ASSISTENT PER LA PREPARACIÓ DE DISTRIBUCIONS

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per

Josep Costa Sanmartí

i dirigit per

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, *Juny* de *2011*

El sotasignat, ***Jordi Pons Aróztegui***,

professor de ***l'Escola d'Enginyeria de la UAB***,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per

Josep Costa Sanmartí

I per a que consti firma la present. Sabadell, ***Juny*** de ***2011***

Signat: ***Jordi Pons Aróztegui***

El sotasignat, ***Xavier Urtasun Villanueva de Unit4,***

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva supervisió per

Josep Costa Sanmartí

I per a que consti firma la present. Sabadell, ***Juny*** de ***2011***

Signat: ***Xavier Urtasun Villanueva***

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte:

Assistent per la preparació de distribucions

Autor: *Josep Costa Sanmartí*	**Data:** *Juny 2011*

Tutor: *Jordi Pons Aróztegui*

Titulació: *Enginyeria tècnica en Informàtica de Sistemes*

Paraules clau (mínim 3)

- Català: distribucions, automatització, paquets, instal·lacions.
- Castellà: distribuciones, automatización, paquetes, instalaciones.
- Anglès: distributions, automations, packages, facilities.

Resum del projecte (extensió màxima 100 paraules)

- ***Català:***

  Aquest document és la memòria del projecte de final de carrera realitzat en un conveni entre l'empresa Unit4 i la UAB. Unit4, es dedica a oferir solucions ERP per la gestió global de les empreses i serveis associats.

  El concepte de distribució dins de l'empresa, es tradueix al paquet d'objectes (taules, formularis, classes, etc...), que s'entreguen al client i que solen contenir o bé versions inicials o bé algunes actualitzacions de versions ja existents. Aquest sistema, és utilitzats per tots els sectors de l'empresa i la seva realització actual és completament manual. El projecte es centrarà en realitzar un conjunt d'eines per facilitar les diferents tasques implicades en la generació de distribucions.

- ***Castellà:***

  Este documento, es la memoria del proyecto de final de carrera realizado en convenio entre la empresa Unit4 y la UAB. Unit4, se dedica a ofrecer soluciones ERP para la gestión global de las empresas y servicios asociados.

  El concepto de distribución dentro de la empresa, se traduce al paquete de objetos (tablas, formularios, clases, etc...), que se entregan al cliente y que suelen contener, o bien versiones iniciales o bien actualizaciones de versiones ya existentes. Este sistema, es utilizado en todos los sectores de la empresa y su realización actual es completamente manual. El proyecto se centrará en realizar un conjunto de herramientas para facilitar las distintas tascas implicadas en la generación de distribuciones.

- ***Anglès:***

  This document, you will write the memory of the final project draft of an agreement made between the company Unit4 and UAB. Unit4 is dedicated to providing ERP solutions for the global management of the companies and associated services.

  The concept of distribution within the company means the package of objects (tables, forms, classes, etc…), which are delivered to customers and usually contains the original versions or updates of existing versions. This system is used in all sectors of the company, and this actual implementation is completely manual. The project will focus on making a set of tools to facilitate the different functions involved in the generation of distributions.

Taula de continguts

Índex de figures

Índex de taules

CAPÍTOL 1: INTRODUCCIÓ

Es definirà el concepte de distribució de paquets i el motiu pel qual l'empresa en necessita automatitzar el seu procés. Parlarem també, de l'empresa en la qual realitzem el projecte, comentant-ne les motivacions que se'ns presenten així com els objectius laborals i acadèmics. Finalment, farem una breu descripció de l'estructura que tindrà aquesta memòria.

1.1. Distribucions, estat actual

El projecte s'ha realitzat en la filial espanyola de l'empresa Unit4 situada a Barberà del Vallès, en un conveni entre la companyia i la UAB que té una durada d'unes 560 hores.

El projecte es titula: "Assistent per la preparació de distribucions", i consisteix en automatitzar el procés de preparació per la generació de paquets que s'han de distribuir als clients. Aquest sistema és utilitzat en tots els departaments de l'empresa; i actualment, no existeix cap aplicació que englobi les diferents tasques que es realitzen en el procés de distribució, fet que comporta un gran cost en quan a temps.

Abans de continuar, cal parar-nos a fer una pregunta: ***què és una distribució?***

Una distribució és el paquet d'un producte en concret, com per exemple el de finances, que conté tot el necessari per què es pugui executar de forma automàtica en el client final. Aquests paquets contenen components (conjunt de taules, de formularis, de classes...) i processos (necessaris per la instal·lació de la distribució). Finalment, esmentar que les distribucions poden ser, o bé versions noves o bé actualitzacions de versions ja existents.

El projecte el centrarem en realitzar un conjunt d'eines per facilitar les diferents tasques implicades en la generació de components de la distribució, tenint en compte en tot moment, que cada usuari l'utilitzarà de manera diferent segons el departament.

La idea principal és crear una independència total entre les eines, però que treballin dins d'un entorn comú, on tindrem definides unes variables globals. D'aquesta manera aconseguirem una aplicació molt manejable durant i després de la realització del projecte.

1.2. L'empresa: Unit4

Unit4 Ibèrica (Euronext: U4AGR) és la filial espanyola, de la multinacional d'origen holandès Unit4, del sector de les Tecnologies de la Informació que desenvolupa i comercialitza software empresarial, i ofereix serveis professionals que ajuden a les organitzacions dinàmiques a gestionar els seus negocis i adaptar-se al canvi amb facilitat, rapidesa i de manera rentable.

Unit4 Ibèrica disposa d'oficines en 17 països europeus, així com en altres 7 països repartits per Nord Amèrica, Asia, el Pacífic i Àfrica i activitat comercial en altres territoris. La seva seu central es troba a Sliedrecht (Països Baixos).

El nostre projecte es desenvoluparà en la seu que la companyia té a Barcelona, ubicada a Barberà del Vallès i en col·laboració amb l'equip de desenvolupament.

1.3. Motivacions

Sempre he pensat que una enginyeria ha de tenir el que s'anomena "pràctiques en empresa", ja sigui com a conseqüència d'una assignatura o per tal de realitzar el projecte de final de carrera. Així que poder fer el projecte en un conveni amb una empresa, em va semblar una oportunitat perfecte per diferents aspectes:

- Et permet endinsar-te en el món laboral i observar el funcionament d'una gran multinacional del sector.
- És una manera d'obtenir experiència, molt útil en el moment de sortir al món laboral.
- Personalment, em permet aprofitar l'assignatura del projecte per fer-ho de la forma que crec més encertada, a la vegada que puc acabar una petita part de la carrera.

1.4. Estructura de la memòria

La memòria del projecte, es divideix en 6 capítols, descrits a continuació:

- **Capítol 1:**

  En aquest capítol definim el projecte d'una manera global. En primer lloc parlem d'en què consisteix i presentem l'empresa on l'hem realitzat. Després, expliquem les motivacions per desenvolupar aquest projecte d'una manera satisfactòria.

- **Capítol 2:**

  En aquest capítol veurem l'estudi de viabilitat, que constarà de les següents parts: introducció, lògica del sistema, objectius del projecte, requisits funcionals i els no funcionals. A continuació, tindrem un apartat amb algunes alternatives que proposem, i per acabar, farem una conclusió de l'estudi de viabilitat.

- **Capítol 3:**

  En aquest capítol generem el pla de projecte, que constarà: de WBS on definirem les fases del projecte (des de l'assignació del projecte fins a la seva defensa), seguit d'un anàlisi complet dels recursos i del calendari per dur-lo a terme en les dates establertes. El següent pas, serà fer una avaluació dels riscos que comporta la realització d'un projecte acadèmic i la seva posterior catalogació. Finalment realitzarem el pressupost i una conclusió del pla de projecte.

- **Capítol 4:**

  En aquest capítol analitzarem el procés a realitzar per tal de fer-ne un disseny complet i desglossar el "problema" que ens ocupa en quelcom més senzill a tractar. El seu resultat seran les eines o processos de distribució que se'n derivaran i el seu funcionament entre si.

- **Capítol 5:**

  En el capítol 5, explicarem l'entorn de desenvolupament utilitzat així com les diferents eines aplicades en el procés d'implementació, com són: la pròpia plataforma Java, els fitxer XML, Command Line i l'Apache ANT.

- **Capítol 6:**

  En aquest apartat observarem l'estil de codificació utilitzat així com les diferents proves, unitàries, d'estrès i d'altres, fetes per assegurar el correcte funcionament de l'aplicació.

- **Capítol 7:**

  Finalment, en aquest darrer capítol, detallarem aquells objectius que s'han superat en contra dels no aconseguits, així com algunes possibles millores que podríem arribar a implementar en un futur. Per acabar, veurem un petit gràfic amb les desviacions que s'han produït segons les hores estimades des de bon principi; i una valoració personal.

- **Annex A .- Glossari:**

  En aquest annex, detallarem aquells conceptes esmentats durant la memòria del projecte per a que quedin clars i es pugui comprendre la totalitat del document.

- **Annex B .- Contingut del CD:**

  En aquest darrer annex, inclourem aquells elements que es troben en el CD que s'entrega conjuntament amb la memòria del projecte.

  A part de la memòria del projecte, s'ha generat un manual d'usuari per l'empresa on s'especifica com utilitzar la nostra aplicació, així com cada una de les tasques creades.

CAPÍTOL 2: ESTUDI DE VIABILITAT

L'estudi de viabilitat és un document que proposa solucions de diferents tipus per tal de resoldre el problema que s'analitza en el nostre projecte. A través del mateix, podrem avaluar les garanties de dur a terme el projecte.

El seu objectiu, és fer una avaluació dels beneficis i costos sobre diferents àmbits com: l'econòmic, el tècnic, el legal i l'operatiu, per tal de determinar la viabilitat del projecte, si es pot dur a terme o no.

2.1. Estudi de la situació actual

Es descriurà la situació actual del sistema que ens ocupa i identificarem els usuaris i/o personal del sistema. Aquesta informació ens permetrà fer un diagnòstic per tal de determinar-ne els objectius del projecte amb els seus respectius requisits funcionals i no funcionals. Finalment, observarem el mercat per intentar trobar alternatives i determinar quina és la millor solució.

2.1.1. Context

Actualment, realitzar una distribució dins l'empresa és un procés molt elaborat on l'encarregat ha de realitzar les diferents parts del procés de forma manual. Inicialment, s'ha de fer l'exportació de tots els elements –des d'una taula, fins als formularis o llistes, passant per les classes Java- que el client necessitarà per fer funcionar en un futur la seva aplicació d'ekon. Un cop hem realitzat les exportacions, s'obrirà l'aplicació anomenada "karat distributions generator" on l'hi indicarem components, regles i informació general també de forma manual per acabar generant la distribució.

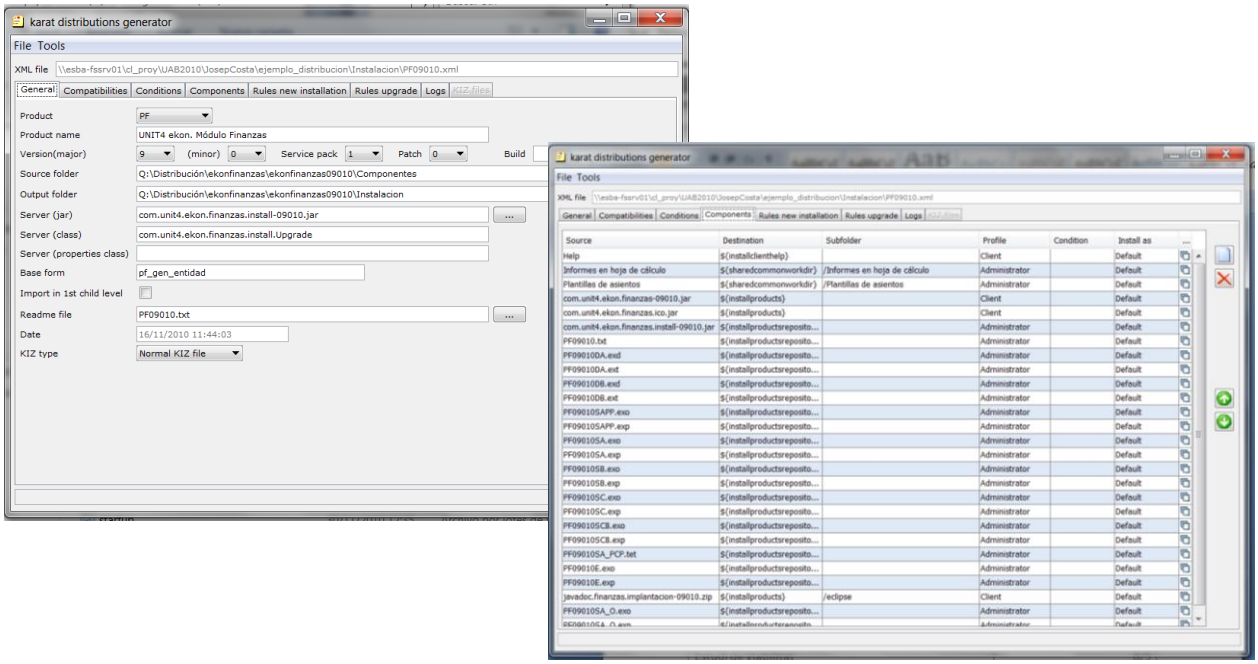


Figura 1 .- "karat distributions generator"

2.1.2. Lògica del sistema

En el gràfic següent, s'observa la lògica del sistema actual on es reflecteix tot el procés que es du a terme per generar una distribució.

