UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Estratègies de visualització de l'estructura de requisits d'un producte software

Autor: Guillem Medina Martínez

Director: Dr. Patricio Letelier Torres

Setembre de 2012

Treball Final de Màster presentat per acomplir amb els requisits finals per a l'obtenció del títol de Màster en Enginyeria del Software, Mètodes Formals i Sistemes d'Informació, de la Universitat Politècnica de València, 2012.

Agraïments

Vull expressar el meu sincer agraïment a Patricio, el meu director de tesi, 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. Sou els que sabeu exactament tot el que hem passat, i sense vosaltres la meua experiència universitària no haguera sigut el mateix.

Als meus pares

Taula de Continguts

Taula de Figures

Capítol 1. **Introducció**

1.1. Motivació

Gestionar els requisits d'un producte software en projectes amb una certa envergadura, pot ser una tasca verdaderament complexa. Ja siga definint els requisits mitjançant casos d'ús, histories d'usuari o proves d'acceptació, la seua evolució i manteniment tendeix a ser extremadament complexa per la pròpia naturalesa canviant del software. Mantindre la consistència entre els diferents elements, així com la completesa i la traçabilitat entre aquests no es una tasca fàcil si no s'utilitzen les eines adequades per a la seua correcta gestió.

Les tècniques i ferramentes actuals, ens permeten mantindre una evolució constant dels requisits software amb l'ús de cicles de vida sobre ells. Açò ens permet gestionar els requisits de manera àgil dins del procés de desenvolupament actual, però ens fa perdre el punt de vista sobre el desenvolupament global del producte. Una correcta anàlisi i representació de l'estructura de requisits del producte ens pot aportar la informació necessària per satisfer les dos vessants.

Tindre un coneixement profund de l'estructura de requisits del nostre programa, ens pot ajudar a prendre decisions més informades en relació a la gestió del nostre desenvolupament. Si seguim la dita popular *una imatge val més que mil paraules* i l'apliquem a aquest context, vindríem a reflectir que l'ús de diversos diagrames de visualització ens poden aportar amb un sol colp d'ull una gran quantitat d'informació útil en diverses fases del desenvolupament. Mostrar l'estat actual del testeig dels requisits pot ajudar a l'equip de testers a definir millor les seues activitats futures, una visualització de l'organització dels requisits i de les relacions entre ells pot beneficiar els analistes a l'hora de definir o modificar les especificacions, i analitzar el grau de connexió dels contenidors de requisits, pot ajudar al product manager a detectar quines parts del nostre sistema son més o menys crítiques a canvis, per tal de centrar els esforços del grup de desenvolupament en refactoritzar-les.

En aquest projecte es presenten alternatives de visualització de l'estructura de requisits software per tal de ser conscients de l'estructura interna del programa que estem desenvolupant i descobrir les relacions que hi ha entre els distints elements que el conformen. Aquestes representacions ens permeten expressar les relacions jeràrquiques

naturals de la composició del software, les relacions d'interdependència entre requisits mitjançant l'ús d'entitats de domini i analitzar la importància dels diversos contenidors de requisits en termes de proves d'acceptació i proves de sistema.

Aquesta tesi final de màster 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. Aquest projecte es pot definir com la segona fase d'un treball d'investigació i implementació que ha durat dos anys.

En la primera fase del projecte l'objectiu 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. En acabar aquest treball i amb el fruit de l'experiència diària es varen detectar certes carències respecte a l'explotació de la informació aconseguida. Es per això que es va definir una segona fase on s'intentaren trobar alteratives de visualització que aporten informació rellevant i que ens ajuden a gestionar d'una manera més òptima i informada els requisits. 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 de l'estat actual de la ferramenta de visualització de la gestió de requisits i anàlisi de les seues limitacions o carències.
- Definició i implementació de vistes d'anàlisi de dades sobre l'estructura de requisits.
- Investigar l'estat de l'art sobre les alternatives de visualització de l'estructura de requisits i explorar la visualització en terme de grafs.
- Desenvolupament d'un mòdul de visualització basat en grafs.
- Avaluació del mòdul amb dades reals extretes de la nostra ferramenta de suport al desenvolupament software TUNE-UP.

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 els elements que actuen com a infraestructura o base teòrica per al nostre projecte: la metodologia i ferramenta TUNE-UP, els processos de desenvolupament Test-Driven Requirements Engineering (TDRE) i una visió teòrica de l'anàlisi d'impacte de requisits.

El capítol 3 es centra en la situació inicial del projecte. Ens mostra la implementació i integració dels conceptes teòrics del capítol anterior, aprofundeix en les problemàtiques o les limitacions que hem anat trobant sobre aquest model i defineix un seguit de característiques que es voldrien aconseguir.

El capítol 4 fa un repàs general a les tècniques de representació i visualització de requisits en les ferramentes actuals, i s'analitzen les noves idees i línies d'investigació sobre visualització que podem utilitzar en el nostre projecte.

El capítol 5 divideix els problemes analitzats en diferents vistes o representacions on cada una es centra en un punt de vista diferent i mostra la implementació de cada un d'ells.

En el capítol 6 ens centrem en els diagrames de visualització basats en treemaps i sobretot en grafs, els seus avantatges i inconvenients, i analitzem les diferents alternatives existents a nivell d'aplicacions o de components.

El capítol 7 especifica el treball fet en l'anàlisi del component de visualització basat en grafs, mostrant les diferents parts que hi actuen i la variabilitat que ens aporta.

El capítol 8 defineix la creació d'un mòdul basat en el component yFiles.NET, on es mostraran casos d'estudi reals aplicats a la nova visualització.

El capítol 9 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.

Capítol 2. **TUNE-UP**

El treball realitzat en aquesta Tesis de Màster s'engloba dins del context de la metodologia i ferramenta Tune-up de desenvolupament software. Aquest projecte esta basat en la idea de definir una metodologia de treball, intentant aportar les millors característiques de l'àmbit de desenvolupament tradicional, i les idees més interessants de les metodologies àgils. Dins d'aquest context i centrant-nos en la gestió de requisits, es defineixen les Proves d'Acceptació (PAs) com a l'element central de la definició dels requisits, organitzats jeràrquicament en contenidors (nodes) i relacionades entre elles mitjançant relacions de traçabilitat entre termes o entitats de domini especificats en aquestes PAs.

Per tal de tindre una idea més elaborada sobre la metodologia Tune-up i els mòduls més relacionats amb aquest treball, s'inclouen a continuació unes seccions extretes i traduïdes de la Tesis de Màster de Maria Company Bria i del meu propi Projecte Final de Carrera, que defineixen la primera fase d'aquest projecte d'investigació de dos anys i que ens aporten una visió més profunda dels diversos elements que actuen com a base o infraestructura per als elements de la tesis actual.

2.1. Metodologia i ferramenta

(Capítol extret i traduït de la tesis de Maria Company [1])

TUNE-UP és una metodologia desenvolupada a partir del treball dia a dia en una PIME de desenvolupament de software amb vocació de millora contínua del procés. TUNE-UP s'ha definit davall el marc de diversos projectes universitat-empresa. TUNE-UP ha aconseguit la seua maduresa amb més de sis anys d'aplicació i refinament aplicant-se en projectes industrials, en l'àmbit acadèmic i també com a objecte de publicacions d'investigació.

TUNE-UP s'inspira en l'essència de **PSP** (Personal Software Process) [8], on es destaca que la base de l'èxit radica en una disciplina de treball i productivitat individual centrada en la gestió dels compromisos. Pel que fa a la gestió del temps i de priorització del treball personal TUNE-UP està alineada amb els principis de **GTD** (Getting Things Done)[9] i **The Seven Habits of Highly Effective People**[10]. Per a l'organització i fàcil accés al treball d'un agent TUNE-UP aporta una innovadora variació de **sistema Kanban**, el qual s'integra en un simple però potent mecanisme de workflows flexibles, que permet orquestrar de forma automàtica gran part de la col·laboració necessària entre activitats.

TUNE-UP ajuda en cada moment del projecte a respondre a la pregunta: *aconseguiré acomplir amb els terminis d'entrega de les meues tasques? o serem capaços de complir amb els terminis d'entrega al client?* Estes simples preguntes inquieten qualsevol gestor o participant d'un projecte. No comptar amb una resposta encertada, i sobretot oportuna, comporta en la majoria dels casos greus complicacions en el projecte.

Per què una altra metodologia i una altra ferramenta si ja existeixen alternatives disponibles?

Després de treballar tant amb metodologies tradicionals **(RUP i Mètrica)** com àgils **(XP i SCRUM)**, i utilitzat moltes ferramentes de suport per a diferents activitats en el procés de desenvolupament (ferramentes per a modelatge o per a proves, IDEs per a programació, etc.), la principal lliçó apresa és que els llenguatges de programació i notacions, així com les seues ferramentes de suport, encara que contribueixen a l'èxit d'un projecte no són tan decisius com ho és el procés de desenvolupament en si. No obstant això, sent les metodologies les protagonistes, les propostes actuals continuen llunyanes de la realitat quotidiana d'un equip de desenvolupament. D'una banda, les metodologies tradicionals d'entrada tenen l'impediment de ser massa genèriques i àmplies, la qual cosa requereix un gran esforç de configuració i implantació en l'equip, el qual pot ser inviable d'abordar. D'altra banda, les metodologies àgils parteixen d'una visió massa simplificada del que és un procés de desenvolupament i particularment del manteniment. Un producte en producció requerirà manteniment. Amb una metodologia àgil el procés de generació inicial d'un producte pot resultar accelerat però plantejant-se una perspectiva seriosa de manteniment no constitueix un enfocament adequat, ja que cada vegada es va fent més imprescindible portar un registre i control dels canvis i del comportament implementat en el producte. Esta informació en les metodologies àgils no és gestionada (ni tan sols emmagatzemada) després de la implementació. Normalment les metodologies tradicionals proposen una planificació també tradicional, és a dir, basada en les tècniques genèriques utilitzades en tot tipus de projectes. Aquestes tècniques són durament criticades i queden en evidència les seues carències per a realitzar un seguiment del projecte quan el model de procés és iteratiu i incremental. *Qui ha provat amb èxit, en este context, gestionar adequadament un projecte software usant un diagrama Gantt?* Les metodologies àgils han sabut abordar este aspecte i un dels seus punts més destacables és precisament la planificació, utilitzant tècniques com ara els diagrames **Burn-down**.

Un projecte de desenvolupament o manteniment de software té per objectiu l'aconseguir una entrega reeixida del producte, és a dir, una versió operacional que puga ser explotada i rendibilitzada pel client. Per a aconseguir este objectiu TUNE-UP inclou els elements següents:

- **Model de procés iteratiu i incremental** per al desenvolupament i manteniment del software. El treball es divideix en unitats de treball (WUs) que són assignades a versions. Les versions són freqüents i de curta duració, entre 3 i 6 setmanes depenent del producte.

- **Procés de desenvolupament dirigit per les proves** (Test-Driven). La definició d'una WU és bàsicament l'especificació de les proves d'acceptació acordades amb el client. A partir d'ací, tot el procés gira al seu entorn, s'estima l'esforç d'implementar el comportament associat i d'aplicar les proves, es dissenyen les proves i implementa el dit comportament, i finalment, s'apliquen les proves sobre el producte per a garantir la seua correcta implementació.

- **Workflows flexibles per a la coordinació del treball associat a cada unitat de treball**. Els productes, segons les seues característiques, tenen associats un conjunt de workflows que són utilitzats per a realitzar WUs. Cada WU segueix el flux d'activitats del workflow. Sota certes condicions es permet saltar cap avant o cap arrere en el workflow, així com canvis d'agents assignats i inclús canvi de workflow. Per exemple, les típiques situacions de retreball en el desenvolupament de software ocasionades per la detecció de defectes poden originar salts arrere no explícits en el workflow. Altres facilitats quant a flexibilitat, és permetre treballar en paral·lel o inclús afegir activitats no previstes en la definició del workflow. Esta flexibilitat alhora evita que l'especificació del workflow es complique afegint explícitament totes les situacions possibles. Els workflows en TUNE-UP actuen com guies essencials del procés però evitant les rigideses usuals de la tecnologia i ferramentes específiques en l'àmbit de workflows. En els workflows i per mitjà del seu refinament es va plasmant la millora contínua de procés.

- **Planificació i seguiment continu**. En tot moment ha d'estar actualitzat l'estat de les versions, de les WU, i del treball assignat als agents. El cap del projecte pot actuar oportunament amb aquesta informació, prenent decisions com ara: negociar l'abast de la versió amb el client, aconseguir més recursos, redistribuir càrrega de

treball entre agents, canviar els terminis de la versió, moure WUs entre versions, etc.

- **Control de temps**. Els agents registren el temps que dediquen a la realització de les activitats, el qual es compara amb els temps estimats en cada una d'elles, i detecten oportunament desviacions significatives. Açò permet als agents gestionar més efectivament el seu temps, millorar les seues estimacions i oferir al cap del projecte informació actualitzada de l'estat de la versió.

TUNE-UP és una metodologia que incorpora aspectes de metodologies àgils i de metodologies tradicionals. Les dos primeres característiques (procés iteratiu i incremental, i procés centrat en les proves d'acceptació) classifiquen a TUNE-UP com a metodologia àgil, no obstant això, les altres característiques estan més pròximes del que seria una metodologia tradicional.

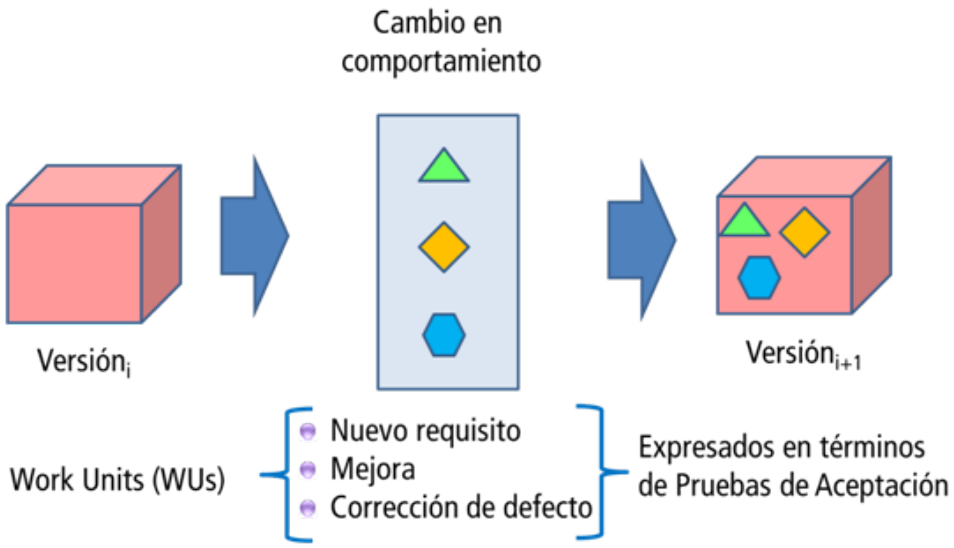


Figura 1. Estratègia Global de TUNE-UP

La Figura 1 il·lustra l'estratègia global de desenvolupament proposada per TUNE-UP. Una nova versió del producte inclourà canvis de comportament respecte de la versió anterior. Aquests canvis de comportament poden ser en general **Nous requisits**, **Millores de requisits** existents o **Correccions de defectes**. En TUNE-UP tots estos tipus de canvis de comportament s'especifiquen de forma homogènia com PAs. Així, d'una banda des del punt d'organització del treball tot tipus de canvi que està en una versió s'aborda de forma integrada quant a planificació i assignació de recursos. D'altra banda, des de la perspectiva de l'especificació del canvi i la seua posterior verificació, en compte de tindre artefactes

diferents (requisits/canvis/correccions i PAs), es té només un artefacte anomenat **Unitat de Treball** (WorkUnit, WU) expressat en termes de PAs.

TUNE-UP està dirigit per les PAs. És un enfocament TDD però en l'àmbit més general de la gestió del projecte i especialment dels requisits del producte. Tal com s'il·lustra en la Figura 2, i a manera d'exemple (perquè els rols podrien variar segons siguen els recursos disponibles i les exigències del projecte), l'*Analista* defineix amb el Client els canvis en el producte en termes de WUs i els seus corresponents PAs. El *Programador* escriurà codi per a satisfer aquestes PAs i finalment el *Tester* aplicarà proves per a assegurar que el comportament del sistema satisfà les PAs.

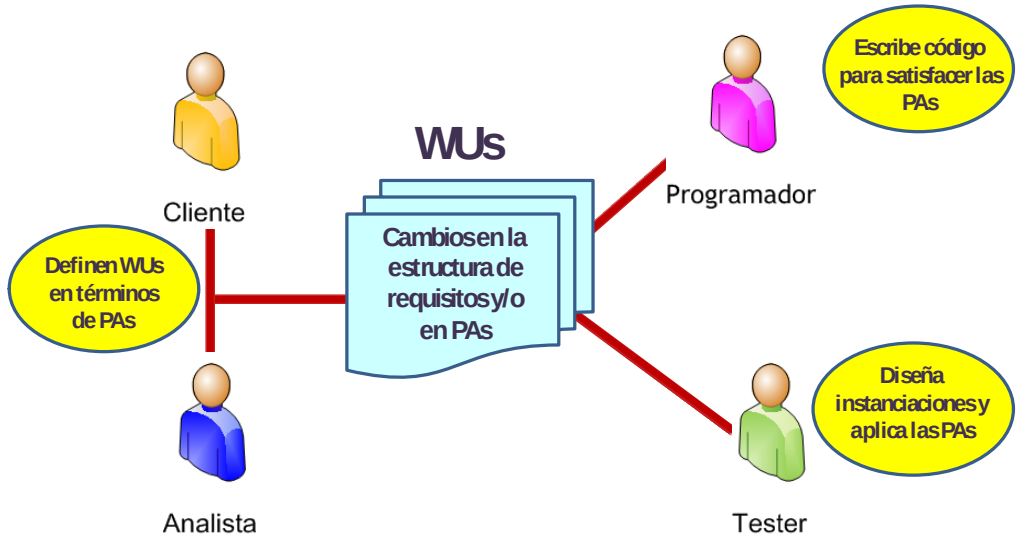


Figura 2. Procés dirigit per les PAs

Depenent del tipus de WU i del projecte en el qual ha de dur-se a terme el canvi de comportament d'un producte, pot ser necessari realitzar diferents activitats i en diferent ordenament temporal. En TUNE-UP quan es crea una WU se li assigna el workflow més apropiat. La Figura 3 mostra un **workflow** bàsic per al desenvolupament d'una WU, en el qual s'han definit quatre rols i deu activitats. TUNE-UP permet crear i modificar workflows segons es requerisca. En TUNE-UP es poden gestionar diferents productes amb diversos equips de desenvolupadors. Cada producte té associat certs workflows els quals poden compartir-se entre diferents productes, així com les activitats poden ser utilitzades en diferents workflows. Les activitats de cada workflow poden variar significativament depenent de factors com ara: quantitat i especialització d'agents participants, validacions o negociacions predeterminades amb el client, característiques del producte (necessitat de migració, traducció, etc.), nivells i activitats de proves (unitàries, d'integració,

d'acceptació, proves de regressió, automatització de proves), etc. El procés de millora contínua es plasma en el refinament dels workflows.

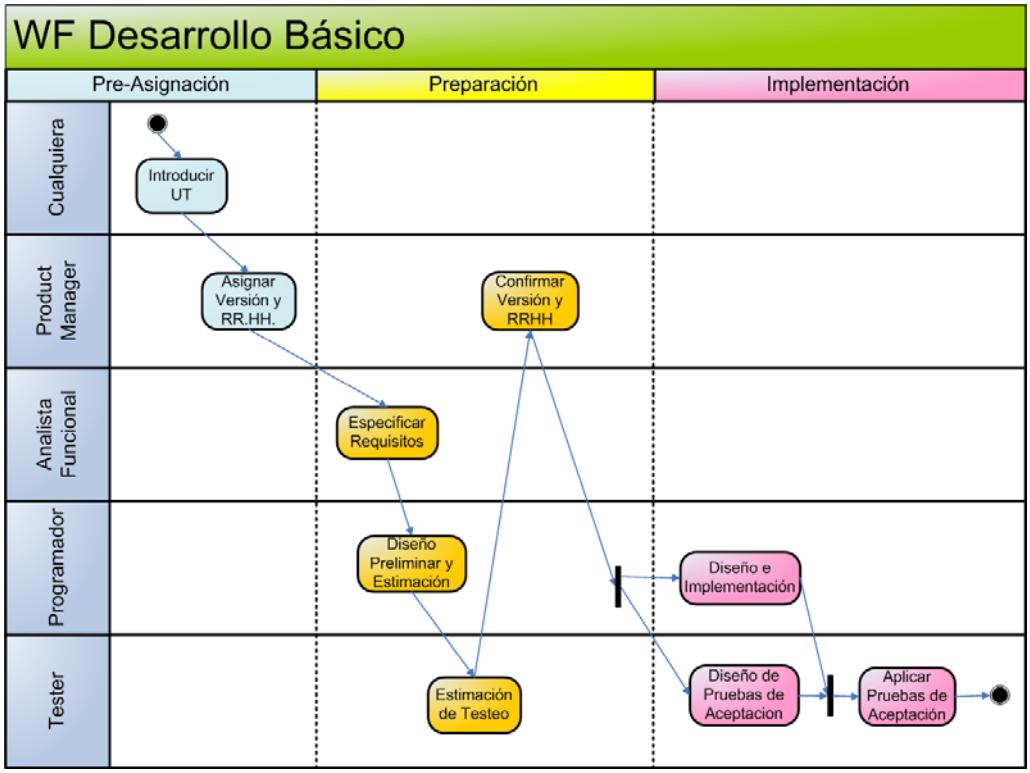


Figura 3. Un exemple de Workflow simple

En TUNE-UP s'utilitza la següent terminologia per a referir-se a comportament no desitjat del producte: Fallada, Defecte i Error. La **fallada** és el símptoma de comportament no desitjat que manifesta el producte en la seua utilització. El **defecte** és el que produeix la fallada. Els defectes poden estar en el codi del producte o en qualsevol de les seues especificacions, per exemple potser el codi estiga correcte respecte de la prova, però la prova pot no ser correcta. Finalment, **l'error** és l'acció humana (realitzada o no realitzada pels desenvolupadors), és el que va originar el defecte. En la Figura 4 s'il·lustra com en TUNE-UP es du a terme un tractament integral de les fallades. Tota fallada s'especifica com una PA amb un KO continguda en una WU. En cas que la fallada es resolga en una mateixa WU (siga esta de qualsevol tipus), es durà a terme el corresponent retreball (tornant a realitzar l'anàlisi o programació requerida). Si la fallada no és abordada en la WU on es detecta o simplement si la detecció de la fallada es fa fora del context d'una WU, llavors es crearà una nova WU de tipus fallada que es durà a terme en la versió actual o en alguna versió futura.

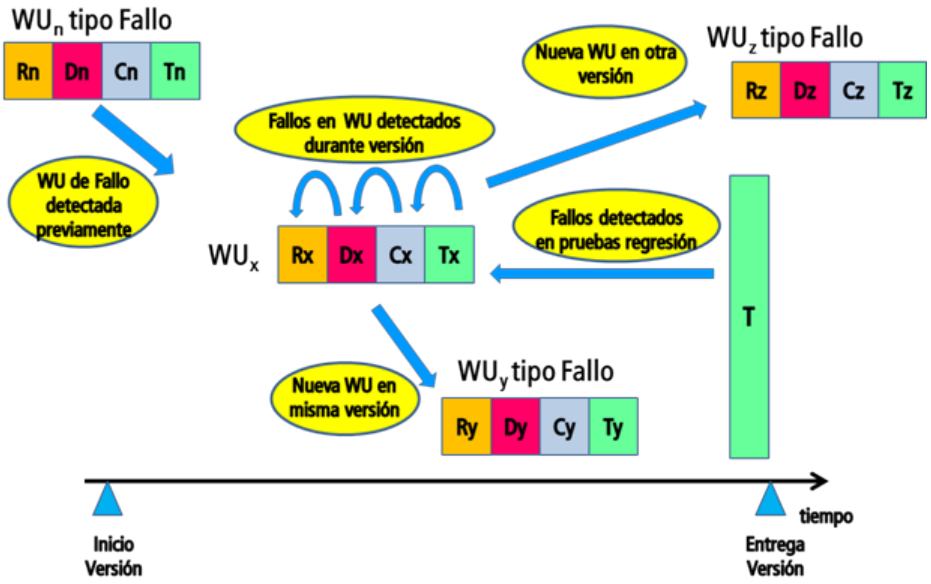


Figura 4. Gestió de Fallades en TUNE-UP

TUNE-UP software Process és el nom de la ferramenta de suport per a l'aplicació efectiva de la metodologia TUNE-UP. La ferramenta està formada per quatre mòduls principals: **Personal Planner (PEP)**, **Work Unit Mànager (WUM)**, Versió **Contents & Tracking (VCT)** i **Requirements Mànager (REM)**. A continuació es descriu breument cada un d'estos mòduls..

Personal Planner (PEP)

És el punt d'entrada a l'espai de treball de l'agent. Quan un agent inicia la seua jornada laboral, accedeix al PEP (Figura 5) per a veure el treball en el qual participa. Este treball correspon a les activitats que tinga assignades en les WUs no acabades. En el **Kanban** (grid de l'extrem superior esquerre de la Figura 5) l'agent pot determinar fàcilment en el PEP quines activitats té pendents o en progrés (actives o pausades). A més pot distingir aquelles activitats que tenen pendents o en procés altres membres de l'equip i que, o bé estarien per arribar-li, o ja ha finalitzat, i han continuat en activitats posteriors en el workflow. L'agent ha de seleccionar la WU en la què treballarà. Un aspecte clau per al projecte és que els agents es dediquen a les activitats encertades d'acord amb les seues prioritats. El PEP ofereix una varietat de facilitats de filtrat i ordenament d'informació perquè l'agent puga determinar les prioritats del seu treball i realitzar una correcta elecció. El Kanban resumeix les comptabilitzacions de les unitats de treball segons l'activitat i estat

en què es troben, i en el grid de la dreta de la Figura 5, mostra informació de les activitats de dites WUs i inclou: producte, versió, descripció de la unitat de treball, estat de l'activitat actual dins del workflow, etc.

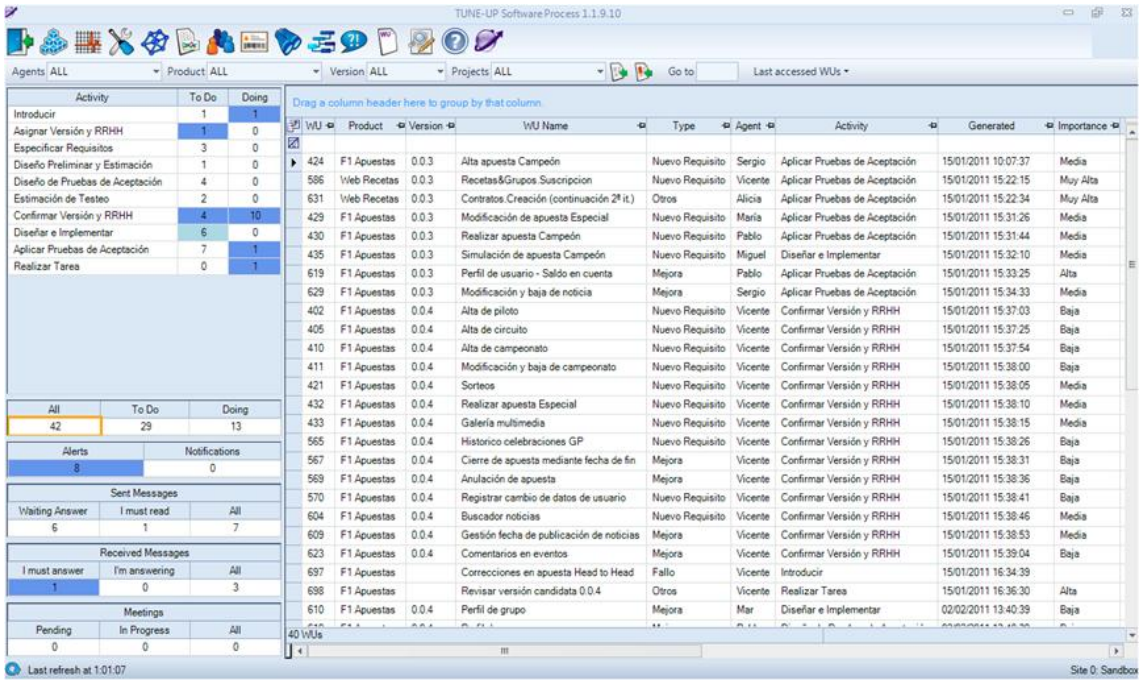


Figura 5. Personal Planner (PEP)

Work Unit Mànager (WUM)

Quan l'agent decideix la WU i l'activitat en què desitja treballar, accedeix amb ella al **WUM** (Figura 6). En la fitxa de la part superior del WUM es mostren les dades generals de la WU, i en la part inferior, un conjunt de pestanyes amb informació més específica. En la pestanya **Tracking** es mostra la llista d'activitats del workflow per les que ha passat la unitat de treball. Cada activitat està associada a un registre de seguiment que inclou: l'activitat, l'agent que la porta a terme, l'estat en què es troba, etc. A més, en esta pestanya, amb els botons **Record, Pause i Finish**, l'agent pot controlar el registre de temps, i activar, interrompre o finalitzar l'activitat. En acabar una activitat, la WU continua la seua workflow o pot a petició de l'usuari fer un salt a una altra activitat del workflow. Altres facilitats per a gestionar una WU i que es troben en altres pestanyes del WUM són: **documentació** (per a la WU, per a apartats específics del producte, per a missatges com a adjunts i per a execucions de PAs per exemple per a indicar passos de reproducció d'una fallada), **missatges** per a comunicar-se amb altres agents en el context de la WU, **reunions**

per a registrar temps i acords aconseguits, **relacions** amb altres WUs, detall dels temps **registrats** i **estimats** per a cada activitat, i informació de les parts **del producte que són afectades per la WU**.

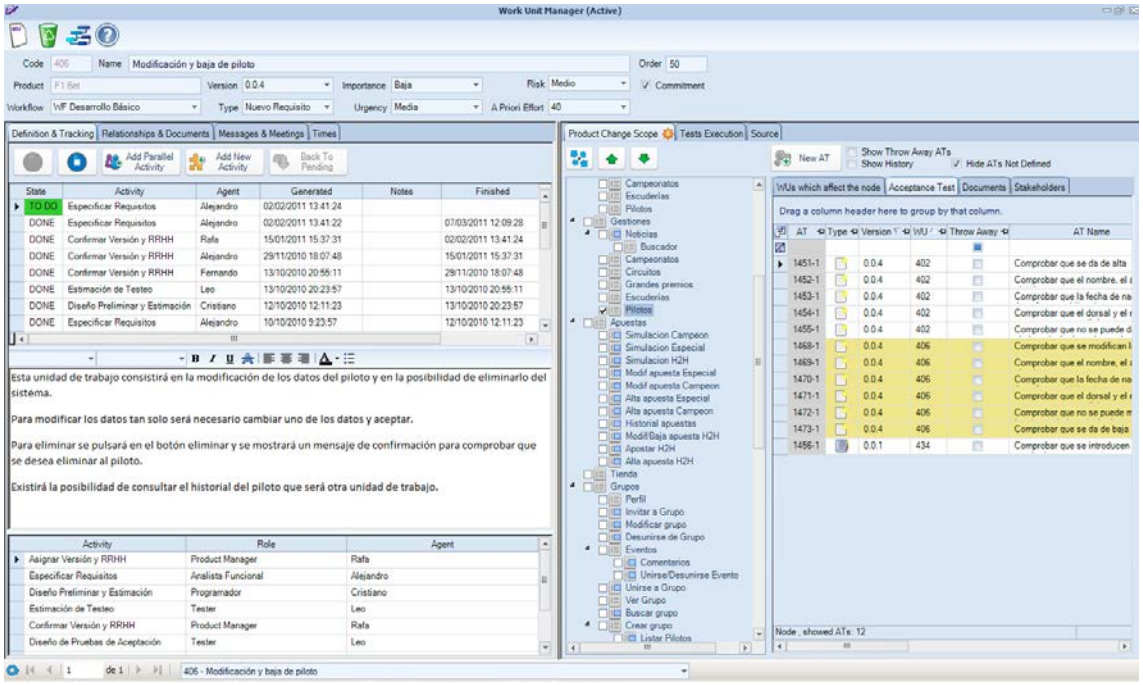


Figura 6. Work Unit Mànager (WUM)

Requirements Mànager (REM)

El **REM** és una de les innovacions que ofereix TUNE-UP pel que fa a la gestió del producte i els seus requisits. En TUNE-UP, el desenvolupament i manteniment del producte està dirigit per les PAs que constitueixen l'especificació del comportament del producte. L'estructura del producte és un **graf acíclic dirigit** mostrat com un treeview, com es mostra en la part esquerra de la Figura 7. Cada node d'esta estructura és un contenidor de comportament expressat com PAs. Així, al seleccionar un node, en la pestanya **'WUs which affect the node'**, podem conèixer tota la seua història en termes de WUs que l'han afectat, l'estan afectant o l'afectaran en futures versions. De manera semblant, en la pestanya **'Acceptance Test'** es pot consultar les PAs que han sigut definides en aquest node i com han canviat. A més, cada node en la pestanya **'Documentació'** pot tindre associada documentació addicional en termes de fitxers de qualsevol tipus. Finalment cada node en la pestanya **'Stakeholders'** pot tindre especificats els stakeholders per als quals el node és rellevant.

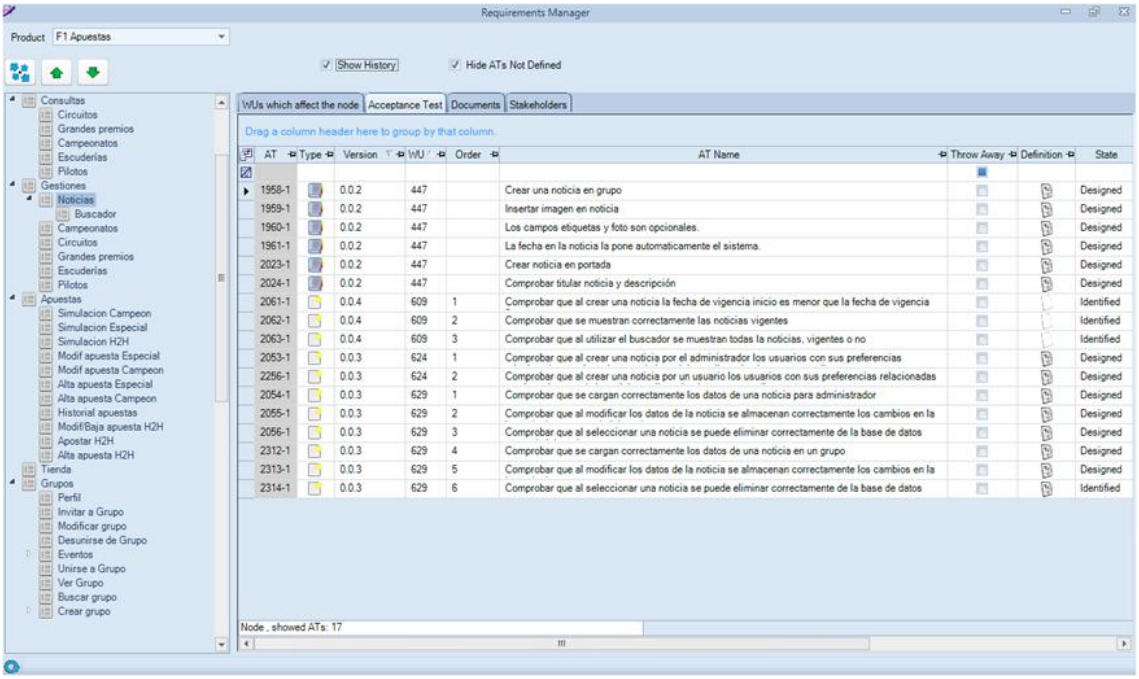


Figura 7. Requirements Mànager (REM)

Versió Contents & Tracking (VCT)

En este mòdul trobem facilitats per a la planificació i seguiment de les versions. En la Figura 8 es mostra el contingut de **la pestanya 'Agent Workload'**, en ella es pot conèixer en qualsevol moment la folgança simple dels agents respecte de les seues activitats en una versió del producte. En esta interfície s'ofereixen potents mecanismes de filtres i agrupacions per columnes. Quan una versió té problemes de folgança, en esta mateixa interfície el **Product Mànager** pot canviar l'agent assignat a l'activitat (per a balancejar la càrrega d'un agent) o canviar de versió alguna unitat de treball. Altres alternatives són modificar la data de terme de la versió, assignar més recursos humans a la versió o dividir unitats de treball per a realitzar-les incrementalment en diverses versions. A més disposem d'altres pestanyes que tenen gran utilitat a l'hora de planificar una versió com **'Relationships'**, **'WUs in Versió'** i **'Affected Requirements'**.

