change Impact Analysis to Support Sustainability. Univertity of Cambrigde, UK. 7th Annual Conference on Systems Engineering Research 2009 (CSER 2009).

7. Dick, J. (2005), “Design Traceability”, IEEE Software 22(6), pp.14-16

8. Bohner, S.A. & Arnold, R.S. (eds) (1996), Software Change Impact Analysis, Los Alamitos, California, USA. IEEE Computer Society Press.

9. Ambler, S. (2002), Agile Modeling: Effective Practices for Extreme Unified Process, New York, New York, USA: John Wiley & Sons.

10. Marante, M. Company, M. Letelier, P. Suarez, F. Gestión de requisitos basada en pruebas de aceptación: Test-Driven en su máxima expresión. XV Jornadas de Ingeniería del Software y Bases de Datos 2010.

11. Documentació i recursos de les webs neo4j.org i neotechnology.com, empresa creadora de la base de dades de grafs Neo4j.

12. Marante, M. Letelier, P. Suarez, F. TUNE-UP:Seguimiento de proyectos software dirigido por la gestión de timepos. XIV Jornadas de Ingeniería del Software y Bases de Datos 2009.

13. Angles, R. and Gutierrez, C. 2008. Survey of graph database models. ACM Comput. Surv. 40, 1, Article 1, (February 2008)

14. Mârtensson, M. Neo4j .NET Client over HTTP using REST and json. http://blog.noop.se/archive/2010/04/16/neo4j-net-client-over-http-using-rest-and-json.aspx

15. Wilhelmsson, P. Neo4RestNet. Client .NET de comunicació amb el servidor REST de Neo4j per a la versió 1.0. https://bitbucket.org/2hdddg/neo4restnet/

16. Conwell, J. Neo4jRestSharp. Client .NET de comunicació amb el servidor REST de Neo4j per a la versió 1.0 http://neo4jrestsharp.codeplex.com/