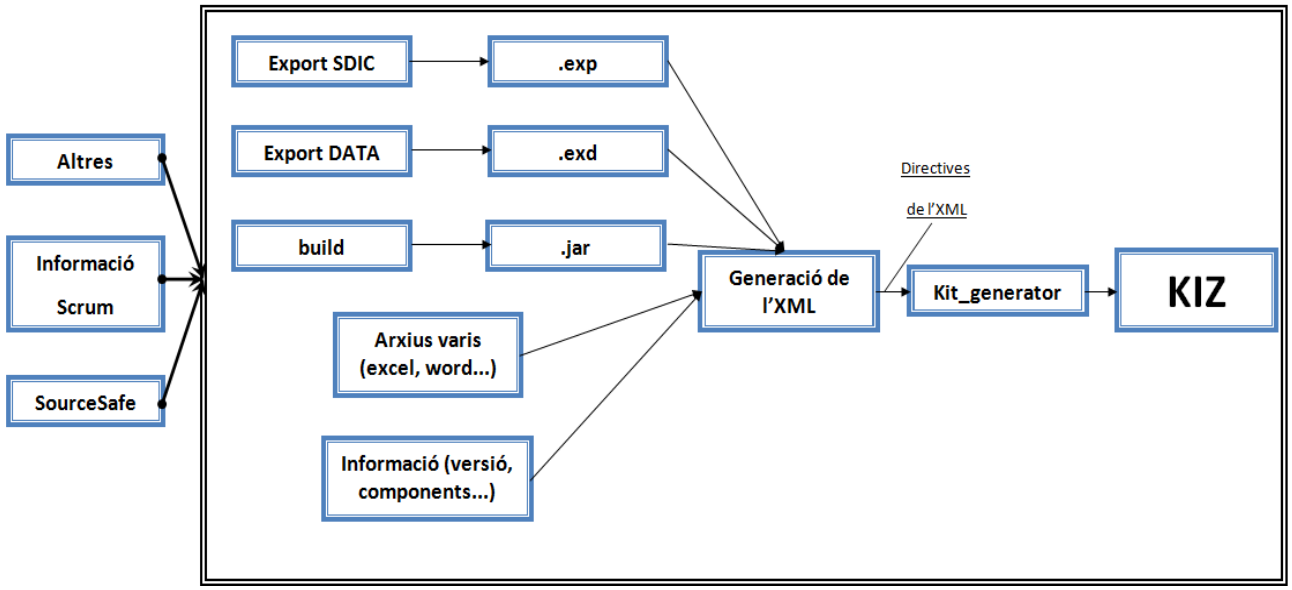


Figura 2 .- Diagrama de la lògica del sistema actual

De forma global, podem veure que el procés de distribució (enquadrat) està alimentat per un conjunt de factors (altres, informació Scrum i SourceSafe), dels quals se'n obté la informació sobre quins elements s'han de distribuir.

A partir de la naturalesa d'aquests elements, començarà el procés de distribució en si, que consta de:

- Exportació SDIC i DATA a través del Karat Studio.
- Generació de JARS de classes a partir d'una altra aplicació.

Un cop obtinguts aquests components, que són la unitat bàsica d'una distribució, s'obre l'aplicació "karat distributions generator" que serà l'encarregada de generar el KIZ després d'haver informat tots els aspectes rellevants, per obtenir la distribució i l'XML.

Solució proposada:

L'aplicació constarà d'un conjunt d'eines independents entre si però que treballaran en un mateix entorn comú on estaran definides les variables (directoris, dades de connexió a la base de dades, data de control...) necessàries per realitzar el procés de preparació d'una distribució

amb èxit. Aquesta llibertat entre les eines, permetrà una gran flexibilitat a l'hora de modificar, afegir o eliminar alguna d'elles en un futur.

L'execució controlada d'aquestes eines permetrà generar els components necessaris per realitzar una distribució, a partir de pocs elements indicats o generar parts d'elles de forma individual.

2.1.3. Usuaris i/o personal del sistema

Un cop, el sistema estigui completament finalitzat, el sistema només serà utilitzat per un únic grup d'usuaris:

Descripció	Responsabilitat
Generador de distribucions	La seva responsabilitat serà la de generar components de les distribucions, a través dels processos generats en l'aplicació.

Taula 1 .- Usuaris del sistema amb la seva responsabilitat.

2.1.4. Diagnòstic del sistema

A l'hora de fer un diagnòstic sobre un sistema en concret, hem de valorar-ne en primer moment les deficiències i després les millores.

Les deficiències que presenta el sistema actual són:

- El sistema és completament manual.
- L'usuari ha d'utilitzar eines diferents per generar una distribució.
- El temps dedicat a les distribucions és elevat pel fet d'haver de seleccionar manualment els objectes i les classes a exportar.
- El fet de fer-se de forma manual, implica una elevada possibilitat de errors.

Després de veure el seguit de deficiències que té l'actual sistema, anem a enumerar les millores que podem proporcionar:

- ✓ Englobar tot el procés en una sola eina.
- ✓ Dotar aquesta eina de diferents tipus d'explotacions com podrien ser, una aplicació simple i funcional o una tipus comanda (.bat). En cas de que hi hagués temps i fos factible, també una a través de l'ANT.

- ✓ Poder fer una distribució indicant només els paràmetres globals.
- ✓ Fer passos de la distribució de forma separada.
- ✓ Disminuir el temps i els errors en el procés de distribució.

2.2. Objectius del projecte

Els objectius d'aquest projecte són:

1. Tenir una eina que permeti a l'empresa Unit4 ajudar a generar una distribució de forma automàtica indicant pocs paràmetres.
2. Poder realitzar parts del procés de preparació d'una distribució separadament.
3. Proporcionar a l'eina diferents maneres de interactuar amb ella.
4. Disminuir el temps que es dedica a preparar cadascuna de les distribucions.
5. Disminuir la gran dependència que té l'eina actual.
6. Poder aconseguir una única eina que sigui utilitzada per tota l'empresa.
7. Definir un estàndard de seguiment del procés de distribució.
8. Realitzar informes sobre les distribucions realitzades.
9. Disminució de les errades humanes.

2.2.1. Catalogació dels objectius

Un cop esmentats els objectius del projecte, procedirem a catalogar-los segons la seva prioritat:

Objectiu	Crític	Prioritat	Secundari
O1	**X**		
O2	**X**		
O3			**X**
O4	**X**		
O5	**X**		
O6		**X**	

O7		**X**	
O8	**X**		
O9	**X**		

Taula 2 .- Catalogació dels objectius (grau d'importància)

2.3. Requisits del projecte

L'especificació de requisits és una descripció completa del comportament del sistema que es desenvoluparà. Aquests requisits els podem dividir en funcionals i no funcionals.

2.3.1. Requisits funcionals

Un requisit funcional és aquell que defineix el comportament intern del software. Tot projecte, ha de tenir una llista de requeriments funcionals, i la d'aquest projecte és:

- RF1: distribucions diàries indicant pocs paràmetres.
- RF2: realitzar individualment alguns processos de la distribució.
- RF3: generació dels elements necessaris en una distribució (.KIZ i XML).
- RF4: generació d'un LOG per controlar quines parts del procés s'han superat de forma satisfactòria.
- RF5: realització d'una aplicació senzilla i eficient per fer les distribucions.
- RF6: realimentació de variables.
- RF7: configuracions XML personalitzades segons els usuaris.
- RF8: interfície per línea de comandes (Wrapper Command Line).

2.3.2. Requisits no funcionals

Les restriccions del sistema ens determinen punts que s'han de tenir en compte a l'hora de realitzar el projecte però que no afecten a les funcionalitats que tindrà aquest. En l'entorn web les restriccions del sistema són les següents:

- RFN1: l'aplicació haurà de seguir les normatives marcades per l'empresa.
- RFN2: els recursos utilitzats per l'aplicació hauran d'anar a mida de l'entitat.
- RFN3: l'eina ha de ser extensible a totes les àrees on s'han de realitzar distribucions dins

l'empresa.

- RFN4: ha de ser un projecte perfectament documentat.
- RFN5: cada un dels mètodes generats per realitzar una distribució ha de ser completament independent dels altres.

2.3.3. Restriccions del sistema

Quan parlem de restriccions del sistema, ens referim a aquells punts que hem de tenir en compte a l'hora de realitzar el projecte però que de cap manera afectaran a les funcionalitats que tindrà el nostre sistema. Les restriccions sobre el sistema seran:

- El projecte ha d'estar finalitzat com a màxim el 28 de juny de 2011.
- L'aplicació s'ha de desenvolupar íntegrament en la plataforma Java.
- El projecte ha de tenir una duració d'unes 560 hores realitzades dins de l'empresa.

2.3.4. Catalogació i priorització dels requisits

En aquest apartat, catalogarem i prioritzarem els diferents requisits enumerats amb anterioritat.

Tipus	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RNF1	RNF2	RNF3	RNF4	RNF5
Essencial	X	X	X		X	X	X		X	X	X	X	X
Condicional				X									
Opcional								X					

Taula 3 .- Catalogació dels requeriments funcionals i no funcionals (grau d'importància)

A continuació, relacionarem els objectius i els requisits.

Objectius	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RNF1	RNF2	RNF3	RNF4	RNF5
Objectiu 1	X		X	X					X	X	X	X	X
Objectiu 2		X		X					X	X	X	X	X
Objectiu 3					X			X			X		
Objectiu 4					X	X	X	X		X			X

Objectiu 5					X	X	X	X		X			X
Objectiu 6					X	X	X		X	X	X		
Objectiu 7						X	X		X		X		X
Objectiu 8				X					X		X		
Objectiu 9					X	X	X	X		X			X

Taula 4 .- Relació entre els objectius i els requeriments funcionals i no funcionals

2.4. Alternativa i solució proposada

Al tractar-se d'una aplicació utilitzada internament per l'empresa, el número d'alternatives queda molt reduït. És a dir, no hi ha la possibilitat de buscar productes externs que puguin aconseguir les funcionalitats que nosaltres busquem, sinó que la única opció és generar nosaltres mateixos l'aplicació.

On si que podem trobar un punt de discrepància, és en la manera com desenvolupar l'aplicació. I les opcions que tenim són:

- **Realitzar una aplicació amb la plataforma Java desenvolupada amb Java Swing, completament independent de l'entorn Karat.**
- Realitzar una aplicació que es pugui incloure en l'entorn de Karat.

Finalment, vam decantar-nos per la primera opció pel fet que crèiem que seria molt més senzill accedir tan a l'XML per realitzar la interfície d'usuari com el fet d'integrar l'aplicació en qualsevol màquina client.

2.5. Conclusions

Com a conclusió a l'apartat d'Estudi de Viabilitat, hem de tenir en compte dos aspectes importantíssims. Per una banda, els beneficis que esperem treure'n amb el desenvolupament d'aquest projecte, i per últim els inconvenients que se'n deriven.

- Beneficis:
    - Realització d'una eina conjunta que pugui ser utilitzada per la preparació de distribucions.
    - Diferents maneres per explotar l'eina.
    - Tenir una eina unificada que realitzi tots els processos de les distribucions.
    - Disminució del temps i de l'error humà a l'hora de fer un KIZ.
    - Possibilitat de modificar l'eina en un futur.

- Inconvenients:
    - Recel per part dels usuaris, per la costum d'utilitzar el procés antic.
    - Discrepàncies de l'aplicació pel fet de que segons l'usuari, la manera de fer distribucions varia.

Un cop observats les avantatges i els inconvenients, i tenint en compte els riscos que es poden ocasionar, podem dir que aquest projecte és **VIABLE**.

CAPÍTOL 3: PLA DE PROJECTE

El pla de projecte és un document que recull el conjunt d'activitats que permeten desenvolupar, executar i controlar el nostre projecte. El document inclou les tasques i els punts de control del projecte, els recursos del mateix, el seu calendari, l'avaluació de riscos i el pressupost del projecte. Per realitzar-lo hem utilitzat una metodologia lineal seguint els processos descrits en el PMBOK (2008).

3.1. WBS (Work Breakdown Structure)

En aquest apartat llistarem totes les fases del projecte, les seves activitats relacionades i més concretament les tasques del projecte.

3.1.1. Fases i activitats del projecte

Fases	Descripció
Iniciació	La fase d'iniciació inclou les activitats de definició del projecte, assignació i matriculació.
Planificació	És l'apartat que inclou l'Estudi de Viabilitat i el Pla de Projecte.
Anàlisi	Anàlisi de requisits funcionals i no funcionals. L'arquitectura del sistema.
Disseny	Inclou el disseny de la capa de dades, de control i de la interfície. També es farà un disseny dels tests.
Desenvolupament	Fase de desenvolupament de l'aplicació.
Test i proves	Fase de prova del sistema. Tests unitaris i d'integració.
Implantació	L'aplicació s'instal·la en el seu entorn real. Inclou la formació d'usuaris.
Generació de documents	Fase de generació de documents del projecte. Inclou manuals i memòria del projecte.
Tancament del projecte	Fase de tancament. El director del projecte signa l'acceptació i tancament del projecte.
Defensa del projecte	Defensa del projecte davant la comissió.