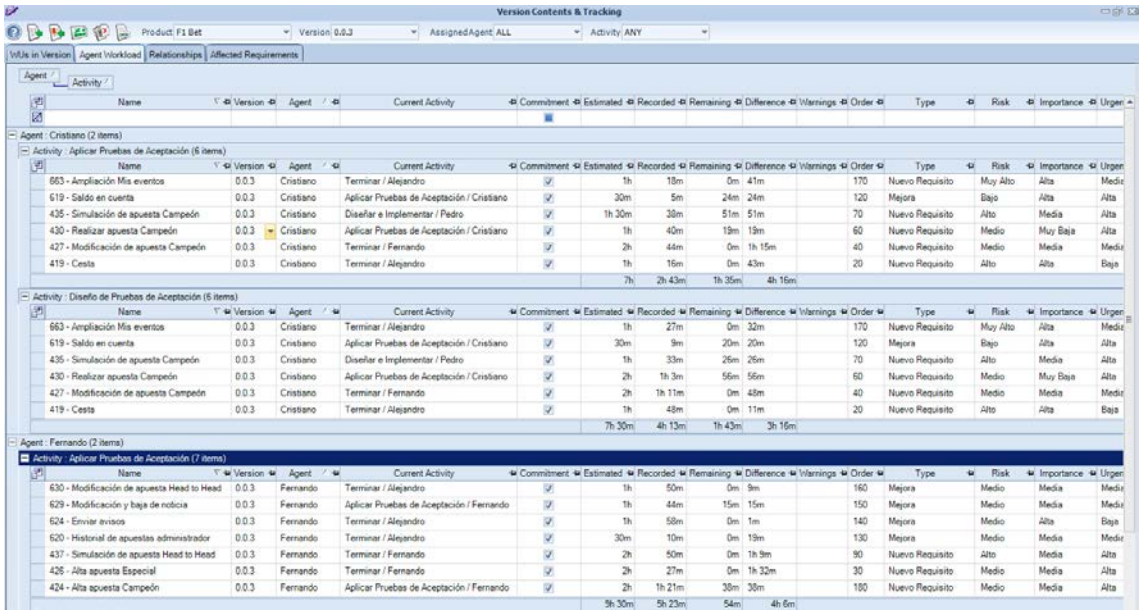


Figura 8. Versió Contents & Tracking (VCT)

2.2. Test-Driven Requirement Engineering (TDRE)

(Capítol extret i traduït de la tesis de Maria Company [1])

L'enfocament **Test-Driven Development (TDD)**[11] es basa en que les proves han de dirigir el desenvolupament del producte software. El TDD s'ha aplicat essencialment en l'àmbit de la implementació, particularment seguint el plantejament "*no escriure codi fins a disposar de les proves que l'han de satisfer*". El TDD ha tingut un fort impuls amb les metodologies àgils, les quals ho incorporen entre les seues pràctiques essencials.

En productes software industrials, quan s'utilitzen pràctiques d'enginyeria de requisits, majoritàriament estan suportades per llenguatge natural, la qual cosa comporta els ja coneguts inconvenients relatius a ambigüitat. No obstant això, la necessitat de validació pot més que els avantatges que pot oferir una especificació més formal i rigorosa dels requisits. El client ha de poder llegir i comprendre els requisits per a així poder donar la seua conformitat. Les tècniques més populars per a especificació de requisits es resumeixen en **Casos d'Ús** (utilitzats en metodologies tradicionals, com RUP [12]) i Històries **d'Usuari** (fitxes de XP [13] o representacions semblants en altres metodologies àgils com Scrum [14]). Si bé els elements Actor (o tipus d'usuari) i Requisits (Cas d'Ús o Història d'Usuari) poden visualitzar-se i manipular-se gràficament o en fitxes, la descripció de cada requisit s'elabora essencialment de forma textual en llenguatge natural.

Per definir els requisits és clau involucrar el client. L'acord/contracte amb el client i la planificació del desenvolupament del producte haurien d'estar basats en els requisits que es pretenen incorporar en el producte. Els requisits són l'objectiu a aconseguir, és a dir, és el que espera el client que el producte software satisfaça.

El **model en V** considera les proves com un procés que corre en paral·lel amb l'anàlisi i el desenvolupament, en compte de constituir una fase aïllada al final del procés. En la representació gràfica clàssica del Model en V (vore Figura 9), les fases de desenvolupament de software apareixen a l'esquerra (des de requisits fins a la implementació d'unitats de codi) i els corresponents nivells de proves a la dreta (des de Proves Unitàries fins Proves d'Acceptació).

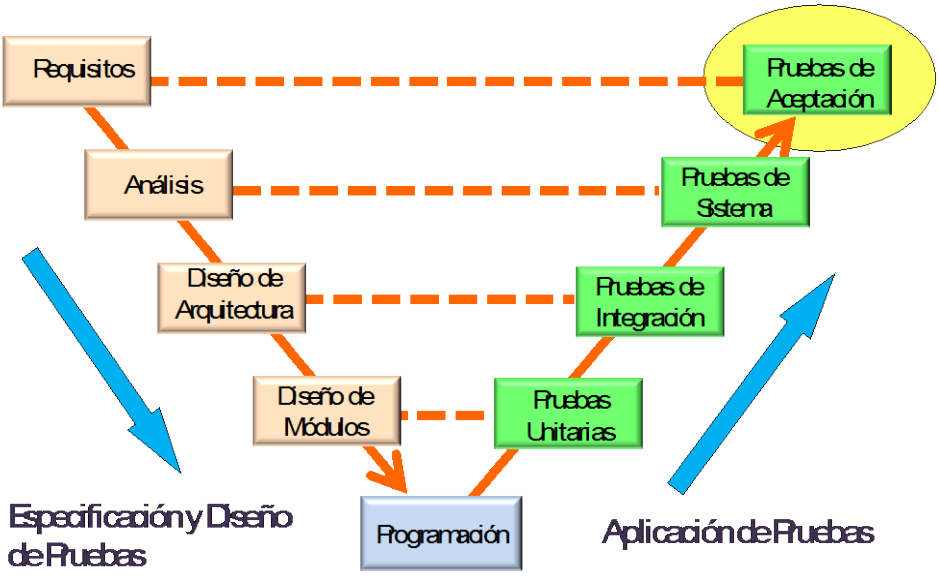


Figura 9. El Model en V

Partint dels requisits o del disseny (a l'esquerra) s'han de planificar i preparar els nivells de proves corresponents (a la dreta). En general, cada activitat de proves a la dreta valida l'activitat enfrontada de l'esquerra. També en general, els nivells superiors de proves en el diagrama estan basats en caixa **negra** (proves basades en les especificacions) i els nivells inferiors estan basats en caixa **blanca** (basades en l'estructura interna dels components del sistema).

La majoria de les metodologies es basen en este model de proves, no obstant això, aquelles que aposten decididament per models de processos iteratius i incrementals (RUP, entre les metodologies tradicionals, i totes les metodologies àgils) utilitzen de forma més explícita les proves **de sistema** i proves **d'acceptació**. L'única diferència entre estos dos tipus de

proves és l'agent que les crea, les d'acceptació són creades pel client i les de sistema són creades pel desenvolupador. Encara que la majoria de metodologies coneixen i utilitzen les Proves **d'Acceptació**, en molt poques les utilitzen com a fil conductor del procés de desenvolupament software. En estes metodologies utilitzen les Proves d'Acceptació com a criteri **d'èxit de cada iteració**. Açò ens portaria a classificar estes metodologies com a Test-Driven, en un context més de planificació i desenvolupament global. No obstant això, en elles no hi ha una analogia entre el Test-Driven de l'àmbit d'implementació i el que seria **Test-Driven en l'àmbit dels requisits i de la planificació**. El plantejament en este últim àmbit seria "*no incorporar un requisit a una iteració (o al pla en general) sense haver-hi abans establert les seues Proves d'Acceptació*".

En les metodologies populars (tradicionals o àgils), els artefactes **Requisit i Prova d'Acceptació** són tractats de forma separada i abordats en diferents activitats i moments del desenvolupament. Una vegada establerts els requisits, es defineixen les Proves d'Acceptació per a ells. A més, usualment estan involucrats dos rols també diferents, Enginyer de Requisits (Analista o Analista Funcional) i Tester (és el tester especialitzat en Proves d'Acceptació, no en Proves Unitàries i d'Integració, les quals normalment són responsabilitat del programador).

La nostra proposta es denomina **Test-Driven Requirements Engineering (TDRE)** per referir-nos a TDD però en l'àmbit dels requisits i la planificació. TDRE integra els artefactes, activitats i rols associats a l'especificació i validació dels Requisits i de les Proves d'Acceptació. El concepte de Requisit es converteix en un contenidor de Proves d'Acceptació, i són estes les que adquireixen protagonisme com a especificació de cada requisit. Una de les huit "bones característiques" que recomana la **IEEE 830** [15] per a una especificació de requisits es refereix que cada requisit ha de ser "**Verificable**". En la nostra proposta els requisits són verificables perquè estan especificats com a Proves d'Acceptació.

Si bé, com a especificació de requisits, les Proves d'Acceptació poden patir els mateixos inconvenients que els requisits tradicionals(en utilitzar el llenguatge natural per a especificar-les), una Prova d'Acceptació per la seua granularitat i verificabilitat és una unitat que determina de forma precisa una part del comportament del sistema. Així, altres característiques recomanades per l'estàndard IEEE 830 (Correcció, No ambigüitat,

Completesa, Consistència, Orde per importància o estabilitat, Modificabilitat i Traçabilitat) també es veuen afavorides.

Encara que els requisits i les activitats associades han anat fent-se espai entre els aspectes essencials d'un projecte de desenvolupament de software, per a les proves no ha passat igual i només equips de desenvolupament amb una certa maduresa les integren com a peça imprescindible en els seus processos de desenvolupament. Hi ha dificultats per a la introducció d'una "cultura", disciplina i pràctiques de proves en un equip de desenvolupament. Entre els obstacles o males estratègies podem destacar: carència d'un procés de desenvolupament que íntegre les activitats de proves amb la resta d'activitats, sobrevaloració de l'automatització de les proves (i particularment les Proves Unitàries) com a objectiu immediat o únic, i el poc èmfasi a "rendibilitzar" l'esforç invertit en proves. L'anterior fa que la iniciativa d'incorporació de proves a més de ralentitzar-se quant a resultats, vaja perdent força en el seu procés d'implantació en l'organització.

En TDRE l'estratègia és implantar una "cultura" de proves basada en el seu aprofitament com a fil conductor del procés de desenvolupament. El nostre enfocament TDRE està incorporat en la metodologia **TUNE-UP** i en la seua ferramenta de suport **TUNE-UP software Process**.

Per a comprendre com els requisits poden ser especificats mitjançant les Proves d'Acceptació començarem amb un exemple. Considerem el requisit "Retirar diners" en el context d'un caixer automàtic. Una típica especificació narrativa podria ser la següent:

"El client ha de poder retirar diners del caixer en quantitats seleccionables. Sempre rep un comprovant, llevat que el caixer es quede sense paper. Quan es tracta d'un client preferencial pot retirar més diners dels que té en el seu compte, però se li ha d'advertir que se li cobraran interessos. El client hauria de poder cancel·lar en qualsevol moment abans de confirmar la retirada de diners. Les quantitats haurien de poder servir-se amb els bitllets que en eixe moment tinga el caixer i no s'haurien d'acceptar altres sumes. Seria convenient avisar quan el caixer estiga realitzant operacions internes mostrant un missatge: Els diners retirats del compte ha de poder comprovar-se en els registres de moviments del compte…".

La Figura 10 il·lustra algunes alternatives d'especificació per a este requisit (incloent l'anterior descripció narrativa com a opció per defecte). Les icones reflecteixen la conveniència de cada alternativa d'especificació.

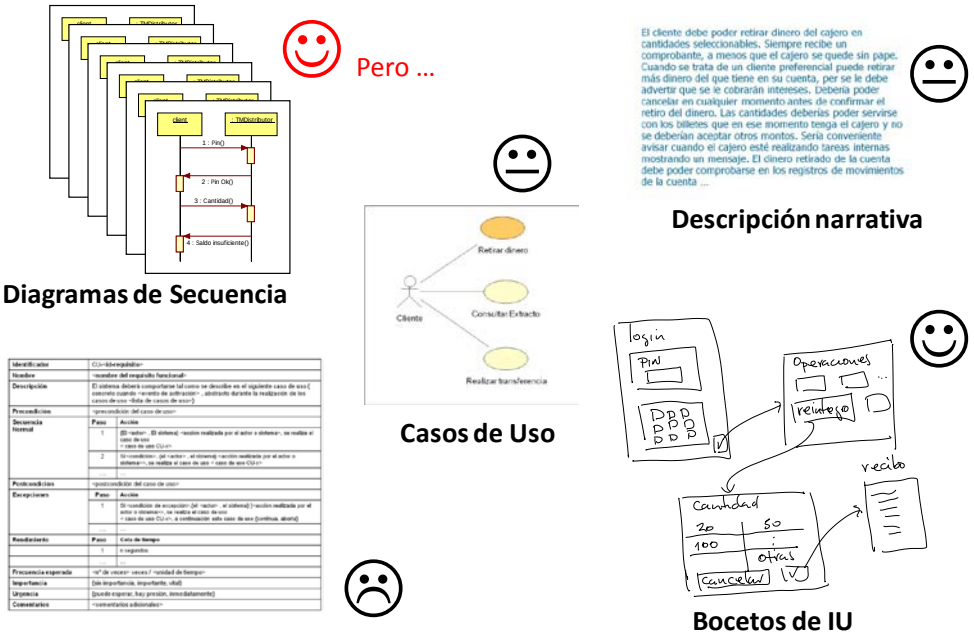


Figura 10. Alternatives populars per a l'especificació de requisits

Elaborar un ***Diagrama de Seqüència*** per a definir cada escenari d'execució del requisit pot paréixer interessant, no obstant això, en general no resulta apropiat per la gran quantitat de diagrames generats. Resulta més interessant la identificació dels escenaris que la il·lustració de cada un d'ells en un diagrama.

La descripció ***narrativa*** no és descartable, almenys per a donar una breu definició del requisit centrant-se a definir els conceptes involucrats (amb la idea d'un glossari o d'un senzill Model de Domini).

Un Model ***de Casos d'Ús*** no és mala idea, especialment per a organitzar i visualitzar els requisits d'un sistema nou. No obstant això, un Model de Casos d'Ús no és apropiat per a il·lustrar l'estructura de requisits detallada d'un producte software en situacions de manteniment a més llarg termini, ja que un producte software de grandària mitjana pot tindre milers de requisits. La visualització i gestió de gran quantitat de requisits necessita de mecanismes més apropiats.

Els ***esbossos*** (visualitzacions molt preliminars) de la Interfície d'Usuari (IU) són sempre benvinguts perquè són una ferramenta efectiva de comunicació i validació amb el client, el

qual pot fer-se una idea del producte. En este context de requisits no ha de pretendre's realitzar el disseny final de les IUs sinó més aïna panells amb un cert àmbit de dades, sense aprofundir en tipus de controls d'interfície o qüestions d'estètica de formularis/pàgines.

Les ***plantilles*** són una de les alternatives d'especificació més usades per a Casos d'Ús. Les plantilles són elegants i proporcionen una sensació d'orde en l'especificació. No obstant això, en general resulten contraproduents ja que tendeixen a donar un tractament uniforme quant al nivell de detall per a tots els requisits. En aquells molt simples es tendeix a incloure coses òbvies o irrellevants només per a poder cobrir tots els apartats de la plantilla. Quan un requisit inclou diversos (o molts) escenaris, l'intent per sintetitzar tots els escenaris en una plantilla (que només ofereix passos i excepcions) porta normalment a especificacions enrevessades.

El nostre enfocament TDRE aposta per especificar els requisits usant els elements que es mostren en la Figura 11: una breu **descripció narrativa** que estableix els conceptes i motivació del requisit**, esbossos de la IU** (si és procedent) i una llista **de Proves d'Acceptació** (PAs). Estes PAs es refinaran des de simples frases que donen nom als escenaris, fins proves dissenyades, llistes per a ser aplicades o per a automatitzar-se i aplicar-se en regressió. Així, els requisits actuen com a contenidors per a les PAs.

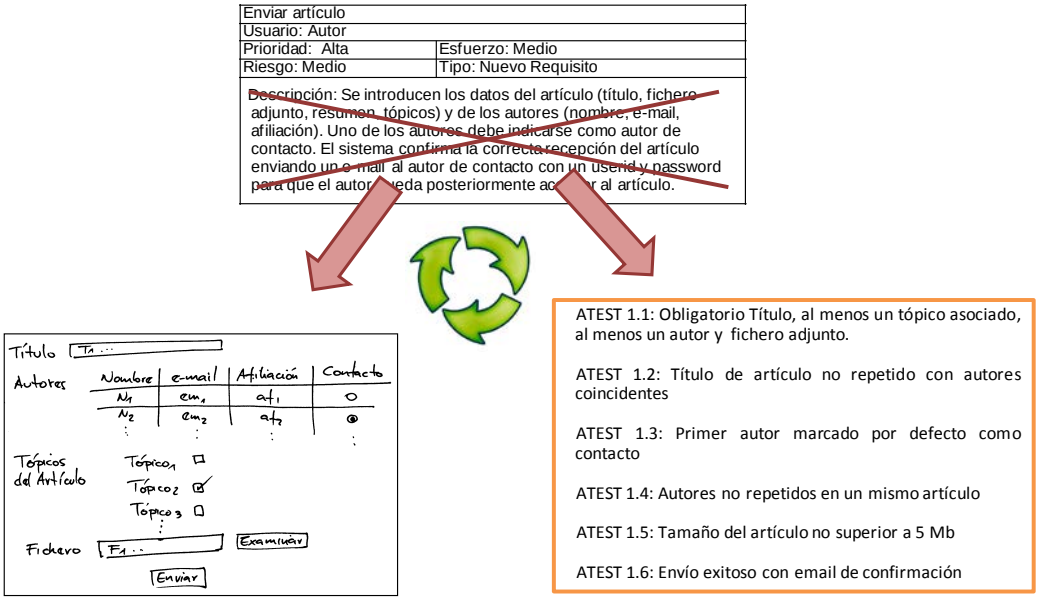


Figura 11. Especificació de requisits en TDRE

Depenent del requisit, podria ser útil utilitzar altres formes d'especificació amb caràcter complementari, per exemple un **Diagrama d'Activitat** si el comportament associat al

requisit és de caràcter algorítmic o un **Diagrama d'Estats** si el comportament inclou habilitació o deshabilitació d'accions d'acord amb l'estat del sistema. La premissa essencial és pragmatisme respecte de l'especificació, amb la qual cosa no es descarta l'ús combinat d'alternatives d'especificació, però el criteri primordial ha de ser el rendibilitzar l'esforç en especificació i facilitar-ne el manteniment. D'altra banda, des del punt de vista d'esforç de manteniment, especialment quant a consistència, és important no abusar de duplicacions d'especificacions en diferents mitjans de representació.

Les milers de PAs que fàcilment pot arribar a tindre un producte software han d'organitzar-se adequadament per a poder ser gestionades de forma eficient. Un **graf dirigit** (Figura 12) és una representació adequada per a realitzar un refinament per nivells. Aquest graf permet tant la visualització de relacions de descomposició com de dependència entre requisits. Així, cada node és un requisit funcional o no funcional. Els arcs entre nodes estableixen relacions pares-fills (mitjançant els quals es fa una descomposició dels requisits) o relacions de dependència del tipus "nodes que afecten nodes" (estes relacions són clau per a realitzar anàlisi d'impacte). Els nodes de la part més alta podrien, per exemple, seguir la classificació de característiques definides per la **ISO/IEC 9126** com a punt de partida per a la definició dels requisits del sistema.

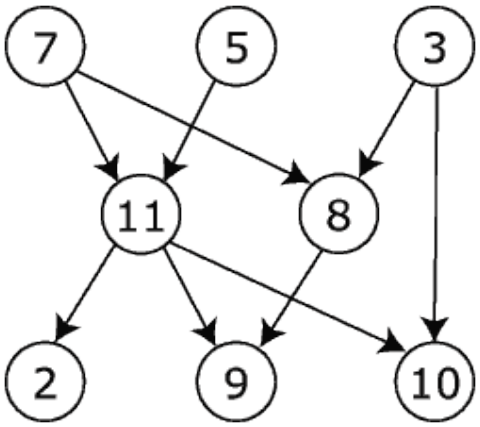
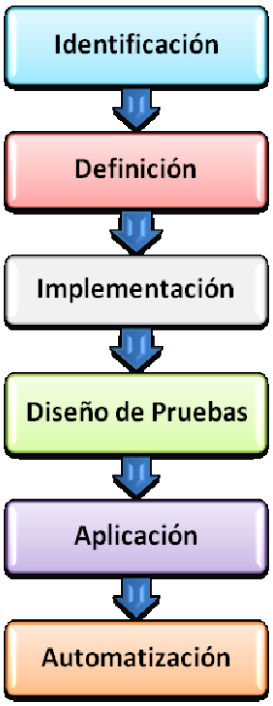


Figura 12. Representació d'un graf acíclic dirigit

La PA quan va passant pel procés de desenvolupament software va canviant d'estats, depenent dels recursos que dispose l'empresa tindrem més o menys agents interactuant amb les PAs al llarg del procés. La situació ideal seria tindre almenys tres rols diferents: **l'Enginyer de Requisits** (a nivell d'empresa conegut com a Analista funcional o Analista de Negoci), el **Programador** i el **Tester**. Si per falta de recursos no es tingueren els tres

rols, el mateix agent que analitza podria programar o el mateix que analitza podria testejar, el convenient és que no prove el mateix agent que ha implementat.

Els estats pels quals passa una PA s'expliquen a continuació:



- **Identificació:** Es capturen els requisits negociant amb el client i s'identifica el nom de les PAs. Quan es tenen les PAs identificades ja es poden prendre decisions de planificació.

- **Definició:** S'especifica el contingut de la PA i es fa l'anàlisi d'impacte que altres parts del producte podria afectar el canvi)Una. vegada avaluem l'anàlisi d'impacte i es valida amb el client podem decidir si el canvi que es defineix en la prova és acceptat o no.

- **Implementació:** Durant la implementació el programador gràcies a les PAs (es podrien considerar com un contracte Analista-Programador) té un criteri d'èxit mesurable ja que escriu codi amb la idea de satisfer la PA. Una vegada ja ha passat totes les PAs ha acabat d'implementar.

- **Disseny de Proves:** El tester és l'encarregat de buscar les diferents combinacions de dades adequades per a les diferents instanciacions de la PA.

- **Aplicació:** S'aplica la PA i es valida la implementació.

- **Automatització:** Fins a este nivell tot és manual, a partir d'ací si l'empresa té suficients recursos i es va a incloure l'automatització de la prova es podrien aplicar proves de regressió **automatitzades**

Al llarg d'este treball ens centrarem més detalladament en la identificació i definició de les PAs, la qual cosa està més associat al marc de treball de l'analista. El disseny i execució de les PAs és responsabilitat del tester i no ho veurem en detall.

Seguint amb l'exemple anterior, el requisit "**Retirar diners**" podria ser un node de l'estructura de requisits. Els noms que identifiquen els seus PAs podrien ser:

1. *Reintegrament usant quantitats predefinides habilitades*

2. *Reintegrament amb quantitat introduïda per client*
3. *Reintegrament saldo < quantitat*
4. *Cancel·lació d'operació*
5. *No disponibilitat de bitllets*
6. *No disponibilitat de paper per a rebut*
7. *Reintegrament saldo < quantitat amb client preferencial*
8. *Excedit temps de comunicació amb sistema central*
9. *Avís d'operacions internes del caixer*
10. *Excedit temps d'espera per a introducció d'acció*

Una PA té com a propòsit demostrar al client el compliment parcial o total d'un requisit del software. Les característiques d'una PA són:

- Una PA descriu un escenari (seqüència de passos) d'execució o un ús del sistema des de la perspectiva del client. Les PAs cobreixen des d'escenaris típics/freqüents fins als més excepcionals
- Pot estar associada a un requisit funcional o requisit no funcional. Un requisit té una o més PAs associades
- Una PA pot tindre infinites instanciacions (execucions amb valors concrets)

La definició d'una PA se separa en quatre apartats: Condició, Passos, Resultat Esperat i Observacions.

- **Condició.** És opcional, i s'utilitza per a establir condicions prèvies abans d'aplicar els passos de la prova. Normalment es refereixen a l'estat de la BD abans d'executar la prova i/o la navegació necessària en la IU per a localitzar-se en el punt adequat per a realitzar la prova (quan no siga obvi)
- **Passos.** Són les accions d'interacció de l'actor amb el sistema. Quan són diverses accions, estes poden posar-se en una llista numerada. Han de ser simples, de l'estil "seleccionar…", "introduir...", evitant fer referència explícita a controls d'interfície

o qualsevol tecnicisme que dificulte la seua validació amb l'usuari

- **Resultat esperat**. És l'efecte de les accions d'interacció de l'actor. Cada acció pot provocar un o més resultats. És important que quan es tracte de missatges a l'usuari s'incloga el text com a part del resultat esperat, així el programador disposa de la aquesta informació ja validada amb el client

- **Observacions**. És un apartat opcional, són recomanacions que l'analista creu convenient fer-li al tester i/o programador

Les condicions i/o resultats esperats poden establir-se en l'àmbit del requisit contenidor de la PA o d'altres requisits. Açò, com indicarem més avant, permetrà establir relacions de dependència entre requisits. A continuació es presenta com a exemple la definició de la PA "Reintegrament *saldo* < *quantitat*".

És important que l'analista establisca un ordre en les proves, des de les corresponents a escenaris més típics/freqüents fins aquells més excepcionals. Este orde servirà de guia d'implementació als programadors i al treball del tester. El programador hauria d'escriure el codi amb la idea de satisfer les PAs de forma incremental, fins a implementar tota la funcionalitat i superar totes les PAs. Per la seua banda, i addicionalment, el programador podria realitzar testeig unitari i d'integració per a les peces de codi que escriga. Aquestes Proves Unitàries i d'Integració, no sempre estan relacionades directament amb els requisits ja que el seu propòsit s'orienta més a verificació del software que a la seua validació (les PAs validen la implementació respecte dels requisits del client). No obstant això, en casos en els quals les Proves Unitàries i d'Integració es deriven o tenen una relació més directa amb els requisits, es veuen afavorides perquè en TDRE els requisits estan especificats com a proves. D'altra banda, el tester dissenyarà, aplicarà, i documentarà una o més execucions instanciades (amb valors concrets) per a cada PA.

Les PAs es rendibilitzen durant el procés de desenvolupament de software perquè permeten:

- Especificar i validar els requisits.

- Valorar adequadament l'esforç associat a la incorporació d'un requisit. És més senzill valorar l'esforç que requereix la satisfacció de cada PA.

- Negociar amb el client l'abast del sistema en cada iteració de desenvolupament. Les PAs introdueixen un nivell de granularitat útil per a negociar l'abast amb el client. Un requisit no necessita implementar-se "tot o res", pot fer-se de forma incremental postergant la satisfacció de certes PAs.

- Guiar als desenvolupadors en la implementació ordenada del comportament associat a un requisit. Si bé les PAs no són tan exhaustives com les Proves Unitàries generen gran part de la cobertura de codi. Açò no significa necessàriament prescindir de la resta de nivells de proves del Model V, sinó que en un context de recursos limitats, les PAs haurien de ser les proves essencials o mínimes que haurien de definir-se i aplicar-se al sistema.

Identificar oportunitats de reutilització. El detall i precisió proporcionat per les PAs permet identificar en l'especificació de requisits possibles duplicacions de comportament que donen origen a oportunitats per a la seua reutilització (lògica de l'aplicació).

2.3. Anàlisi d'impacte de requisits

(Capítol extret del PFC de Guillem Medina Martínez [2])

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 i 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 13, les tècniques d'anàlisi d'impacte es poden classificar en tres tipus: *Traçabilitat, Dependència i Experimental.* [16]

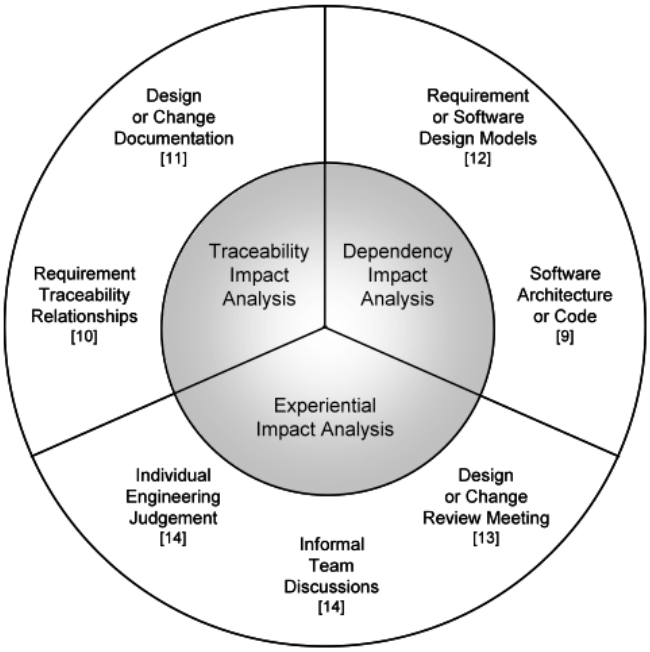


Figura 13. Tipus de tècniques d'anàlisi d'impacte. [17]

Seguint les definicions i principals característiques d'aquestes tècniques donades per articles acadèmics: [17] [18] [19] [20]

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.[4] 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.

Capítol 3. **Estat inicial de l'estructura de requisits**

Per tal d'entendre els avanços aportats en aquest projecte cal analitzar inicialment, l'estat del mòdul de gestió de requisits de la ferramenta de suport de la metodologia TUNE-UP. Com s'ha presentat anteriorment, aquesta eina es basa en el paradigma TDRE [3] de gestió de requisits. Des d'aquest punt de vista, es representen els requisits com un conjunt organitzat de **Proves d'Acceptació** (PAs) que representen un escenari d'execució o un ús del sistema des del punt de vista de l'usuari, i seran precisament aquestes les que marcaran la guia base del desenvolupament software afectant totes les seues fases.

3.1. Anàlisi del Gestor de Requisits

En un procés normal de desenvolupament seguint aquesta metodologia, es defineix un **Work Unit** (WU), [7] que marcarà el context de treball sobre el que es van a realitzar les millores, les solucions d'errors o la implementació de nous requisits sobre el projecte a desenvolupar. Aquesta definició es fa sobre el **Work Unit Manager** (WUM) que podem vore en la Figura 6 i que serà la part central del software sobre la que Analistes, Programadors i Testers controlaran el seu treball relacionat amb el WU sobre el que estiguen treballant en aquest moment.

Una vegada definides les guies o principals idees a desenvolupar en la WU corresponent, aquesta anirà evolucionant dins d'un workflow de treball prèviament definit que, resumint, marcarà el treball que analistes, programadors i testers han de portar a terme en aquest context. Inicialment, l'analista s'encarregarà de les tasques **d'elicitació, anàlisi dels requisits i especificació** d'aquests en forma de proves d'acceptació que hauran de ser **validades** i acceptades pels clients ja que representen les necessitats d'aquests i es poden vore com un contracte entre les dos parts. A continuació, aquestes PAs serviran com a base per a que els programadors **desenvolupen** el codi necessari per tal de **satisfer** aquestes proves, i per altre costat que els testers defineixen les Proves de Sistema (PS) que són combinacions de dades basades en les PAs amb les que **verificaran** a posteriori la validesa o acceptació del codi implementat en relació a les proves d'acceptació definides.

Tot i que el treball dels rols principals esta centrat en l'ús del WUM, no és en aquest mòdul on es porta a terme la gestió de requisits. La definició, gestió, evolució i manteniment de les PAs es la tasca central del **Requirements Manager** (REM) que podem vore en la Figura 7. Aquest mòdul és principalment utilitzat pels analistes i els permet definir el

comportament de l'aplicació que estem desenvolupant en termes de proves d'acceptació. Així i tot, des del WUM tenim accés a la informació de requisits relacionades a la WU corresponent, mitjançant la pestanya **Product Change Scope,** aquesta permet el treball dels programadors i testers sense eixir d'aquest mòdul del programa.

Per accedir al REM ho podem fer des de dos modes de treball diferents: **l'Standard Mode** i el **New WU Preparation Mode.** Encara que la informació mostrada en aquests és més o menys la mateixa, la primera esta pensada per tal de consultar el comportament passat, actual i futur del producte sense trobar-nos en el context de cap WU per tal de tindre més informació a l'hora de prendre decisions, evitant solapaments o inconsistències en els requisits del programa. En canvi, el segon es centra en la idea de definir quines parts de l'estructura dels requisits del producte es voran afectades i especificar les propostes d'evolució dels requisits que es faran en el context d'una WU determinada.

Independentment del mode amb el qual accedim a la informació, el REM està format principalment per dos vistes sincronitzades entre elles. En la part esquerra trobem un graf acíclic dirigit o arbre que representa la denotada estructura principal del requisits. Aquest arbre està format per nodes que son contenidors lògics de requisits, és a dir, contenidors de PAs, i aquests estan organitzats jeràrquicament dins d'aquest diagrama representant de manera lògica l'estructura del software. Esta estructura permet organitzar d'una manera més òptima i estructurada el conjunt de requisits, definint els diversos mòduls o seccions lògiques del programa i assignant les diferents PAs a cada contenidor. Seguint aquesta idea, en el mode New WU Preparation Mode, cada node de l'arbre té associat un check per tal de definir si es veu afectat per la WU actual i definir a continuació els canvis que s'han de fer sobre les PAs. En seleccionar un node de l'arbre, la part esquerra del REM s'actualitza mostrant un grid amb el conjunt de PAs contingudes per aquest node que representen el comportament actual del nostre programa. A banda d'aquesta informació, també trobem referències a l'històric d'aquestes PAs o a l'històric d'actuació de WUs respecte al node seleccionat, documents i IUs associades als nodes o a alguna PA en concret o inclús quins Stakeholders estan interessats en cada un d'estos requisits.

Quan un analista es disposa a especificar els requisits d'una WU, inicialment detecta quins nodes de l'arbre de requisits van a resultar afectats o si és el cas, modifica l'estructura de l'arbre afegint, eliminant o reestructurant nodes. I a continuació, comença la tasca d'especificació dels requisits. Respecte a les PAs actuals contingudes en cada node,

l'analista pot definir-ne de noves (funcionals o no funcionals), pot fer modificacions sobre les actuals, eliminar-ne o marcar-ne alguna per tal de que s'apliquen dins de la WU com a proves de regressió, acotant així el temps i l'esforç a l'hora de validar i verificar el comportament definit en la WU.

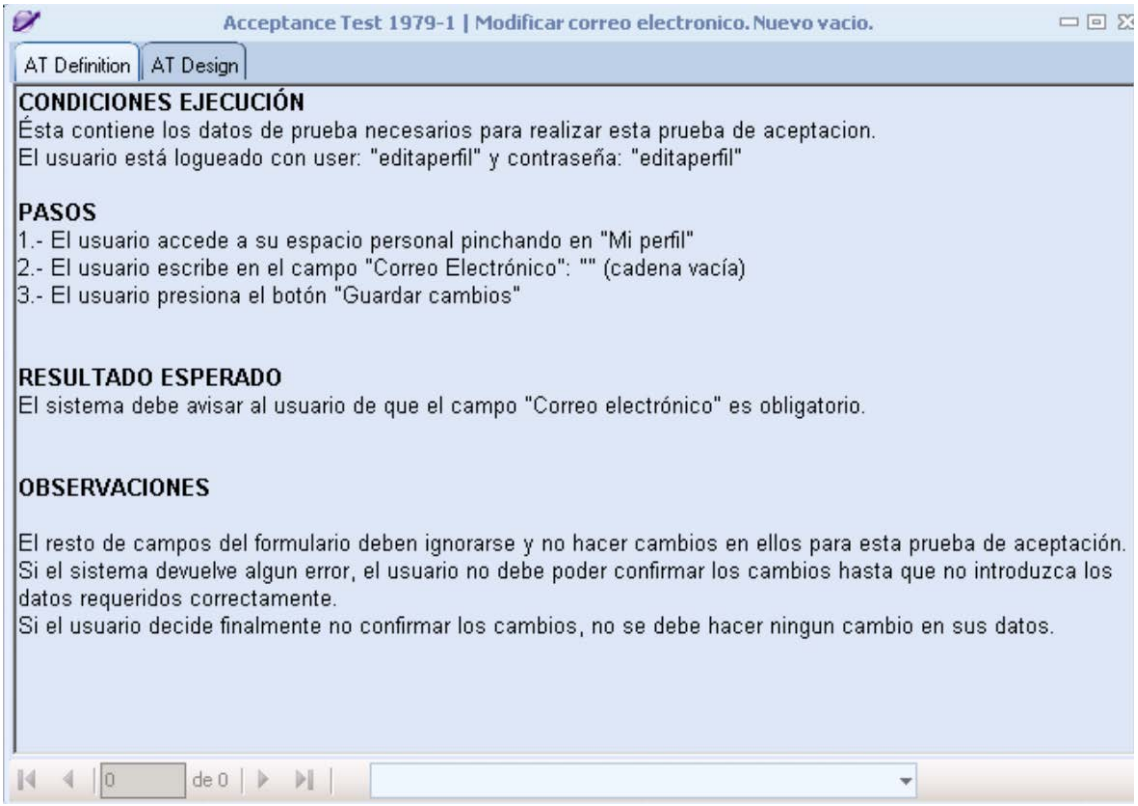


Figura 14. Formulari d'Acceptance Test

Com podem en la Figura 14, a l'hora de crear o modificar PAs accedim al **formulari d'Acceptance Test** on s'especifiquen de manera textual els requisits de l'aplicació. Aquest apartat esta dividit en seccions per tal de fer la lectura més clara i tindre una estructura clara i homogènia comuna a tota l'aplicació. En una primera secció de **Condicions** es denota la navegació que s'ha de fer sobre el programa per arribar a la situació que estem especificant i es defineix la situació de les fonts de dades abans de començar l'aplicació de la PA. A continuació, definim els **Passos** d'execució que volem representar en la prova i marquem els **Resultats Esperats** en aplicar aquests passos. Finalment es complementa la plantilla amb una secció **d'Observacions** on es poden fer anotacions rellevants en relació a la PA definida però que no tenen cabuda en la resta de seccions.

A banda de la pròpia definició, cada una de les PAs tenen definit un estat d'aplicació dins del context de una WU concreta. Aquest es pot vore dins la pestanya **Test Execution** del

WUM (Figura 15) i és utilitzat pels programadors i testers per definir l'estat de cada prova en el seu nivell de testeig determinat. Cada nivell de testeig aplicat es registra independentment i permet visualitzar aquestes aplicacions com un històric de sobre el context d'una PA.

Existeixen 5 nivells d'aplicació sobre PAs[1]: Inicialment l'estat definit és el buit que representa la no aplicació de la prova. Si l'última aplicació ha estat exitosa l'estat apareixerà marcat amb un check d'aprovació amb fons verd, en canvi si s'ha trobat algun defecte, es marcarà amb una aspa amb fons roig. Existeixen a més dos nivells més fruit de la interacció entre nivells de testeig diferents, en primer lloc el símbol d'exclamació sobre fons groc representa que la prova s'ha aplicat amb èxit, però s'ha de tornar a aplicar a conseqüència d'una modificació sobre aquesta o com a sospita de que hi ha una altra prova posterior que pot afectar el seu funcionament. Per últim l'estat representat per un símbol d'interrogació amb fons blau denota que la prova s'ha aplicat amb èxit, però que en algun nivell de testeig posterior s'ha trobat algun defecte i s'hauria de tornar a aplicar.

Figura 15. Pestanya Test Execution

Com he comentat amb anterioritat, el treball ací presentat constitueix la segona fase d'un projecte més ampli. En la primera fase [1][2], es va afegir un nou nivell d'informació rellevant sobre l'estructura dels requisits, amb relacions horitzontals entre nodes que representen la interacció entre requisits dins el context d'un anàlisi d'impacte. La idea es centra en detectar automàticament les relacions d'interdependència existents entre els

requisits, integrant-ho dins de la pròpia metodologia TUNE-UP per tal de reduir o eliminar el cost de recollir, gestionar i actualitzar les relacions de traçabilitat existents.

Per tal d'aconseguir esta detecció automàtica, cal inicialment definir quins seran els criteris, les característiques o els elements que determinaran les relacions. En la nostra proposta són els termes o entitats de domini les que marcaran este comportament. En l'empresa amb la que treballem conjuntament ja existia el concepte de model d'entitats de domini com a conjunt abstracte de conceptes relacionats amb el domini de l'aplicació, però aquest estava pobrament definit, no s'integrava dins de la ferramenta de treball i per tant no es podia extraure informació útil d'aquest element, complicant la tasca dels analistes.

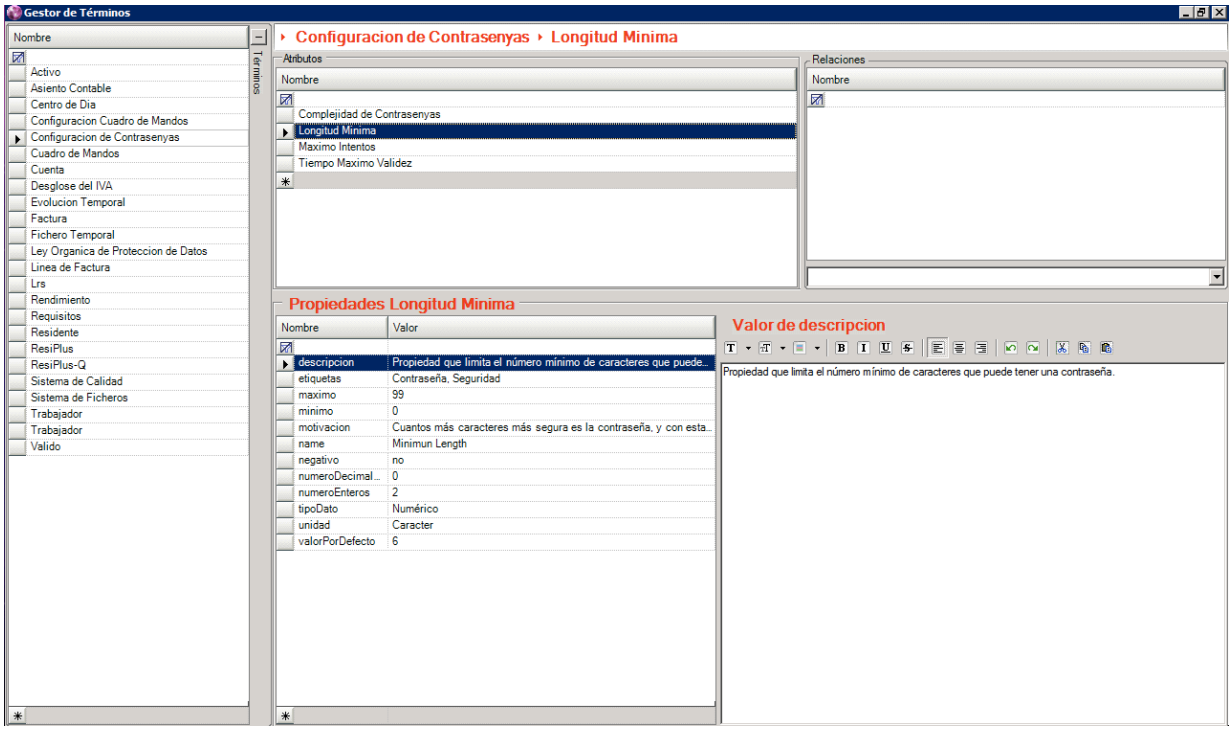


Figura 16. Gestor de Termes o Entitats de domini

Partint d'aquesta idea, es va decidir definir l'anàlisi d'impacte entre requisits basat en relacions de traçabilitat existents entre les proves d'acceptació i les entitats del domini de l'aplicació. Aquetes entitats de domini s'especificaran directament pels analistes en la descripció de les proves i serà el propi sistema l'encarregat de gestionar i crear les relacions de traçabilitat entre els elements, ja siga a nivell de PA o fins i tot a nivell de node.

Per tal de portar a terme aquesta idea, es van crear diversos mòduls d'infraestructura per suportar-ho. Per començar, com podem vore en la Figura 16, es va crear un gestor de termes integrat en la ferramenta que permet als analistes definir el model d'entitats de domini, utilitzant operacions bàsiques de creació, modificació i eliminació de termes. Aquest gestor a més, permet visualitzar la informació de cadascun dels elements servint com a guia o diccionari per als desenvolupadors, estableix un estàndard de definició de les entitats per mantindre un nivell d'homogeneïtat més bo i permet crear relacions entre entitats de domini que podran ser explotades posteriorment en la propia anàlisi d'impacte.

Tota aquesta informació introduïda i gestionada pel gestor, s'emmagatzema internament en una **Base de Dades Orientada a Grafs** (BDOG). En aquesta fase del projecte es va decidir utilitzar aquest tipus de sistema pel seu interès en l'àmbit de la investigació, la seua gran flexibilitat a canvis i sobretot perquè el domini de dades sobre el que estàvem treballant es podia modelitzar intuïtivament com un graf, on els seues nodes podien representar PAs, nodes i entitats de domini i on les arestes mostrarien precisament les relacions jeràrquiques entre elements o les relacions de traçabilitat que utilitzades com a base en l'anàlisi d'impacte de requisits. Per al nostre treball i després d'una comparativa entre les diferents alternatives existents ens vam decantar per Neo4j com a BDOG ja que es tractava de l'opció més madura i que ens permetia una comunicació basada en servicis REST, solucionant-nos els possibles problemes d'incompatibilitat tecnològica. En la Figura 17 podem vore una xicoteta instància que ens mostra com s'emmagatzema la informació seguint aquest paradigma.

Com hem comentat anteriorment i es pot observar, en aquesta base de dades no solament es registra la informació referent a les entitats de domini, si no que també es contemplen referències als nodes i les PAs del gestor de requisits (emmagatzemades en una base de dades relacional), pel que va ser necessari crear un mòdul de sincronització entre les dos seccions del projecte per tal que les referències a estos elements estigueren actualitzades constantment, havent de fer actualitzacions en crear nodes o PAs, en eliminar algun dels elements, en moure una PA a un altre node o en modificar alguna de les característiques emmagatzemades també a la BDOG com el nom, l'identificador o el més important el tipus de PA sobre la que estem actuant (PA actual o Proposta).

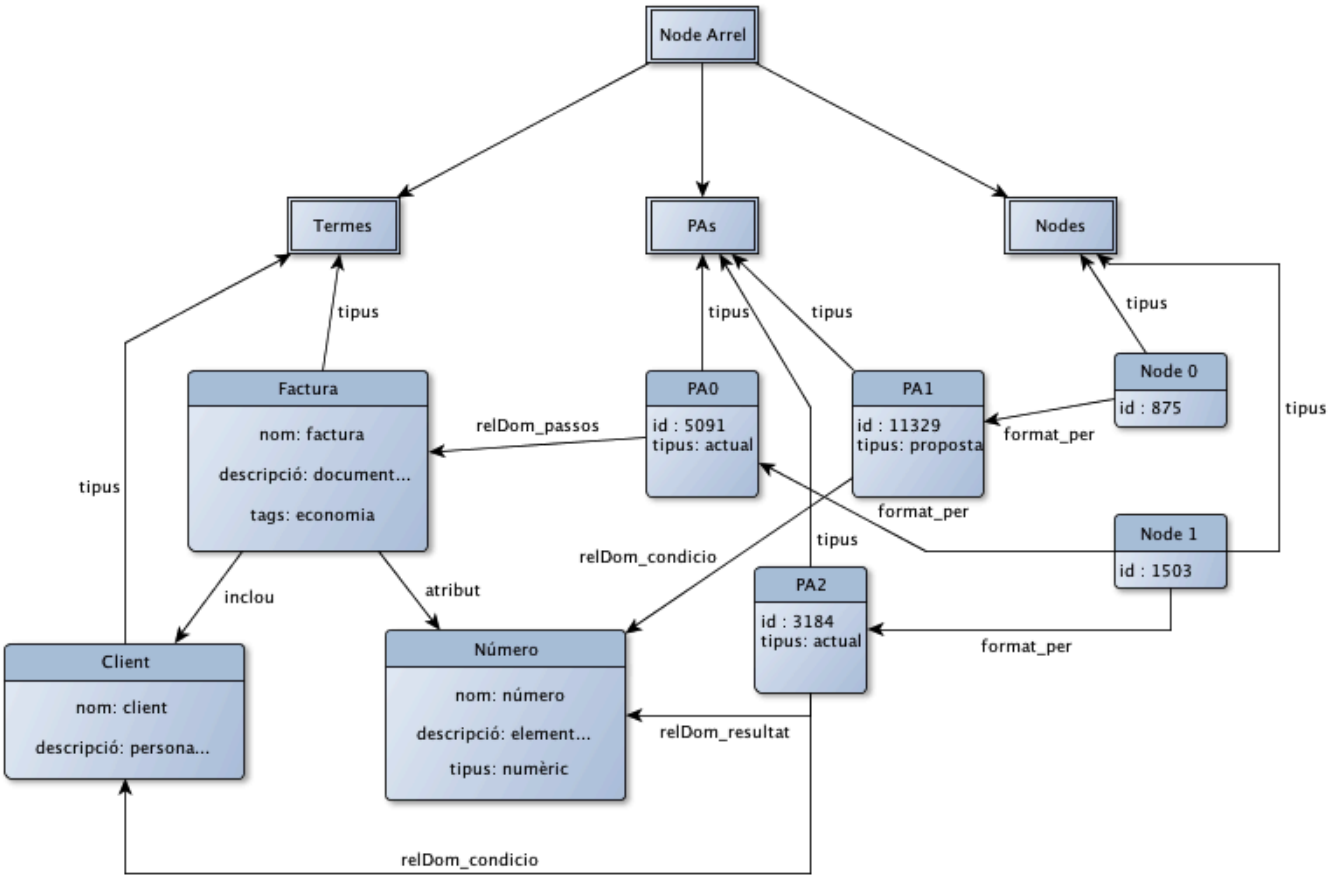


Figura 17. Instància de la base de dades de grafs

A banda d'aquests apartats, calia oferir una eina simple per tal que els analistes pogueren especificar les entitats que afecten a cada PA dins de la ferramenta actual i sense que suposes un esforç addicional. Per tal d'aconseguir-ho vam basar-nos en un mecanisme *intellisense* com el que ens ofereixen la major part de les ferramentes de suport per al desenvolupament (Microsoft Visual Studio, Eclipse,...) que mitjançant un quadre de text expansible amb una combinació de tecles ens permet navegar i seleccionar elements dins del model d'entitats de domini definit anteriorment en el gestor de termes.

En seleccionar un element a l'*intellisense*, aquest apareixerà reflectit en l'especificació textual de la PA amb un estil característic. Aquesta diferència textual serà precisament la que s'utilitzarà per parcejar la descripció i detectar les relacions de traçabilitat existents entre la PA definida i les Entitats seleccionades i actualitzar així la base de dades de grafs amb la nova informació. Una vegada definit com es crearan automàticament les relacions de traçabilitat entre els diferents elements i vistos els mòduls creats per tal de donar suport

a aquest treball, únicament queda per mostrar com es visualitzarà i s'explotarà aquest anàlisi d'impacte.

Es va definir l'anàlisi d'impacte a dos nivells de granularitat: una proposta inicial centrada en l'anàlisi de la relació de dependència entre PAs i un segon anàlisi a nivell de nodes. Un exemple de la visualització a nivell de PA el podem observar en la Figura 18 on veiem com una prova es relaciona amb altres dos. En la imatge es pot vore com per a cada una de les relacions es remarca informació com ara a quin node pertanyen les PAs afectades o quina entitat del domini ha relacionat les dues proves.

Amb la implantació d'aquest projecte en un entorn de producció s'esperava que aportés un gran conjunt de relacions de traçabilitat dins del context d'un producte software real i així poder refinar les condicions i filtres utilitzats per a l'anàlisi, però amb la implantació d'aquest i altres mòduls hem detectat certes carències en la representació de l'estructura de dades i hem aparcat momentàniament aquest projecte per buscar alternatives i aportar un valor afegit a la gestió dels requisits software.

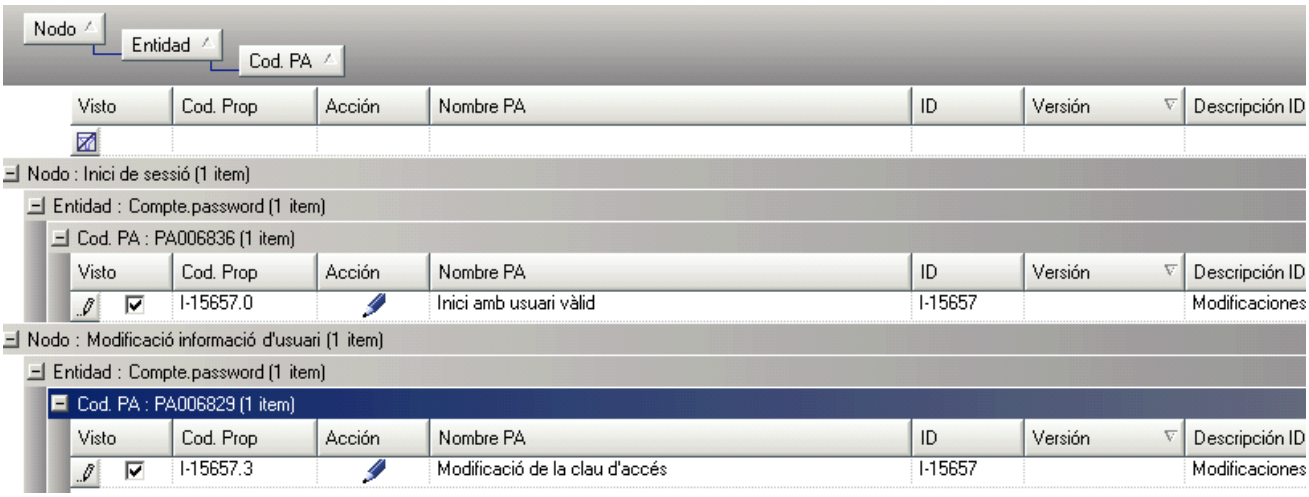


Figura 18. Vista de l'anàlisi d'impacte sobre una PA

Deixant l'anàlisi d'impacte de banda, i centrant-nos en un projecte paral·lel però també relacionat amb la implementació de la ferramenta, s'han definit altres conceptes en relació al manteniment de la definició de requisits [21], com son el concepte de PA no definida o faltant o de refactorització de PAs i Nodes.

Es possible que en determinades situacions, les especificacions queden obsoletes o incoherents amb el comportament real de la ferramenta, ja siga perquè s'ha començat a utilitzar la metodologia en un projecte iniciat amb anterioritat o per exemple perquè degut a les necessitats temporals del projecte s'haja decidit reduir temporalment l'esforç dedicat a la fase d'anàlisi com malauradament passa en algunes situacions. Es per això que s'han definit una sèrie de ferramentes per als analistes per tal de que en cas de trobar-se en casos de inconsistències o falta de definicions puguen deixar constància d'aquest fet en la pròpia ferramenta. Aquesta ferramenta els permet actuar adequadament, seguint la metodologia TUNE-UP, en trobar-se en alguna situació com les abans esmentades i deixant un registre més consistent en l'eina per tal que el product manager puga analitzar millor on i com s'han utilitzat els recursos de què disposa en una versió determinada.

3.2. Problemàtica existent

En el punt anterior hem vist quines son les ferramentes que disposem per tal de gestionar el cicle de vida dels requisits d'un producte en la ferramenta de suport a la metodologia TUNE-UP. L´ús diari d'aquestes ens ha donat l'oportunitat de descobrir algunes limitacions o carències existents i començar a plantejar millores que puguen aportar solucions simples però efectives a algunes situacions concretes.

Per tal d'estudiar les necessitats i possibles millores a implementar vaig a separar els resultats trobats en funció de si son problemes relacionats amb la representació actual de l'estructura de requisits en forma d'arbre o si per contra son carències detectades sobre la informació extreta de l'estructura i que podrien ser esmenades amb la incorporació de nous mecanismes.

Com ja s'ha comentat amb anterioritat, l'estructura de requisits en la ferramenta es basa en un graf acíclic no dirigit on cada node actua com a contenidor lògic de PAs. Aquest està representat en forma d'arbre jeràrquic utilitzant un element comú de les interfícies gràfiques d'usuari com es el Treeview i parteix d'un node fictici que representa el programa sobre el que estem definint els requisits software. L'aplicació actual, ens permet gestionar els requisits de diversos programes i inclús compartir requisits entre diferents productes. Per a cada node se'ns mostra el seu nom, el seu codi identificador i en un panell auxiliar informació addicional sobre ell com per exemple una descripció sobre les característiques d'este node i dels requisits que conté.

Aquesta representació és molt útil i intuïtiva ja que se'ns mostra d'una manera molt visual la representació jeràrquica dels nodes, i ens facilita la identificació i navegabilitat de les diferents parts de l'aplicació que es poden vore afectades en una determinada WU. Malauradament, amb l'ús constant d'aquesta ferramenta i després de gestionar diversos productes software de diferent envergadura, s'han trobat certes limitacions o problemes d'optimització relacionats amb ell.

Per començar cal destacar que, en projectes amb una certa grandària, la representació arbòria dels requisits creix amb una magnitud tan elevada que dificulta la tasca d'identificar els nodes afectats com es pot vore en la Figura19, però sobretot, complica el treball posterior d'especificació de requisits en proves degut a la complexitat de l'arbre. Esta **complexitat**, a banda de distraure l'atenció dels analistes i baixar la seua productivitat, també té un alt cost computacional a l'hora de carregar l'estructura, navegar entre elements i fer cerques o altres operacions sobre l'arbre.

Per tal d'evitar el problema del temps de càrrega de l'arbre de requisits, s'han portat a terme diferents alternatives d'optimització. Inicialment es varen modificar els algoritmes de càrrega així com les estructures de dades utilitzades per al seu funcionament, però encara que les millores respecte a la versió inicial varen ser notables, els **temps de càrrega** per a certs contextos de l'arbre (producte, WU) seguia sent massa elevat. Finalment i després de diversos intents, es va decidir per centrar-se en una estratègia incremental de la càrrega de l'arbre. En el mode Standard, únicament es carreguen els nodes de primer nivell de l'estructura dels productes seleccionats, és a dir, aquells que tenen com a pare el node producte, i es van afegint nodes a l'estructura en funció de la necessitat o a l'expandir un node determinat. En el mode amb context WU, es va decidir expandir únicament aquelles seccions de l'arbre on es trobaren els nodes afectats per una determinada WU i en una segona optimització únicament aquells nodes que no tingueren altres nodes ascendents entre els afectats.

Tot i que aquestes solucions van tindre el resultat esperat, reduint considerablement el temps de càrrega, molts dels problemes associats a la complexitat d'aquesta vista persistien i a més, s'havia **perdut visibilitat** respecte a l'estructura global del producte.

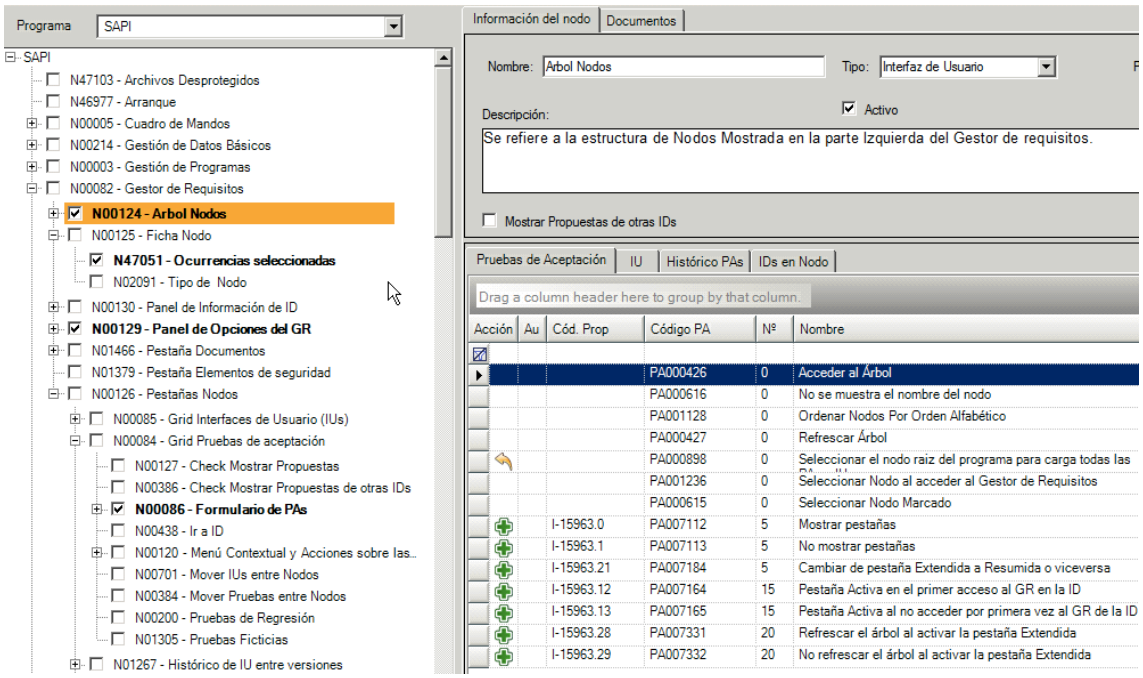


Figura 19. Representació arbòria de l'estructura de requisits d'un programa.

A més d'açò, el que pot representar el major problema o repte d'aquesta vista, està relacionat directament amb la naturalesa de l'estructura utilitzada: en haver utilitzar un graf acíclic no dirigit no existeix cap limitació per poder definir nodes amb múltiples pares. Aquesta característica té molt de sentit des del punt de vista de l'analista, ja que podem trobar-nos un mateix conjunt de funcionalitats en diferents parts del software que siguen comuns independentment de la seua localització. Per tant, açò ens permet reutilitzar requisits sense haver de redefinit tota la funcionalitat. Tot i això, aquesta eina que inicialment s'havia pensat per ser utilitzada en casos molt particulars, va començar a estendre's fins arribar per exemple a que un node i els seus descendents de un projecte real podia ser accedit des de 63 pares diferents.

Aquest ús de la ferramenta té implicacions directes en el significat d'algunes característiques de l'estructura de requisits. Per començar cal destacar que des d'este nou punt de vista, anem a trobar **nodes repetits** en distintes parts de l'arbre d'un producte i aquest fet té conseqüències en diverses activitats i opcions de l'arbre. La visibilitat de l'estructura es veu afectada, ja que al repetir la mateixa informació diverses vegades la visualització s'ofusca i perd claredat, a més, si això ho ajuntem a la ja elevada complexitat de l'arbre, podem no saber sobre quina instància del node estem treballant. Una altra qüestió a tindre en compte és el nou nivell de complexitat que aporta a les cerques sobre

nodes o sobre PAs, a l'hora de detectar totes les instàncies d'un mateix node o a l'hora de fer certes operacions com ara aquelles relacionades amb la reestructuració de l'estructura de l'arbre.

Aquest punt de vista també comporta una complexitat afegida per als analistes, ja que en la seua tasca de detecció dels nodes afectats per una determinada WU, han de tindre en consideració totes les instàncies del node, és a dir, totes les parts del programa on podem trobar la funcionalitat especificada per els requisits del node repetit. Aquesta qüestió ens mostra algunes carències de la ferramenta actual en relació a aquest fet, com per exemple l'opció demanada pels analistes de l'empresa, de poder diferenciar i especificar en algunes situacions quines instancies del node volem que afecten o sobre les que volem que s'apliquen les modificacions fetes sobre la WU, sense haver d'aplicar obligatòriament tots els canvis a tot el conjunt d'instàncies.

Un altre problema que hem extret del treball diari amb el mòdul de gestió, és la limitació en la visualització de l'històric de l'estructura de dades de requisits. Com hem destacat altres vegades en este treball, **l'evolució constant** del software és una de les seues característiques que també s'aplica a l'estructura dels seus requisits, per tant poder mostrar l'evolució que ha tingut aquesta pot aportar-nos informació sobre el perquè de molts canvis o decisions sobre ella.

Si bé és cert que la ferramenta conté un registre històric dels moviments de l'estructura (creació, eliminació o moviment de nodes) amb informació sobre el moment i l'agent que han portat a terme el canvi, aquesta informació no és tan visual com alguns casos ho poden requerir. Per exemple, és comú trobar-se que en el context d'una WU concreta es varen definir unes propostes de modificació i regressió sobre les PAs actuals existents d'un determinat node i que aquest ha sigut canviat de lloc en una altra WU o inclús d'una altre versió del producte. Aquest tipus de moviments poden deixar xicotetes inconsistències en la redacció de l'especificació d'una prova que pot portar als analistes a malentesos. En l'actualitat els analistes en cas de vore aquest tipus de comportament poden accedir al formulari de l'auditoria de l'arbre per detectar perquè o com s'ha produït, però no tenen cap eina per visualitzar com era l'estructura de requisits en el moment en que es van definir els canvis per a aquesta WU..

Com a últim problema destacat dins de la categoria d'aquells relacionats amb la representació actual de l'estructura de requisits trobem la **inexistent representació de**

l'anàlisi d'impacte de requisits en l'estructura arbòria actual. Si intentàrem afegir també la informació relacionada amb les relacions de traçabilitat entre PAs i nodes, la complexitat i l'opacitat de l'arbre augmentarien considerablement, i serien contraproduent per als nostres propis interessos en la gestió de requisits. De totes maneres, encara que ho intentàrem aplicar, ens trobaríem que per la pròpia naturalesa acíclica del graf representat en el Treeview, seria impossible afegir la major part d'aquest tipus de relacions.

Encara que no cal que tota aquesta informació es trobe dins del mateix diagrama, el fet de que la visió de l'anàlisi d'impacte no es mostre en la finestra principal i que es situe en canvi com una ferramenta addicional al propi curs habitual de la metodologia i sobretot la seua representació poc visual com es pot vore en la Figura 20, fan que la seua explotació i per tant la seua utilitat es vegen reduïdes considerablement.

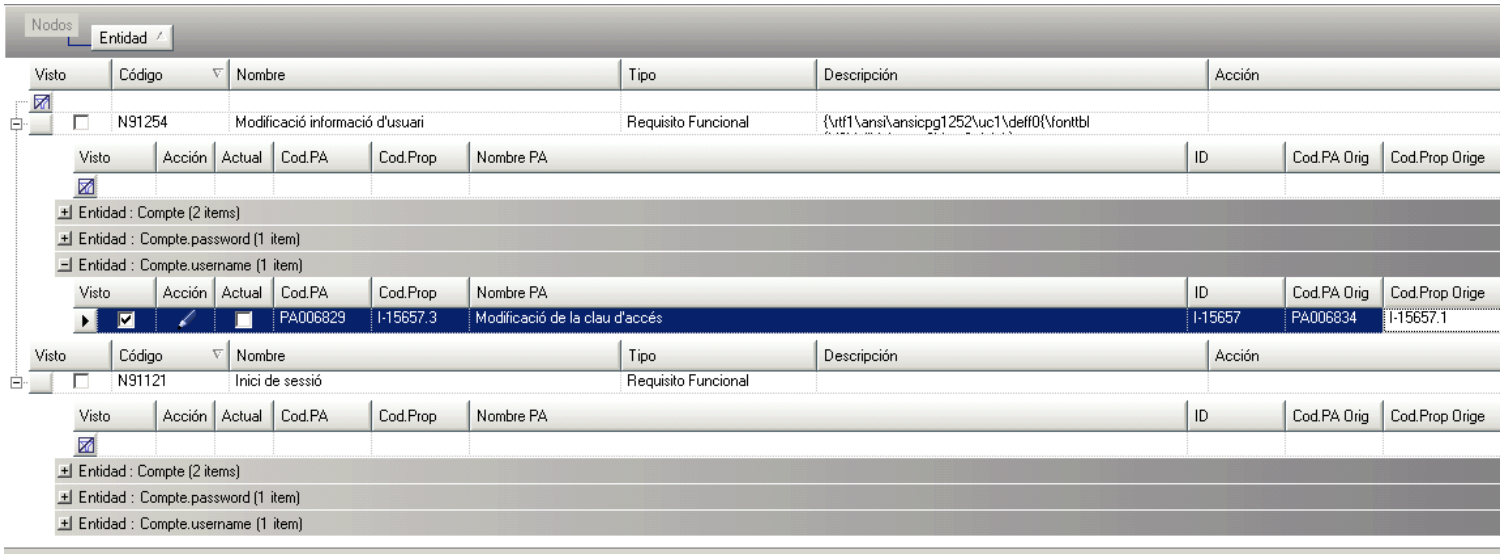


Figura 20. Formulari d'anàlisi d'impacte sobre un node.

Cal destacar que inicialment es va decidir representar aquesta informació mitjançant l'ús de grids multinivell, ja que el gran esforç de la primera fase del projecte es va centrar en crear la infraestructura necessària per aconseguit una anàlisi d'impacte automàtic i no ens vàrem detindre tant en la part de visualització i explotació de la informació aconseguida fins que no s'aportaren els primers resultats del projecte en producció. A més, aquest tipus de visualització era el més econòmic en termes de temps d'implementació degut a l'experiència dels membre de l'equip en aquest tipus de control i sobretot per l'ús constant que feien els analistes de diagrames amb controls semblants i que els evitava una corba

d'aprenentatge pronunciada sobre la nova visualització. Encara així, com es pot vore en la imatge superior, la visibilitat de la informació dista molt de ser clara, concisa i d'aportar-nos algun tipus de coneixement útil amb un simple colp d'ull, pel que seria interesant buscar alternatives més adequades per al nostre context.

Canviant de categoria respecte a les carències de l'aplicació actual, ens centrarem ara en les situacions analitzades on ens ha fet falta alguna informació relativa a l'estructura del producte i on hem trobat que no tenim una anàlisi general de la informació continguda o algun tipus d'estadística que ens puga ajudar a respondre a preguntes senzilles o a encaminar-nos en la direcció correcta a l'hora de prendre decisions.

Aquest tipus d'informació pot ser útil a agents encarregats de tasques molt diferents amb tipus de rols completament distints. Des del punt de vista de l'analista de la ferramenta, hem necessitat de cert tipus d'informació concreta que era inexistent o difícil de recol·lectar amb les opcions actuals. Per un costat, conèixer el nombre de proves d'acceptació que conté cada node pot ser útil per entendre la importància que té cada part del software que estem desenvolupant o veure sobre quines seccions hem dedicat més esforç, però sobretot ens pot deixar entreveure nodes sobrecarregats amb una quantitat desmesurada de proves que dificulta la tasca de gestionar els requisits i que a més dilueix el sentit del node com a contenidor lògic de proves amb característiques semblants o que afecten a un mateix elements. Si tinguérem alguna eina que ens permetés detectar aquest tipus de comportaments, podríem organitzar millor els requisits creant subnodes amb comportaments més específics dins d'un subgraf millor organitzat sota el node superpoblat. Aquest tipus de refactorització és realment útil, de la mateixa manera que ho és la refactorització de codi, ja que ens facilita les futures tasques d'especificació i manteniment de requisits.

Una altra qüestió a tindre en compte des d'aquest punt de vista, és el nombre de camins distints per arribar a un node. Com comentàvem anteriorment, un dels grans problemes que existeixen en la representació actual és la possibilitat de tindre nodes amb diversos pares, aquesta ferramenta tan útil en certes ocasions ens pot crear complicacions i pèrdues de rendiment degut al alt cost computacional de calcular tots els camins possibles de cada node, informació necessària per a algunes de les operacions sobre nodes. Tindre una vista o representació que ens mostre els nodes amb més instàncies, ens pot ajudar a analitzar si realment aquestes instanciacions són correctes, és a dir, si realment estem enfront de

requisits que s'han de tindre en compte en distintes parts del programa o si per contra estem davant d'un intent de reutilització de nodes erroni que incrementa el nombre i el nivell de complexitat de les proves afectades.