Taula 5 .- Fases i activitats del projecte (nom i descripció)

3.1.2. Diagrama WBS

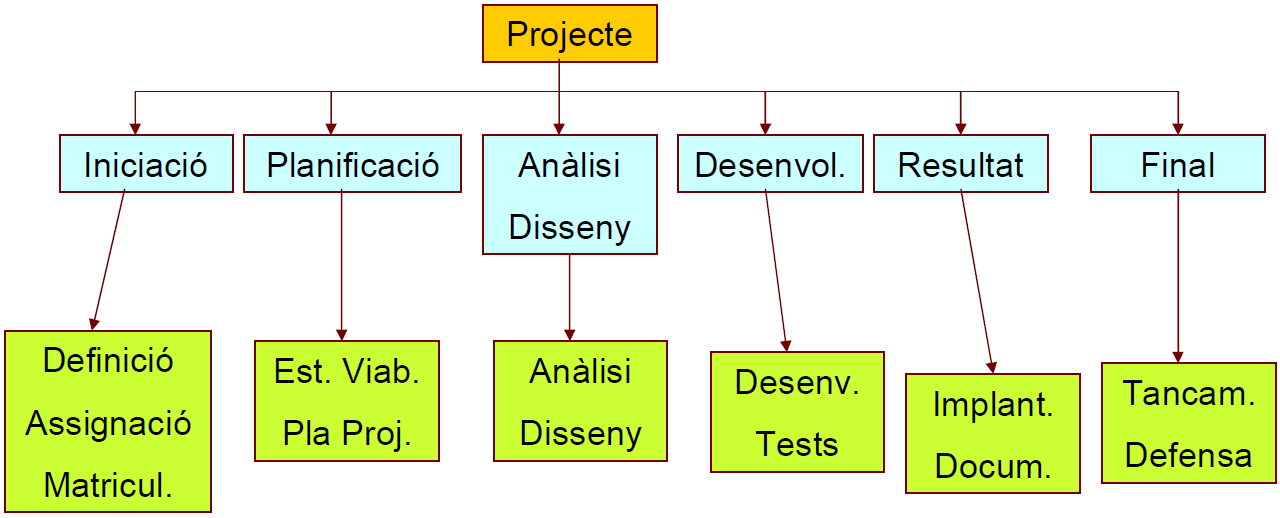


Figura 3 .- Diagrama WBS del projecte

3.1.3. Milestone

Veurem una breu descripció en forma de taula, en la que relacionarem tots els punts de control del projecte segons de quin tipus són, i la data que prevista de finalització.

Nom	Descripció	Data
Iniciació	Matriculació	15/10/2010
Estudi de Viabilitat	Aprovació	22/11/2010
Pla de Projecte	Aprovació	26/11/2010
Anàlisi	Aprovació	02/12/2010
Disseny	Aprovació	14/12/2010
Tancament	Acceptació	24/06/2011
Defensa	Avaluació	08/07/2011

Taula 6 .- Milestone. Fases del projecte i la seva ata de finalització

3.2. Anàlisi de recursos

3.2.1. Recursos del projecte

Quan parlem de recursos d'un projecte, ens referim en una totalitat, als recursos humans necessaris per dur a terme un projecte en l'àmbit del desenvolupament de software. En el nostre cas, els recursos humans són: el cap de projecte, l'analista, el tècnic de proves i el programador; i els seus costos associats són:

Recursos humans	Cost
Director de projecte	54,00 €/hora
Cap de projecte	36,00 €/hora
Analista	24,00 €/hora
Programador	15,00 €/hora
Tècnic de proves	15,00 €/hora

Taula 7 .- Recursos del projecte amb el seu cost per hora

Cal recordar que al tractar-se d'un projecte acadèmic, les figures de l'analista, el programador i el tècnic de proves estaran realitzades pel propi estudiant. Tot i que sempre, amb un control dins l'empresa.

Per altre banda, les figures del director i cap de projecte les realitzarà el tutor del projecte, designat per l'empresa.

Aquests recursos esmentats anteriorment, no participaran en totes les tasques, sinó que treballaran en unes en concret:

a) **Director de projecte (DP):** participarà en l'inici del projecte, la planificació, en totes les aprovacions i en la generació de documents, tancament i defensa del projecte.
b) **Cap de projecte (CP):** participarà en les mateixes fases que el director de projecte però d'una forma molt més activa.
c) **Analista (A):** participarà en l'anàlisi i disseny de l'aplicació d'una forma activa; i també en la d'implementació.
d) **Programador (P):** participarà activament en el desenvolupament de l'aplicació i en bona part de tot el tema d'implementació.

e) **Tècnic de proves (TP):** la seva participació està enfocada en una de les tasques finals com és la de test i proves.

3.3 Calendari del projecte

Com ja hem dit anteriorment, es tracta d'un projecte desenvolupat en un conveni amb l'empresa Unit4 i que estarà comprès entre el novembre de 2010 i el juny de 2011. On realitzarem un horari de 4 hores diàries.

Per poder fer una correcta planificació del projecte amb el seu conjunt de tasques i subtasques, hem utilitzat l'eina Microsoft Project 2010 pel seguiment i control del projecte.

3.3.1. Dependències

En tot el procés s'utilitzarà el **model lineal.** Això significa que cada fase no comença fins que no s'ha completat la fase anterior. En la fase de desenvolupament es preveu un model àgil de tal manera que el disseny, el desenvolupament i el test segueixin un model iteratiu, dividit per cada procés de distribució.

3.3.2. Quadre de tasques del projecte

Nº	Nom de la tasca	Inici	Final	Recursos	Pr.
1	**Assistent per la preparació de distribucions**	**18/10/10**	**28/06/11**		
2	Inici del projecte: assignació, matriculació i formació	18/10/10	12/11/10	DP[15%]; CP	
3	**Planificació**	**15/11/10**	**26/11/10**		**2**
4	Estudi de Viabilitat	15/11/10	18/11/10	CP	2
5	Aprovació de l'Estudi de Viabilitat	22/11/10	22/11/10	CP	4
6	Pla de Projecte	22/11/10	26/11/10	CP	5
7	Aprovació del Pla de Projecte	26/11/10	26/11/10	CP	6

8	**Anàlisi de l'aplicació**	**26/11/10**	**02/12/10**		**3**
9	Anàlisi de requisits (cassos d'ús)	26/11/10	29/11/10	A	7
10	Anàlisi de dades (base de dades)	30/11/10	01/12/10	A	9
11	Anàlisi de seguretat i legalitat	01/12/10	01/12/10	A	10
12	Documentació de l'anàlisi	02/12/10	02/12/10	A	11
13	Aprovació de l'anàlisi (Punt de Control)	02/12/10	02/12/10	CP; A[50%]; DP[10%]	12
14	**Disseny de l'aplicació**	**03/12/10**	**14/12/10**		**8**
15	Disseny de la base de dades	03/12/10	06/12/10	A[80%]; P[20%]	13
16	Disseny modular de l'aplicació	06/12/10	08/12/10	A[80%]; P[20%]	15
17	Disseny de l'interfície, ajuda en línea	08/12/10	09/12/10	A[80%]; P[20%]	16
18	Disseny de proves (test)	09/12/10	10/12/10	A[80%]; P[20%]; TP[20%]	17
19	Documentació del disseny	10/12/10	13/12/10	A; P; TP	18
20	Aprovació del disseny	14/12/10	14/12/10	CP; DP[10%]	19
21	**Aplicació gràfica**	**14/12/10**	**07/02/11**		**14**
22	Disseny de la interfície	14/12/10	16/12/10	A; P	20
23	Desenvolupament de la interfície	16/12/10	31/12/10	P	22
24	Proves unitàries	04/01/11	04/01/11	TP	23
25	Documentació	07/02/11	07/02/11	CP; DP[10%]	24
26	**CoreAPI Principal**	**08/02/11**	**29/04/11**		**21**
27	Tasca sense UI	08/02/11	10/02/11	P[75%]; TP[25%]	25
31	Command Line	10/02/11	25/02/11	P[75%]; TP[25%]	27
35	Crear Directoris	25/02/11	01/03/11	P[75%]; TP[25%]	31
39	Copiar Fitxers	01/03/11	04/03/11	P[75%]; TP[25%]	35

43	Generar plantilles a partir de taules d'informació	04/03/11	17/03/11	P[75%]; TP[25%]	39
47	Exportació SDIC	17/03/11	21/03/11	P[75%]; TP[25%]	43
51	Exportació DATA	21/03/11	23/03/11	P[75%]; TP[25%]	47
55	Generar JAR	23/03/11	31/03/11	P[75%]; TP[25%]	51
59	Generar JAVADOC	31/03/11	08/04/11	P[75%]; TP[25%]	55
63	Generació de plantilles a part de filtres	11/04/11	20/04/11	P[75%]; TP[25%]	59
67	Comparació d'arxius	20/04/11	22/04/11	P[75%]; TP[25%]	63
71	Opcions XML	25/04/11	29/04/11	P[75%]; TP[25%]	67
75	**Wrapper línea de comandes**	**29/04/11**	**19/05/11**	**P[75%]; TP[25%]**	**26**
79	**Test i proves**	**20/05/11**	**09/06/11**		**75**
80	Proves d'integració	20/05/11	07/06/11	TP	78
81	Documentació del desenvolupament i test	07/06/11	08/06/11	TP	80
82	Aprovació del desenvolupament i proves (Punt de Control)	08/06/11	09/06/11	CP; DP[10%]	81
83	**Generació de documents (memòria del projecte)**	**09/06/11**	**23/06/11**	**CP; DP[10%]**	**79**
84	**Tancament del projecte**	**23/06/11**	**24/06/11**	**CP; DP[10%]**	**83**
85	**Defensa del projecte**	**24/06/11**	**28/06/11**	**CP**	**84**

Taula 8 .- Quadre de tasques amb la seva data d'inici i finalització, les dependències i els recursos.

3.3.3. Calendari temporal

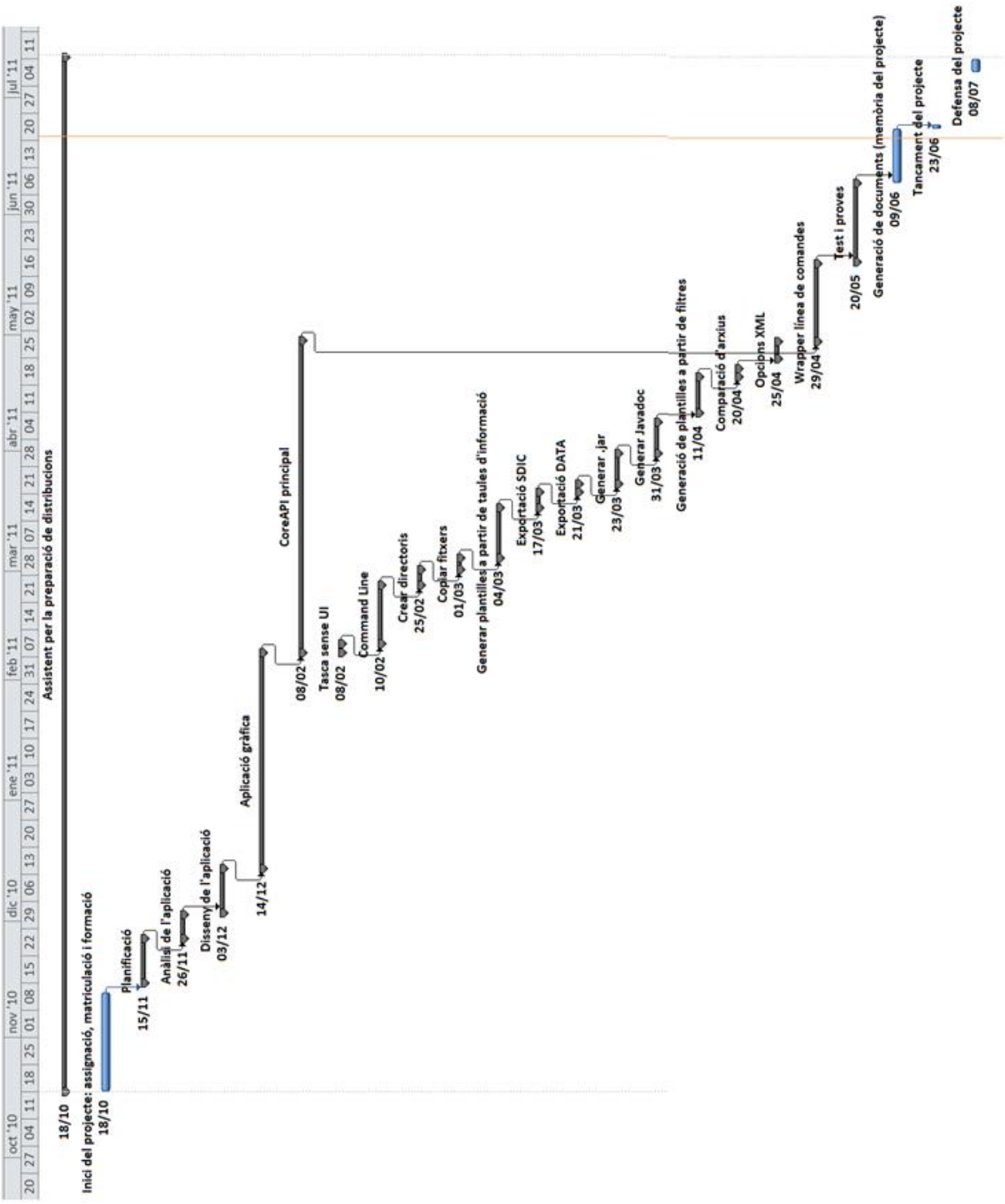


Figura 4 .- Diagrama de Gantt del projecte

3.4. Avaluació de riscos

Qualsevol projecte té una sèrie de riscos associats. Això significa, que poden succeir un seguit de problemes que duguin a un endarreriment de la data d'entrega del nostre projecte.

3.4.1. Llista de riscos

A continuació presentem els principals riscos que pot comportar el desenvolupament del projecte. En la següent taula determinarem en quina fase es poden produir els errors i quines conseqüències se'n derivarien:

Rec.	Títol	Localització	Conseqüències
R1	Planificació temporal optimitza	Pla de projecte	No s'acaba en la data prevista, augmenten els recursos.
R2	Manca d'alguna tasca	Pla de projecte	No es compleixen els objectius del projecte.
R3	Canvi de requisits	Estudi de viabilitat Anàlisi	Endarreriment en el desenvolupament i resultat.
R4	Equip del projecte massa reduït	Pla de projecte	Endarreriment en la finalització del projecte, no es compleixen els objectius del projecte.
R5	No es fa correctament la fase de test	Desenvolupament Implantació	Manca de qualitat, deficiències en l'operativitat, insatisfacció usuaris, pèrdua econòmica.
R6	Abandonament del projecte abans de la finalització	Qualsevol fase	Pèrdues econòmiques, frustració.

Taula 9 .- Llistat de riscos i les conseqüències que se'n deriven

3.4.2. Catalogació de riscos

La catalogació dels riscos esmentats en l'apartat anterior, segons la seva probabilitat d'aparèixer i la gravetat del seu impacte es veuen en la taula número 10.

Riscos	Probabilitat	Impacte
Risc 1	Alta	Crític
Risc 2	Alta	Crític
Risc 3	Alta	Marginal
Risc 4	Alta	Crític
Risc 5	Alta	Crític
Risc 6	Baixa	Catastròfic

Taula 10 .- Catalogació dels riscos

3.4.3. Pla de contingència

Un cop analitzada la taula anterior, hem de mirar les solucions que hauríem d'adoptar en cas de que es produís algun dels riscos indicats. Aquestes solucions són les següents:

Riscos	Solució que s'ha d'adoptar
Risc 1	Ajornar alguna funcionalitat, afrontar possibles pèrdues, fer una assegurança.
Risc 2	Revisar el Pla de projecte i modificar-ne la planificació.
Risc 3	Renegociar amb el client, ajornar funcionalitat, modificar la planificació i pressupost.
Risc 4	Demanar un ajornament, negociar amb el client, afrontar pèrdues.
Risc 5	Dissenyar el test amb antelació, realitzar tests automàtics, negociar contracte de manteniment, donar garanties, afrontar pèrdues econòmiques.
Risc 6	No té solució.

Taula 11 .- Solucions ha adoptar segons els riscos produïts

3.5. Pressupost

3.5.1. Estimació cost de personal

Un cop hem comptabilitzat les hores realitzades per cada membre de l'equip i tenint en compte el seu preu per hora, detallarem mitjançant una taula, el cost total per cada membre així com el cost total del projecte.

Recursos humans	Total hores	Total cost
Director de projecte	17,3 hores	934,20 €
Cap de projecte	174 hores	6.264,00 €
Analista	45,8 hores	1.099,20 €
Programador	282,4 hores	4.236,00 €
Tècnic de proves	103,8 hores	1.557,00 €
Total:	**568,1 hores**	**14.090,40 €**

Taula 12 .- Estimació del cost segons el personal del projecte.

3.5.2. Estimació cost dels recursos

En aquest apartat farem una valoració dels recursos utilitzats en el nostre projecte, que sempre aniran a càrrec de l'empresa. Els costos són:

Recursos	Llicència	Cost
Microsoft Office 2007	Individual i OEM	129,99 €
Eclipse IDE for Java Developers	Pública general GNU	0,00 €
OpenProj	Pública general GNU	0,00 €
SQL Developer	OTN License	0,00 €
TOTAL		129,99 €

Taula 13 .- Estimació del cost dels recursos

3.6. Conclusions

Un cop s'han observat totes les dades dels diversos costos relacionats al projecte es pot establir el cost total d'aquest.

Cost de desenvolupament del projecte	14.090,40 €
Cost dels recursos utilitzats	129,99 €

Taula 14 .- Taula resum dels costos

Aquest projecte té un cost força elevat, tot i que l'empresa en farà un ús intern que posteriorment pot ser utilitzat per desenvolupar una aplicació que es pugui vendre al mercat i treure'n un benefici econòmic.

Tot i que, el cost del projecte seria l'indicat en la taula anterior, el cost real per l'empresa és el sou del becari, en aquest cas 2.352 €.

Un cop realitzat el pla de projecte hem obtingut un anàlisi complert del nostre projecte, on sabem per exemple: el cost del projecte, els riscos que es poden produir, el personal que el portarà a terme, etc. Aquest document ens permetrà desenvolupar, gestionar i controlar el projecte.

CAPÍTOL 4: ANÀLISI I DISSENY

L'anàlisi i disseny ens facilitarà la informació necessària per comprendre el projecte en si, d'una manera més interna. Per tal d'aconseguir-ho, veurem aspectes com l'arquitectura de l'aplicació, la interfície d'usuari i els processos de distribució (funcionalitat i comunicació entre ells).

4.1. Arquitectura de l'aplicació

L'arquitectura de l'aplicació que hem presentat per desenvolupar el nostre projecte, com ja hem comentat en la introducció, consta d'un entorn global on es definiran aquelles variables que poden ser utilitzades per tots els processos. Dins d'aquest entorn estan definits els diferents processos.

Aquests processos tenen com a característica més important el fet de que són independents els uns dels altres. Això proporcionarà que si en un futur es necessités modificar algun procés en concret, només hauríem de modificar el procés en qüestió. L'altre avantatge que ens atorgarà és que permetrà executar un procés individual per generar una part de la distribució, o a través d'una execució de forma conjunta i coordinada poder generar parts importants d'un KIZ, indicant pocs paràmetres.

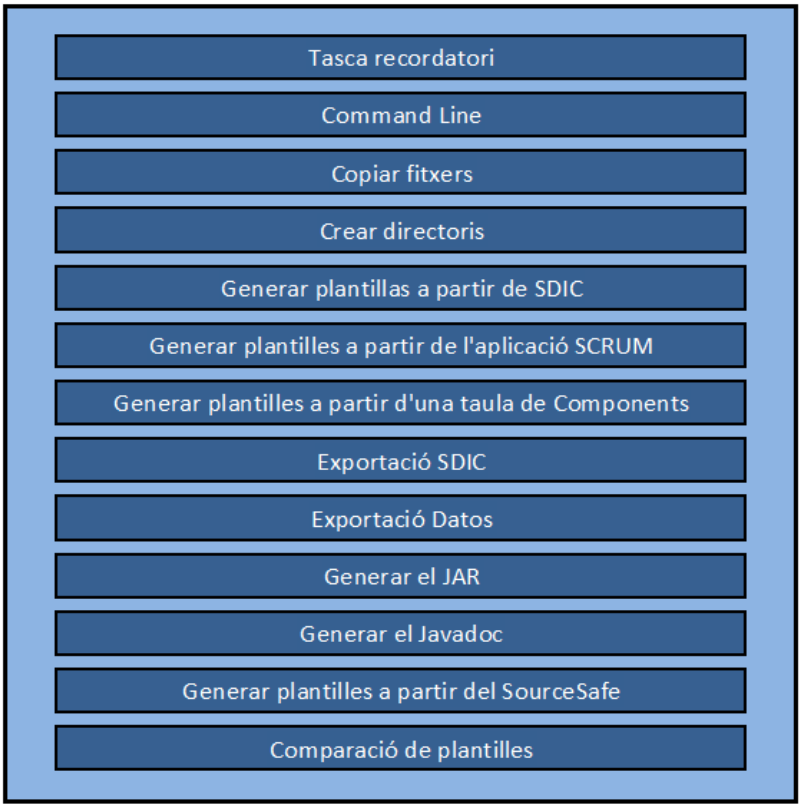


Figura 5 .- Arquitectura de l'aplicació

4.2. Interfícies d'usuari

La idea inicial era generar la funcionalitat de l'aplicació a través de classes Java com hem comentat en punts anteriors, i després tenir diferents maneres per poder-ho explotar:

1) **Aplicació gràfica:** desenvolupat amb Java Swing, amb la que podrem interactuar amb els processos indicant manualment els diferents paràmetres.

Per dur-la a terme hem utilitzat las classes MigLayout i BorderLayout per tal d'organitzar els diferents elements (botons, labels, títols...) en la finestra de treball. A continuació mostrem un parell d'imatges de l'aplicació, la figura 6 forma part de la pantalla principal i la figura 7 de la interfície d'un procés en concret.

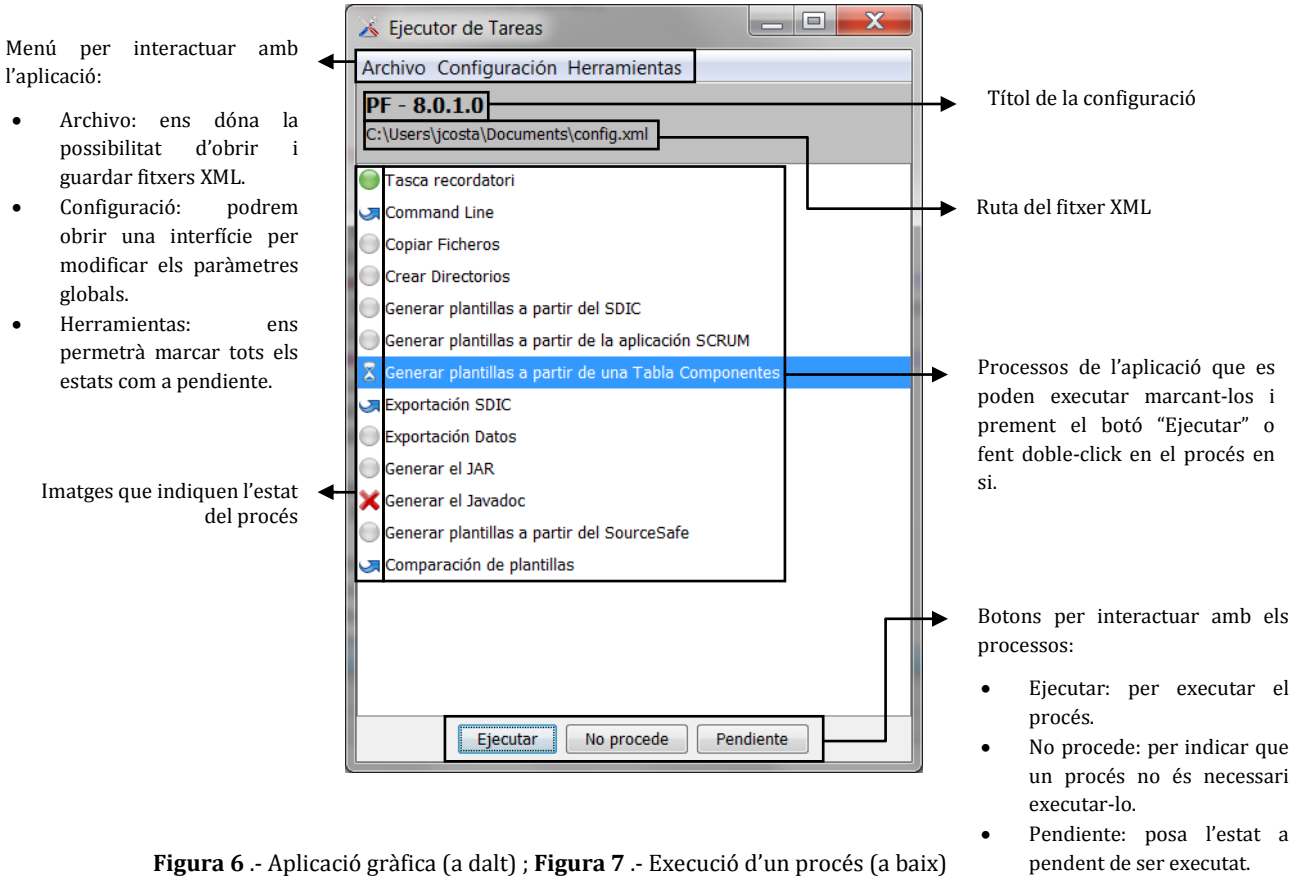
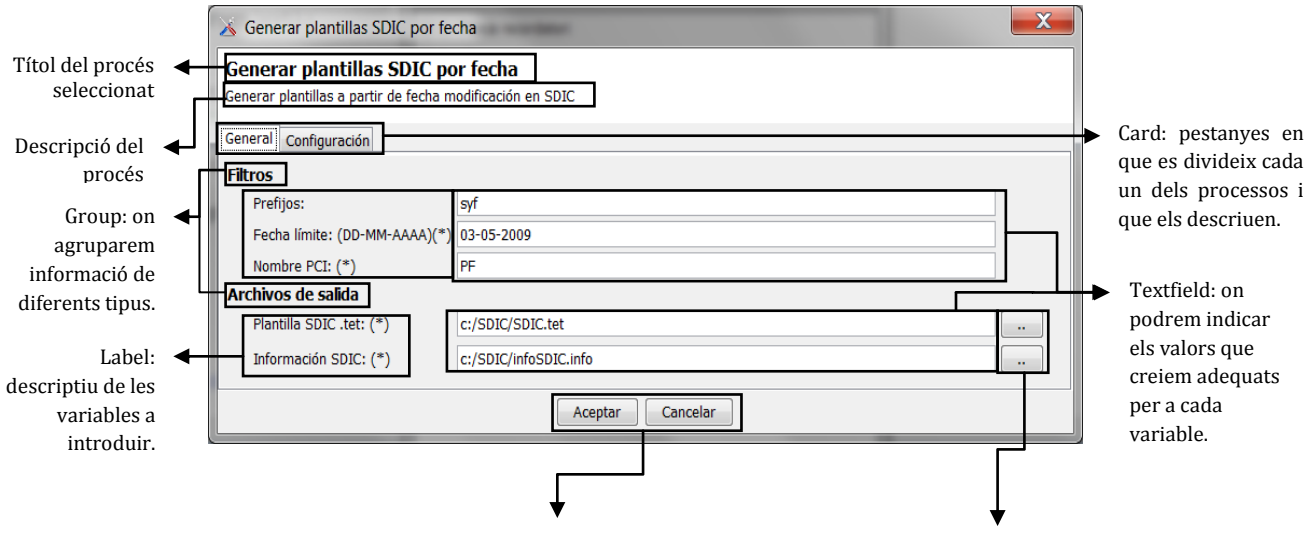


Figura 6 .- Aplicació gràfica (a dalt) ; **Figura 7** .- Execució d'un procés (a baix)



2) **Wrapper Command Line:** per tal de desenvolupar l'explotació del sistema (coreApi), mitjançant línies de comandes. Inicialment, aquesta explotació del coreApi havia de ser principal i prioritària, però veient com es treballa actualment vàrem decidir que passés a ser una fase ***next*** del projecte.
   Una fase ***next*** significa que es farà després d'haver finalitzat els procediments prioritaris com puguin ser la creació del coreApi i de l'aplicació gràfica.