Quan parlàvem en el punt anterior del projecte portat a terme en relació a la millora de les ferramentes de manteniment de requisits, ens centràvem en conceptes com refactorització de proves o proves faltants per tal de representar estats incoherents entre els requisits de la nostra aplicació i la seua corresponent especificació i que així quedés constància d'un treball que els analistes ja estaven portant a terme, però sense comptar amb les ferramentes adequades. Amb la introducció d'aquests elements, s'ha millorat considerablement el manteniment dels requisits, però hem trobat a faltar una visió general sobre l'estat de les inconsistències de l'anàlisi. Si tinguérem algun tipus de representació centrada en aquesta informació ens permetria planificar millor el treball dels analistes i pot ser una eina molt útil per al product manager a l'hora de planificar el treball del seu equip envers una nova versió del software, ja que si en aquesta anem a dedicar-nos a millorar una part concreta del programa tindre una visió sobre l'estat de l'anàlisi pot marcar la diferència respecte al treball a realitzar en les seues fases inicials.

Per últim, i canviant el punt de vista o millor dit el rol més interessat amb la possible millora de la representació del requisits, caldria destacar els testers. Actualment l'equip de testeig no es veu plenament involucrat amb l'estructura global del requisits d'un programa. En canvi, la seua visualització de les proves d'acceptació es defineix a nivell d'una WU determinada, amb la mostra local del conjunt de propostes que han de testejar, ja siga manualment o automàticament en eixe context determinat. Tot i que la tasca de decidir on centrar els esforços en una determinada versió recau en la figura del product manager, pensem que el responsable de l'equip de testers pot aportar coneixements més experts en la direcció de l'estratègia de testeig global de l'aplicació i seria interesant tindre una visió general de l'estat del testeig sobre l'especificació de requisits.

Amb aquestes carències o millores ja detectades, podem començar a definir quines seran les nostres estratègies per intentar solucionar els problemes i donar un valor afegit als usuaris de la nostra metodologia i ferramenta de suport per al desenvolupament software. Per acabar aquest capítol i a mode resum, voldria remarcar les limitacions detectades:

- Sistema limitat a un únic tipus de representació de relacions.

- Problemes de rendiment i visió enfront programes de gran envergadura.
- Informació repetida. L'ús de la reutilització de nodes fa que un node reutilitzar aparega repetit en totes les seues ocurrències.
- Carència d'informació històrica o d'un element navegador d'estructures antigues del producte.
- Deficient representació dels elements de l'anàlisi d'impacte que minimitzen la seua explotació.
- Inexistents vistes estadístiques d'anàlisi sobre els requisits.

Capítol 4. **Visualització i anàlisi de dades**

4.1. Principis

Abans de començar a analitzar les possibles solucions als problemes detectats en el capítol anterior, trobem que seria una bona idea analitzar les diverses alternatives existents en matèria de visualització de dades i visualització de dades massives, per tal d'extraure idees claus i bones pràctiques que puguen ser aplicades al nostre model concret. Encara que els diversos estudis provinguen d'àmbits d'ús molt diferents amb finalitats i pràctiques diverses, pensem que es poden discernir una sèrie de premisses o idees centrals que poder aplicar en la creació dels nous mòduls de visualització.

Seguint aquesta idea, anem a analitzar múltiples articles de investigació relacionats amb les temàtiques que interessen, estudiarem les diferents alternatives existents en producció i sobretot ens centrarem en les ferramentes i metodologies actuals de visualització i gestió de requisits.

Cal destacar que la disciplina centrada en la visualització de requisits es troba actualment en una fase d'àmplia expansió. En els últims anys s'han escrit diversos articles al respecte i fruit d'aquest interès creixent, es va decidir organitzar a partir de l'any 2006 el primer taller exclusivament centrat en aquesta temàtica, el Requirement Engineering Visualization Workshop (REV) en el context de la Requirement Engineering Conference (RE) i que es va celebrant any rere any. Gran part dels articles que hem analitzat, es situen dins el marc d'aquesta conferència, però també hem volgut incloure idees extretes d'altres articles i tesis amb temàtiques que hem trobat interesants.

Aquest interès, naix de la facilitat i rapidesa que aquest tipus de visualització d'artefactes ens aporta per entendre i compartir el coneixement sobre els requisits entre els diferents agents implicats. Encara així, ens trobem davant un gran problema difícil d'abordar degut entre altres coses a la seua falta d'estructura general i a la gran diferència existent entre la representació tradicionalment textual i una possible representació visual que ens complica la transformació entre les dos.[22]

Segons Marchese et al [23], la visualització és el procés que involucra la recol·lecció, el processat, el renderitzat, l'anàlisi i la interpretació de les dades o requisits, o dit d'una altra manera [29], és l'acció de presentar la informació en la pantalla. La seua finalitat principal es centra en facilitar l'enteniment del producte per fer millorar el comportament dels desenvolupadors a diferents nivells del procés de creació i es precisament per açò que hem decidir posar els nostres esforços en este tema.

En aquest capítol, analitzarem inicialment les alternatives que podem trobar actualment en les ferramentes de gestió de requisits, intentarem extraure les idees principals que ens puguen ser útils en el nostre treball actual i estudiarem les diverses línies d'investigació futures existents.

4.2. Visualització actual de requisits

Quan parlem de visualització de requisits o de la seua estructura interna en el món empresarial actual, hem de fer una clara associació a UML i a les ferramentes CASE de suport basades en aquest estàndard. Aquestes eines són àmpliament utilitzades en àmbits de treball i han anat madurant en el temps fins oferir diverses opcions de visualització i anàlisi molt útils. Entre elles podem destacar DOORS [1] i Requisite Pro [2], actualment propietat de IBM ambdues, IRQA [3] i Open Source Requirement Management Tool (OSRMT)[4] que com el seu nom ens indica es tracta d'un projecte de desenvolupament de software lliure.

La idea base sobre la que treballen, és la definició textual de requisits mitjançant casos d'ús. Aquests són textos seccionats, molt concrets i específics, que representen escenaris que els distints agents de l'aplicació han de poder portar a terme amb èxit per satisfer les necessitats dels clients. Junt a aquesta idea, s'utilitzen també diagrames de Casos d'ús que permeten visualitzar d'una manera més gràfica el comportament general de cada un d'estos requisits, definint aquesta representació a distints nivells de profunditat en funció de les necessitats dels agents. En aquesta representació del comportament dels requisits, també s'utilitzen moltes vegades diagrames de seqüència que permeten mostrar a un nivell més

[1] www.ibm.com/software/awdtools/**doors**/

[2] *www.ibm.com/software/awdtools/req**pro**/*

[3] *www.visuresolutions.com/**irqa**-requirements-tool*

[4] ***source**forge.net/projects/osrmt/*

baix, el comportament dels requisits especificats. Aquesta opció pot resultar especialment útil per als programadors per la proximitat entre la seua concepció i la seua manera de pensar i concebre el software. Podem vore dos exemples d'aquests tipus de representació en la Figura 21.

Figura 21. Exemples de Diagrama de Casos d'ús i Diagrama de seqüència de la ferramenta DOORS. [26]

A l'hora d'estructurar els requisits, aquestes ferramentes ens aporten bàsicament dos visions clarament diferenciades. Per un costat tenim una vista tradicional caracteritzada per la representació seguint l'estàndard dels diagrames de casos d'ús, on es mostren els requisits codificats seguint les especificacions UML com es pot vore en la Figura 21. En aquesta, a més, s'organitzen els requisits dins de paquets amb una certa organització jeràrquica, però sempre mantenint la granularitat dels casos d'ús en un nivell d'abstracció molt més alt del que podem necessitar nosaltres en la nostra ferramenta. Aquesta és precisament la raó per la que inicialment, la metodologia TUNE-UP va decidir allunyar-se

dels casos d'ús, ja que, encara que aquests permetien especificar els requisits d'una manera molt simple i al mateix temps completa, estava pensada des d'un punt de vista inicial en el desenvolupament d'un producte. En canvi en la nostra metodologia, més centrada en la gestió i manteniment d'un producte de gran envergadura, la gran granularitat d'aquest tipus d'artefactes no ens faciliten la tasca d'addició de nous requisits, millora dels actuals o solució d'errors.

Per altre costat, com veiem en la Figura 22 associada a la ferramenta OSRMT, però que també podem trobar en la resta de ferramentes enunciades anteriorment, es defineix la segona visualització esmentada, com un TreeView jerarquitzar amb els requisits de la ferramenta. Encara que esta representació puga semblar pareguda a la que presentem nosaltres en la ferramenta TUNE-UP, el plantejament sobre el que es basen és completament diferent. En aquest cas, es tracta d'una representació de requisits en forma de casos d'ús (escenaris textuals), que podem vore organitzats en el grid de la dreta de la imatge, com un únic contenidor global on es troben tots els requisits. Per altre costat, en la part dreta de la imatge, veiem com este conjunt, s'organitza amb certa simplicitat en una estructura jeràrquica basada en carpetes, però sense poder reutilitzar requisits (en el nostre cas PAs) dins d'aquesta estructura.

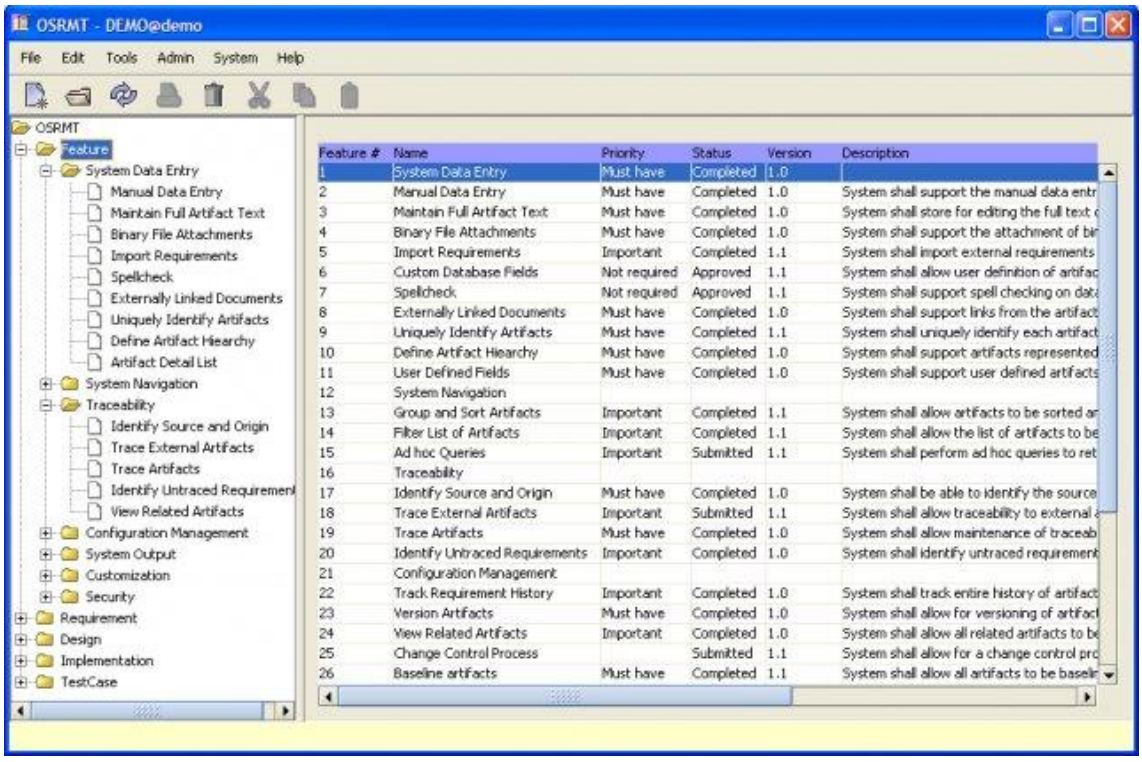


Figura 22. Estructura de requisits de la ferramenta OSRMT.

El principal problema d'aquest tipus de representació dual de les ferramentes de gestió de requisits més conegudes, és la falta de sincronització o inclús d'equivalència entre les dues visions. En l'actualitat no existeix una equivalència automàtica entre els artefactes que es mostren en el diagrama de casos d'ús i les especificacions textuals organitzades en el TreeView. Tot i que es podria portar a terme una cert control de manera manual, les ferramentes no ens donen cap tips de suport per a gestionar aquesta carència. A més, amb l'ús d'aquests tipus de diagrames, a la pràctica els agents intenten representar el màxim d'informació possible en un mateix context amb l'esperança de no haver d'escriure tant. El resultat, en canvi, és una representació enrevessadament complicada que ocupa més que el text equivalent i que és més difícil de llegir i extraure coneixement [28].

Per altre costat, un altre tema que nosaltres tenim molt present com a conseqüència de la primera fase del nostre projecte, és la representació de les relacions de traçabilitat entre requisits per tal de poder portar a terme un anàlisi d'impacte clar i útil. En aquestes ferramentes tant la definició de les relacions com la propia anàlisi, es fan mitjançant l'ús d'una matriu de traçabilitat com la que podem vore en la Figura 23. Aquesta tècnica tradicional tot i tindre els seus avantatges, com pot ser la seua simplicitat i la lleugera corba d'aprenentatge, resulta incomprensible i difícil de manejar amb un conjunt mitjà o elevat de requisits (almenys amb la granularitat amb la que nosaltres treballem). Per tant, hem de buscar alternatives que fent ús del nostre sistema automàtic de relacions de traçabilitat, ens permeten visualitzar i analitzar millor la informació aconseguida.

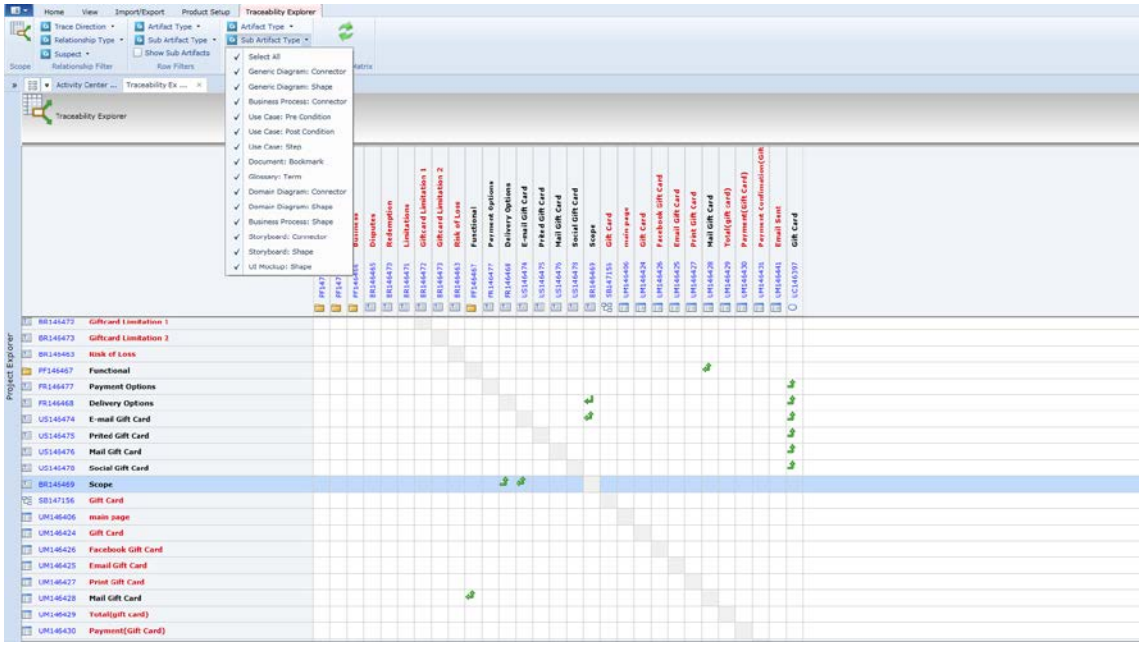


Figura 23. Anàlisi d'impacte mitjançant matrius de traçabilitat

Després de vore les propostes de visualització més utilitzades, i tenint en compte les necessitats del nostre sistema, hem decidit que no anem a seguir estes característiques. Encara així, en la ferramenta DOORS, hem trobat una vista estadística com la que veiem en la Figura 24 on es mostren una sèrie de mètriques molt bàsiques respecte a l'estructura de requisits. Aquesta perspectiva, tot adaptant-la a les nostres necessitats i millorant la informació que ens aporta, pot ser una poderosa ferramenta per tindre una idea global de la situació actual del nostre producte.

Figura 24. Estadístiques de Casos d'ús de la ferramenta DOORS. [26]

4.3. Alternatives existents

Abans de començar a analitzar les diverses alternatives que hem trobat en articles d'investigació, tesis i altres documents diversos, m'agradaria començar per enumerar o distingir les 5 categories genèriques de visualitzacions de dades existents[22]:

1. **Visualització tabular** és aquella formada per un conjunt d'interseccions de files I columnes on emmagatzemar típicament informació textual.

2. La **Visualització relacional** consisteix en una col·lecció de nodes i connectors que descriuen o indiquen la relació entre components o un sistema, però no mostren implícitament la relació d'ordre inherent de les operacions del sistema.

3. **Visualització seqüencial** és aquella que manté un ordre d'operació entre distintes parts d'un sistema o respecte als usuaris i el propi sistema. Aquest és el tipus tradicional que podem vore en els flow charts o en els diagrames d'estats.

4. La **Visualització jeràrquica** implica la descomposició del sistema en parts, com s'utilitza típicament en l'aproximació de modelat basat en objectius.

5. **Visualització quantitativa/metafòrica,** és aquella representada normalment en forma de gràfics de barres, circulats o altres figures que representen la informació, a més, també s'inclouen en esta categoria tècniques més sofisticades que utilitzen metàfores visuals o altres estratègies semblants com ara l'ús de colors, formes i grandària com a elements diferenciadors.

Partint d'aquesta categorització inicial, podem anar analitzant totes les alternatives que trobem en estos 5 grans grups i anar definint els seus aspectes positiu, negatius i si poden ser útils per al nostre cas concret.

Les visualitzacions analitzades en el punt anterior, podrien ser incloses dins de la categoria de representació tabular: per un costat tenim el grid de casos d'ús i per l'altre la matriu de traçabilitat de requisits. Com hem dit adés, aquest tipus de representació utilitza elements molt simples i naturals per tal de mostrar una quantitat ingent de dades de caràcter textual, però al mateix temps no ens dona una visió global o abstracta d'allò que volem representar. Aquest tipus de visualització, caracteritzat sobretot en la figura del grid, és forçament utilitzat en la nostra ferramenta, però no per a les necessitats actuals del projecte.

L'altra vista que hem vist anteriorment, els diagrames de casos d´ús, es podrien englobar dins la categoria de visualització relacional. Aquest tipus de representació és altament intuïtiva i ens permet vore amb un simple colp d'ull l'estructura i sobretot les relacions existents entre diferents elements. Tot i que els diagrames de casos d'ús no s'aproximen a les nostre necessitats actuals, la idea sobre la que es basen sí.

Les representació basada en grafs [32], [27], [29], [30], [31] com les de la Figura 25, poden mostrar d'una manera natural, les relacions que hi ha enter tots els artefactes que afecten a la gestió de requisits. De fet, la representació teòrica d'aquesta interacció entre elements és defineix precisament com un graf i pensem que aquesta pot ser una solució molt interesant per mostrar diverses característiques del nostre sistema. Així i tot, sempre cal recordar que si

representem un nombre elevat de dimensions en un únic diagrama, aquest pot ser molt complicat de desxifrar, afectant a la pròpia naturalesa d'aquesta figura matemàtica.

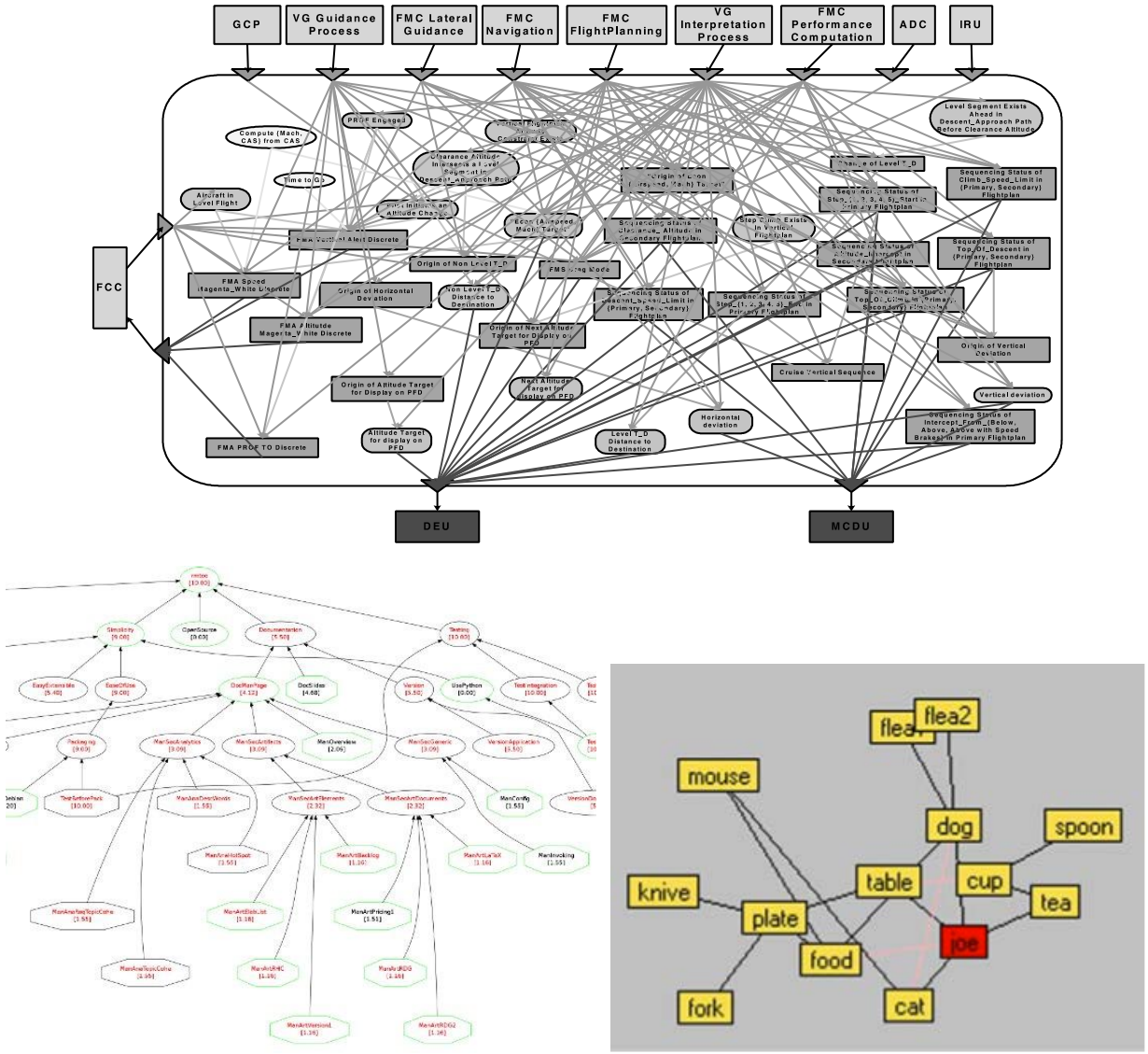


Figura 25.Representacións basades en grafs.

De totes maneres, si veiem les imatges podem detectar la simplicitat i l'eficàcia que aquesta representació pot tindre a l'hora de mostrar l'estructura d'un producte i les seues diverses relacions. Aquest tipus de visualització serà abordat amb més deteniment en el Capítol 6, on analitzarem les alternatives existents, la potència del seu ús i les ferramentes o components existents que podem utilitzar.

En aquesta tesi de màster, no ens hem centrat molt en el tipus de visualització seqüencial. Tot i que estos diagrames ens poden aportar informació molt útil sobre el nostre sistema des d'un punt de vista temporal, en aquesta tesi no hem centrat els nostres esforços en la

representació de l'ordre de certs elements. Així i tot, trobem que aquest tipus de diagrames poden ser molt útils per a futures millores en el nostre sistema. Per exemple, ens permetria representar l'evolució d'una prova d'acceptació al llarg del temps mostrant-nos l'evolució i la motivació d'aquestos canvis o en un altre cas, ens podria especificar l'ordre d'implementació natural d'una WU prenent com a base les relacions existents entre elles. De totes maneres, aquestes idees les deixarem per a futurs projectes i no entraré en més deteniment sobre aquesta categoria de visualització.

La següent categoria, es una vella coneguda nostra. La visualització jeràrquica és la base que utilitzem actualment per a mostrar l'estructura de requisits, i com hem pogut vore anteriorment també és la que utilitzen les ferramentes més conegudes per organitzar els casos d'ús com es pot vore en la Figura 22 o en la Figura 19. Sobre aquest tipus de diagrama ja hem fet un anàlisi en el capítol 3, i únicament voldria destacar que encara que ens aporta un gran nombre d'avantatges a l'hora d'organitzar els diferents elements, hem detectat una sèrie de carències sobre la nostra pròpia implementació. Així i tot, analitzarem els problemes sorgits, intentant traure el màxim partit als avantatges d'aquesta estructura jerarquitzada, minimitzant al mateix temps els problemes detectats. En el capítol 5 ens centrarem precisament en aquest tipus de representació.

Finalment ens fixem en la representació quantitativa/metafòrica. Aquesta categoria té dos vessants força diferenciades i fruit d'aquesta característica, anem a analitzar-les per separat.

Per un costat, al parlar de visualització quantitativa, es fa una clara referència a l'anàlisi o visualització de les dades des d'un punt de vista estadístic. Ja siga mitjançant gràfics de distints tipus com el de barres o el circular, o mitjançant taules amb la informació quantitativa com hem pogut vore en la Figura 24 (aquesta seria una mescla entre visualització tabular i quantitativa), aquest tipus de representació ens mostra dades de caràcter intern sobre l'estructura de requisits, i pot ser molt útil a l'hora d'analitzar certes característiques del sistema o de prendre decisions.

Per altre costat, la visualització metafòrica que esta relacionada molt estretament amb l'anterior, ens pot aportar visibilitat i variabilitat a les representacions mitjançant l'ús de codis de color, formes, grandàries i altres elements. Encara que normalment aquesta idea va en conjunt amb la visió quantitativa, hem trobat alternatives que utilitzen idees d'altres categories en conjunció amb aquesta idea.

Per exemple, com podem vore en la Figura 26, es mostra una matriu de gestió [29] on s'utilitzen característiques tabulars més pròpies de representacions com la matriu de traçabilitat, junt a elements metafòrics com és l'ús de colors o millor dit de la intensitat d'un color, per tal de representar el grau d'interrelació entre els elements oposats de la taula. De totes maneres, no hem trobat una aplicació directa d'aquest tipus de representació en la nostra metodologia.

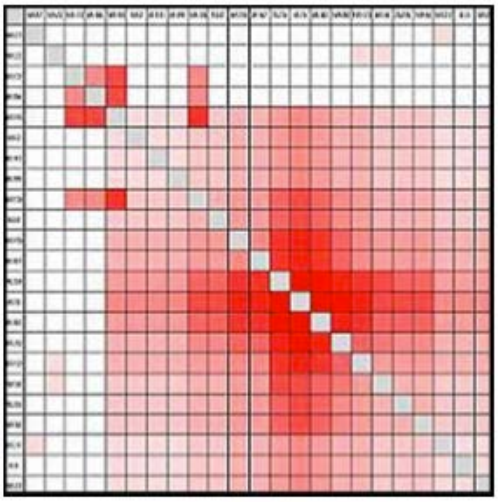


Figura 26. Matriu de gestió o conveni

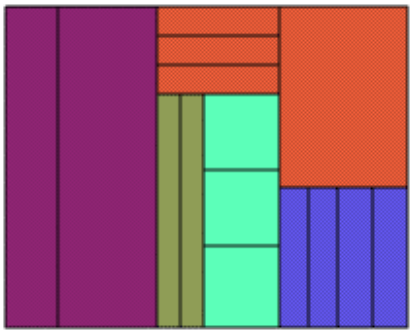


Figura 27. Representació TreeMap [32]

També m'agradaria remarcar un exemple interesant que utilitza d'una manera molt intel·ligent característiques de diverses categories: la representació TreeMap. Aquesta s'engloba dins de la categoria de visualització metafòrica, ja que utilitza els colors per diferenciar distints tipus d'elements i grandàries (pel que caldria destacar la seua vesant quantitativa) per representar la importància de cada element dins del conjunt a representar. Per altre costat, aquest tipus de representació també utilitza internament una jerarquia d'elements per construir aquesta visió fet que també ens aporta informació o part dels avantatges ja explicats. Es per això que aquest tipus de representació pot ser molt interesant per a la nostra ferramenta i ens endinsarem amb més detall en el capítol 6.

Finalment, una vegada analitzats els distints tipus de representacions i detectades aquelles que ens poden ser útils en el nostre projecte, m'agradaria destacar una idea que apareix en la pràctica totalitat d'articles analitzats: Moltes vegades caiguem en la temptació d'intentar representar en un únic diagrama una quantitat massa gran de dades. Açò dificulta la futura tasca d'extracció de coneixement i per tant lleva intuitivitat i força al propi element. Per tant, hem de ser ben conscients que volem representar en cada situació, contextualitzant la informació i sense por a múltiples visualitzacions.

Capítol 5. **Anàlisi i representacions múltiples**

A partir del que hem vist en el punt anterior, volem centrar-nos en la vella idea tan present en l'àmbit de la informàtica i en el món algorítmic que podem resumir amb l'eslògan "divideix i venç". Moltes vegades a l'hora d'encarar un problema com aquest on hem de tindre en compte diversos punts de vista o on volem tindre en consideració molts aspectes, tendim a crear sistemes excessivament complexos que perden part de la seua funcionalitat per culpa precisament del gran nombre d'opcions, dades i representacions que volem oferir en un mateix context o espai de treball. Es per açò que seguint aquesta idea, hem decidit separar les distintes millores aportades a la ferramenta de gestió de requisits en diverses vistes amb un comportament similar, lògic i coherent amb el procés de desenvolupament de la metodologia.

A l'hora de presentar les solucions creades en aquest treball, he decidit separar l'explicació de la implementació en dos parts ben diferenciades, que s'ajusten a les diferents fases d'actuació del projecte. En aquest capítol es mostraran les diverses implementacions que s'han portat a terme en el producte al llarg d'aquest any i que estan implementades en la versió de producció actual del software de suport a la metodologia TUNE-UP.

Per altre costat, en capítols posteriors, ens centrarem en l'estudi de dos alternatives més teòriques o experimentals en l'àmbit de la representació: els diagrames basats en grafs i la representació TreeMap aplicada a l'àmbit de l'estructura de requisits [21]. Inicialment presentarem de manera teòrica la base d'aquestes alternatives de visualització i mostrarem i analitzarem les diverses alternatives existents que hem trobat. Una vegada feta l'elecció, mostrarem la implementació d'un mòdul, sobre el que aplicarem un xicotet cas d'estudi i analitzarem les possibles millores o variacions que es poden portar a terme en la integració amb la ferramenta de gestió de requisits.

Tornant al tema que ens ocupa en este capítol, cal destacar que s'ha dividit la interfície gràfica principal del mòdul de gestió de requisits en tres vistes diferenciades, però integrades al mateix temps dins del mateix context de treball. Cada una d'aquestes vistes té una funcionalitat ben diferenciada i s'utilitzaran en passos diferents dins de la tasca de cada un dels rols que l'utilitzen.

5.1. Vista general

Abans de començar, cal recordar que totes les vistes que s'explicaran en aquesta i altres seccions s'engloben dins del mòdul de gestió de requisits que podem vore per exemple en la Figura 19. Per tant, no cal oblidar el comportament general que s'ha explicat en el capítol 3, ja que es la base del nostre treball sobre la que es defineixen les nostres ferramentes addicionals, sobretot tenint en compte que s'ha intentat respectar la interfície inicial del mòdul de requisits, centrant tots els esforços de millora sobre la secció dreta del gestor, és a dir, sobre l'arbre de requisits. Per tant, es manté el comportament del panell de la dreta del formulari que mostra informació sobre les PAs, IUs i altres elements relacionats amb l'especificació de requisits, en funció de quin element tenim seleccionat en el panell de l'esquerra. El panell de la dreta, tot sent sincers, si que ha sofert canvis en la seua visualització i comportament com podem vore en la Figura 28. En ella s'han afegit noves pestanyes que aporten funcionalitat a la ferramenta, com tota aquella part del software relacionada amb la refactorització i manteniment dels requisits, mesures d'accés natural a la informació millorant la navegabilitat o com podem vore en la imatge, la creació d'un barra de ferramentes que actua sobre les proves d'un node seleccionat, mostrant les eines disponibles que podem utilitzar sobre una determinada prova o proposta. Així i tot, sobre aquests temes no entrarem en detall, ja que tot i haver millorat la navegació i eficiència de la ferramenta, no s'inclouen directament en la proposta d'aquesta tesi final de màster.

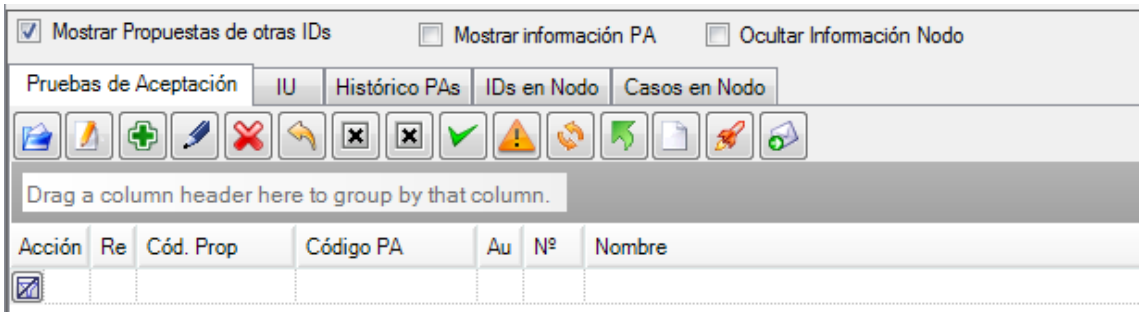


Figura 28. Barra de ferramentes del Gestor de requisits

La vista general del gestor de requisits, no és altra que l'antiga visió de l'arbre amb l'estructura de requisits ja presentada amb anterioritat. En ella, sota un node que representa el programa sobre el que s'especifiquen els requisits, es mostra un graf acíclic no dirigit mitjançant l'ús d'un Treeview que representa els contenidors de proves d'acceptació que anomenem nodes. Com s'ha comentat anteriorment, aquesta vista mostra els nodes en totes

les seues possibles instanciacions (recordem que un node pot tindre múltiples pares) i per tant la informació que ens mostra no es tan clara com ens agradaria. En aquest cas, en compte de reestructurar la visualització actual i substituir-la per una altra molt més completa i nova, s'ha decidit deixar-la com esta ja que els agents que interactuen amb la ferramenta estan molt acostumats a treballar amb aquest tipus de visualització i seria contraproduent posar molt d'esforç en una nova ferramenta que els complicarà el seu treball amb una corba d'aprenentatge relativament costosa.

Encara que com ja hem dit anteriorment aquesta vista té algunes deficiències, ha quedat demostrat en el transcurs del temps que també pot resultar una bona eina de treball, és per això que hem decidit deixar-la com a vista inicial del mòdul, però al mateix temps, hem acotat les tasques de la metodologia que es deurien realitzar en aquest context.

S'escau recordar que existeixen dos modes d'accés al gestor de requisits, i açò no ha sigut modificat en la nostra proposta. És precisament aquesta vista sobre la que reposa el treball d'analitzar l'estat actual de la ferramenta de cada producte en el mode Standard del nostre programa de suport, i encara que s'ha afegit altra funcionalitat que explicarem en seccions posteriors d'este capítol, es aquesta vista la que continua sent la peça clau.