3) **Wrapper Apache Ant:** si entenem que l'Apache Ant s'utilitza per la realització de tasques mecàniques i repetitives, i sabent que el procés de generar una distribució es realitza a partir de fer els mateixos passos cada cop, vàrem creure que seria una bona idea realitzar una explotació a partir de l'Apache Ant. Com en el cas anterior, es tracta d'una fase ***next*** del nostre projecte.

4.3. Processos de distribució

En moltes ocasions, durant aquesta memòria, hem parlat i parlarem de processos de distribució. Un procés de distribució és cadascuna de les parts en que es divideix el procés de preparació d'un KIZ i que té una funció en concret.

Un procés de distribució pot està format per:

- Una classe Java: pot no tenir-ne. Aquest cas és el que anomenem una tasca recordatori.
- Paràmetres: en cas de no tenir-ne, s'executarà la classe sense paràmetres.
- UI: en cas de no tenir-ne, no es podran editar els paràmetres i s'executarà la classe amb els paràmetres guardats en el fitxer de configuració, sempre que en tingui. En cas contrari, el procés també haurà de tenir paràmetres.

Resum de successos dins de l'Executora de Tasques, dels possibles cassos:

	UI SÍ	UI NO
CLASSE SÍ	- Obrir la interfície d'usuari. - Acceptar: execució de la classe amb els paràmetres del UI. - Cancel·lar: no fa res, manté l'estat.	- Missatge de confirmació. - Sí: executa la classe amb els paràmetres guardats en el fitxer de configuració. - No: no fa res, manté l'estat.
CLASSE NO	- Obrir la interfície d'usuari. - Acceptar: marca la tasca com a completada. - Cancel·lar: no fa res, manté l'estat.	- Missatge de confirmació. - Sí: marca la tasca com a completada. - No: no fa res, manté l'estat.

Taula 15 .- Resum de successos dins l'aplicació

A continuació passem a fer una petita definició de cada un dels processos del nostre projecte indicant la classe Java que l'executa i els paràmetres que s'han d'indicar en cadascun d'ells.

Tasca buida

No es tracta d'un procés en concret, sinó que ens referim a "Tasca buida" quan parlem de coses que necessitem recordar en la utilització de l'aplicació. Al tractar-se d'un procés, com qualsevol altre, es podrà marcar com a finalitzat en el moment desitjat. Per utilitzar aquest procés no hem d'indicar cap classe en la seva definició.

- Classe associada: NoClass (classe associada als processos que no tenen classe associada en la seva definició), s'associa internament.
- Paràmetres: el procés no tindrà cap paràmetre associat.
- Exemple: si hem de recordar que cal realitzar una còpia de la distribució en un directori, podem crear una tasca buida indicant aquest fet i marcar-la com a correcta un cop realitzada.

Copiar fitxers

Té com a objectiu realitzar 4 còpies com a màxim, de fitxers o directoris. Es podrà copiar un fitxer sobre un altre ja existent, un fitxer en un directori de l'ordinador, o un directori en un altre directori ja existent.

- Classe associada: CopyFiles
- Paràmetres: el procés tindrà 8 possibles paràmetres: 4 elements a copiar i 4 destinacions, que podran ser fitxers o directoris, tan uns com els altres.

Crear directoris

Té com a objectiu crear 4 directoris. Es crearan els directoris sempre que el disc existeixi. En el cas que la ruta no existeixi, anirà creant subcarpetes fins a obtenir el directori indicat. Per exemple:

C:\Jcosta\Mis documentos\UAB2010\Aplicació

Si la ruta C:\Jcosta\Mis documentos existeix però no hi ha cap carpeta UAB2010; el que farà el nostre procés serà crear-nos una carpeta anomenada UAB2010 que tindrà una subcarpeta anomenada Aplicació.

- Classe associada: CreateDir
- Paràmetres: el procés tindrà 4 possibles paràmetres que seran obligatòriament 4 directoris.

Command Line

Té com a objectiu executar una línea de comandes MS-DOS. Aquesta línea de comandes s'executarà sempre que existeixin els fitxers indicats, en cas que s'hagi indicat algun fitxer.

- Classe associada: CommandLine
- Paràmetres: la línea de comandes (obligatòria), i de forma opcional: un possible directori on es trobi l'executor de la comanda, i els dos fitxers que han d'existir per executar el Command Line.

Comparació de plantilles

Té com a objectiu comparar dues plantilles d'elements. El procés crearà un fitxer indicant aquells elements que es troben en una plantilla i en l'altre no, i viseversa.

- Classe associada: CompareTemplates
- Paràmetres: les dos plantilles a comparar, i la ruta absoluta (directori + nom del fitxer) del resultat. Tots són obligatoris.

Exportació SDIC

Té com a objectiu realitzar l'exportació d'elements del SuperDiccionari filtrats per un nom únic del producte.

- Classe associada: ExportSDIC
- Paràmetres: la configuració Karat per connectar-se (nom de la configuració, usuari i contrasenya). el nom del producte (utilitzat com a filtra), la plantilla amb els elements a introduir i el nom que se li vol donar al fitxer resultant (.EXP). Tots són obligatoris.

Exportació DATA

Té com a objectiu realitzar l'exportació de taules de dades.

- Classe associada: ExportDATA
- Paràmetres: la configuració Karat per connectar-se (nom de la configuració, usuari i contrasenya), la plantilla amb les taules que s'han d'incloure en l'exportació i el nom que se li vol donar al fitxer resultant (.EXD). Tots són obligatoris.

Obtenir la última versió del Source Safe

Té com a objectiu obtenir la última versió del Source Safe, per a què qualsevol funció que necessitem obtenir del SourceSafe, sigui de la versió més actual possible.

- Classe associada: LastSSVersion
- Paràmetres: la configuració al Source Safe per connectar-se (carpeta del SS al que volem accedir, usuari i contrasenya), el directori on es troba el SS.exe per poder treballar contra el Source Safe, el nom de la carpeta SS de la que volem obtenir la última versió i el directori on volem guardar el resultat.

Generar plantilles a partir del SourceSafe

Té com a objectiu generar plantilles de classes segons uns filtres indicats sobre el SourceSafe. Extraurem 4 plantilles diferents: una amb la informació de les classes que compleixin els filtres indicats, una altra amb la informació sobre cada una de les classes, una tercera amb les classes que s'han eliminat i finalment un fitxer de LOG per saber si la tasca s'ha executat amb èxit.

- Classe associada: TemplateSourceSafe
- Paràmetres: el procés tindrà com a paràmetres: la configuració al Source Safe per connectar-se (carpeta del SS al que volem accedir, usuari i contrasenya), el directori on es troba el SS.exe per poder treballar contra el Source Safe, la ruta completa (ruta + nom) de les diferents plantilles, i els filtres que seran: una data límit (s'agafaran aquelles classes modificades més tard que aquesta data), una carpeta del SS i un conjunt de tipus de fitxers.

Generar plantilles a partir del SDIC

Té com a objectiu generar una plantilla d'elements del SuperDiccionari (SDIC) després d'aplicar-li un conjunt de filtres. A més a més d'aquesta plantilla .TET, també obtindrem una plantilla d'informació sobre els elements que compleixin els filtres.

- Classe associada: TemplateSDIC
- Paràmetres: la configuració Karat per connectar-se (nom de la configuració, usuari i contrasenya), la ruta completa de les plantilles com en la funció explicada anteriorment, i els filtres que seran: una data límit (s'agafaran aquells elements modificats més tard que aquesta data), un prefix per filtrar el nom dels components i el nom del PCI (únic).

Generar plantilles a partir del SCRUM

Té com a objectiu la generació d'un conjunt de plantilles segons la informació extreta de l'aplicació SCRUM i complint amb un conjunt de filtres. Les plantilles que es generaran són: una de components SDIC (.TET), una de dades (.TED), una de classes (.TXT) i una d'altres possibles components (.TXT). Per cadascuna d'aquestes plantilles es generarà la seva corresponent plantilla d'informació.

- Classe associada: TemplateSCRUM
- Paràmetres: la configuració de l'SCRUM de Karat per connectar-se (nom de la configuració, usuari i contrasenya), la ruta completa de les plantilles com en algunes de les funcions anteriors (ruta + nom), i els filtres que seran: el producte de Karat del qual volguem obtenir la informació, la secció i las històries; aquestes dos últimes són variables internes de l'aplicació de l'SCRUM.

Generar plantilles a partir de la Taula de Components

Té com a objectiu la generació d'un conjunt de plantilles segons la informació extreta d'una taula de components on es guarda informació sobre els components que ha d'incloure cada distribució. Les plantilles que es generaran són: una de components SDIC (.TET), una de dades (.TED), una de classes (.TXT), una d'altres possibles components (.TXT) i una d'elements eliminats. Per cadascuna d'aquestes plantilles es generarà la seva corresponent plantilla d'informació.

- Classe associada: TemplateComponents
- Paràmetres: la configuració de Karat per connectar-se (nom de la configuració, usuari i contrasenya), la ruta completa de les plantilles com en algunes de les funcions anteriors (ruta + nom), i els filtres que seran: el producte de Karat del qual volguem obtenir la informació i la versió d'aquest producte.

Generar el Jar o el Javadoc

Té com a objectiu la generació d'un Jar o un Javadoc depenen de la tasca de l'aplicació cridi a la classe. Es tracta d'una única classe que pot ser cridada des de 2 tasques diferents. Per diferenciar si s'ha de realitzar un Jar o un Javadoc, ens fixem amb si s'ha indicat la variable JarName per generar un Jar, o si s'ha indicat JavadocName per generar un Javadoc. El document generat vindrà filtrat per una plantilla de classes. En cas de no indicar-ne cap es farà el Jar o el Javadoc de tots els elements dins del directori base.

El procés depèn d'un fitxer XML que nosaltres distribuïm, però pot ser executat a partir de qualsevol altre fitxer complint uns requisits indicats en el Manual d'usuari (Annex 2).

- Classe associada: Create
- Paràmetres: la ruta absoluta (ruta + nom del fitxer) del fitxer XML que defineix l'ANT que realitzarà el Jar o el Javadoc, la versió del producte del que realitzarem la funció, el nom del producte, el directori base on es troba la informació que volem generar, un apartat on podrem modificar algunes variables del fitxer XML que defineix l'ANT i els directoris de Java i de l'Ant per poder executar-los i també els corresponents directoris de les llibreries que es necessiten pel funcionament del Jar o el Javadoc.

  A més a més, tindrà com a paràmetre d'entrada la plantilla de classes (opcional), i la ruta dels directoris temporals (on es realitzarà el treball previ) i el directori final (on es trobarà el resultat). I finalment, indicarem el nom del Jar si volem generar un Jar o el del Javadoc en cas contrari, a part d'indicar un fitxer de Log per controlar l'execució.

Una altre procés que s'ha generat és el de crear el Jar i el Javadoc en el mateix instant mitjançant una única tasca. El procés que es segueix és el mateix que l'anterior però en aquest cas indicarem tan el nom del Jar com el del Javadoc per obtenir els dos resultats.

4.4. Comunicació entre processos

Com ja hem vist anteriorment, la única via de comunicació entre processos és l'entorn global que comparteixen. Per tan, en cas de que vulguem comunicar un procés de distribució amb un altre ho podem fer utilitzant les variables globals.

Per exemple: indiquem que la generació de plantilles amb una de les funcions específiques es guardi en un directori i amb un nom en concret, tot això indicat en una variable global; i que després, aquesta variable sigui l'entrada d'un altre procés (Export SDIC).

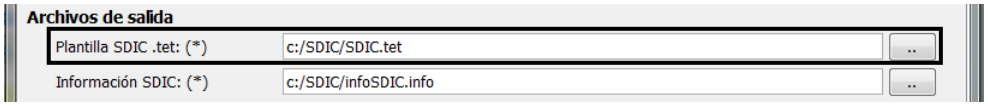
Explicació gràfica de l'exemple anterior:

- Definició XML: existeix una variable global anomenada TEMPLATE_SDIC que guarda la ruta completa (ruta + nom de la plantilla).