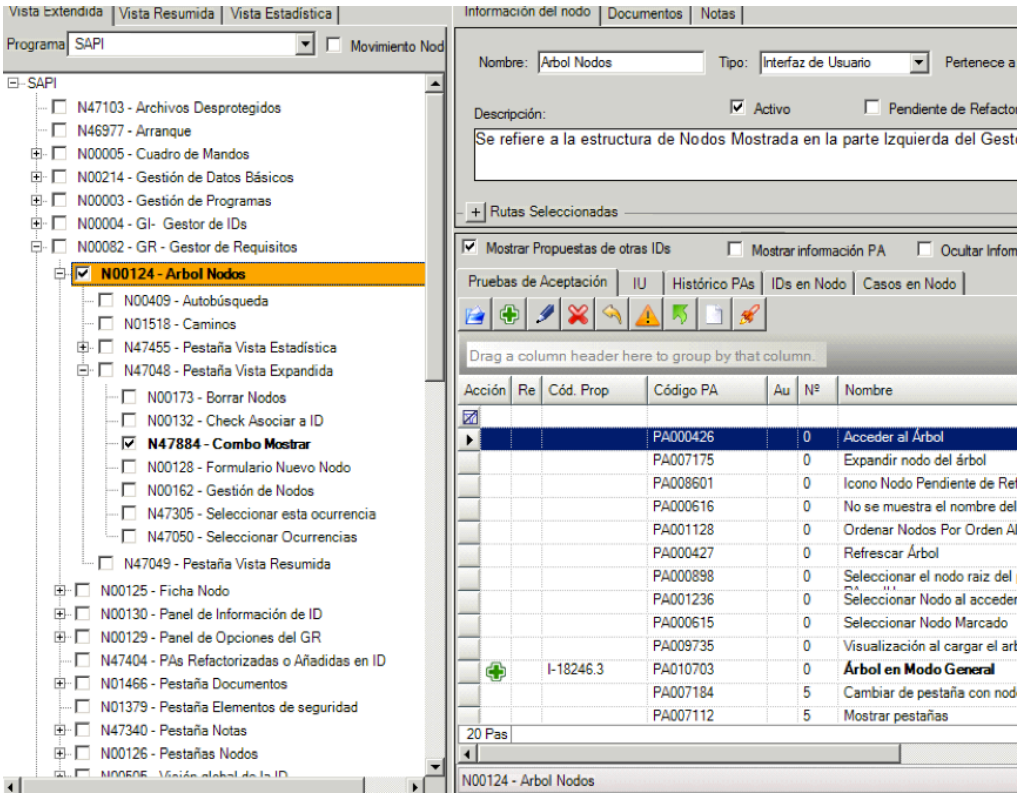


Figura 29. Vista General del gestor de requisits en el context d'una WU.

En el mode relacionat amb el context d'una WU, en canvi, si que s'han fet modificacions significatives de la metodologia de treball, remarcant d'una manera més clara les diferents fases de l'anàlisi i especificació dels requisits software. En aquesta proposta, la vista general o vista estesa, es centra principalment en la tasca inicial que han de fer els analistes de detectar els nodes que van a vore's afectats per la WU que volem implementar.

Junt a la representació de l'estructura de requisits en arbre jerarquitzat, també comptem amb un buscador de nodes i PAs integrat que ens facilitarà la tasca abans descrita i seguim tenint totes les opcions de creació, modificació i eliminació de nodes, a més s'ha activat la funcionalitat de moviment de nodes (amb el conseqüent moviment de proves associat) que únicament teníem habilitat en el mode Standard del gestor de requisits.

Tot i que no s'ha restringit la funcionalitat actual de la visualització i que encara es pot procedir a fer altres tasques d'anàlisi donant més flexibilitat a la pròpia ferramenta, els analistes de l'empresa després d'una xicoteta introducció de les noves funcionalitats, han assimilat amb naturalitat la finalitat de cada una de les vistes implementades. Gran part d'aquesta acceptació esta relacionada amb el bon rendiment i la senzillesa de les noves vistes que simplifiquen el seu treball i els permet centrar-se en el que és més important en cada fase de l'anàlisi de requisits.

Encara que com hem dit anteriorment aquesta vista roman igual que abans de començar aquesta tesis, s'han afegit un conjunt de ferramentes automàtiques o semi-automàtiques per tal de tindre una integració completa amb les noves representacions, però al mateix temps intentant que no afecten o que ho facen d'una manera lleu a la dinàmica de treball dels agents de la ferramenta. Aquestes modificacions s'explicaran amb més detall en la secció següent, ja que l'explicació perdria sentit si abans no es comentaren altres elements relacionats amb la vista utilitzada per a l'especificació textual de les proves d'acceptació.

5.2. Vista resum

En analitzar els diversos problemes que se'ns presentaven, el primer sobre el que vàrem decidir posar la nostra atenció va ser aquell relacionat amb la gran complexitat de la representació arbòria dels requisits, centrats en la figura de nodes amb diferents pares.

Aquest tipus de representació no solament dificulta la visió general de l'arbre de nodes amb la repetició d'informació sense una visualització, sinó que afecta el rendiment d'algunes opcions del mòdul amb l'addició d'un càlcul complex com és el dels múltiples

camins o instàncies que té un node i els seus descendents. Es per això que necessitàvem alguna solució parcial, que sense una gran pèrdua d'informació respecte al context dels nodes afectats per una WU, ens permetera centrar els esforços en la definició i especificació de requisits.

Abans d'abordar el problema dels múltiples pares, es va decidir basar-se en la representació arbòria que ja coneguem i amb la que els agents estan familiaritzats, però intentant reduir els problemes detectats amb anterioritat, mantenint al mateix temps els seus avantatges i les característiques que millor ens ajuden. Per fer-ho, es va simplificar el nombre de nodes que havien de carregar-se en el treeview com es pot vore en la Figura 30, mostrant únicament aquells que es veuen directament afectats per la WU i els seus antecessors per tal de donar una perspectiva més àmplia sobre aquests nodes.

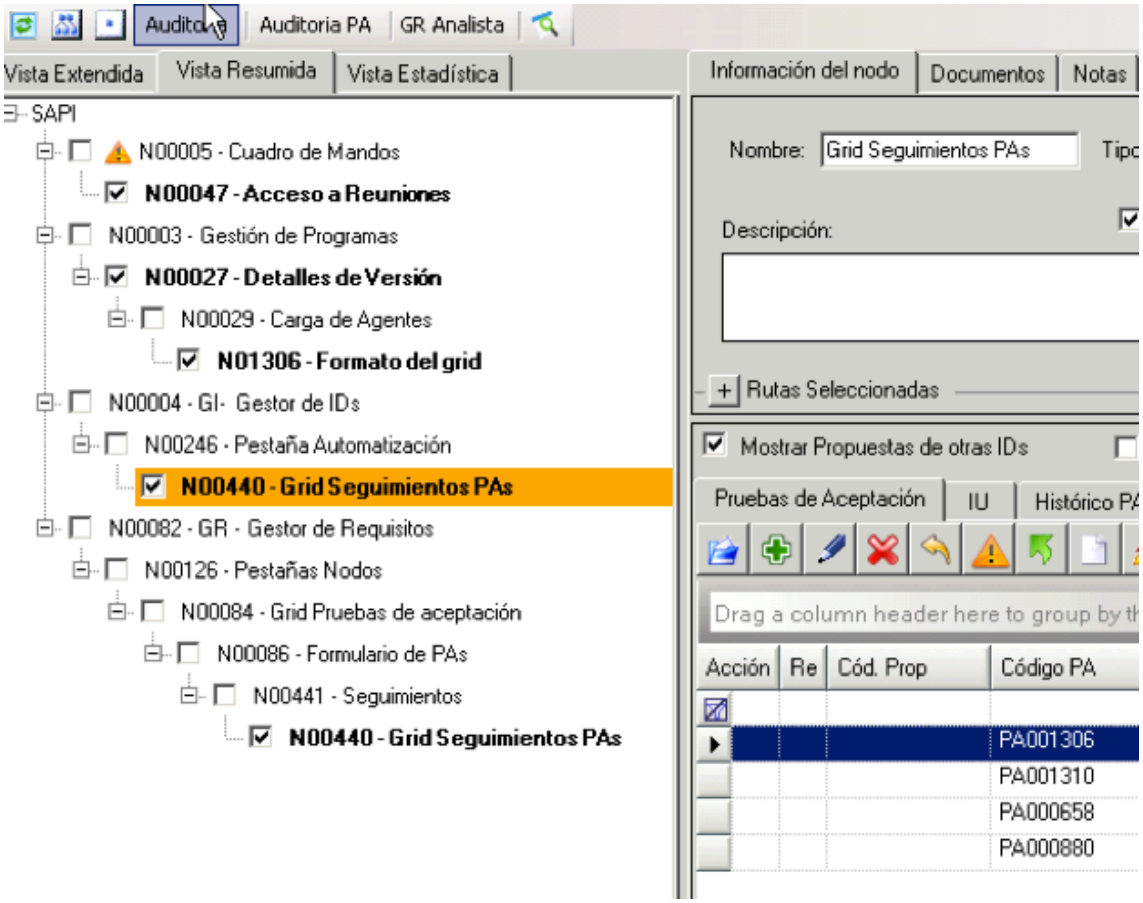


Figura 30. Vista Resumida de una WU del Gestor de Requisits

Cal destacat que aquest tipus de millora, solament té sentit després d'haver fet una separació de l'espai de treball com la feta, amb l'elecció de dividir la tasca d'anàlisi en una primera fase de detecció de nodes afectats que es faria sobre la vista general i una segona

centrada en l'especificació de requisits, que es portaria a terme en aquesta nova vista simplificada.

Aquesta primera millora, ja ens començava a aportar grans millores en quant a eficiència dels agents al treballar sobre la ferramenta i marcava una bona base sobre la que treballar per introduir noves modificacions. Però com cal recordar, en l'estructura de requisits dels programes més importants de la companyia, existeixen nodes amb varies desenes de instancies diferents que seguien provocant un nombre massa elevat de camins a considerar amb tots els problemes que açò comporta. Es convenient remarcar que tots els nodes que es seleccionen com a afectats, no solament seran utilitzats pels analistes per definir les proves d'acceptació, si no que també són utilitzats per programadors i testers per tal d'acomplir les seues respectives tasques. Amb açò últim en ment, pot resultar clar el problema d'eficiència que suposa que un programador o un tester haja de comprovar el comportament d'allò que esta implementant en les 63 instàncies del mateix node (cas més elevat, però al mateix temps node molt utilitzat). Es per això, que es va començar a plantejar l'ús diferenciat de les diferents instàncies d'un node en el context d'una WU.

Aquesta interpretació o millor dit aquesta elecció, ens suposa un problema inicial de caràcter teòric. Tenint en compte que un node es un contenidor de requisits i que aquests han d'acomplir-se en tots els llocs on hagen sigut definits, el fet de fer una selecció entre ells, ens podria portar a inconsistències en l'especificació, si resultara que per no comprovar el comportament del programa en un determinat context aquest no es complís o ho fes de manera parcial.

De totes maneres, tenint açò en compte, en la pràctica hem pogut observar que en la major part dels casos, els canvis que es produeixen sobre una determinada instància afectarà d'igual manera, si no a totes les altres instàncies, a un conjunt d'elles. Per tant, si donem l'opció de definir un conjunt d'instàncies claus, podem estar reduint de manera considerable el treball que els agents han de portar a terme, sense afectar a la qualitat dels requisits. A més, en ser una selecció opcional de les instàncies, en cas de que un agent es trobe enfront una part crítica del programa, aquest podrà seleccionar totes les instàncies d'un node, és a dir, com si no s'hagués portat a terme aquesta optimització.

Per tal d'aconseguir aquesta millora, hem hagut de modificar lleument el workflow natural de treball dels analistes seguint la metodologia TUNE-UP i s'ha hagut de crear un conjunt d'infraestructura que permetera fer-ho de manera simple i no excessivament intrusiva. A

més, una vegada implementat i amb els bons resultats que aquesta optimització ha aportat, s'ha decidit utilitzar aquesta vista en altres contextos de la ferramenta per tal de tindre una millor integració global amb el programa i aportar informació relacionada a diferents tipus d'agents. En definitiva, hem intentat fer que el coneixement arribe a tots aquells que els puga ser útil.

Per començar l'explicació sobre el treball realitzat, cal centrar-se primer en la decisió inicial de l'emmagatzemament i la representació de les instàncies utilitzades en cada una de les WU. Una instància d'un node ve definida inicialment per la ID i el nom del node al que representa, però aquesta informació no es suficient si el que volem es diferenciar cada una d'aquestes instàncies. Com no existeix el concepte d'identificador d'instància, ja que aquest element no s'ha tingut en compte d'una manera explícita en la metodologia que seguim, hem de trobar alguna manera d'identificar-les i que ens aporten informació suficient per a la seua futura manipulació i ús.

Seguint aquesta idea, vàrem crear una taula en la nostra base de dades relacional, per tal d'emmagatzemar de manera diferenciada els elements i tindre així un identificador per a cada instància. Aquesta taula inclou a més l'identificador del node al que representen, el que ens permet fer l'associació entre els dos elements (node – instància), el identificador de la WU on es veu afectada la instància determinada i el que és més important, una cadena de caràcters representant el camí de la instància des del node inicial de l'arbre fins a ell mateix. Aquesta decisió es va prendre degut a que, la major part dels càlculs pesats que es portaven a terme en relació a aquest problema, sorgien del complex càlcul dels camins d'un determinat node, i amb aquesta informació, aquest càlcul podia ser simplificat de manera poc costosa per al nostre projecte. La representació utilitzada per a definir esta cadena de caràcters es defineix com un conjunt de números ordenats jeràrquicament (seguint l'ordre de l'arbre) que representen els identificadors dels nodes, separats per comes.

Una vegada definida l'estructura de dades que anàvem a utilitzar per emmagatzemar la informació, era el torn de decidir com s'anava a interactuar amb la ferramenta. Per tal d'aconseguir un ampli consens, es va definir una proposta inicial, i es van fer diverses reunions amb els analistes, que varen acabar amb una millora més madura que satisfeia a la majoria d'actors implicats.

Com ja es mostrava anteriorment en la Figura 30, la representació que els anàvem a definir era un arbre molt simplificat que els mostrava una fotografia de les instàncies dels nodes que es veien afectades per una determinada WU. Aquesta informació, es calculava mitjançant la informació emmagatzemada en la nova taula d'instàncies o camins que permetia que aquesta vista es carregara quasi immediatament, independentment de la complexitat o quantitat de nodes que s'havien vist afectats. Aquesta millora en el rendiment va ser gratament rebuda pels analistes, que amb la confirmació dels bons resultats, es van encoratjar a fer noves propostes o necessitats.

El primer que calia solucionar era com s'anaven a definir les instàncies dels nodes que afectaven a una determinada WU, es per això que es va decidir modificar lleugerament el significat de la primera etapa de l'anàlisi de requisits, per que representara les instàncies afectades, i no únicament els nodes afectats com feiem fins eixe moment.

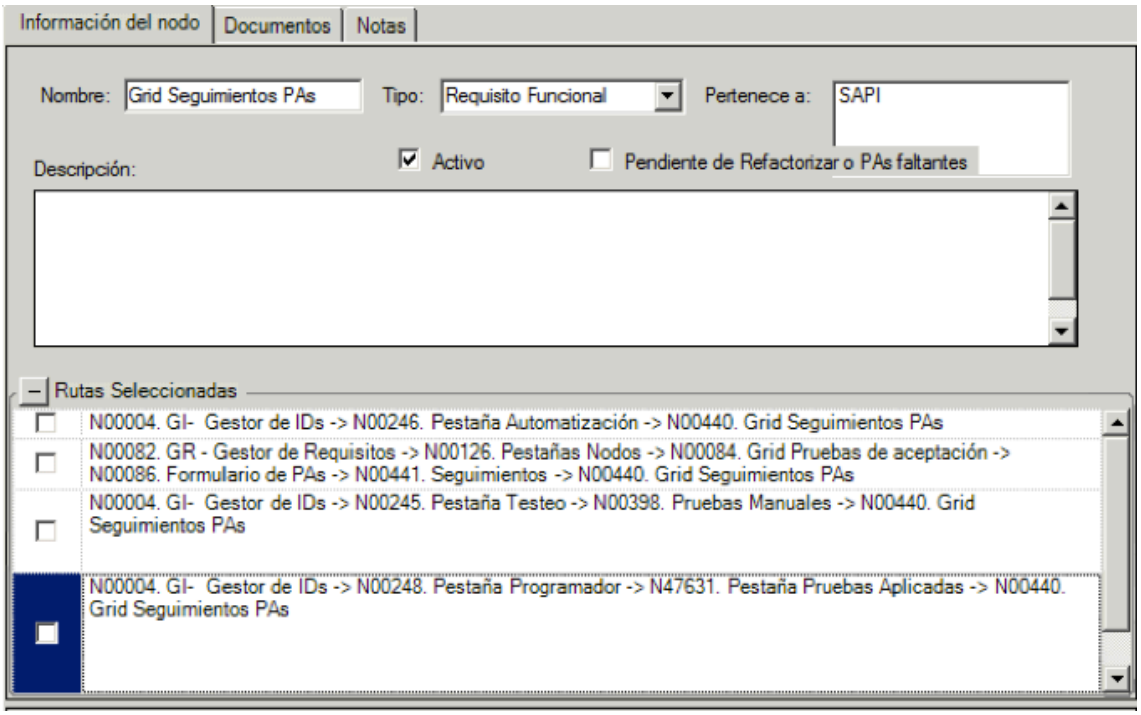


Figura 31. Informació de les instàncies d'un node que afecten una WU

Fruit del consens, es va decidir que en el moment que un agent marques el check d'un node per tal de definir-lo com afectat en la WU que esta analitzant, la instància concreta sobre la que s'havia fet check també es marques com a afectada. Aquesta mesura ens ajuda a simplificar el treball dels analistes, perquè encara que hi ha un bon conjunt de nodes amb múltiples camins, la majoria segueixen tenint únicament una instància. Amb aquesta decisió doncs, no és necessari modificar el mode de treball dels agents per a la major part

de les situacions i els redueix considerablement la corba d'aprenentatge d'aquestes noves modificacions.

Per tal de poder distingir amb facilitat sobre quins nodes s'ha de fer el treball extra de selecció d'instàncies, es va crear una nova vista en el panell d'informació del node com es pot vore en la Figura 31, que mostra de manera textual els camins que representen a cada una de les instàncies de node i quines d'elles estan actualment afectades per la WU. Amb aquesta informació a la vista, es varen definir les opcions de selecció d'instàncies sobre el menú contextual dels nodes ja marcats com a afectats en el nostre context. Per tant, si un node ja ha sigut seleccionat, mostrarà les opcions "Seleccionar aquesta instància del node" (si no ho estava ja) o en ens donarà l'opció general de "Selecció d'instàncies".

En elegir aquesta última opció, se'ns obre un xicotet formulari de selecció de requisits que com es pot vore en la Figura 32, consisteix en un grid amb els distints camins que representen les instàncies d'un determinat node i on per cada fila del grid tenim un check que ens permet marcar o desmarcar un determinat camí. A banda d'açò s'han afegit dos botons d'ajuda que ens permeten marcar o desmarcar totes les instàncies de manera automàtica., encara que si decidim desmarcar-les totes, no podrem eixir d'aquest formulari fins que almenys no hi haja una seleccionada. Per tal d'eliminar la relació entre una WU i totes les instància d'un node, cal desmarcar directament el node des de l'arbre, que després d'un missatge amb l'alerta pertinent, eliminarà automàticament totes les relacions existents.

Amb les primeres proves que es varen fer sobre aquestes millores, es va detectar una problemàtica, que utilitzaríem a posteriori per solucionar un dels problemes o carències detectades i ja comentades en el capítol 3 d'aquesta tesis.

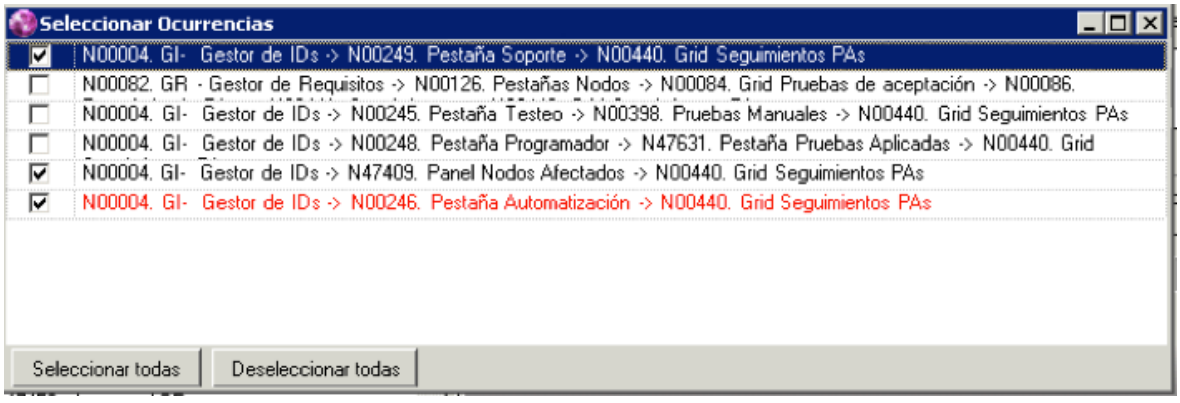


Figura 32. Formulari de selecció d'ocurrències

Una de les qüestions que varem detectar després d'aplicar les millores anteriors, va ser que la nostra estructura de dades de representació no estava sincronitzada amb l'evolució de l'estructura dels requisits. No solament no havíem creat codi per tal de modificar els camins degut a un moviment o altre tipus de reestructuració de l'arbre, si no que aquests no eren gens fàcils de portar a terme. Cal destacar que quan un node es mou, caldria analitzar totes aquelles instàncies del node que afecten a totes les WUs definides en la nostra ferramenta i recalcular-les per obtindré els camins actualitzats, i encara així, això solament resoldria una xicoteta part, ja que caldria actualitzar també totes aquelles instàncies on aquest node haja aparegut com a part del seu camí identificador.

Com es pot imaginar, aquesta es una tasca massa complexa que en cas d'executar-se, castigaria en excés la aparentment simple tasca de moure un node dins l'estructura de l'arbre. Per tant i per tal d'evitar una pèrdua considerable de rendiment en la reestructuració dels nostres requisits, calia trobar alguna altra eixida al problema. I va ser analitzant les problemàtiques existents en aquell moment en l'arbre, que ens vàrem donar compte que podíem resoldre dos problemes amb un únic i senzill canvi en el punt de vista de la resolució.

Tornant a les carències o limitacions que havíem definit en el capítol 3, cal recordar la impossibilitat que teníem amb la ferramenta actual de poder vore l'estructura de l'arbre en el moment de definició d'una determinada WU. Açò era important, ja que degut als constants canvis que poden sorgir en l'estructura dels requisits d'un producte, es poden crear certes inconsistències entre la descripció d'una prova degut a la seua descontextualització respecte al moment de la seua definició. Es per això que tenint aquest problema en ment, vérem canviar lleugerament el significat de la vista resumida del gestor de requisits per tal que ara representara *una fotografia de les instàncies dels nodes que es veuen afectades en una determinada WU en l'instant que van ser analitzades.*

Aquesta lleugera modificació en la seua definició, ens soluciona el problema de l'actualització de l'estructura, ja que en aquest cas ja no serà necessari portar-la a terme. De tota manera, es possible que en determinades situacions siga interesant actualitzar la informació respecte a la representació actual de l'estructura, o simplement necessitem algun mecanisme per ser conscients de l'obsolència d'alguns dels camins mostrats en aquesta nova vista.

Per tal de donar suport a aquesta última idea, es va decidir modificar el comportament dels formularis on es mostren totes les instàncies d'un determinat node (Figura 31 i Figura 32). En carregar aquesta informació, es mostren totes les instàncies actuals del node amb un tipus de lletra normal, mentre que si alguna de les instàncies que tenim emmagatzemades en la taula de camins no es troba entre aquestes, la marquem en roig per mostrar que és un camí antic i actualment obsolet. Aquest tipus d'informació, permet els analistes decidir si volen deixar la vista com una fotografia de l'estat anterior o si per contra volen actualitzar-ho a un estat més actual, tot comprovant les definicions de les proves. A més, també permet a testers i programadors situar millor el context de les proves en cas d'haver-hi alguna modificació.

Finalment, una vegada ja teníem tota la infraestructura necessària per a l'ús d'aquesta nova ferramenta, calia actualitzar la informació emmagatzemada actualment en la base de dades. Per a això, es va crear un xicotet programa o script independent que s'encarregara de recórrer la taula que contenia els nodes afectats per cada WU, i per cada un d'ells, calcular tots els camins que representen les seues instàncies emmagatzemant-los en la nova taula d'instàncies. Cal destacat que es va utilitzar el criteri inicial de seleccionat tots els camins possibles, ja que no existia cap criteri inicial per a la seua reducció, i gràcies a les noves ferramentes, els analistes podien aplicar els seus propis criteris en un futur.

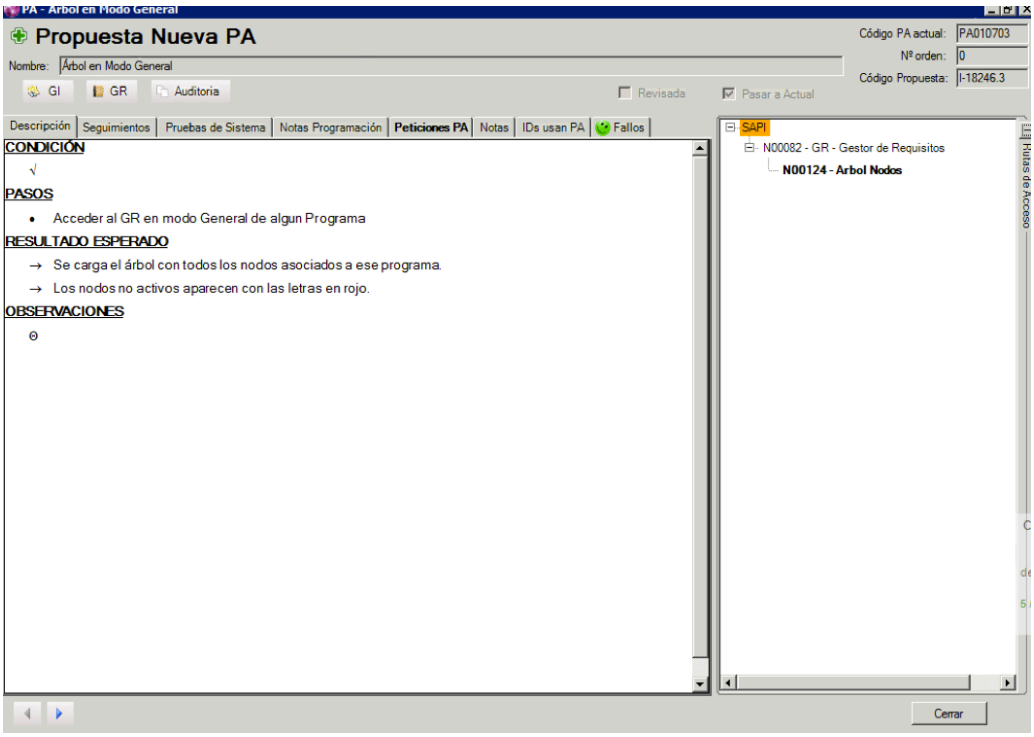


Figura 33. Integració de la vista resumida en el formulari de proves

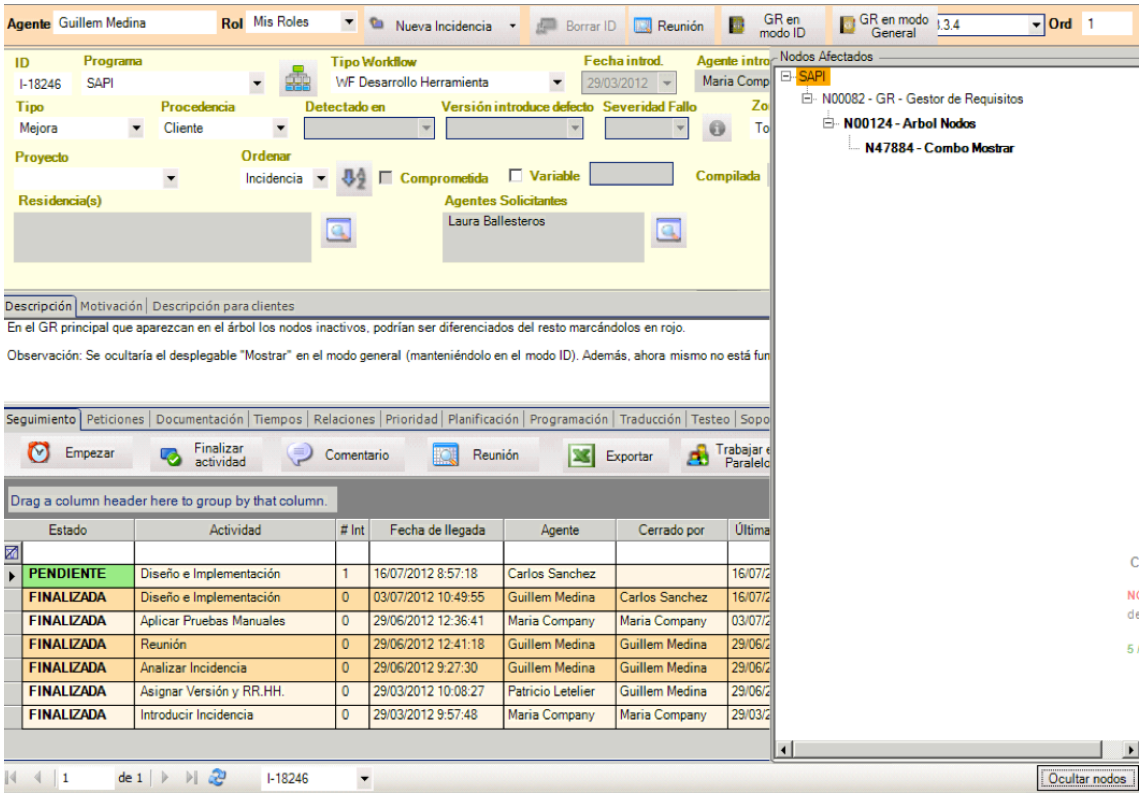


Figura 34. Integració de la vista resumida en el Gestor de WUs

Amb aquest conjunt de canvis fets, ja teníem tota la funcionalitat necessària per donar per finalitzada aquesta vista del gestor de requisits, però amb la bona acollida i resultats aconseguits, vàrem decidir integrar-la millor amb la resta de la ferramenta.

Per començar i per tal que les transicions foren més naturals entre les distintes vistes del gestor de requisits i com les dos representacions mantenen grans similituds amb la visualització arbòria, es va decidir mantindre el node seleccionat al canviar de vista. En el cas d'anar de la vista general a la resumida solament ho mantindríem si el node estiguera representat en ella, seleccionant el node programa en cas contrari. A més, en carregar el arbre, es manté l'última vista seleccionada sobre la que es va treballar en la WU corresponent, així si la fase de selecció de nodes ja s'ha completat, la càrrega del gestor serà molt més ràpida. Una altra qüestió que també s'ha cuidat és que les vistes únicament es carreguen una vegada al navegar entre elles, però tenint en compte al mateix temps que, en cas d'haver una modificació en la vista general relacionat amb les instàncies que afecten a una WU, es recarregue la vista resumida, encara que com ja hem dit anteriorment aquest càlcul es pràcticament automàtic.

Altres qüestions que s'han considerat en aquestes últimes millores ha sigut la integració amb la resta de formularis de la ferramenta TUNE-UP, es pera això que com podem vore en la Figura 33 i en la Figura 34, aquesta informació es mostrarà en altres parts del programa. En la primera podem vore com se'ns mostra en un panell ocultable del formulari de proves d'acceptació i que permet a analistes, testers i programadors a contextualitzar d'una manera simple i efectiva la descripció escrita de la prova. En la segona, utilitzant el mateix tipus de panell, donem accés a esta informació des del gestor de WUs, per vore així amb un simple colp d'ull els nodes que li afecten, sense tindre que obrir el gestor de requisits.

Aquesta nova vista ha sigut sens dubte una gran millora dins de la ferramenta. Amb el temps que porta en producció, hem pogut observat que la productivitat dels diferents agents ha millora i sobretot la satisfacció d'aquests envers els nous canvis fan que estiguen més predisposats a aportar noves idees per a la millora de la nostra eina.

5.3. Vista d'anàlisi de dades

Una vegada ja teníem solucionats una gran part del problemes que ens havíem trobat en la representació arbòria de l'estructura de requisits, va arribar al nostre coneixement una altre carència existent en el programa, que aquesta vegada naixia de la necessitat de l'equip de testers. Degut a la necessitat d'analitzar i prendre certes decisions respecte a la direcció i la distribució del seu treball, els tester necessitaven tindre algun tipus de visió general del programa que estaven desenvolupant actualment. Aquesta informació, centrada en el nombre de proves de sistema i proves d'acceptació que havien i no havien estat automatitzades, devia ajudar-los a analitzar l'estat actual del testeig, prenent així decisions més conscients i informades.

Fins a aquell moment, era possible visualitzar a nivell de node, un gran grapat d'informació relativa a les proves d'acceptació contingudes en ell, però no existia cap representació general que ens permetera analitzar l'estat global, ni tan sols una vista on poder vore l'estat de les proves de sistema en el context de l'estructura de requisits. Seguint les dos idees bàsiques aplicades al punt anterior, és a dir, basar-se en representacions ja conegudes pels agents i dividir el problema en diverses vistes per resoldre els problemes plantejats, vàrem presentar una proposta que podem observar en la Figura 35.

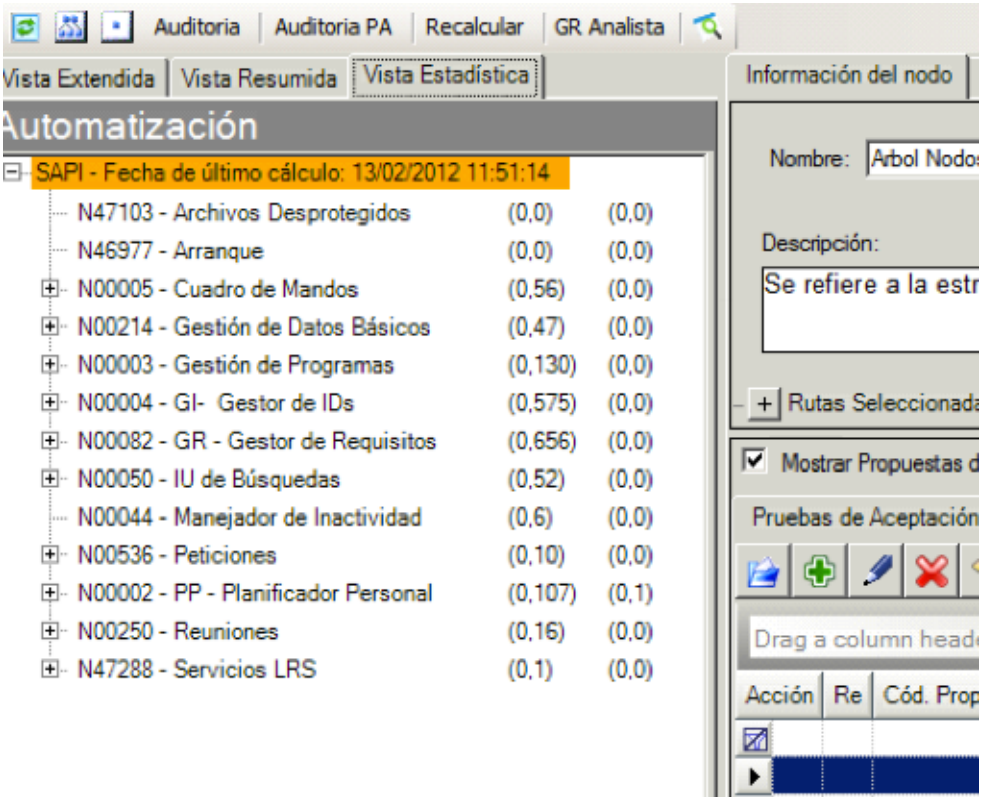


Figura 35. Vista estadística sobre l'estat del testeig d'un programa

En aquesta, ens basem en la mateixa estructura de l'arbre de requisits que hem estat utilitzant durant tot aquest temps i que es àmpliament coneguda pels agents de l'empresa, però centrant la seua informació en les necessitats actuals de l'equip de testeig. Cal destacar que en utilitzar aquesta visualització mantenim certes propietats molt útils com són la **navegabilitat**, la **contextualització dels elements en el esquema global**, a més de tota la **informació rellevant dels nodes** del programa analitzat.

A més, per aconseguir aportar la informació demanada, es varen afegir un conjunt de columnes associades a cada node on mostrar les estadístiques i dades que els pogueren ser útils. Aquesta informació va ser mostrada en parells de nombre relacionats, amb tooltips d'informació associats a mode de llegenda.

La informació mostrada, com ja s'ha comentat adés, naix de les necessitats dels testers i es per això que es va arribar a un consens amb ells per tal de que la nova vista fora el més útil possible. En un principi es va decidir que volien tindre informació sobre el nombre de PAs de cada node distingint aquelles que havien sigut automatitzades i aquelles que encara no ho havien sigut. Seguint aquesta idea, es va fer el mateix amb les proves de sistema utilitzades pels testers i també es va decidir mostrar el número de KOs i OKs sobre l'últim nivell de testeig aplicat de les proves d'un node. A més d'açò, es mostren el número de WUs de tipus Fallada que han afectat un node relacionades amb el nombre d'altres tipus de WUs. Finalment i en relació a una ferramenta de gestió de fallades implementat per l'equip de testeig i que actualment hem integrat en la nostra eina, mostrem el número de fallades pendents de cada node junt al número de fallades ja solucionades.