  <TEMPLATE_SDIC>c:/SDIC/SDIC.tet</TEMPLATE_SDIC>

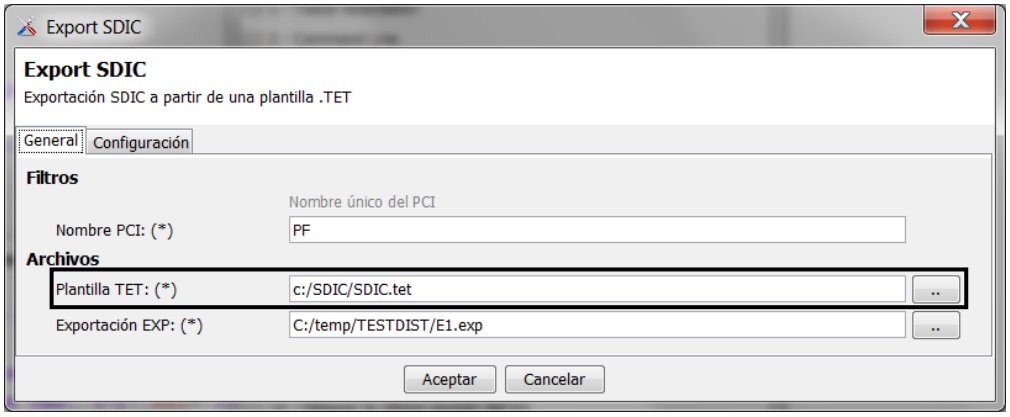
- En l'aplicació gràfica:

  Com ja hem explicat, es pot accedir a qualsevol variable global des de tots els processos de distribució, i això és el que fem en el nostre cas:

  Es tracta d'una captura del procés "Generar plantilles a partir del SDIC" i que en l'apartat de paràmetres té definit "Plantilla SDIC .tet: (*)" amb el següent valor: %TEMPLATE_SDIC%. D'aquesta manera accedeix a la variable global.

  El procés realitzarà una plantilla SDIC que es guardarà en el fitxer indicat per TEMPLATE_SDIC. Per altre banda, en el procés ExportSDIC indiquem que la plantilla d'elements d'SDIC que volem incloure en l'exportació es troba també en el valor de la variable global TEMPLATE_SDIC.



D'aquesta manera hem enllaçat dos processos de l'única manera possible. La sortida d'un procés ha estat l'entrada de l'altre.

Aquest aspecte es pot utilitzar per agilitzar l'ús de l'aplicació o per realitzar dues tasques a la vegada, ja que el llistat de tasques té multi selecció. Si s'utilitza aquest últim cas, és molt important mantenir l'ordre d'execució dels processos.

CAPÍTOL 5: IMPLEMENTACIÓ

La implementació en aquest cas consisteix en definir el "com" del projecte. És a dir, quines aplicacions s'han utilitzat per desenvolupar-lo i de la manera que ho hem realitzat; per tal de tenir-ne una percepció clara.

En aquest capítol trobarem l'explicació tan de la funcionalitat, com de l'execució de la interfície, passant, fins i tot, per la manera que tenim de controlar els errors.

5.1. Entorn de desenvolupament

Per implementar i escriure el nostre codi, utilitzarem l'entorn de desenvolupament Ecplipse que ens permetrà dur a terme la nostra aplicació en Java.

5.1.1. Desenvolupament en Java

Java és un llenguatge de programació orientat a objectes, desenvolupat per Sun Microsystems a principis dels anys 90. El llenguatge en si mateix agafa molta sintaxis de C i C++, però té un model d'objectes més simples i elimina eines de baix nivell, que solen portar molts errors, com la manipulació directa de punters o memòria.

La implementació del nostre projecte està dividida en 2 parts ben diferenciades: per una banda la interfície gràfica que serà executada amb Java Swing i alimentada per XML – Reflection, i per altra banda, com ja hem dit anteriorment, cada funció necessària en el procés de distribució serà una classe Java implementada amb la interfície Runnable.

Aquestes classes tindran totes els mateixos 4 mètodes:

a) setParameters(String data): mètode que servirà per passar paràmetres a la classe en concret. Aquests paràmetres vindran indicats com a paràmetres d'entrada per un String en format XML, de la següent manera:

   <Parameters>

   <Nom_Variable> Valor_Variable </Nom_Variable>

   <Nom_Variable> Valor_Variable </Nom_Variable>

   </Parameters>

b) run(): és on es troba la funcionalitat del procés de distribució.
c) succed(): és l'encarregat d'enviar a l'aplicació si el run() s'ha executat satisfactòriament, true o false.
d) getException(): s'encarrega d'enviar a l'aplicació els errors produïts en la classe en concret.

Codi de la definició de la Interfície RunnableXml:

```
public interface RunnableXml extends Runnable {
/**
* Pasaremos variables a nuestras funciones.
*/
public void setParameters(String parameters);
/**
* Permite indicar el estado final de la función.
*/
public Boolean succed();
/**
* Es llamada en caso de aparecer una excepción.
*/
public Exception getException() throws Exception;
/**
* Da funcionalidad a la clase.
*/
public void run();
}
```

La crida a aquestes classes que defineixen les tasques de l'aplicació es farà a través del SwingWorker. La classe SwingWorker ens permet realitzar una tasca que tardi un temps considerable i a la vegada ens permet modificar elements en la nostra pantalla principal, i tot això en el mateix temps d'execució. El SwingWorker, consta de dos mètodes:

- doInBackground(): és el mètode on es troba la funció que pot tardar bastant de temps. Tot i que, pot ser utilitzada també per retornar algun valor, en el nostre cas no serà necessari. En ell serà on realitzarem l'execució de cada una de les classes.
- done(): aquest mètode s'executarà justament al finalitzar l'altre. Aquí, controlarem si la classe s'ha executat correctament o no, i en cas negatiu mostrarem l'excepció per pantalla.

5.2. Desenvolupament XML

XML de l'anglès e**X**tensible **M**arkup **L**anguage (<llenguatge de marques extensible>), és un metallenguatge d'etiquetes desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació de l'experimental SGML, i permet definir la gramàtica de llenguatges específics. Per tant, XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. És un sistema que no ha nascut només per la seva aplicació a Internet sinó, com en el nostre cas, és una molt bona eina per realitzar un intercanvi d'informació estructurada.

La utilització que tindrà l'XML en el nostre projecte és transcendental, amb ell ho definirem tot: tan les variables globals del sistema, com les variables específiques de cada procés, com la definició de cada interfície (manera com veurem l'aplicació).

L'arquitectura del fitxer XML serà la següent:

<Configuration>

<Variables>

<Variable />

</Variables>

<Uis>

<Ui />

</Uis>

<Tasks>

<Task>

<Paramaters>

</Parameters>

<Ui>

</Ui>

</Task>

</Tasks>

</Configuration>

En aquesta estructura podem dividir uns components a comentar:

- Variables: en aquest apartat es definiran les variables globals del nostre sistema que podran ser utilitzades pels processos de distribució. Per exemple:

  <PRD>PF</PRD> ; <nom_variable> valor_variable </nom_variable>

  Aquí introduïm el concepte de Reflection, que significa que podrem utilitzar el valor de qualsevol variable en qualsevol punt del nostre fitxer XML. Això ho aconseguirem escrivint %nom_variable%. En l'exemple indicat seria: <Versió> %PRD% 9.0.1.0 </Versió>, i el valor de la variable Versió seria: PF 9.0.1.0.

- Uis: definirem les interfícies d'usuari per poder modificar les variables globals. En l'apartat <Ui>, ens defineix com s'estructurarà gràficament la nostra aplicació.

```
<Card Caption="Informació 1">
        <Group Caption="Versión">
                <Input Caption="Producto:  " Ref="//Variables/PRD" Type="Text"/>
        </Group>
</Card>
```

Figura 8 .- Visualització de la definició de UI en format XML.

En l'apartat Input, definim el camp descriptiu que ens permet saber què hem d'introduir, la referència al paràmetre i finalment el tipus de valor que es demana (text, directory, un fitxer, una contrasenya...).

- Tasks: aquí es trobarà el centre del nostre sistema. Serà on es definiran els processos de distribució i estarà format en primer lloc pels paràmetres necessaris per executar el procés (seguint la forma de l'etiqueta Variables), i a continuació la definició de la seva interfície d'usuari; seguint en ambdós casos els procediments indicats .

5.3. Desenvolupament intern

En aquest apartat, explicarem com desenvoluparem internament els diferents processos i aprofundirem sobretot en aquells elements que necessitarem per dur-los a terme.

En tots els processos de distribució que s'inclouen en el nostre desenvolupament, utilitzem diferents importacions de Java que ens permetran realitzar diferents operacions: des de llegir o escriure fitxers per les plantilles (java.io), fins a diferents tipus d'utilitats com llistes o mapes; passant per una classe interna de l'empresa que ens permetrà parcejar els fitxers XML.

5.3.1. Command Line

Algunes parts d'aquests processos s'han realitzat mitjançant comandes MS-DOS (Command Line), com per exemple la funció pròpiament dita "Command Line" que permet executar una línea de comandes des de la nostra aplicació Java o la "TemplateSourceSafe", ja que es pot accedir al SS per comandes MS-DOS.

En un principi, per executar una comanda MS-DOS s'utilitza la següent línea:

"Runtime.getRuntime().exec(comanda);"

Però ens vàrem adonar que en alguns cassos no en teníem prou amb una línea de comandes sinó que havíem de realitzar un fitxer per lots (.bat), i a través de l'anterior línea executar aquest fitxer. Els passos que hem seguit en el segon cas (tenir més d'una línea de comandes), són els següents:

1) Escriure el fitxer per lots amb totes les comandes MS-DOS.
2) Tancar el fitxer.
3) Executar el fitxer amb el getRuntime().
4) Esborrar el fitxer.

5.3.2. Apache ANT

Apache Ant és una eina utilitzada en programació per la realització de tasques mecàniques i repetitives, normalment durant la fase de compilació i construcció (build). És, per tant, un software per processos d'automatització de compilació, similar al "Make" però desenvolupat en llenguatge Java i que requereix la plataforma Java. Aquesta eina té l'avantatge que no depèn d'ordres shell de cada sistema operatiu, sinó que es basa en fitxers de configuració XML.

Nosaltres hem utilitzat l'Apache Ant en els processos: "CreateJar" i "CreateJavadoc". El que farem és modificar el fitxer XML de creació del Jar o del Javadoc a partir de les variables demanades a l'usuari. El fitxer XML està realitzat pensant en que a partir de la introducció d'aquestes variables base que indica l'usuari es puguin generar altres variables necessàries per la seva execució, com per exemple: el nom del jar dependrà del nom del projecte.

5.4. Control d'errors

El control d'errors en el nostre projecte està sempre controlat a partir d'un mètode getException(e); on "e" és l'excepció que s'ha produït. Al tractar-se d'una aplicació gràfica ha de tenir un control d'errors molt exhaustiu i complert en tots el sectors de l'aplicació, tan en la pròpia com en cada un dels processos. Li hem de proporcionar al usuari la possibilitat d'observar quin error s'ha produït per poder solucionar-lo en un futur.

Aquestes excepcions es poden produir en qualsevol moment de l'execució del sistema, i el seu origen pot ser:

- Un error produït per alguna errada en un try-catch, com per exemple al obrir sessió.
- Un error indicat per nosaltres, per controlar alguna part del funcionament, com per exemple per obligar a l'usuari que indiqui les variables que són obligatòries.

El mètode getException(), treballarà de forma diferent segons si es troba en l'execució de la finestra principal, o en algunes de les execucions secundàries com podria ser algun procés o els paràmetres globals. En el primer cas, el mètode serà un ***void*** que generarà la finestra de diàleg perquè l'usuari pugui detectar que s'ha produït un error i a continuació finalitzarà l'execució d'aquell procés en concret. En canvi, quan la crida a l'excepció es produeixi en algun dels altres cassos, el mètode serà un ***Exception***, és a dir que retornarà l'excepció que li arribi per poder esser recuperada en un moment futur.

CAPÍTOL 6: CODIFICACIÓ I PROVES

En aquest capítol explicarem l'estil de codificació que hem utilitzat pel desenvolupament de la nostra aplicació i les diferents proves que hem realitzat. Aquestes proves les hem dividit en tres tipus diferents: proves unitàries, proves d'integració i proves d'estrès.

6.1. Estil de codificació

Pel que respecta a l'estil de codificació, hem intentat en tot moment realitzar un codi simple, clar i senzill d'entendre. Es tracta d'una aplicació tancada, però que es pot augmentar o disminuir de funcionalitats, és a dir, afegir o treure tasques en l'Executor de Tasques.

Això ha portat a que es segueixin unes normes a l'hora de codificar:

- ✓ Agrupar les funcionalitats de forma lògica.
- ✓ Cada classe i mètode té una finalitat específica.
- ✓ Les classes s'agrupen d'una forma lògica i ordenada en diferents paquets:
    - El paquet ***com.unit4.ekon.genkit.coreapi*** agrupa les classes que donen funcionalitat a les tasques de l'aplicació.
    - El paquet ***com.unit4.ekon.genkit.utilities*** agrupa les classes que ajudaran als diferents processos a fer algunes funcions, com per exemple executar un fitxer BAT o parcejar els fitxer XML.
    - El paquet ***com.unit4.ekon.genkit.xmlgui***, conté la classe que analitza el fitxer XML de configuració de cadascuna dels processos i ho transforma en aplicació gràfica.
    - El paquet ***com.unit4.ekon.genkit.exec***, conté les classes que permeten generar i treballar amb l'aplicació.

- ✓ Assignació de noms descriptius per classes, mètodes i objectes dins del codi.
- ✓ Documentació del codi a nivell de classe.

Una de les parts més importants que trobem en l'estil de codificació és la interfície que hem utilitzat per generar les classes del coreApi (explicat en l'apartat d'implementació). I en ella pren gran importància com controlem els errors. Per cada possible error que genera la nostra aplicació o l'usuari, indicant erròniament algun paràmetre o realitzant algun procés desconegut per l'aplicació s'ha de mostrar un missatge d'error específic per a que l'usuari pugui saber on es troba l'error i posar-li remei.

6.2. Proves unitàries

Les proves unitàries és la forma de comprovar el correcta funcionament d'un mòdul del codi. Això serveix per a que cada mòdul funcioni correctament per separat. Després, amb la prova d'integració es podrà assegurar el correcte funcionament del sistema i del subsistema en qüestió.

Les proves unitàries es van dividir en dos apartats: per una banda, les proves realitzades a l'explotació inicial de l'aplicació (pantalla principal) i les proves a l'explotació de cada procés de distribució internament; i per altre banda, les proves realitzades a cada una de les classes del coreApi.

- **Proves unitàries en l'aplicació gràfica.**

  El sistema inicial tenia un aspecte de funcionament correcte però a mida que anava passant el temps i les execucions, es produïen nous errors i noves limitacions en alguns aspectes.

  Els errors que es van produir i que no estaven controlats de bon principi van ser:

  - El fitxer de configuració estava mal definit, per exemple no es trobava algun element.
  - A l'executar alguna tasca que tenia el seu nom definit també en una altre, s'executaven les dues.
  - A l'obrir un fitxer més d'un cop, provocava la desactivació dels botons de l'aplicació.

  Algunes limitacions que van anar apareixen són:

  - De bon principi no es controlaven els possibles casos en que es volia tancar l'aplicació o obrir un nou fitxer de configuració i no s'havia guardat prèviament. Els casos són:
    - En cas de tancar i que hi hagi una modificació dels estats inicials de les tasques o dels paràmetres globals, l'aplicació demanarà a l'usuari si vol guardar el fitxer de configuració.
    - En cas de voler tancar l'aplicació i trobar-se alguna de les tasques en execució, l'aplicació demanarà si es vol guardar el fitxer de configuració. Si la resposta és afirmativa, es guardarà l'estat d'aquell procés com a incorrecta.
  - Control en l'execució de les tasques: una tasca no es pot executar si ja s'està executant.

- **Proves unitàries del coreApi.**

  Cada cop que es finalitzava un procés de distribució es realitzaven proves unitàries per comprovar el seu correcte funcionament. En el nostre cas, es tractava de comprovar si el resultat que havia de generar el procés era correcta.

  Les proves que es van realitzar en els diferents processos van ser sempre les mateixes: anar provant les tasques repetides vegades indicant qualsevol tipus de paràmetres. Això ens va permetre veure com reaccionava la nostre aplicació.

A continuació realitzarem un llistat dels processos que havien generat errors destacables després de realitzar-li les proves unitàries, enumerant quines proves es van realitzar i com ho vàrem solucionar.

- Crea directoris:

  L'únic problema que va sorgir va ser quan vàrem descobrir que no controlàvem els paràmetres. Això significa que inicialment podíem indicar un directori del qual, no existís la seva arrel i la tasca sortia com a realitzada correctament quan realment no realitzava la seva funció.

  La solució va ser assegurar-nos de que existia l'arrel de la ruta que indicàvem en el procés i en el cas que no existís, mostrar un missatge d'error.

- Copiar fitxers:

  Les proves del procés es basen en la funcionalitat del Command Line "xcopy", que en limita que lògicament no es pot copiar un directori en un fitxer i per tant vàrem haver de controlar els elements d'origen i els de destí i quan ens trobéssim en el cas de que l'origen fos un directori i el destí un fitxer, mostrar un missatge d'error.

- Generar plantilles a partir del SourceSafe:

  Les proves del procés es centren a l'hora de fer la connexió al SourceSafe mitjançant les línies de comandes. El que va succeir és que no es podia anar cridant a les comandes una a una, sinó que s'havia de generar el fitxer BAT i executar-lo llegint les línees. Aquest fet feia que no poguéssim controlar si la connexió al SourceSafe s'havia realitzat correctament ja que en cas de realitzar una connexió errònia el procés es quedava penjat.

  La solució va ser realitzar una connexió que generés un resultat ràpid, i que passat un temps prudencial és comprovés si s'havia generat aquest resultat en forma de document.

Un cop realitzades aquestes proves, ens vàrem donar compte que el control que fèiem de les variables d'entrada era quasi inexistent i això feia que es produïssin errors inesperats en l'aplicació. Errors que no sabíem d'on procedien ja que l'error que generava l'aplicació era un missatge en blanc. Per

solucionar aquest problema, en cada try-catch, en l'apartat on es generava l'error vàrem cridar a "e.printStackTrace();" perquè ens mostrés a la consola de l'Eclipse d'on procedia l'error.

Per no tornar-nos a trobar amb aquest error, vam haver de realitzar una revisió de cada un de les variables d'entrada comprovant que el tipus de valor coincidís amb el tipus desitjat i en cas contrari, mostrar el missatge d'error corresponent.

6.3. Proves d'integració

Les proves d'integració són aquelles que es realitzen un cop s'han aprovat les proves unitàries. Consisteix en realitzar proves per verificar que un gran conjunt de parts del software funcionin de manera conjunta, en el nostre cas la part de l'aplicació gràfica i l'execució de cada una de les tasques.

De la relació entre l'aplicació i les tasques se n'encarrega la classe SwingWorker que té com a funció principal cridar a l'executora de la interfície d'usuari i realitzar la funció corresponent. La relació entre la classe Java i l'aplicació es veurà en els següents casos:

- Canviar l'estat de la tasca: quan s'inicia el procés canviant l'estat inicial a "en execució" i un cop finalitzat el procés marcant-lo com a "completat correctament" o "completat de forma errònia".
- Recuperar els missatges d'error enviats des de cada procés dins de l'aplicació.

En aquest punt de comprovació del sistema, no ens hem trobat cap problema destacable.

6.4. Altres proves

En aquest apartat inclourem un conjunt de proves que hem realitzat a part de les ja esmentades per tal de veure com responia l'aplicació i d'assegurar-nos el seu correcte funcionament. Les podem dividir en dos:

- ✓ Una prova d'estrès que va consistir en deixar provar l'aplicació a algú sense coneixements sobre l'aplicació i veure com reaccionava aquesta als diferents casos en que es trobés.
  Un cop acabada l'aplicació, la vàrem deixar en mans d'un company per a què la provés sense donar-li cap tipus d'informació i els resultats que vàrem obtenir són:
  - S'havia de controlar l'execució de les tasques.

- S'havia de realitzar un control dels paràmetres en cada procés.

✓ Proves des d'un altre punt de vista: un cop realitzades les diferents proves per part del realitzador del codi, es va passar l'aplicació al tutor del projecte dins l'empresa Unit4 que es va encarregar d'anar provant tan l'aplicació com cada un dels processos un a un, trobant errors que s'havien pogut escapar fins aquell moment.

6.5. Integració de l'aplicació a l'ordinador

Per a la integració del Karat distributions és necessari distribuir, als treballadors de Unit4 que necessitin l'aplicació per realitzar la preparació dels seus KIZ, el mòdul que hem realitzat, i que conté:

- Karat_distributions: és el Jar de la nostra aplicació.
- Directori del Apache Ant: és un directori necessari per executar fitxers ANT. S'utilitza en l'execució del processos per a la generació dels Jar i els Javadoc.
- miglayout-3.7.4-swing: és un Jar extern a l'aplicació que necessitem tenir per visualitzar l'aplicació de forma correcte.
- Config.xml: fitxer de configuració imprescindible per poder explotar l'aplicació al complert, amb les seves interfícies d'usuari i executar les tasques.
- Kit_tasks: fitxer BATCH (.bat), que al executar-lo s'obrirà la nostra aplicació.
- Manual d'usuari: necessari i imprescindible on expliquem els diferents processos de distribució així com la manera per utilitzar l'aplicació i personalitzar-la.
- Manual d'instal·lació: fitxer TXT on s'indiquen els passos per instal·lar l'aplicació i les seves dependencies.

Per poder utilitzar la nostra aplicació, hem de copiar el directori karat_distributions en qualsevol directori local del nostre sistema, i editar el valor de la variable KARATHOME en el fitxer kit_tasks indicant el directori karat del nostre ordinador. Per exemple:

```
SET KARATHOME="C:\Program Files (x86)\karat"
```

CAPÍTOL 7: CONCLUSIONS

Es detallaran aquells objectius que s'han aconseguit i els que no, també anomenarem algunes de les possibles ampliacions i millores de l'aplicació. Tot seguit, realitzarem un anàlisi de les possibles desviacions que s'han produït, sobre el temps dedicat a cada apartat, comentant-ne les més destacables;. Per acabar, veurem una valoració personal sobre el projecte en general.

7.1. Objectius aconseguits i no aconseguits

Un cop arribats a aquest punt, s'ha de fer valoració del treball realitzat. En aquest apartat, mirarem quins objectius, dels indicats inicialment s'han dut a terme i quins no, comentant-ne els motius.

Hi ha un conjunt d'objectius que podem considerar que s'han complert però que no en tindrem la certesa total fins que l'aplicació estigui integrada en un equip de treball i s'utilitzi de forma regular. Aquests objectius són: ***disminuir el temps que es dedica a fer cada una de les preparacions de distribucions, disminuir la gran dependència que té l'eina actual, disminució de les errades humanes.***

Podem dir, que l'aplicació compleix els requisits funcionals bàsics, que són:

- Tenir una eina que permeti a l'empresa Unit4 ajudar a generar una distribució de forma automàtica indicant pocs paràmetres.
- Poder realitzar parts de la distribució separadament.
- Aconseguir una eina que sigui utilitzada per tota l'empresa.
- Definir un estàndard de seguiment del procés de distribució.

Aquest últim punt s'ha complert amb escreix pel fet de poder personalitzar l'eina i escollir en cada cas l'estàndard de seguiment.

L'únic objectiu que no s'ha complert és el de realitzar ***informes sobre les distribucions realitzades***. El motiu és que vàrem considerar innecessari realitzar informes sobre cada una de les distribucions ja que en cada moment indicàvem si una tasca s'havia realitzat correctament o no de forma gràfica en l'Executora de Tasques.

Finalment, comentem que l'objectiu de ***proporcionar a l'eina diferents maneres d'interactuar amb ella*** ha complert la part obligatòria, és a dir fer l'aplicació gràfica. Les altres possibles maneres que havíem pensat per explotar el coreApi i que estaven definides com a objectius NEXT (explotació per línea de comandes i explotació a través de l'ANT) no s'han pogut aconseguir per falta de temps.

7.2. Possibles ampliacions i millores

Com ja hem observat en el punt anterior, la nostra aplicació compleix els objectius bàsics que vàrem definir de bon principi. Però a mida que s'anaven realitzant els diferents processos de distribució s'anaven descobrint possibles ampliacions i millores, com:

- Generar el fitxer XML de directives d'instal·lació: una de les parts que es necessita per generar la distribució física són les directives d'instal·lació. Aquestes directives defineixen un conjunt de instruccions que s'han d'executar a l'hora d'instal·lar la distribució en el client.
  Per tan, una de les millores podria ser el fet de generar aquestes directives d'instal·lació.

- Obrir el "karat distributions generator":

  Tot i que actualment, a través de la tasca "Command Line" podem realitzar aquesta funció però no explotar-la al 100%. La funcionalitat que tindria aquesta millora seria la d'obrir l'aplicació indicant el fitxer de definició anomenat directives d'instal·lació per seguidament poder generar la distribució.

- Generar plantilles a partir d'ekon SCRUM:
  Tot i que actualment ja existeix aquest procés, no està essent utilitzat per tothom i podria ser que s'haguessin de realitzar algunes millores en un futur.

- Diferents maneres d'explotació:
  Com ja hem comentat, unes de les millores que es podrien realitzar són les de realitzar les dos explotacions ja esmentades: explotació per línea de comandes i per Apache ANT.

7.3. Desviacions respecte la planificació

Comparant els processos de distribució que havíem de generar i el que finalment hem generat, trobem algunes diferències que s'han de comentar ja que són elements nous que s'han generat o uns que pensàvem que generaríem i no hem generat. Aquestes diferències són:

- El procés que vàrem anomenar "Obrir XML", que consistia en generar el fitxer XML que es genera amb el KIZ no s'ha pogut realitzar pel simple fet que no hem tractat en cap moment la generació del KIZ en concret, perquè l'aplicació "karat distributions generator" no ens permetia realitzar les funcions que nosaltres volíem. S'ha parlat amb els companys de desenvolupament

per modificar aquesta aplicació i poder treballar amb ella.

- El procés "generar plantilles a partir de filtres" s'ha dividit en 3 processos diferents segons l'aplicació d'on se'n treu la informació. Aquests nous processos són: generar plantilles a partir del SourceSafe, generar plantilles a partir del SDIC i generar plantilles a partir del producte SCRUM.

A continuació presentem una comparació entre les hores previstes inicialment i les hores reals dedicades en les diferents etapes del projecte i una altra sobre les tasques del coreApi.