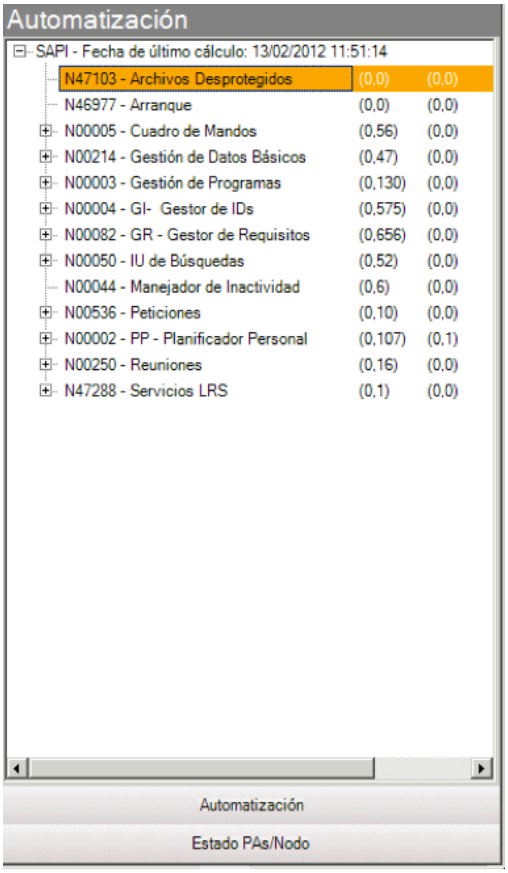


Figura 36. Panell d'informació estadística

Tota aquesta quantitat d'informació, ajuda als testers a mostrar el seu treball amb més perspectiva al product manager, així com a definir quin serà el seu treball futur o sobre quines parts del programa volen centrar-se a continuació. De totes maneres, el que vàrem trobar en començar a aplicar aquesta idea, van ser dos problemes més que requerien solució.

Per un costat, ens trobem el problema de la contextualització. Cal recordar que estem mostrant tota aquesta informació en el context d'un node i per altre costat, aquest node forma part d'una estructura jeràrquica d'altres contenidors. Si mostrem la informació del node aïllada de la resta, açò ens dona una idea clara de la situació d'un node concret, però es possible que no siguem conscients d'ell ja que aquest node, el podem trobar ocult com a successor d'uns altres que no ens hagen cridat l'atenció. Es per això que després d'alguna que altra discussió sobre el tema, es va decidir modificar el significat d'aquesta informació

per tal de que no represente únicament la informació d'un node aïllat, si no que mostre el número total d'elements si sumarem tots els seus descendents.

Aquesta idea a la vegada ens portava dos problemes més, però ens donava un context sobre la informació increment els avantatges. El primer es tractava de la pèrdua de la informació aïllada, cosa que es podia solucionar amb relativa facilitat afegint eixa informació com a element addicional a les dades ja mostrades. Per altre costat el segon problema, l'únic que feia era agreujar considerablement el segon problema que encara no havíem comentat, és a dir la falta d'eficiència i l'alt cost temporal dels càlculs.

Amb aquesta última modificació, els càlculs que s'havien de fer sobre els nodes es van tornar força costosos, i açò provocava que es fera impossible mantindre aquesta vista actualitzada o que es fera el càlcul cada vegada que s'accedira a ella. Una possibilitat podia ser intentar emmagatzemar aquesta informació en una taula de la base de dades, però açò haguera implicat grans modificacions en el codi, ja que necessitaríem actualitzar la informació cada vegada que hi haguera un canvi a nivell de node, PA o PS i com hem dit els càlculs que es fan sobre certs productes gestionats en el nostre programa poden durar durant alguns minuts. Així que finalment vàrem arribar a una solució a mig camí entre les dos opcions que va satisfer a totes les parts implicades.

Per un costat, es va crear un algoritme que una vegada trobades totes les fulles de l'arbre, calculava les dades de cada un d'ells i propagava la informació cap als nodes superiors. Aquesta construcció bottom-up, encara que va millorar el rendiment del càlcul respecte a l'anterior versió iterativa seguia sense donar uns resultats amb un temps suficientment curt com per tornar a replantejar-se les coses, pel que es varen prendre dos noves modificacions. Per un costat, es va definir el càlcul com un fil independent d'execució que permetera als agents seguir treballant en la resta de l'aplicació al mateix temps. A més, es va afegir un sistema de notificació que ens avisara amb un pop-up de que el càlcul ja havia finalitzat introduint un link al formulari amb la informació actualitzada.

Finalment, i després de parlar amb els testers per validar les noves propostes, ens adonarem que tot i que la precisió de la informació era important, la veritat és que el que realment necessitaven eren dades del mateix ordre de magnitud que els donara una idea general. Per tant, seguint aquesta idea, es va optar per canviar el funcionament general d'aquesta vista, es va tornar a la idea d'emmagatzemar la informació de l'estat de testeig en una taula. Aquest canvi però, seguia les propostes anteriors, pel que emmagatzemava l'últim càlcul que s'haguera fet amb el fil d'execució independent. Per tant, es va decidir que inicialment es mostraren les dades de la taula ja calculades per tal d'obrir la vista quasi immediatament, mostrant en quin moment (data) es va fer l'últim càlcul i permetent així recalcular la informació en cas de trobar-nos amb informació obsoleta o molt antiga.

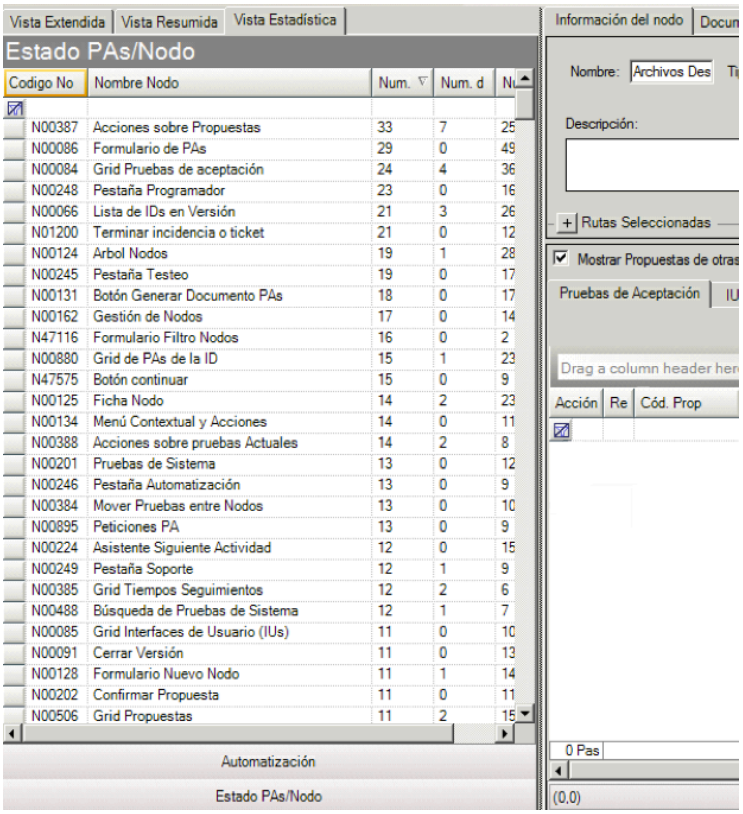


Figura 37. Vista de l'estat de PAs i Nodes

Aquesta mesura va donar bons resultats i es va decidir seguir la mateixa idea per aportar noves millores respecte a altres temes. Es per això que com es pot vore en la Figura 36 i la Figura 37 es van afegir altres panells d'informació sobre la mateixa vista estadística o d'anàlisi. Entre elles cal destacar la vista que anomenarem d'estat de PAs i Nodes.

Tornant a les possibles millores que encara no s'han vist cobertes amb aquestes modificacions i que vàrem definir en el capítol 3, i tenint en ment els mecanismes utilitzats per mostrar l'estat del testeig d'un producte software, hem definit una vista que ens permet analitzar dades relacionades amb la informació emmagatzemada per la ferramenta de gestió de requisits.

En aquest cas, hem abandonat l´ús del control treeview per mostrar totes aquestes dades, ja que pensem que en aquest cas l'ordre per al seu estudi no esta relacionat amb la pròpia

jerarquia dels elements, si no més bé amb les dades de cada un dels nodes. Es per això que hem definit un grid, on cada fila correspon a un node de l'arbre amb opció a filtrar i ordenar segons els criteris que nosaltres trobem més adients.

En aquesta nova vista que podem vore en la la Figura 37, es mostra un gran nombre d'informació distinta. El número de PAs i el número de propostes sobre WUs de cada node ens dona informació sobre l'estat de l'organització de les PAs, el que ens pot ajudar a detectar nodes sobrecarregats o que estiguen definits de manera incorrecta i ajudar així a la seua futura refactorització. A més, també podem mostrar per exemple el nombre de instàncies que té un node, cosa que ens pot ajudar a analitzar les situacions extremes i vore si realment s'està fent un ús correcte d'aquesta ferramenta.

Aquest és solament un subconjunt de la informació que podríem mostrar en aquest tipus de vista analítica. Hem començat amb la implementació d'aquests casos com a conseqüència de situacions concrets a resoldre, però en un futur es poden estudiar altres situacions on aquest tipus de tècniques puguen resultar útils.

Capítol 6. **Alternatives de visualització: diagrames**

En el capítol anterior, hem millorat i completat la visualització existent del gestor de requisits, basant-nos en la combinació de tècniques estadístiques i de simplificació sobre l'estructura actual d'arbre. En canvi en aquest, anem a abordar dos noves tècniques que creguem que ens poden aportar una gran visibilitat i que amb un poc de treball i maduresa podrien integrar-se en la nostra ferramenta per aporta un grau més d'informació que podríem utilitzar en un futur anàlisi semiautomàtic de l'estructura de requisits.

En concret, en aquest capítol anem a analitzar la figura gràfica del TreeMap. Encara així, no entrarem en tant de detall, ja que existeix un projecte paral·lel al nostre dins del propi grup d'investigació que s'ha encarregat de presentar una altre treball centrat completament en aquest tipus de diagrama. Per altre costat, parlarem de la representació gràfica basada en grafs que serà la peça central dels pròxims 3 capítols, analitzarem les diferents alternatives existents en el mercat i elegirem una d'elles per crear un mòdul de visualització amb dades d'un producte real.

6.1. Diagrames Treemap

Aquesta primera tècnica [35][36], va ser creada a principis dels anys 90 per Ben Shneiderman per tal de representar l'estructura interna (a nivell de fitxer) d'un disc dur i poder detectar així quins elements ocupaven més espai i podien ser eliminats en cas de faltar-ne. Actualment, aquesta tècnica s'està utilitzant en nombrosos camps, i s'utilitza principalment per representar ordre i importància sobre objectes que estan relacionats mitjançant relacions de contenció.[21]

Aquesta tècnica de visualització, com ja hem comentat en el capítol 4, es pot classificar dins de varies de les categories especificades.[22] Per un costat, es tracta d'una tècnica de visualització massiva de dades basada en jerarquies on es representen els elements en funció de les relacions pare-fill existents entre ells. Per altre costat, aquest diagrama també es pot categoritzar com a representació quantitativa/metafòrica, ja que amb l'ús de caixes per nivells (unes dins d'altres) i de característiques com el tamany o el color d'aquestes es permet remarcar la seua importància i inclús la seua tipologia.

Per tal de generar aquest tipus de diagrama, necessitem crear una estructura de dades on emmagatzemar tota la informació que el control TreeMap utilitzarà. En el cas del control

ofert per la companyia Infragistics que utilitzem en el desenvolupament de la nostra ferramenta, necessitem definir un DataSet amb tota la informació necessària. En esta estructura de dades cal que creem un DataRow per cada element que volem que aparega en el diagrama, on definirem els identificadors dels seus elements antecessors, així com elements quantitatius respecte a les propietats que volem analitzar.

Amb aquest tipus d'informació, el control és capaç de crear l'estructura en arbre per representar la jerarquia de caixes. A més, amb la resta de informació i definint les propietats que volem analitzar es poden caracteritzat les caixes en distints colors per diferenciar categories o es pot definir el tamany de cada caixa en funció d'una escala numèrica respecte a alguna propietat en concret.

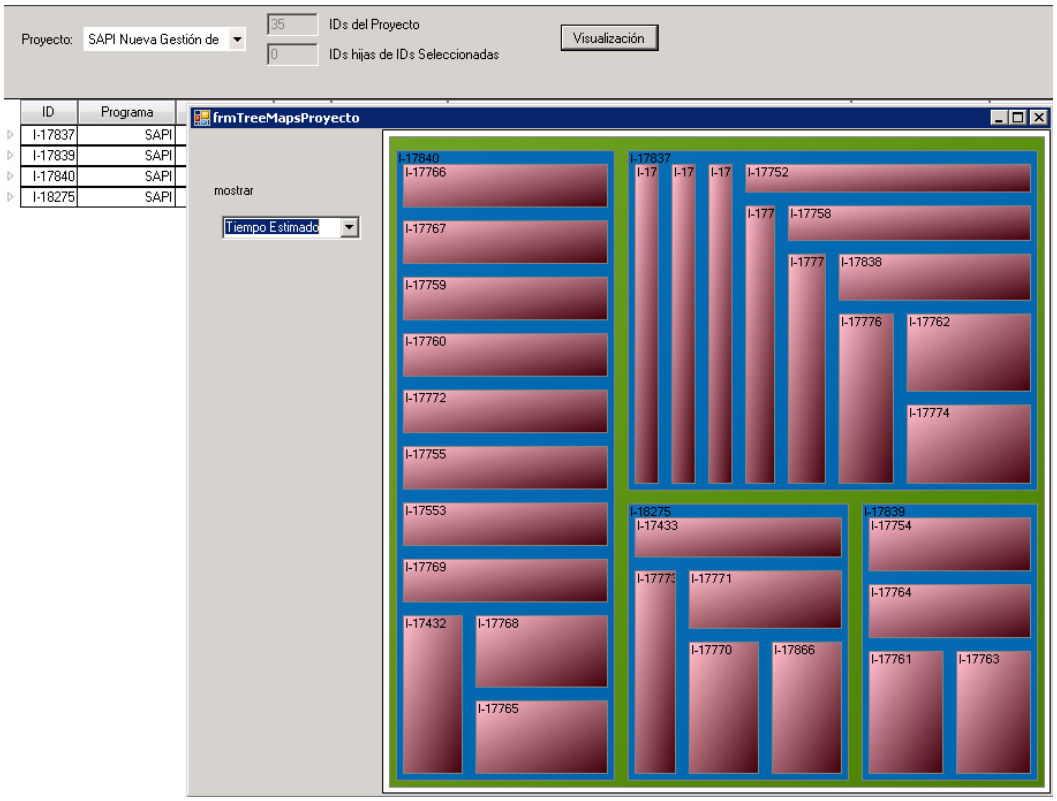


Figura 38. Representación TreeMap aplicado a un conjunto de WUs de un programa [21]

Aquest tipus de representació ens podria resultar molt útil per tal de mostrar l'estructura de requisits des de diversos punts de vista. Per exemple podríem representar l'estat del testeig d'una aplicació, mostrant les PAs i/o nodes amb distint color o tonalitat per definir el percentatge de proves automatitzades o ja aplicades satisfactòriament sobre un producte. També podria resultar útil per detectar les zones d'un programa més susceptibles a canvis

en funció del nombre de relacions de traçabilitat existents i representades pel tamany de les caixes.

Encara que puga resultar una tècnica molt útil per al nostre projecte, hem decidit centrar-nos en la representació basada en grafs. Pensem que aquesta és la més adient respecte al nostre sistema i sobretot, perquè com hem dit amb anterioritat, existeix un projecte paral·lel al nostre, que es centra en la representació de TreeMaps. En la Figura 38, podem vore un petit prototipus d'aquest projecte on s'aplica aquesta tècnica a l'àmbit de la gestió de projectes dins de la metodologia TUNE-UP. Esperem així, que els coneixements i experiències que es puguen aconseguir en aquest altre projecte també puguen ser aplicades en un futur al nostre àmbit concret de gestió de requisits i visualització de la seua estructura.

6.2. Diagrames basats en grafs

6.2.1. Idees generals i requisits

La finalitat que intentem aconseguir en este capítol no es altra que trobar una tècnica de visualització que ens permeta representar de la manera més natural, completa i intuïtiva, l'estructura global de l'estructura de requisits. Des d'aquest punt de vista, pensem que els controls i ferramentes de visualització basats en grafs poden tindre un paper destacat precisament perquè són la representació natural i teòrica del nostre sistema.

Cal destacar, que la nostra estructura de requisits esta basada en dos tipus d'entitats, per un costat les PAs que representen escenaris concrets sobre el funcionament de la ferramenta, i per altre els nodes, que són contenidors de PAs i que representen d'una manera abstracta els requisits software. Junt a estos elements, existeixen una sèrie de relacions entre ells que denoten els seus vincles d'herència i les relacions de traçabilitat utilitzades en l'anàlisi d'impacte del requisits software. Tots estos elements poden ser fàcilment encarnats mitjançant els objectes principals dels grafs: els nodes i els arcs.

Un altre aspecte a tindre en compte en relació a aquest tipus de representació és precisament la gran base teòrica que té en els seus fonaments. Cal recordar que el graf és una estructura matemàtica utilitzada per primera vegada en 1736 per Leonhard Euler i a partir d'aquest moment ha sigut àmpliament estudiada i utilitzada en diversos camps científics i tecnològics. Aquesta característica, no solament permet visualitzar les dades

amb l'ajuda d'algoritmes molt eficients, si no que a més, en un futur ens permetrà tindre una sèrie de ferramentes amb les que analitzar el nostre sistema. Algoritmes com els relacionats amb camins més curts, capacitat de les relacions, cerca per profunditat o inclús cobertura del graf ens poden ser molt útils en tasques posteriors d'anàlisi.

Si comparem aquest tipus de visualització amb el TreeMap vist anteriorment, podem vore que aquest ens dona més flexibilitat, ja que amb el primer control solament podem representar relacions de tipus contenidor-contingut i en aquesta tècnica podem mostrar tantes relacions i de tants tipus com nosaltres decidim. Per altre costat, ens aporta un nivell de complexitat que amb grafs d'un cert tamany pot resultar poc intuïtiu i visual com podem vore en la Figura 39.

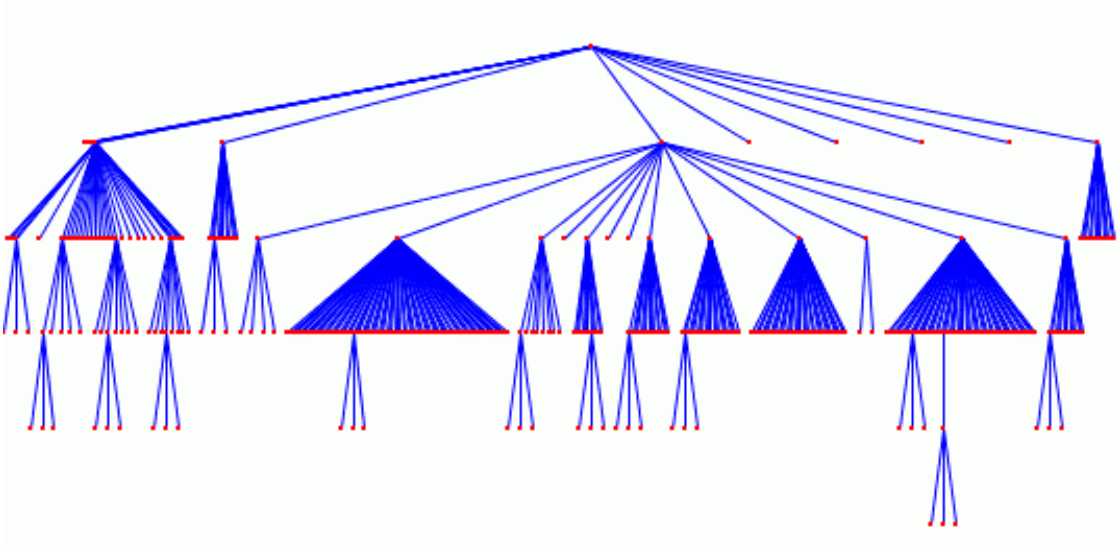


Figura 39. Representació basada en grafs d'un sistema amb grna nombre de dades.

Precisament la grandària del graf o el número d'elements que volem representar pot resultar un gran repte a l'hora d'utilitzar aquesta tècnica [32]. Per un costat la càrrega de nombrosos elements pot afectar al rendiment del sistema degut al cost dels càlculs relacionats amb ell, a més pot arribar al límit d'elements de la nostra plataforma de visualització. De la mateixa manera, l'anàlisi posterior sobre una estructura basada en grafs, és molt més costos a mesura que augmentem l'espectre de cerca. Estos problemes, es poden vore reflectits en un dels reptes més grans associats a la visualització de grafs: els layouts.

Quan ens referim a layouts, estem parlant de la definició ja siga manual o automàtica de la posició dels elements del graf sobre la superfície de visualització. En aquest problema, no solament hem de tindre en compte les coordenades sobre les que situarem cada node, si no

que a més, hem de definir la forma i tamany dels arcs per tal d'aconseguir una visualització intuïtiva i clara on no trobem per exemple arcs que s'entrecreuen o nodes superposats.

Actualment existeixen un gran nombre d'algorismes matemàtics centrats en la tasca de resolució d'este problema. En funció de l'estratègia o tipus de representació que intenten proposar, podem trobar nombroses investigacions centrades en algoritmes de layouts automàtics circulars, jeràrquics, hiperbòlics, radials, ortogonals o inclús mitjançant representacions 3D. També existeixen propietats o característiques que molts d'ells intenten aconseguir o parametritzar com ara l'opció de mantindre una distància costant entre nodes, aconseguir arcs amb la mateixa llargària o l'intent de reducció de l'àrea de representació del graf. Un gran nombre de les ferramentes actuals de visualització de grafs, ja ofereixen la implementació d'algunes d'aquestes estratègies que podem aplicar de manera senzilla a les nostres estructures. Aquest és un gran avantatge, ja que ens permet centrar-nos directament en l'aplicació d'estes eines a la millora de la nostra ferramenta, sense haver de dissenyar a priori un algoritme que satisfaça les nostres necessitats.

A banda del repte que suposen els layouts, un altre aspecte a destacar sobre el que també existeixen una sèrie d'investigacions, és la navegabilitat i visibilitat del graf. Aquesta característica fa referència a la facilitat de moure la nostra visió a través del graf, mostrant la informació contextualitzada i centrant-se en aquelles parts més importants en cada situació concreta.

Entre les tècniques clàssiques, cal destacar l'ús del Zoom i les estratègies de gestió del focus [32]. El zoom consisteix bàsicament en modificar el punt de vista de l'observador, apropant o allunyant els elements a base de transformacions i escalats sobre els artefactes representats i sobre la pròpia superfície de representació. Existeixen dos vessants diferents a l'hora d'implementar este tipus d'eina. La geomètrica o tradicional esta basada en transformacions simples d'escalat basant-nos en una relació de transformació definida pels usuaris. Per altre costat la semàntica, no solament fa un canvi respecte a la superfície mostrada, sinó que a més també aplica una escala de detalls en relació a la proximitat amb els elements. Un exemple d'aquest tipus de zoom el podem vore en ferramentes de visualització de mapes com ara Google Maps, on en augmentar el zoom sobre una àrea determinada, se'ns mostra un nombre més elevat d'informació relacionada com ara carrers o llocs d'interès.

Ambdós tipus de tècniques ens poden resultar molt útils a l'hora d'analitzar una secció concreta del nostre sistema degut a la grandària de la representació global de l'estructura d'un producte software. La tècnica semàntica a més, ens pot ajudar a separar la informació per nivells, i mostrar únicament certs nodes o relacions des de la vista inicial i anar afegint-ne de manera incremental. Per exemple, podríem mostrar les relacions jeràrquiques entre requisits inicialment, en fer zoom sobre un conjunt de nodes determinats mostrar també les relacions de traçabilitat entre ells i finalment en tornar a aproximar-se representar les PAs contingudes com a nodes associats a un requisit determinat.

A banda d'aquesta tècnica clàssica també existeixen altres com aquelles relacionades amb la posició del focus de la visualització. Les estratègies normals que s'apliquen en la major part de les ferramentes, estan basades en el moviment de la càmera per part dels usuaris i alguna opció més com l'opció de centrar la vista seguint uns criteris. Normalment estos criteris fan referència a la visualització completa de l'estructura representada o al fet de centrar la vista en relació a un grup determinat d'elements, encara així també hem trobat alternatives molt interesants que podríem aplicar en un futur. Per exemple, l'ús de la coneguda tècnica fish-eye [37] , que agranda l'àrea sobre la que posem el focus mitjançant la transformació dels elements ja siga de tamany o de distància en relació a la proximitat.

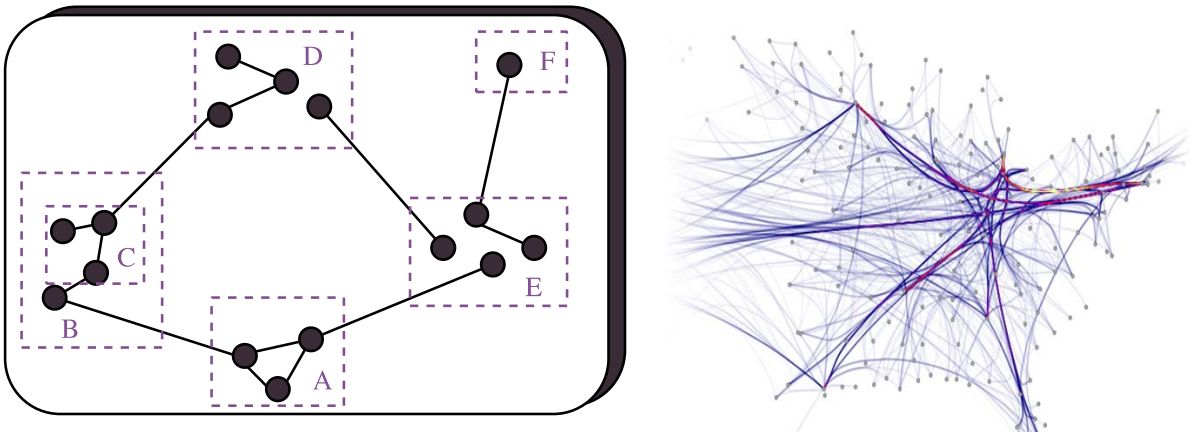


Figura 40. Representació d'un graf amb diversos tipus d'agrupacions.

Un altre estratègia molt utilitzada en l'àmbit de la visualització basada en grafs, és el clustering o agrupació d'elements. Aquesta tècnica ens permet reduir el número de nodes i arcs que es mostraran en el diagrama mitjançant el descobriment o la classificació d'elements amb característiques semblants. Aquest tipus de tècnica millora la visibilitat general del graf reduint la precisió o exactitud de la informació mostrada. Este tipus de

tècnica es sol aplicar sobre nodes, i mitjançant l'ús d'una sèrie de mètriques es troben de manera automàtica agrupacions amb trets pareguts. Una altra aplicació comú que podria ser molt útil per les característiques del nostre sistema, és l'agrupació feta per nivells. Utilitzant les relacions jeràrquiques existents entre els nodes de l'estructura de requisits, podríem carregar el graf de manera incremental, reduint de manera considerable el nombre d'elements mostrats inicialment de manera similar a com ho fem actualment en l'arbre de la vista general del gestor de requisits. També m'agradaria destacar l'existència d'alguna proposta d'agrupació d'arcs [38], que tot i ser menys coneguda, pot resultar molt útil com podem vore en la Figura 40, per tal de detectar o remarcar relacions o conjunt de nexes rellevants segons una sèrie de criteris establerts mitjançant l'ús de codis de color o grossor.

Aquesta personalització que acabem de comentar, és un dels grans avantatges que ens proporciona este tipus de visualització. La possibilitat d'utilitzar colors, tonalitats, tamanys i diverses formes, ens dona una gen flexibilitat i llibertat que permeten representar molts tipus d'informació diversa en un mateix control. Encara així, cal tindre sempre present que si es recarrega en excés un diagrama, aquest perd visibilitat i per tant cal utilitzar aquestes propietats amb coherència.

6.2.2. Alternatives

Una vegada analitzades les característiques teòriques d'aquest tipus de representació, volem fer una lleugera avaluació de les ferramentes i controls actuals per vore quines funcionalitats ens aporten.

Cal recordar que la nostra finalitat és crear un mòdul de visualització amb la informació de l'estructura de requisits dels productes software que es gestionen amb la nostra ferramenta TUNE-UP. Seguint aquesta idea valorarem la flexibilitat de les distints alternatives per tal de representar totes aquelles característiques que volem remarcar en la nostra estructura, el nivell de visibilitat i llegibilitat dels diagrames resultants, la quantitat de layouts automàtics disponibles, les opcions addicionals que ens proporciona cada un dels controls i sobretot la seua facilitat d'integració o de comunicació amb la nostra ferramenta de gestió de requisits.

Per últim destacar que entre les alternatives que anem a analitzar trobem controls d'usuari integrables directament en la nostra ferramenta, així com ferramentes independents encarregades de carregar grafs serialitzats a partir de les dades de la nostra eina.

6.2.2.1. Tulip

El primer element que anem a analitzar és el framework d'anàlisi i visualització de grafs Tulip (http://sourceforge.net/projects/auber). Aquesta opció ens aporta les dos vessants d'ús que comentaven amb anterioritat. Per un costat, es tracta d'una ferramenta independent i multiplataforma amb un gran nombre d'opcions. Ens permet importar i exportar graf mitjançant l'ús de distints formats, generar grafs de manera automàtica, aplicar múltiples layouts de visualització, així com algoritmes de selecció, mesura o test de propietats.

Per altre costat, també es pot utilitzar al mateix temps com una framework de desenvolupament en C++. Així podem crear distintes vistes on generar, editar, interactuar i analitzar les nostres dades. Aquest framework ens aporta un conjunt de llibreries en el mateix llenguatge de programació per al maneig de les estructures de dades i dels algoritmes, així com l'ús d'un conjunt de interfícies definides en Qt i inclús una API de desenvolupament en Python. Cal remarcar que una de les característiques més interesants d'aquesta opció és la possibilitat de definir nous algoritmes mitjançant scripts i plugins que es possible integrar en la ferramenta.

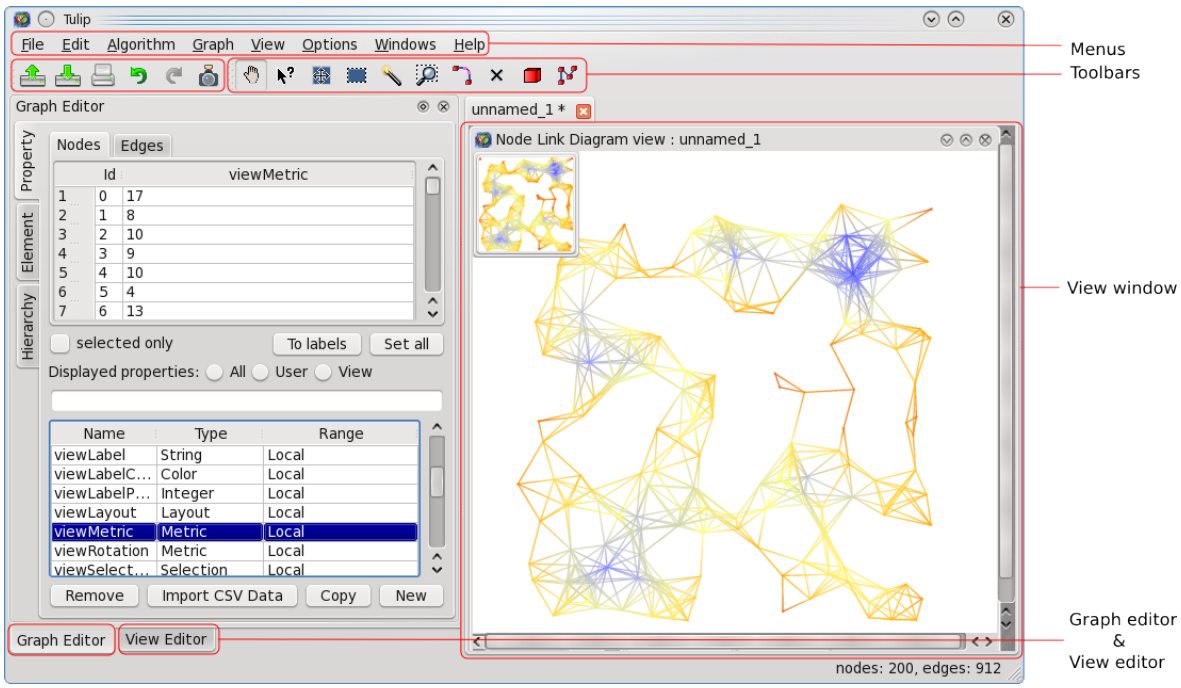


Figura 41. Interfície gràfica de la ferramenta Tulip.

Aquesta alternativa és, sense cap tipus de dubte, una de les més completes que existeixen en l'actualitat. El gran nombre d'algoritmes implementats i de formats compatibles en la

representació dels grafs, junt al fet de tractar-se d'un projecte de software lliure sobre la llicència LGPL, aconsegueix suplir un gran nombre de les necessitats o requisits del nostre mòdul. Encara així, cal destacar que la integració amb la ferramenta no es podria portar a terme fàcilment degut a la diferència de tecnologies existents amb el nostre sistema. Per tant, caldria utilitzar-ho com a ferramenta externa mitjançant la generació de les dades del graf. Un últim detall a considerar, és la maduresa i estabilitat de la ferramenta actual, ja que en fer algunes proves d'ús sobre grafos generats l'aplicació s'ens ha tancat de manera anòmala.

6.2.2.2. Graphviz

Aquest és una altre projecte open source, que naix en els laboratoris d'investigació d'AT&T (www.**graphviz**.org*)*. Inicialment es tracta d'un llenguatge propi de definició de grafs anomenat DOT i una sèrie de ferramentes de renderitzat de grafs utilitzant distints tipus d'algoritmes de layouts. Allò interesant d'aquest projecte és el conjunt de plugins i ferramentes paral·leles que utilitzen aquesta paquet com a motor de renderització. Altres ferramentes que anem a analitzar en esta secció fan ús de les seues propietats o afegeixen suport al seu format propi.

Entre els projectes existents, nosaltres volem centrar-nos en Graphviz4Net que proveeix un conjunt de control WPF sobre el nucli graphviz. Encara així, el projecte encara no resulta tan complet i madur com altres i a més, cal recordar que la nostra ferramenta esta basada en Windows Forms i caldria adaptar la visualització per tal de ser compatibles.

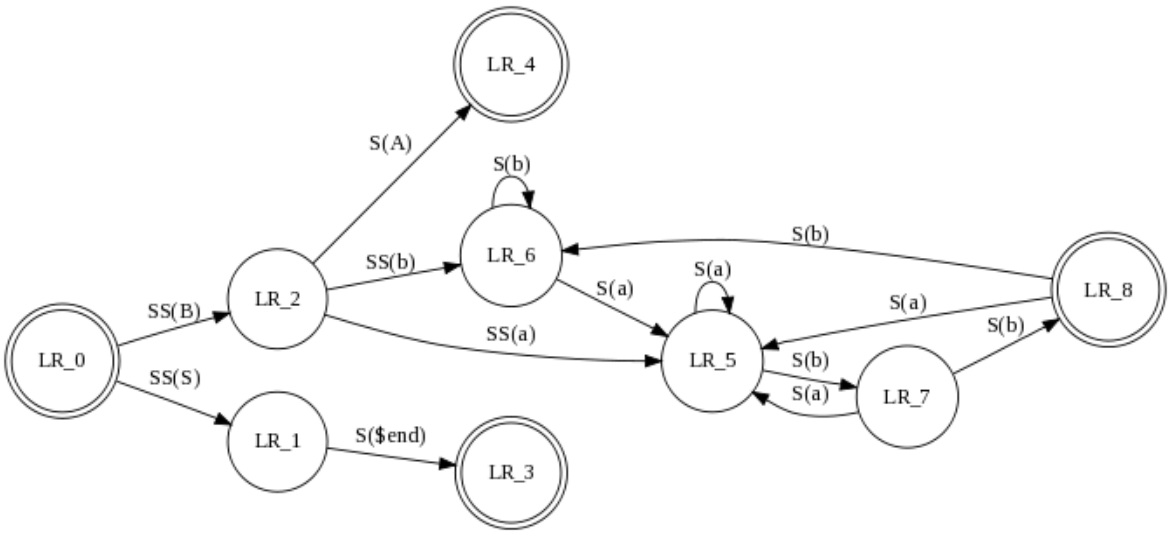


Figura 42. Renderitzat d'un graf mitjançant la ferramenta GraphViz.

6.2.2.3. Gephi

Possiblement aquesta siga la ferramenta més avançada i completa de les existents (https://**gephi**.org). Es tracta d'un projecte open source multiplataforma escrit en Java, amb una forta comunitat de desenvolupadors i projectes associats darrere. Suporta diversos tipus de formats de representació de grafs i es tracta d'una plataforma molt estable i madura que tot i trobar-se encara en la versió 0.8. La gran varietat d'opcions d'aquesta ferramenta tant de navegabilitat, mètriques, anàlisi, filtrat i clustering com diferents formes de visualització amb un gran nombre de layouts fan d'aquesta aplicació una de les més indicades per al nostre mòdul de visualització. Un altre factor a tindre en consideració era el plugin existent de comunicació entre la base de dades de grafs Neo4j, que utilitzarem en la primera part del projecte per emmagatzemar les relacions de traçabilitat dels requisits, amb aquesta ferramenta d'anàlisi de grafs.

Així i tot, aquesta opció no ens permet integrar la visualització dins d'un dels formularis de la nostra ferramenta. Açò resulta important ja que en tractar-se d'una aplicació externa, es perd part de la centralització o fluïdesa de la metodologia que aconseguim en utilitzar la ferramenta de suport TUNE-UP. Per altre costat, per polítiques internes de la empresa amb la que hem treballat, la màquina virtual de Java necessària per executar este software no es troba disponible, pel que caldria modificar les polítiques d'empresa per tal de poder visualitzar tota aquesta informació.

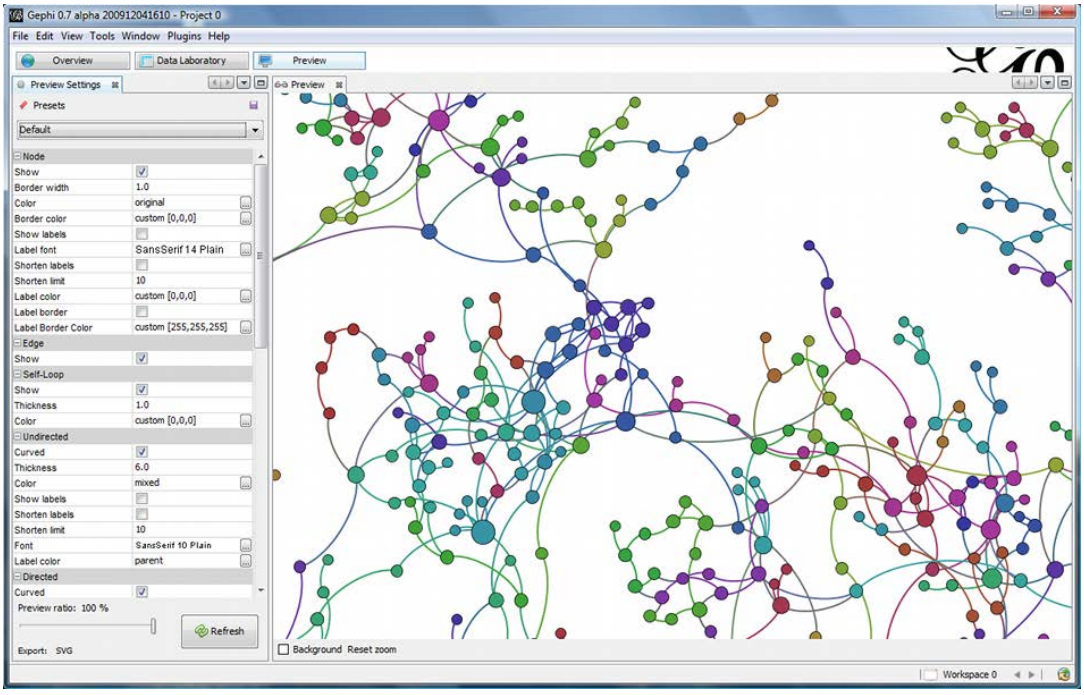


Figura 43. Interfície de visualització de la ferramenta Gephi

6.2.2.4. yFiles.NET

Aquesta ha sigut finalment l'alternativa elegida per tal de fer el nostre mòdul de visualització de grafs (http://www.yworks.com). No ens entretindrem molt en les seues característiques ja que en el capítol següent analitzarem amb més profunditat la seua arquitectura, la seua varietat respecte als algoritmes de layouts i en general les seues propietats. Així i tot cal destacar que es tracta d'un component basat en Windows Forms pel que podem integrar-lo de manera natural amb la nostra pròpia ferramenta. Este ha sigut un dels punts claus que ens han fet decantar-nos per aquesta opció. A banda d'esta facilitat d'integració que ens permetia centrar-nos en analitzar les opcions de visualització, també cal destacar la maduresa d'aquesta llibreria, el gran nombre d'algoritmes d'anàlisis i layout, la seua personalització i filtres i la gran quantitat d'exemples i documentació disponibles. Com a únic element negatiu, la llicència d'ús d'aquesta alternativa té un preu molt elevat, però ens ofereixen una versió de proves mensual sense restriccions que hem anat actualitzant fins acabar el mòdul.

6.2.2.5. Nevron Diagram

Aquesta és una altra opció molt interesant per als nostre interessos, ja que es tracta d'una altra llibreria basada en Windows Forms i per tant té una fàcil integració amb la nostra ferramenta (http://www.nevron.com). Com l'alternativa anterior, Nevron és una companyia encarregada de desenvolupar diversos tipus d'interfícies gràfiques i entre totes les que ofereixen, existeix un paquet anomenat Diagram .NET que inclou representacions basades en arbre i grafs. El preu d'aquesta eina és considerablement inferior al de l'anterior i també ens ofereixen una versió de probes temporal.

Aquesta alternativa ens ofereix un nombre de possibilitats paregudes a les de yFiles, des de layouts automàtics de representació, passant per ferramentes de navegació i zoom, fins a personalització del elements mostrats mitjançant la modificació de formes. Així i tot, la seua característica més

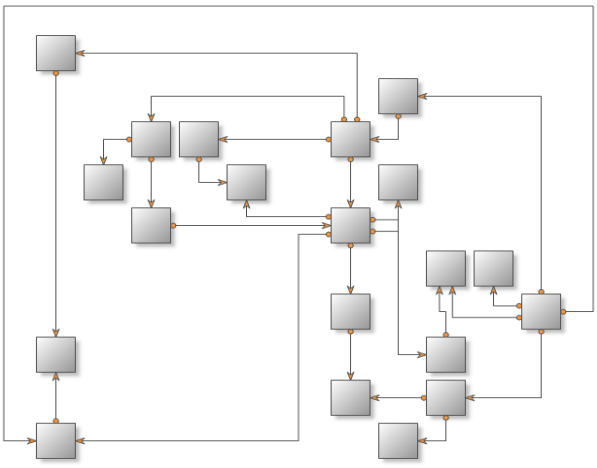


Figura 44. Llibreria de visualització Nevron Diagram

destacada és la compatibilitat existent d'aquesta plataforma respecte a la importació de les dades necessàries per definir el graf. Entre totes les opcions ofertes, destaca la possibilitat de llegir directament la informació d'un DataSet amb una organització concreta. L'únic inconvenient que podem trobar a aquesta opció, està possiblement relacionat amb qüestions de limitacions o escalabilitat, ja que encara que no hem pogut fer proves exhaustives sobre ella, pareix com si la representació estiguera pensada per a un sistema amb pocs centenars o milers d'elements.

El problema que vàrem tindre amb aquesta ferramenta a l'hora d'analitzar-la per a l'elecció, va ser la falta de documentació i les restriccions que existien en aquell moment sobre la versió de proves. En canvi, en escriure este capítol les condicions han canviat i d'haver sigut així podríem haver modificat la nostra elecció en favor d'aquesta llibreria.

6.2.2.6. QuickGraph + GLEE

Aquesta alternativa és un poc diferent a la resta de les analitzades amb anterioritat ja que es tracta de dos projectes diferents on cada un d'ells supleix les carències de l'altre. La primera vegada que ens vàrem plantejar crear un mòdul basat en grafs, estudiàrem l'existència d'alguna llibreria de visualització proporcionada per la pròpia Microsoft: Graph Layout Execution Engine (GLEE) (http://research.microsoft.com/en-us/projects/msagl). Aquesta és una llibreria amb llicència gratuïta, que ens dona la possibilitat de representar grafs mitjançant una API simple i senzilla d'utilitzar. El problema d'aquesta llibreria és que solament ens ofereixen opcions molt bàsiques respecte a la resta d'opcions analitzades, com per exemple la creació de nodes i arcs en una posició concreta i la definició de l'estil dels elements relacionats, en canvi no ens donen suporta layouts automàtics ni a algoritmes d'anàlisi.

Per tal d'aconseguir aquestes funcionalitats, existeixen dos camins: comprar una llicència del Microsoft Automatic Graph Layout (MSAGL) que és una millora que ha fet Microsoft sobre GLEE però amb llicència de pagament o l'opció que nosaltres trobàvem més factible, utilitzar la llibreria QuickGraph.

QuickGraph (quickgraph.codeplex.com*)* és una llibreria escrita en .NET que ens aporta un gran conjunt de ferramentes addicionals. Per un costat defineix una sèrie d'estructures de dades per gestionar i especificar el nostre graf i per altre ens aporta tots els algoritmes necessaris de layouts per visualitzar correctament el nostre sistema així com per analitzar

el graf. Encara així, aquesta llibreria no té capacitat de representació visual del graf, i per això utilitza GLEE o Graphviz com a motor de renderització. A més, també cal destacar que compta amb funcions de càrrega i serialització del graf en format GraphML. Per últim voldria destacar que tot i que aquesta opció té certs avantatges com és la seua gratuïtat o el gran control que tenim sobre tots els elements de la visualització, creiem que el projecte QuickGraph encara no és suficientment madur per a la seua utilització.

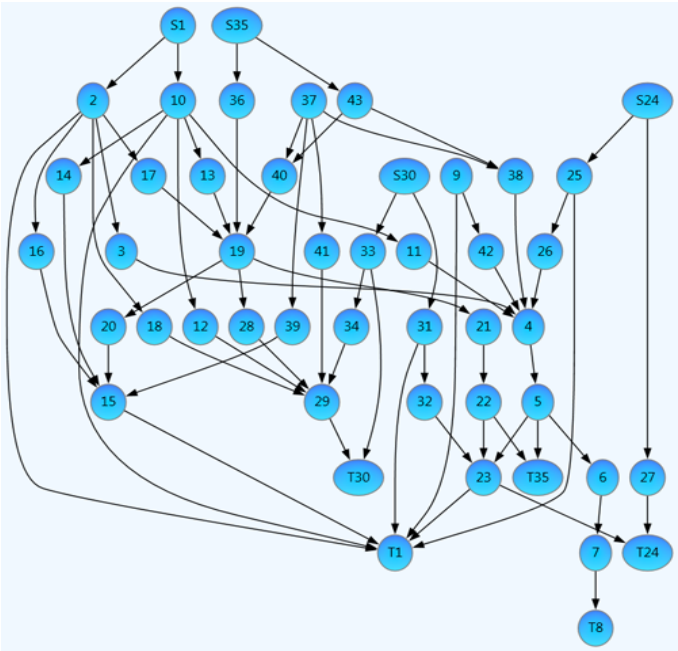


Figura 45. Representació d'un graf mitjançant la llibreria GLEE de Microsoft

Finalment destacar que existeixen també altres projectes analitzats com ara NodeXL, Graph# o aiSee molt semblant a la resta, però que en no aportar cap característica diferenciant i en trobar-se en estats menys madurs o amb una gran falta de documentació, hem decidit no aprofundir molt més en les seues especificacions.

Capítol 7. **Component de visualització basat en grafs: yFiles**

Una vegada ja hem analitzat les diferents alternatives existents en la representació basada en grafs i ja pressa la decisió d'utilitzar el component yFiles.NET per tal de portar a terme el nostre mòdul, és moment d'aprofundir més en aquesta llibreria i en tot allò que ens pot oferir.

yFiles són un conjunt de llibreries o de components de interfícies gràfiques implementades inicialment en Java, però que a poc a poc han anat aportant a altres tecnologies. En el nostre cas, hem elegit utilitzar la versió .NET que ofereixen basada en Windows Forms, ja que la ferramenta de suport a la metodologia TUNE-UP esta desenvolupada en C# i utilitza aquesta mateixa tecnologia per al desenvolupament de la nostra interfície gràfica. Aquesta companyia oferta un gran nombre de interfícies d'usuari, però nosaltres anem a centrar-nos en la representació basada en grafs que queda especificada en els arxius graph viewer i graph layout. Aquesta llibreria és, de llarg, la més completa i madura de les que hem analitzat en el punt anterior, i ens oferta una extensa variabilitat i flexibilitat per poder representar tot allò que volem amb ella. A banda d'açò, també cal destacar que la documentació associada a la llibreria és molt completa i s'ofereix un nombrós conjunt de demos i xicotets exemples molt útils a l'hora d'encarar-se per primera vegada a esta ferramenta.

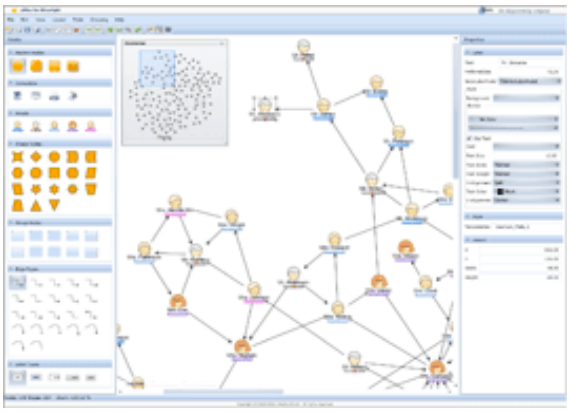


Figura 46. Demo yFiles.NET.

Com hem dit anteriorment, el principal inconvenient de la llibreria yFiles.NET es troba en el seu preu, ja que la llicència anual per a la seua utilització és molt més cara del que nosaltres estàvem considerant. Encara així, ofereixen proves gratuïtes del seu producte per un període de 30 dies (prorrogables) sense restriccions ni grans limitacions sobre la llibreria. Açò ens han permès treballar amb yFiles durant tot este temps i construir un mòdul basat en la nostra ferramenta, amb les millores que esperàvem representar. En el capítol següent es mostrarà amb més deteniment les característiques del mòdul esmentat i captures de pantalla per il·lustrar la seua funcionalitat, encara així, en les properes seccions s'explicaran certes parts utilitzades posteriorment per a la creació del cas d'estudi.

7.1. Arquitectura i estructures de dades

La llibreria yFiles .NET està dividida en dos parts diferenciades, que s'encarreguen de portar a terme totes les tasques necessàries per a la visualització del graf: el graph layout i el graph viewer.

Un dels màxims problemes a l'hora de representar grafs en un component d'estes característiques, és sens dubte, la posició que ha de tindre cada node respecte a la superfície del canvas. Aquest tipus de posicionament pot fer-se de manera manual, però açò seria pràcticament impossible de controlar amb grafs d'una certa grandària com els que nosaltres anem a gestionar. Des de fa uns anys, moltes de les línies d'investigació relacionades amb aquesta disciplina han posat tots els seus esforços en el camp de la creació d'algorismes automàtics d'ordenació de nodes en un layout. Aquest tipus d'investigació es certament costosa, i com la finalitat del nostre projecte no és aquesta si no trobar alternatives de visualització que satisfacen els nostres requisits, hem decidit elegir aquest component entre altres coses per la facilitat d'ús i per la complexitat que tenen els seus algoritmes de layouts.

És precisament en el graph layout on podem trobar la definició de tots estos algoritmes automàtics, entre ells trobem layouts circulars com el que veiem en la Figura 47, jeràrquics i orgànics. A més, aquesta part de la llibreria també és l'encarregada de portar a terme tots els algoritmes d'anàlisi sobre grafs com ara la cerca de camins entre dos nodes, el camí més curt o algoritmes de cerca sobre el graf. Encara que estos elements no van a ser utilitzats en el projecte actual, també ho hem valorat com a una característica positiva, ja que si els resultats aconseguits amb el mòdul són bons, podríem integrar-lo amb l'actual i començar una segon fase centrada en l'anàlisi de requisits mitjançant l'ús de la teoria de grafs i els seus algoritmes relacionats. Finalment destacar que en el graph layout també trobem opcions avançades d'actuació sobre el graf com l'ocultació d'elements o la seua copia parcial o completa i tota aquella part de codi encarregada d'interactuar amb el viewer.

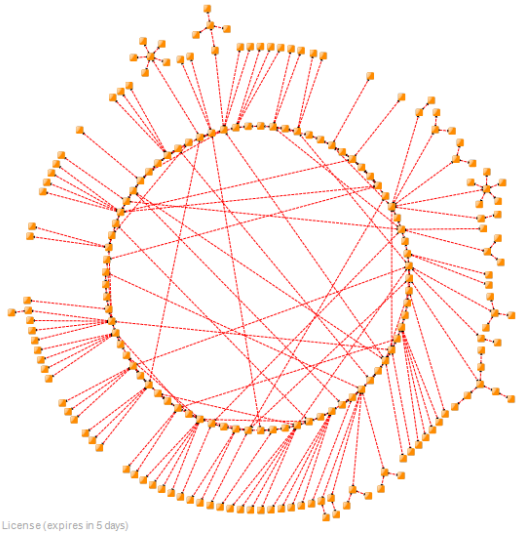


Figura 47. Representació d'un graf amb layout circular

Per altre costat, en la part anomenada graph viewer, és on s'especifica tota la infraestructura necessària per visualitzar i gestionar el graf sobre el component. Cal recordar que aquesta implementació de yFiles està basada en el la API Windows Forms que ofereix Microsoft per al desenvolupament d'interfícies gràfiques per a la seua plataforma/tecnologia. Este tipus de framework, implementa el patró Model –View – Controller (MVC), molt estès en la creació d'interfícies, que basat en un desenvolupament centrat en events ens permet aïllar les diferents capes que s'utilitzen en una interfície gràfica d'usuari. En la Figura 48 podem vore un esquema simplificat de la implementació del patró MVC en la llibreria que estem utilitzant. A continuació especificarem amb més detall cada una d'aquestes parts.

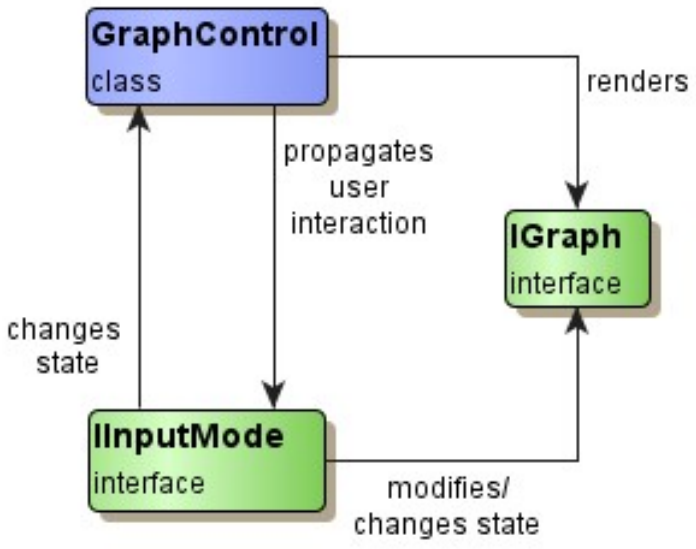
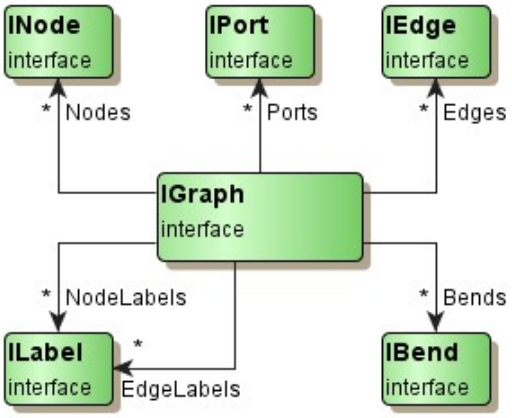


Figura 48. Arquitectura MVC de la llibrera yFiles.NET

El paradigma MVC, com el seu nom indica, separa la implementació de la interfície d'usuari en tres capes ben diferenciades. En primer lloc tenim la capa **Model**, encarregada de mantindre les dades associades al nostre sistema. En la llibreria yFiles aquesta capa està especificada per mitja d'una sèrie d'interfícies que representen una conjunt d'estructures de dades.

L'estructura central de la API és la figura del IGraph, aquest és l'element clau que utilitzarem per definir les característiques del nostre graf i serà l'element que englobe a tota la resta. Seguint el mateix conveni a

l'hora de definir els noms, també trobem els elements INode i IEdge que representen els artefactes bàsics per a la definició del grafs. Un altre element afegit en aquesta visualització és el concepte ILabel, que representa el text que apareix al mig dels nodes com a identificador.



A banda, per a cada un d'estos elements s'han definit una sèrie de classes que representen el seu estil o aparença visual. Gràcies a açò, podem definir la forma que representarà cada element, el color, el tamany, el tipus de lletra (per al cas dels labels associats als nodes i les relacions) i inclús si volem utilitzar alguna imatge com a identificador de node. Aquesta versatilitat és un dels factors que vàrem tindre en alta consideració en seleccionar aquesta llibreria.

Per altre costat, la capa **View** és l'encarregada de mostrar o presentar la informació final als usuaris, a més permet la interacció entre estos i el sistema mitjançant un sistema d'events. En el cas de yFiles, aquesta vista esta definida per la classe GraphControl, que és un classe que hereta de el Canvas de Windows Forms i que aporta a més una sèrie de funcionalitats ja implementades com ara un sistema de càrrega i guardat de grafs. Cal remarcar que aquest control pot funcionar en dos modes diferents, un primer on la funció primària és la de visualitzar un graf, i un segon més complex on a més de mostrar la representació de l'estructura també sens permet modificar-la.

Degut a estes modalitats del control, la capa **Controller** està definida en dos classes distintes a utilitzar en funció de quin mode estem utilitzant. Aquesta última capa, es situa entre los dos anteriors, i s'encarrega de processar les interaccions dels usuaris, especificar la modificació que s'ha de fer sobre el model i per últim actualitzar o renderitzar de nou la vista, en cas d'alguna modificació en les dades.

7.2. GraphML

Com s'ha pogut vore en la secció anterior, yFiles.Net i més concretament el graph viewer, ens aporta una API per tal de definir un graf mitjançant els elements bàsics d'aquesta representació. Tot i que podríem fer ús d'aquesta ferramenta, hem trobat adient utilitzar un altre mètode per definir la nostra estructura.

Si bé és cert que l'API que ens ofereixen és molt senzilla d'utilitzar i especialment completa, a l'inici del projecte desconeixíem si finalment anàvem a utilitzar i integrar aquest component com en la nostra ferramenta. Per tant, allò que volíem era analitzar tota la potència que un control d'este tipus ens podia oferir, però intentant que la major part del nostre esforç fora reutilitzable. És per això que seguint aquesta idea, ens vàrem decidir per utilitzar una altra opció d'entrada de grafs que ens oferia la llibreria yFiles mitjançant el graph viewer: el format GraphML basat en XML.

Aquest tipus de format, es utilitzat per la llibreria com a element serialitzador quan els usuaris decideixen guardar el graf que estan visualitzant/editant. Ens hem decidit per aquest tipus de format, ja que com hem vist amb anterioritat, dos de les ferramentes més avançades que existeixen actualment en aquest camp, Gephi i Tulip, també són capaces d'utilitzar aquest format per a carregar grafs.

Com acabem de dir, GraphML és un format de dades que es basa en la codificació XML per tal d'aconseguir un estàndard comú d'intercanvi de grafs entre ferramentes. Per tant, és un arxiu estructurat amb etiquetes que defineixen els distints elements que formen el graf i les seues propietats com podem vore en la Figura 49.

```
<graph id="G" edgedefault="directed">
  <node id="n0"/>
  <node id="n1"/>
  ...
  <node id="n10"/>
  <edge source="n0" target="n2"/>
  <edge source="n1" target="n2"/>
  ...
  <edge source="n8" target="n10"/>
</graph>
```

Figura 49. Exemple simple d'un fitxer GraphML

Després d'un element inicial XML, trobem l'etiqueta GraphML on es definiran les característiques de l'estàndard utilitzat, que serà el punt de partida per a la resta d'elements del nostre sistema. Dins d'aquest podem definir grafs, que tindran un ID i un tipus (dirigit o no dirigit) associat, nodes amb ID i edges, sobre els que definim un id, un node font i un altre destí, així com també especificar implícitament si es tracta d'una relació dirigida o no. A banda d'aquestes etiquetes, n'existeixen d'altres com ara port o hiperedge que nosaltres no utilitzarem en el nostre sistema degut principalment a la seua pròpia naturalesa.

Per tal d'afegir dades associades a cada un dels elements anteriors, aquest estàndard ens proporcions dos elements relacionats: KEY i DATA. El primer element el podríem caracteritzar com una definició de tipus de dades, a més, té associat un identificador i un àmbit d'acció (node, edge, graph, all,...). Per altre costat, l'element data es defineix dins de les característiques d'un element del graf, té associat l'identificador d'una key que especifica el tipus de les dades i aporta informació addicional sobre l'artefacte. Utilitzant esta especificació podem definir els noms de cada un dels elements o inclús el color o la forma que volem que tinga un determinat node.

Per tal d'utilitzar aquest format amb una certa facilitat, he creat una xicoteta llibreria encarregada de construir, de manera simple, un fitxer seguint l'estàndard graphML. Inicialment l'hem utilitzat per fer proves sobre el control per vore la potència d'aquest i en el capítol següent l'utilitzarem per crear un fitxer a partir de la nostra base de dades relacional. El codi font d'aquesta llibreria el podem vore en l'annex d'aquesta tesis.

7.3. Variabilitat i Layouts

Una vegada ja tenim una manera simple de carregar un graf en el component, anem a analitzar amb més deteniment quines possibilitats i variacions ens ofereix a l'hora de representar la nostra informació. En punts anteriors, ja hem introduït el mecanisme que ens proporciona la llibreria per tal de personalitzar els elements del graf. En el cas de la càrrega mitjançant graphML, cal modificar lleugerament l'estructura del fitxer per incloure dades no estàndards que ens posa a la nostra disposició la ferramenta. Per fer-ho, cal utilitzar una sèrie de tags especials on s'especifica el domini yFiles, crear keys associades a especificacions pròpies de la llibreria i finalment definir valor vàlids per al tipus de dades que anem a utilitzar.

Són moltes les característiques que podem modificar en relació als elements, però anem a destacar aquelles sobre les que ens han paregut més útils. Per un costat, cal parlar de la personalització del tipus i color de lletra que es pot fer sobre els labels presents en nodes i relacions. Encara que nosaltres probablement no utilitzem aquesta característica, està clar que pot resultar molt útil en nombroses situacions, inclús permetent una millor visualització en casos on el text no es puga llegir correctament.

Quant a les relacions, podem fer que es representen per una línia simple, per un arc o fins i tot amb una polilínia. Encara així el que nosaltres hem trobat més interessant és la

possibilitat de personalització respecte al grossor i el color d'estos. Amb aquesta opcionalitat podem distingir entre distints tipus de relacions (jeràrquiques i de traçabilitat per exemple) i inclús remarcar el grau d'importància de les relacions ja siga mitjançant una escala de colors o de tamanys.

Per últim, en relació a l'estil dels nodes hem de destacar la gran flexibilitat existent. Per un costat, podem utilitzar una imatge en compte d'una forma geomètrica per tal de representar-los. Encara que tal vegada en el nostre cas d'estudi no resulta molt útil degut a que solament anem a mostrar inicialment dos tipus d'artefactes, és una molt bona característica que ajuda als desenvolupadors a augmentar la visibilitat i intuïtivitat dels nostres diagrames. Per altre costat, també trobem les mateixes opcions existents respecte a les relacions, on podem modificar el color, la forma, el tamany i inclús la brillantor.

Una de les característiques més interesants que hem trobat, és la possibilitat de filtrar o ocultar nodes en funció d'expressions booleanes. Açò ens permet donar la possibilitat als usuaris de filtrar la informació mostrada seguint una sèrie de característiques del sistema. També cal destacar la gran varietat d'opcions ja implementades que ens proporciona la ferramenta, entre elles podem destacar l'opció d'emmagatzemar i carregar nodes (utilitzat per nosaltres per a la definició del graf), ferramentes de zoom i navegabilitat i inclús opcions per a la definició del focus de la representació. Finalment, l'últim detall a considerar en relació a la potència de visualització del component yFiles, és precisament el que ens va fer decantar-nos finalment per aquesta llibreria, és a dir, els layouts automàtics.

yFiles.NET ens proporciona una sèrie d'algoritmes automàtics per al posicionament dels nodes respecte al GraphControl. A més, ens permet modificar certs paràmetres de visualització com ara la distància mínima o màxima que volem que hi haja entre els distints elements. Encara que existeixen més algoritmes implementats, nosaltres volem posar la nostra atenció sobre els quatre que veiem representats en la Figura 50, ja que pensem que són els que millor ens ajuden a mostrar les característiques del graf.

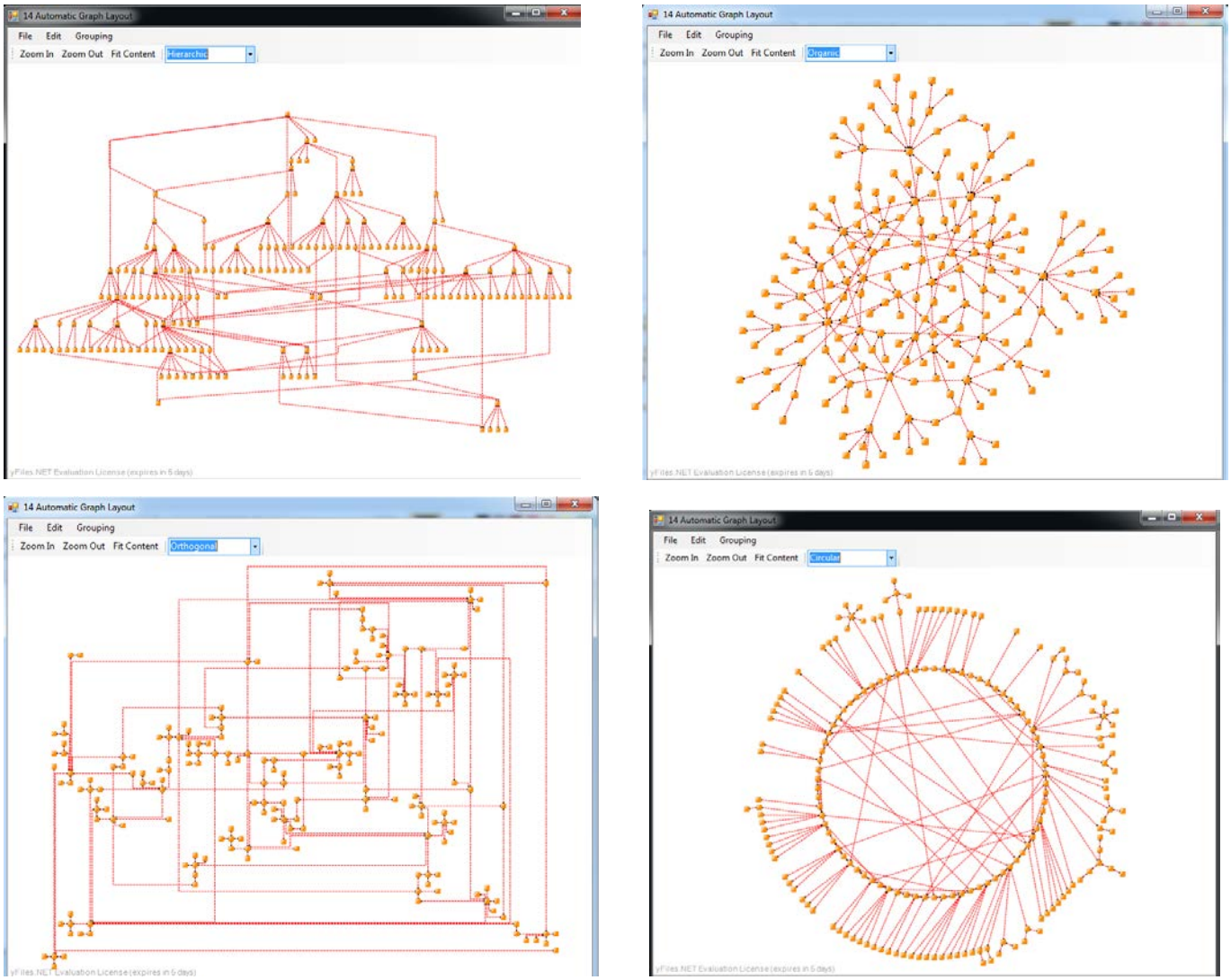


Figura 50. Aplicació de layouts automàtics (jeràrquic, organic, ortogonal i circular)

El layout que més anem a utilitzar o que millor s'apropa a les necessitats del nostre sistema és probablement el jeràrquic, ja que permet mostrar-nos l'estructura de l'arbre, contextualitzant la posició de cada element respecta a la resta. Encara així, en el capítol següent podrem vore una implementació amb dades reals sobre la que s'intentaran mostrar la major part de les característiques ací expressades.

Capítol 8. **Cas d’estudi i mòdul de visualització**

A l’hora de plantejar-se crear un mòdul basat en la representació de la informació mitjançant grafs, hem de valorar primer que és allò que volem aconseguir mostrar. Per un costat, cal aplicar aquest tipus de visualització a casos d’estudi reals que ens permeten analitzar aquest diagrama i extraure conclusions útils i fiables, a més açò ens permetrà trobar les possibles limitacions o restriccions d’aquesta representació. Per altre costat, també seria convenient que experimentarem diverses variants de la visualització per tal de poder aplicar aquest mòdul a diversos aspectes del nostre sistema i vore a quin element representa més adequadament.

En relació als casos d’estudi, hem decidit aplicar aquest mòdul a dos de les ferramentes reals que es gestionen a diari seguint la metodologia TUNE-UP. Per un costat, mostrarem l’estructura interna de l’aplicació SAPI, que es tracta d’una implementació alternativa de la ferramenta de suport a la metodologia TUNE-UP, que hem anat desenvolupant durant anys i que està personalitzada seguint les necessitats de l’empresa amb la que col·laborem. Es tracta d’una aplicació mitjana, amb una gran quantitat de funcionalitats, que és utilitzada a diari pels agents de l’empresa en el seu treball i que segueix cicles de desenvolupament d’aproximadament 3 setmanes.

Per altre costat, també anem a representar l’estructura de l’ERP sociosanitari, que és el principal producte de l’empresa amb la que col·laborem. Es tracta d’un software de gestió sociosanitari que permet gestionar diverses facetes relacionades amb les residències de persones majors, centres de dia i establiments amb tasques paregudes. Aquest software, té un nombre ingent de funcions, mòduls i operacions que permeten simplificar i optimitzar la tasca dels responsables d’estos centres, el que ha propiciat que actualment siga l’aplicació més utilitzada pel sector en l’àmbit nacional. Es tracta per tant, d’una aplicació de gran tamany, amb un nombrós conjunt de requisits en constant evolució i una alta complexitat.

Com hem comentat amb anterioritat, a banda d’estos casos d’estudi reals, també estem interessats en analitzar alternatives locals o més enfocades en una característica concreta. Entre les possibles propostes, seria interesant poder analitzar l’impacte d’un node o un conjunt d’estos sobre la resta del sistema, o inclús poder analitzar tot allò que afecta a una determinada WU d’un sistema.

El primer pas que havíem de fer, era estendre la funcionalitat del constructor d'arxius GraphML estàndard que havia creat a l'hora d'avaluar i analitzar la ferramenta yFiles. Per tal de portar a terme la personalització dels diagrames a representar, calia afegir una sèrie d'etiquetes específiques d'aquest framework per tal de que el graphML loader de la llibreria, carregara la informació correctament. Per tal de fer-ho i gràcies a l'arquitectura de la llibreria implementada, solament havíem d'estendre les interfícies definides per a les dades addicionals de nodes i arcs, redefinint el comportament del mètode ToString() per tal de crear la línia graphML que represente la característica determinada.

El màxim problema que s'ha trobat a l'hora d'implementar aquesta part del projecte ha sigut la falta de documentació existent al respecte, però després d'un tediós anàlisi sobre les opcions de serialització de les respectius demos, he sigut capaç de detectar la sintaxis necessària per a la personalització del nostre sistema. A banda de certes opcions com ara el canvi de color, forma o tamany dels elements mostrats, també hem detectat un mecanisme que ens permet emmagatzemar informació en l'arxiu graphML per tal de poder mostrar-la a posteriori en el control de visualització.

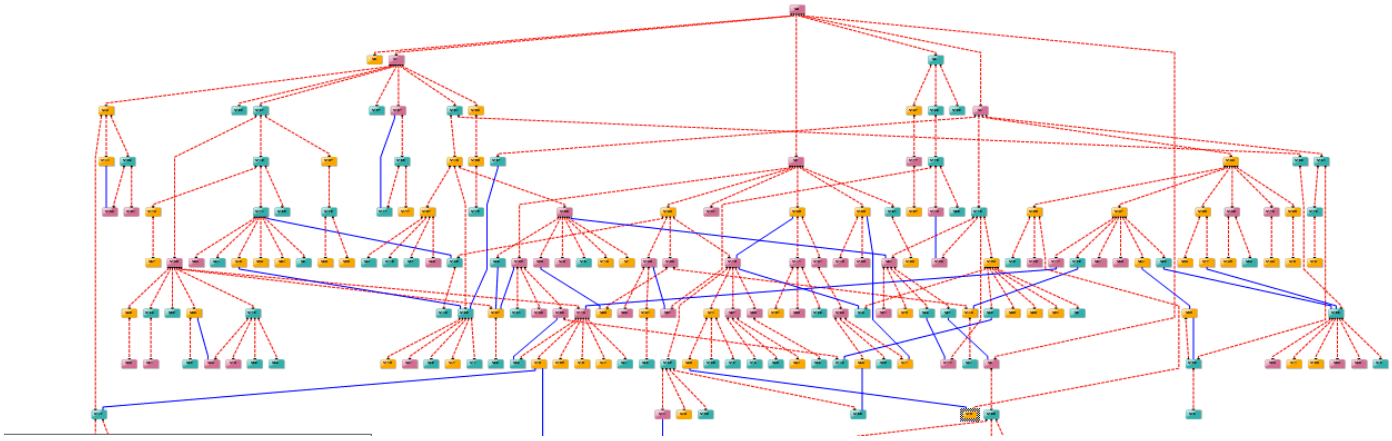


Figura 51. Exemple de graf generat automàticament

Després de millorar el graphML Builder i definir un sistema per tal d'utilitzar la informació addicional emmagatzemada en els arxius generats, s'ha creat un xicotet algoritme de creació d'un graf de 200 nodes en forma d'arbre multinivell amb diferents característiques de personalització, com el que podem vore en la Figura 51. Aquesta primera càrrega, ha permès analitzar la potència de la ferramenta i facilita la posterior tasca de mostra d'informació real, aportant idees per a la millora de les característiques de navegació del mòdul.

Una vegada creada la infraestructura necessària per carregar la informació en el control de visualització basat en grafs, calia centrar-se en aplicar-ho a la informació emmagatzemada en les bases de dades de gestió dels programes anteriorment esmentats. El problema principal que vàrem trobar en aquesta fase està relacionat amb les polítiques de seguretat i privacitat de l'empresa, que fan difícil extraure informació del seus sistemes o instal·lar software de tercers en les seues màquines de desenvolupament. Encara així, hem utilitzat el constructor d'arxius graphML que havíem implementat amb anterioritat, i s'ha pogut integrar parcialment en la ferramenta de suport per a la metodologia TUNE-UP. D'aquesta manera, hem pogut generar i extraure la informació necessària d'aquestes dos aplicacions. En concret, hem extret els nodes i relacions jeràrquiques existents entre ells, centrant-nos en els seus identificadors i noms. Per altre costat, també hem decidit incloure totes les PAs (id i nom també) com a nodes, creant relacions entre contenidors i contingut.

En este cas, hem decidit especificar els nodes i les PAs amb diferents colors, així com les relacions existents entre estos elements amb diferent color i forma per tal de poder distingir amb facilitat els distints elements i per simplificar la tasca de l'opció de filtrat que ens agradaria incloure. La generació d'estos arxius és una tasca relativament ràpida, que ha tardat aproximadament 5 segons per al programa més gran. Per altre costat, l'arxiu corresponent a l'ERP ocupa 5Mb i la de SAPI 1'5Mb. Aquestes mesures poden resultar interesants, ja que una possible aplicació d'aquesta visualització seria l'emmagatzemament versionat de l'estructura de requisits d'un determinat producte. Aquesta millora ens permetria analitzar l'evolució de les nostres aplicacions interactuant i navegant per les distintes versions emmagatzemades.

Tornant a l'especificació del mòdul de visualització de requisits, cal destacar que a l'hora de carregar els grafs generats amb la llibreria yFiles i aplicar els layouts automàtics que havíem seleccionat (jeràrquic, orgànic, circular i ortogonal), hem pogut observar les limitacions existents respecte a la visibilitat d'aquest tipus de diagrama. Com podem vore en la Figura 52, la quantitat d'informació a representar és tan gran i l'espai de representació tan limitat, que no existeix un layout que permeta analitzar de forma simple l'estructura global de les nostres aplicacions.

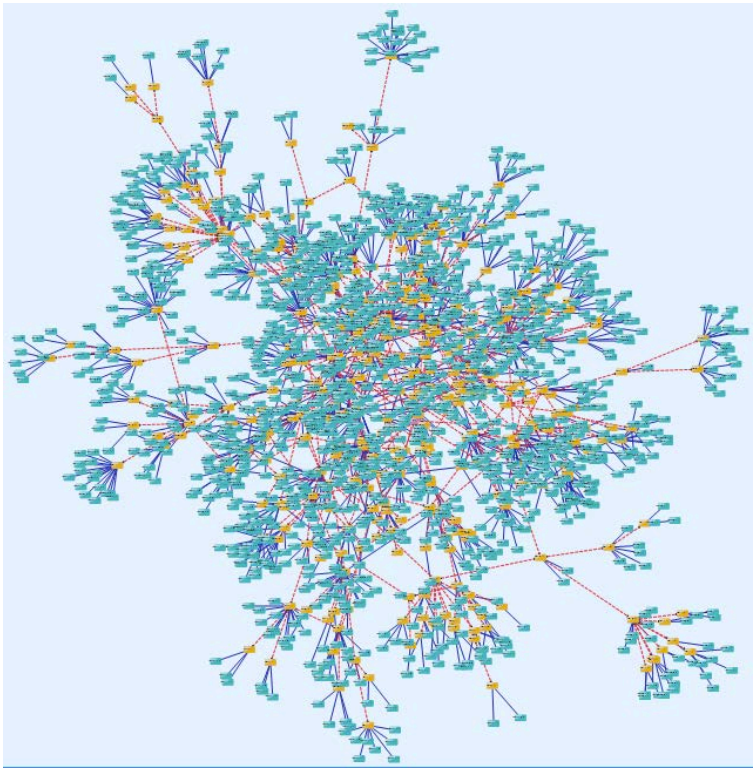
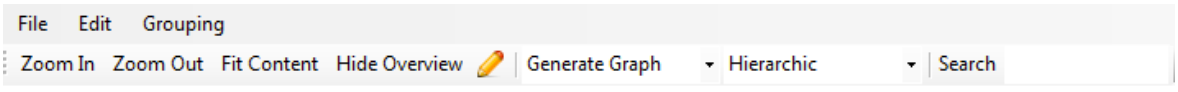


Figura 52. Representació de l'estructura completa de l'ERP amb un layout orgànic

A banda de la immensa quantitat d'elements a tindre en compte, una altra qüestió que vàrem detectar són les carències existents a nivell de navegació. Cal destacar que quan la quantitat d'elements és tan gran que no es poden representar tots en una mateixa pantalla, cal millorar les opcions de navegabilitat al màxim per simplificar la tasca d'anàlisi de les dades mostrades. La primera versió del mòdul, s'encarregava de mostrar simplement la informació completa d'un programa, i gràcies a esta hem discernit que aquest tipus de diagrama és més útil quan podem acotar els elements a representar seguint un context determinat. Seguint aquestes idees, s'ha anat polint el mòdul de visualització mostrant diferents opcions que poden ser molt útils de cara a una futura integració amb la ferramenta.



Com es pot vore en la imatge superior, s'ha creat una barra d'opcions amb diversos elements útils que actuen sobre el graf inferior. Inicialment tenim tres opcions relacionades amb la vista específica del graf i més concretament amb el zoom utilitzat per mostrar la informació (Zoom in, Zoom out, Fit Content). Estes ferramentes permeten analitzar la informació des d'un punt de vista global o per altre costat, centrar-nos en un espai concret

de la representació. A més d'açò, en analitzar els exemples aportats per la llibreria yFiles, es va detecta una opció molt útil anomenada OverView que com podem vore en la Figura 53, es tracta d'un xicotet rectangle situat en aquest cas en el cantó inferior esquerre de la imatge, i que mostra en miniatura la representació completa del graf de requisits.

Aquesta opció, ens permet definir d'una manera molt simple, quina secció del diagrama volem analitzar, amb la conseqüent aplicació automàtica de les ferramentes d'acotació i zoom sobre la vista general de la representació. Després de moltes proves, s'ha pogut observar la seua gran utilitat i senzillesa, i s'ha afegit un botó per tal de mostrar o ocultar aquesta ferramenta en funció de la necessitat de l'agent que esta analitzant la informació.

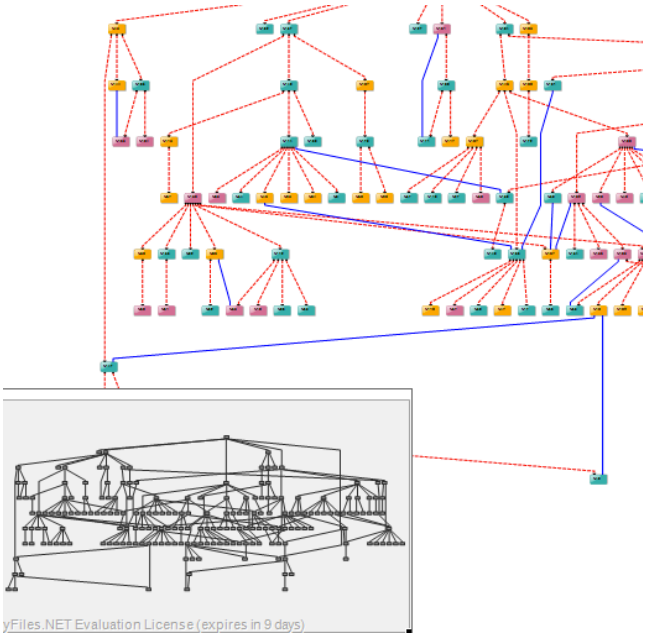


Figura 53. Ferramenta Overview de navegació

Com a últim element per millorar la navegació del diagrama, hem decidit definir dos modes d'accés sobre la visualització de l'estructura. D'aquesta manera, podem filtrar algunes ferramentes que ens ofereix la llibreria i que resulten útils en determinades situacions, però que per contra poden ser una distracció en altres contextos. Per tant, hem separat el seu comportament en dos: el mode d'edició i el de navegació. En el primer, es poden crear nous nodes o relacions amb un simple click en el ratolí. A més existeixen moviments del cursor que faciliten la tasca de moure els diferents elements de lloc o inclús d'afegir punts de desviació sobre els arcs. Per altre costat, en mode de navegació tot i no oferir-nos la possibilitat de modificar el graf, ens facilita molt el moviment de la càmera i l'ús del zoom i es centra detingudament en l'anàlisi de la informació.

En la segona secció de la barra de ferramentes implementada, s'han definit opcions relacionades directament amb la informació que volem mostrar en el diagrama inferior. Inicialment es mostra un combo amb les alternatives que hem definit per a aquest mòdul

concret, és a dir, les quatre opcions de càrrega d'informació que hem decidit estudiar en este exemple.

Aquestes són les alternatives existents:

- Estructura general del programa 1. Es tracta d'un software de gran tamany.
- Estructura general del programa 2. Es tracta d'un software de tamany mitjà.
- Generació semi-automàtica d'una estructura. Crea una estructura de software característica que consta de 200 nodes, i que segueix un model d'arbre molt semblant al dels programes amb els que treballem. Aquesta mesura mostra un programa amb un tamany perfecte per tal de poder explotar totes les característiques del nostre mòdul.
- Exemple d'aplicació sobre una WU. Es tracta de la càrrega d'un xicotet graf com es pot vore en la Figura 54 on es mostren diferents aspectes que podríem representar mitjançant l'ús d'aquest diagrama. En aquest cas, tenim un node central amb un tamany més gran que la resta que representa la WU que volem representar. A partir d'aquesta trobem relacions sobre altres elements relacionats, com ara altres WUs que han sigut marcades amb algun tipus de connexió amb l'actual o tots els nodes de requisits que es veuen afectats per aquesta. Cada element ve distingit amb un color diferent en relació al tipus d'artefacte que representa. Per últim hem volgut mostrar també les PAs creades, modificades o eliminades en aquesta WU, i marcar en diferents colors el seu estat actual de testeig, és a dir, roig si la prova ha fallat, verd si ha passat satisfactòriament o gris si encara no ha sigut aplicada. Encara que no es tracta d'una extracció real d'una WU de ningun dels productes gestionats per la nostra ferramenta, el esquema mostrat representa un cas d'estudi realista que intenta mostrar totes les possibles relacions existents entre estos artefactes i destacar la potència d'aquest tipus de visualització contextualitzada.

Seguint el recorregut per les diferents ferramentes creades per al mòdul, a continuació trobem un llistat amb els diferents layouts que podem aplicar sobre el nostre graf. En el nostre cas hem decidit proposar-ne quatre, que són els mateixos que podem vore aplicats en la Figura 50. Sobre estos algoritmes, la llibreria ens permet modificar alguns paràmetres com ara la separació màxima o mínima existent entre els distints elements. Un altra opció molt interessant és l'opció de propietats per tal de compactar al màxim (horitzontal o

vertical) la representació del diagrama. En el nostre cas, he aplicat una compactació horitzontal, però encara així la quantitat d'informació mostrada resulta difícil d'analitzar.

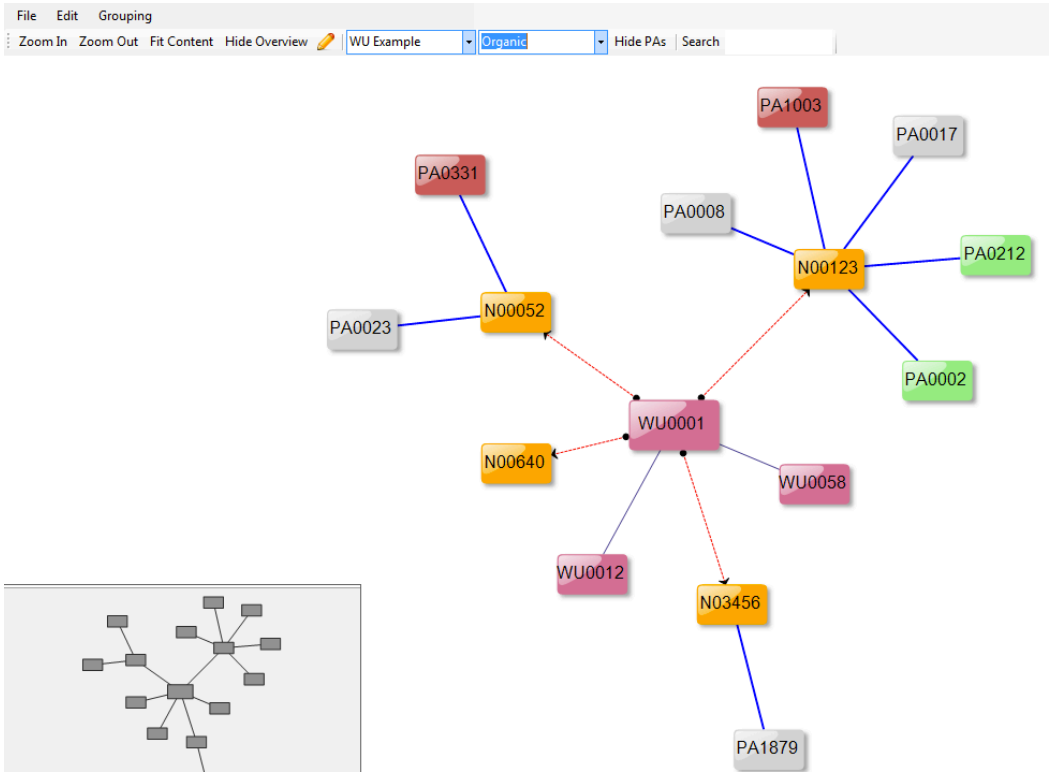


Figura 54. Exemple de representació dels elements d'una WU sobre el proptotipus

El següent element que hem decidit aportar al mòdul és la possibilitat d'amagar o mostrar els nodes que representen a les PAs degut a la seua gran quantitat. En gran part de les anàlisis que podem portar a terme, aquesta informació pot resultar irrellevant i si tenim en compte que la major part dels nodes i arcs representats estan directament relacionats amb aquest concepte, creiem que seria una bona opció poder decidir d'una manera ràpida i simple si volem visualitzar aquesta informació.

Per tal d'aconseguir aquesta funcionalitat, ens basem en un sistema de filtrat existent en la llibreria. Per utilitzar-ho, cal definir un tipus de dades que caracteritza al graf filtrat i marcar quins nodes o arcs volem visualitzar. Aquesta separació es fa mitjançant criteris simples com ara la separació per colors o formes dels conjunt de nodes existents.

Una altra opció que ens ha paregut molt útil, és la de cercar un determinat element del graf i centrar la visualització en ell. Per això s'ha definit un textedit per tal que els usuaris puguen escriure el identificador de l'element que volen trobar. Una vegada llançada la cerca, si l'element existeix dins del graf, es centrarà la imatge i s'aplicarà un determinat

zoom per tal de que l'artefacte aparega centrat en la vista i amb un tamany adequat. La implementació d'aquesta utilitat es basa en el recorregut de tots els elements existents dins de l'IGraph del control, i una vegada detectat el que busquem, aplicar una sèrie de transformacions sobre la vista per tal de centrar la càmera seguint els interessos de l'agent.

Quan teníem implementades aquestes últimes funcionalitats (filtrar PAs i cercar elements), es va detectar que la unió d'estos dos mecanismes ens permetria analitzar una altra vessant molt interessant que havíem plantejat per a futures millores: l'anàlisi centrada en un conjunt reduït de nodes. Seguint questes idees, hem afegit un altre textedit dedicat al filtrat d'elements amb un node determinat com a condició principal.

El funcionament d'aquest potent ferramenta és molt senzill. En primer lloc es recorren tots els nodes per trobar aquell que correspon amb el text escrit per l'usuari, una vegada hem trobat l'identificador d'este element, recorrem tots els arcs existents i marquem aquells que estiguen directament relacionats amb el que busquem mitjançant una propietat dels elements del control anomenada tags. A continuació, apliquem un filtre seguint les mateixes característiques que els filtres explicats anteriorment, centrant-nos en aquells elements que tenen la propietat tag marcada com a certa. Finalment, apliquem els mateixos mecanismes que dèiem en la cerca, per tal de centrar l'element sobre el que estàvem buscant.

Una vegada aplicat el filtre, s'ha modificat l'accés a les ferramentes i es permet desfer els canvis i tornar al graf inicial, tornar a aplicar un layout sobre el nou graf reduït o aplicar una altre nivell de profunditat d'impacte sobre la cerca, en aquest mòdul solament hem implementat l'opció d'anàlisi sobre dos nivells de proximitat respecte al node seleccionat, però seguint el mateix funcionament es podrien oferir molts més.

Per últim cal destacar que la representació inicial del nostre mòdul tenia certes carències en relació a la informació mostrada sobre cada element, per tal de solucionar-ho es varen definir una sèrie de mecanismes per fer més visible i identificable cada element. En concret, en l'actualitat cada node mostra un label amb el seu codi identificador, i s'ha afegit un tooltip a cada element representat que mostrarà el títol o nom de l'artefacte en mantindre el cursor dos segons per damunt de l'element que volem analitzar. Per fer-ho, s'ha creat una nova etiqueta per al graphML, que *mappegem* a continuació mitjançant uns mecanismes que ens proporciona la llibreria per tal de ser assignats a cada element de la representació.

Amb aquesta última funcionalitat implementada s'ha mostrat tot allò que volíem analitzar respecte a aquest tipus de representació. Cal destacar, que tot hi haver trobat importants limitacions en relació al nombre d'elements que es poden mostrar mantenint una bona visibilitat, hem detectat noves aplicacions on aquesta visualització podria resultar veritablement útil.

En carregar el conjunt d'informació relacionat amb dos dels programes que actualment gestionem amb la nostra ferramenta de suport a la metodologia TUNE-UP, s'ha pogut vore que la nostra finalitat principal centrada en la visualització de l'estructura global d'un producte no s'adequa correctament a aquesta ferramenta i per tant no s'ha pogut acomplir. De totes maneres tot aquest treball ens ha servit per valorar aquest tipus de visualització des de diferents punts de vista, i el resultat ha sigut més que satisfactori. La seua gran flexibilitat, personalització, potència i sobretot la seua representació intuïtiva, ens ha permès detectar nous contextos on poder, en un futur, traure a relluir les seues característiques i aprofitar-ho per simplificar la tasca de visualització, detecció i gestió dels artefactes involucrats en el procés de desenvolupament.

Capítol 9. **Conclusions i treball futur**

En l'actualitat, resulta cada vegada més important tindre una bona metodologia de desenvolupament software on la gestió de requisits supose una peça clau i decisòria per tal de millorar l'eficiència i disminuir del cost del manteniment dels productes. Com l'experiència ens diu, el software és un producte extremadament dinàmic que es troba exposat constantment a canvis. Ja siga per tal de solucionar errors existents, realitzar millores sobre alguns mòduls del software o degut a nous requisits definits conjuntament amb els clients, la creació d'aplicacions informàtiques es desenvolupa seguint un procediment iteratiu incremental. Aquesta naturalesa dinàmica és precisament la que defineix la importància de gestionar correctament els requisits software i de conèixer en tot moment allò que el client espera i l'estat de la nostra estructura.

Per tal de millorar la nostra ferramenta de suport a la metodologia de desenvolupament software TUNE-UP, aportant noves idees i madurant-ne altres de noves, ens hem volgut centrar en la visualització de l'estructura de requisits de programes. Pensem que una representació completa i intuïtiva de tots els elements relacionats amb la funcionalitat i característiques de seguretat i qualitat, pot ajudar a tots els distints rols encarregats del desenvolupament d'aplicacions a optimitzar el seu treball, reduint costos en les diferents etapes de creació, centrant-se sobretot en la part més costosa, el manteniment. Per un costat, el Product Manager serà més conscient del desenvolupament general de la ferramenta, i facilitarà la seua tasca de gestió de recursos i pressa de decisions. Per altre costat, els analistes seran més conscients del seu treball des d'un punt de vista global i els aportarà un nou conjunt de ferramentes, facilitant així la seua tasca. Per últim, els testers amb aquesta nova modificació aconsegueixen finalment una visualització clara sobre l'estat del testeig de la ferramenta desenvolupada, no localitzat com fins a eixe moment en el context d'una WU.

Amb aquesta idea en ment, hem dividit el nostre projecte en dos fases de desenvolupament. En un primer lloc, hem analitzat la ferramenta actual de gestió de requisits, separant la informació mostrada en tres vistes amb funcionalitats diferenciades. En la primera, trobem l'estructura de l'arbre clàssica que teníem abans de començar el projecte, però centrant ací únicament la tasca de detecció dels nodes afectats per una determinada WU. En la segona, basant-nos en la idea de simplificació i contextualització, hem creat una nova vista per a la definició de les PAs, on únicament mostrem les ocurrències particulars dels nodes que

afecten al context de la WU. Per últim, hem definit una vista estadística on es mostren diferents mètriques sobre l'aplicació gestionada, ja siga a nivell de PA o node per tal de que els analistes puguen detectar contenidors a refactoritzar, com a nivell de PSs per tal de mostrar als testers l'estat de l'etapa de testeig d'aquesta aplicació.

En la segona fase de desenvolupament, hem decidit centrar els nostres esforços en trobar diagrames de visualització alternatius, que ens permeten representar millor les relacions de traçabilitat existents entre els requisits i al mateix temps ens aporten una major visibilitat i comprensió sobre l'estructura. Entre les alternatives analitzades, ens em volgut centrar en la visualització basada en grafs ja que pensem que és l'estructura lògica i natural de representació d'un sistema interrelacionat com és el nostre. Amb aquesta idea, hem analitzat els diferents components i ferramentes existents i hem creat un mòdul amb un d'ells, concretament yFiles.NET. En aquest exemplar, hem demostrat tota la potència que ens proporciona aquest tipus de visualització i hem aplicat dos casos d'ús reals per tal de vore el seu rendiment enfront situacions verídiques.

En general, el treball realitzat en aquest projecte ha sigut molt satisfactori. Respecte a la divisió del gestor de requisits en vistes, l'acollida entre els agents que utilitzen diàriament la nostra ferramenta en producció ha sigut molt bona. Estos ens han aportat una gran quantitat de feedback positiu respecte al rendiment de la nova vista resumida i contínuament ens donen idees per tal de intentar millorar-la. A més, gràcies a la nova vista d'anàlisi s'han detectat una sèrie de nodes amb un nombre massa elevat de proves d'acceptació que ha donat peu a tasques de refactorització de requisits. De la mateixa manera, la detecció de nodes amb nombroses ocurrències (fins a 63 en algun cas) ha donat peu a replantejar-se l'ús que fan els analistes d'aquesta eina.

En relació a la visualització de l'estat del testeig, comentar que tot i no tindre crítiques o realimentació al respecte, principalment degut al reduït període de temps que porta en producció, creiem que és una vista molt útil que pot ajudar a planificar millor les tasques de l'equip de testeig. A més, el gran interès mostrat per estos per tal d'integrar al màxim possible les seues ferramentes, fan que tinguem bones sensacions respecte a la seua opinió.

Hem de valorar de manera diferent el mòdul de visualització basat en grafs ja que es tracta d'una primera aproximació a aquesta alternativa. Ha de quedar clar que el que intentàvem aconseguir amb aquesta fase del projecte era analitzar la potència d'aquest tipus de diagrama i plantejar-nos la seua integració amb la nostra aplicació, i amb aquest objectiu

en ment, estem molt satisfets amb els resultats obtinguts. Per un costat hem aconseguit una representació del nostre sistema, completament intuïtiva i molt pròxima a la naturalesa del problema plantejat, que amb la gran flexibilitat que la caracteritza ens permet visualitzar i analitzar la informació amb diferents graus de profunditat. Per altre costat, estem gratament sorpresos amb les possibilitats que ens ofereix la personalització d'estils de la ferramenta utilitzada ja que ens obri un ampli ventall de possibilitat a l'hora de representar característiques diverses de la nostra estructura de requisits. Un altre factor a destacar, és el bon rendiment de la llibreria a l'hora de carregar la informació. En el cas d'estudi analitzat, s'ha decidit optar per la representació de dos aplicacions reals de mitjà i gran tamany amb resultats parcialment satisfactoris en tractar-se d'un mòdul en la seua fase inicial.

Pensem sincerament que aquest tipus de representació pot ser molt útil a la llarga com a eina de suport per a la majoria de rols participants en el desenvolupament d'una aplicació software. És per això que el següent pas que es portarà a terme una vegada acabada aquesta tesis, serà integrar aquest mòdul de visualització amb la nostra ferramenta de suport a la metodologia TUNE-UP, tot intentant satisfer a la vegada les necessitats o exigències de cada un dels rols afectats. A més, com mostràvem en el capítol 8, el mòdul es centrava en la visualització completa de l'estructura de requisits d'un producte reflectint tots els tipus de components i relacions, però amb aquest tipus de diagrama, també podem representar diverses vistes més centrades en un context determinat, com per exemple les relacions que afecten a un determinat node o PA.

Com hem pogut vore en els casos d'estudi analitzats sobre el mòdul, aquest tipus de visualització també te les seues limitacions, centrades en aquest cas en la quantitat d'elements que volem representar i en el tamany de la superfície sobre la que plasmarem la representació. És per això, que deuríem centrar part dels nostres esforços futurs en l'anàlisi de diferents vistes o algun tipus de filtres automàtics, que ajuden els distints agents de la ferramenta a trobar la millor visualització possible enfront d'un problema concret. A més, cal destacar que amb l'ús d'aquest mòdul i l'emmagatzemament de l'estructura de dades en arxius graphML, se'ns obri una nova via d'anàlisi històrica de les dades amb ferramentes de navegació i filtrat que ens poden ajudar a estudiar l'evolució del nostre software.

Una de les futures vies d'investigació que hem definit gràcies a l'experiència extreta de la construcció del nostre mòdul, és la possibilitat de modificació i gestió de l'estructura

mitjançant el propi component de visualització. Aquesta propietat que ens aportava la llibreria utilitzada, pot simplificar la tasca dels analistes per tal d'evolucionar amb certa facilitat l'estructura jeràrquica dels requisits, i el que és més important, permetria una entrada directa de dades per tal que analistes i programadors puguen definir relacions de traçabilitat existents entre diferents elements tot i no estar marcades de manera automàtica pel nostre sistema. Esta millora, aconseguiria aportar més informació a l'anàlisi d'impacte de requisits que mitjançant l'experiència dels agents ens ajudaria a madurar la informació aconseguida.

Seguint amb aquesta mateixa idea, cal recordar que la visualització de requisits és una de les vies d'investigació que ens plantejàrem en finalitzar el meu projecte final de carrera, però junt a aquesta, encara queden algunes altres on caldria destacar la millora o el refinament de les condicions de traçabilitat entre artefactes, per tal d'aconseguir un anàlisi d'impacte realment útil. Una altra línia que hem deixat de costat en aquest projecte, per centrar-nos en la visualització basada en grafs, ha estat l'aplicació dels diagrames TreeMap. Tot i això, creiem que aquest tipus de representació també ens pot ajudar en la tasca de perfeccionar la ferramenta i és per això que esperem rebre les experiències extretes del projecte paral·lel del grup per tal d'aconseguir afegir un nou mòdul a la nostra aplicació.

Una altra part del projecte que ens ha donat moltes idees de cara al desenvolupament futur de la nostra metodologia, ha sigut la vista estadística del gestor de requisits. Ser conscients en tot moment de l'estat del nostre treball amb una sèrie d'escales numèriques, és una manera de mesurar el correcte ús que estem fent de les nostres eines. Encara així, la visualització actual de la informació és molt bàsica i rudimentària, pel que caldria definir millor les dades que volem mostrar, crear gràfiques i diagrames addicionals per al suport d'aquesta informació i inclús, profunditzar més en l'anàlisi dels requisits mitjançant l'ús de regles condicionals que generen avisos quan es produïsquen situacions anòmales.

Per últim, m'agradaria destacar que haver treballar durant els últims dos anys en aquest projecte de col·laboració entre empresa i universitat, m'ha donat l'experiència suficient per encarar la nova etapa que tinc al davant amb més confiança. A més, en centrar el meu treball en l'àmbit de l'enginyeria de requisits, he aconseguit una visió més àmplia sobre el desenvolupament software que em dona més recursos i ferramentes per tal d'enfrontar-me a projectes futurs.

Referències

1. Company, M. Tesis Final de Máster: Análisis de Impacto de Requisitos en un proceso de desarrollo centrado en Pruebas de Aceptación. Universitat Politècnica de València. 2011.
2. Medina, G. Projecte Final de Carrera: Explotació d'una base de dades de grafs per a l'anàlisi d'impacte en l'enginyeria de requisits. Universitat Politècnica de València. 2011.
3. 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 (JISBD 2010).
4. Company, M., Letelier, P., Marante, M., Suarez, F. Análisis de impacto en requisitos soportado de forma semi-automática y en un marco de desarrollo TDD basado en pruebas de aceptación. XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2011)
5. Marante, M., Company, M., Letelier, P. Seguimiento ágil de proyectos de desarrollo de software utilizando Gráficas Burn Down. XVI Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2011)
6. Marante, M., Letelier, P., Suarez, F. TUNE-UP: Seguimiento de proyectos software dirigido por la gestión de tiempos. XIV Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2009). (pág. 57- 68), Septiembre 2009.
7. Marante, M., Letelier, P., Suarez, F. TUNE-UP: Un enfoque pragmático para la planificación y seguimiento de proyectos de desarrollo y mantenimiento de software. I Congreso Iberoamericano SOCOTE – SOporte al COnocimiento con la TEcnología (SOCOTE 2009), Noviembre 2009.
8. Humphrey, Watts S. The Personal Software Process. Software Process Newsletter, Technical Council on Software Engineering, IEEE Computer Society, Volume 13, No. 1,, pp SPN 1-3. Sept. 1994.
9. Allen, D. Getting Things Done: The Art of Stress-Free Productivity. Published in Penguin Books 2003
10. Covey, S. The Seven Habits of Highly Effective People. Published Simon & Schuster, Limited. Edition Softcover. 9 . 1989.
11. Haugset, B., Hanssen, G., Automated Acceptance Testing: A Literature Review and an Industrial Case Study, pp. 27-38, Proc. of Agile 2008

12. Kroll P., Kruchten P., Booch G. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professional. 2003.
13. Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley. 2000.
14. Schwaber K., Beedle M. Agile Software Development with Scrum. Prentice Hall, Series in Agile Software Development. 2001
15. IEEE 830-1993, IEEE Recommended Practice for Software Requirements Specifications. 1993.
16. Kilpinen, M.S. The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis. University of Cambridge. Cambridge, UK. 2008.
17. 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).
18. Dick, J. , "Design Traceability", IEEE Software 22(6), pp.14-16. 2005.
19. Bohner, S.A. & Arnold, R.S. (eds), Software Change Impact Analysis, Los Alamitos, California, USA. IEEE Computer Society Press. 1996.
20. Ambler, S., Agile Modeling: Effective Practices for Extreme Unified Process, New York, New York, USA: John Wiley & Sons.2002.
21. Sánchez, C. Tesis Final de Màster: Estrategias para el mantenimiento de un producto software de gran envergadura. Universitat Politècnica de València. 2012.
22. Cooper Jr., J. Lee, S. Gandhi, R. and Gotel, O. Requirements Engineering Visualization: A survey on the State-of-the-Art. REV '09. 2009.
23. Gotel, O., Marchese, F., Morris, S., On Requirements Visualization, *Requirements Engineering Visualization 2007 (REV 2007),* 2007.
24. Naslia, N. And Hashim, K.Requirements Visualization Techiques: A Comparative Analysis. International Conference on Applied Computer Science (ACS'08). 2008.
25. Sud, R. and Arthur, J. Requiremens Management Tools. A qualitative assesment. 2003
26. Alexander, I. Visualizing requirements in UML. Telelogic White Paper. 2001
27. Dulac, N. Viguier, T. Leveson, N. Storey, M. On the Use of Visualization in Formal Requirements Specification. 2002
28. Cockburn, A. Writing Effective Use Cases. Pàgina 233. Addison-Wesley Longman, 2000.

29. Li, Y. Discovering Structure of Data to Create Multiple Perspective Visualization. Master of Engineering Thesis at MIT 2004.
30. Auber, D. Tulip – A huge graph visualization framework. LaBRI-Université Bordeaux. 2004.
31. Bastian, M. Sebastian, H. Gephi: An Open Source Software for Exploring and Manipulating Networks. 2009.
32. Herman, I. Melançon, G. and Scott, M. Graph Visualization and Navigation in Information Visualization: Survey. 2000.
33. Documentation of yFiles.NET component. http://docs.yworks.com/yfilesdotnet
34. The Graph File Format WebPage. http://graphml.graphdrawing.org
35. Vliegen, R. Van der Linden, E. and Van Wijk, J. Visualizing Business Data with Generalized Treemaps. February 24, 2010.
36. Shneiderman, B. Plaisant, C. Treemaps for space-constrained visualization of hierarchies. June 25, 2009.
37. Sarkar, M. and Brown, M. Graphical Fish–eye views of graphs, Human Factors in Computing Systems, CHI ’92 Conference Proceedings, ACM Press, pp. 83–91, 1992.
38. Holten, D.Van Wijk, J. Force-Directed Edge Bundling for Graph Visualization. Eurographics/ IEEE-VGTC Symposium on Visualization, 2009