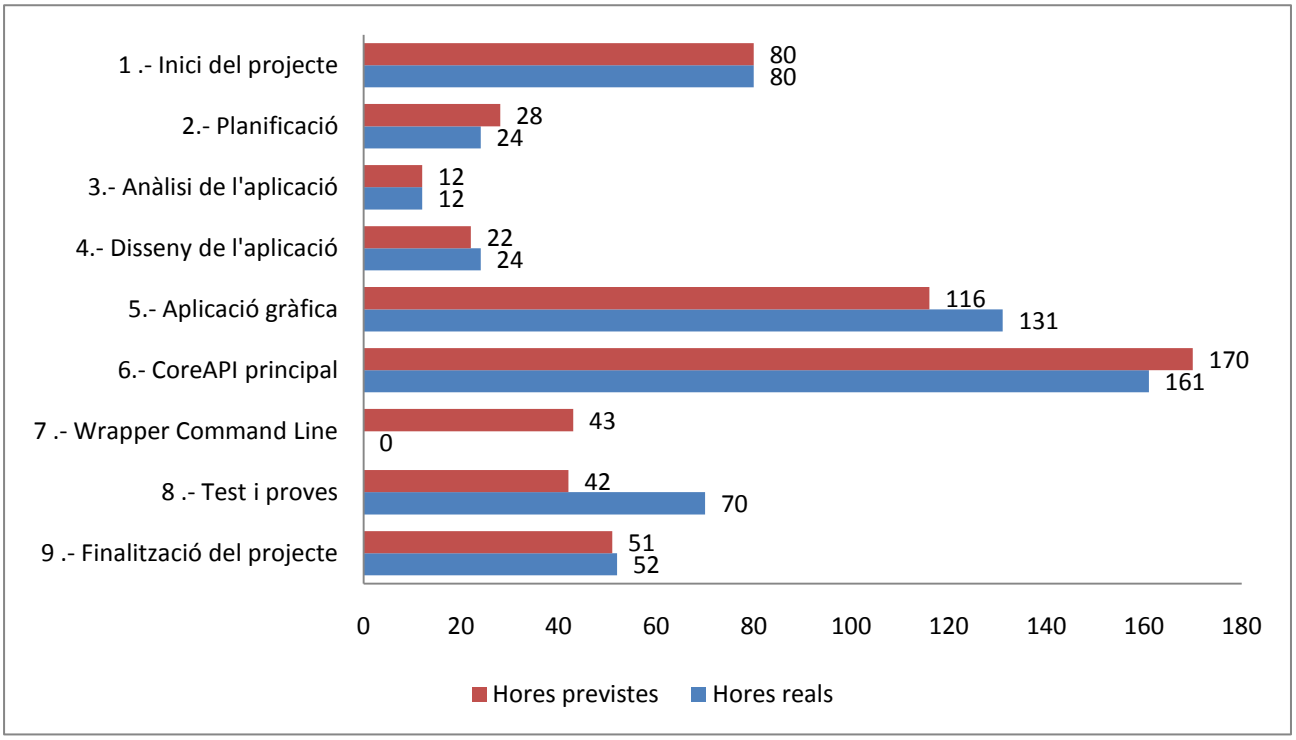


Figura 9 .- Comparativa d'hores per cada fase del projecte

Com podem observar en la figura 9, hi en algunes etapes del projecte hi ha hagut unes desviacions a comentar.

- En l'etapa de realització de l'aplicació gràfica hem necessitat més temps del previst per aconseguir l'objectiu pel fet de que l'aplicació s'havia de generar a partir d'un fitxer XML de configuració.
- Un altre punt que ha rebut modificacions és l'etapa anomenada "Wrapper Command Line" que consistia en poder explotar el coreAPI a partir de línies de comandes. Finalment, es va

desestimar aquesta explotació i destinar aquest temps a la realització dels testos per aconseguir una aplicació que funcionés correctament sense errades.

A continuació veurem una comparativa de dedicació en cada un dels processos de distribució:

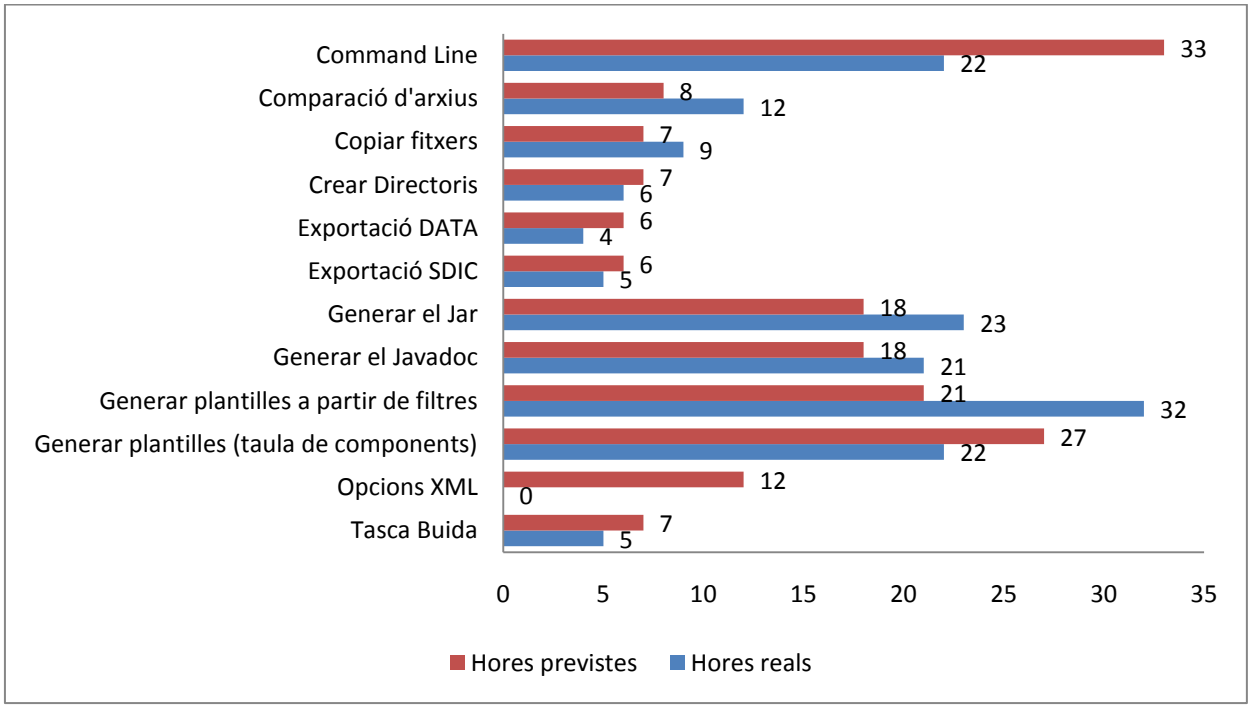


Figura 10 .- Comparativa d'hores per cada tasca

Com es pot observar en la figura 10, hi ha tres tasques que han experimentat canvis importants en quan a hores de dedicació:

- La tasca Command Line ha necessitat menys temps de l'esperat pel simple fet de quan es va realitzar. Al ser una tasca realitzades de les finals, ja havíem vist com generar funcions a partir de línies de comandes.
- La tasca anomenada inicialment "Generar plantilles a partir de filtres", i que com hem vist anteriorment es va acabar dividint en tres tasques, ha necessitat més dedicació per que enlloc de generar una tasca com ens pensàvem inicialment, se n'han generat tres.
- I finalment, la tasca "Opcions XML" no s'ha realitzat ja que s'ha centrat més el projecte en la preparació de la distribució i no en la distribució en si.

7.4. Valoració personal

Durant el meu període en l'empresa Unit4 he pogut conèixer de primera mà el que és treballas en una empresa de desenvolupament de software de primer nivell. D'aquesta manera he pogut complir un conjunt d'objectius personals i professionals/acadèmics. Els professionals o acadèmics són: la finalització del projecte (acadèmic), el coneixement de conceptes com la metodologia Scrum, i finalment augmentar l'experiència en el meu currículum. Personalment he estat capaç de realitzar una feina coordinada i constant, que té com a resultat una eina de treball fàcil d'utilitzar i que pot ser de gran ajuda pels treballadors de l'empresa Unit4.

He tingut la sort de generar un projecte completament en Java, fent només connexions esporàdiques a la plataforma Karat. Aquest fet, m'ha proporcionat la possibilitat de realitzar un projecte Java des de zero complint els objectius inicials del projecte, i trobo que he aconseguit un nivell elevat en el que programació Java es refereix.

Les dificultats al llarg del projecte han sigut moltes i diverses, ja que hi ha moments que sembla que t'encallis i no sàpigues per on sortir. Quan ha passat això he tingut l'ajuda en tot moment del meu tutor dins l'empresa i en casos puntuals dels meus companys de feina, tan becaris com treballadors.

Arribat a aquest punt, és una gran satisfacció haver pogut realitzar aquest projecte que penso que serà utilitzat per l'empresa, però sobretot ha estat una satisfacció haver-lo realitzat obtenint tants beneficis personals.

Bibliografia

A continuació, es detalla la informació que s'ha utilitzat pel desenvolupament del projecte. En aquest cas, es tracta de consultes Web, on inclourem els enllaços i una definició de que hi hem trobat:

1) http://www.wikipedia.org
   Utilitzada en consultes sobre diferents temes (XML, Ant, SourceSafe...).

2) http://intrabus.uab.cat/
   Per la consulta de memòries de projectes ETI de gestió i de sistemes.

3) http://www.chuidiang.com/
   Per exemples de Java, com escriure i llegar fitxers.

4) http://ant.apache.org
   Pàgina web per consultar les diferents tasques de l'Apache Ant.

5) http://www.chuidiang.com/java/herramientas/ant.php
   Pàgina web utilitzada per iniciar-me en l'Apache Ant.

6) http://es.scribd.com/doc/7545519/Manual-JAVA-Swing
   Manual de Java Swing.

7) http://ant.apache.org/manual/Tasks/vss.html#vsshistory
   Pàgina web per consultar les tasques del SourceSafe.

Annex A .- Glossari

Glossari de termes utilitzats en el document.

A

ANT

Eina utilitzada en programació per la realització de tasques mecàniques i repetitives, normalment durant la fase de compilació i construcció. És semblant al Make però desenvolupat en llenguatge Java i que necessita la plataforma Java.

B

Batch (.BAT)

És un arxiu de processament per lots. Es tracta d'arxius sense format, guardats amb l'extensió *.bat que conté un conjunt de comandes DOS. Quan s'executa aquest arxiu bat, les comandes contingudes són executades en grup, de forma seqüencial, permeten automatitzar diferents tasques. Es pot utilitzar qualsevol comanda DOS en un fitxer batch.

BorderLayout

És un Layout que divideix el panel en cinc zones: central, a dalt, altra a sota, a l'esquerra i a la dreta. Podem anar afegint components en les diferents zones i s'aniran situant de forma correcta.

C

CoreAPI

API (Application Programming Interface) és una interfície per la programació d'aplicacions.

El coreApi serà el conjunt de classes que donen funcionalitat a les nostres tasques i que nosaltres explotem a partir de la nostra aplicació gràfica.

D

DATA

És un dels elements que es necessita distribuir en el nostre procés i que està format per taules d'unes bases de dades en concret.

Distribució

És el paquet d'un producte en concret, com per exemple finances, que a l'entregar-se al client pugui ser instal·lat de forma automàtica.

E

Ekon

Solució ERP que Unit4 ofereix als seus clients.

Executora de Tasques

És el nom que pren la nostra aplicació.

F

Fitxer de configuració

És el fitxer XML on definim tota la nostra aplicació (variables globals, interfícies gràfiques i tasques).

J

Jar

Java ARchive – és un tipus d'arxiu que permet executar aplicacions escrites en llenguatge Java.

K

Karat

Plataforma tecnològica pel disseny i implementació d'interfícies d'usuari, desenvolupada en Java per l'empresa Uni4. És independent del sistema operatiu del client.

Karat distributions generator

És l'aplicació que permet a l'usuari de Karat generar la distribució KIZ. En ella s'indiquen aspectes com: dependència sobre altres productes, components que s'han d'incloure i regles.

KIZ

Extensió de la distribució.

L

Layout

En Java, quan fem finestres, és la classe que decideix com es reparteixen els controls dins de la finestra. També és l'encarregat de definir la mida ideal de la finestra en funció dels components que hi hagi a dins.

M

MigLayout

Es tracta d'un tipus de layout molt senzill d'entendre i d'implementar que hem utilitzat per generar els formularis de cada tasca de l'aplicació.

P

Paràmetres globals

Conjunt de paràmetres que es defineixen globalment i que poden ser utilitzats per qualsevol procés en el seu moment d'execució. Estan definits en el fitxer XML de configuració i es poden modificar dins l'Executora de Tasques.

Plantilla .TET

És una plantilla que guarda objectes de SDIC i a partir de la qual es pot generar una exportació d'SDIC.

Procés de distribució

És cada una de les funcionalitats que té el nostre sistema, com per exemple: generar un jar o copiar fitxers.

S

Scrum

Es tracta d'una metodologia de treball que es fa servir a Unit4. Aquesta metodologia guarda informació en una aplicació sobre els components que van modificant, components que nosaltres hem de saber quins són en la tasca "generar plantilles a partir de l'SCRUM".

SDIC

Són les sigles de SuperDiccionari.

SourceSafe

És una eina de control de versions que forma part de Microsoft Visual Studio i que és utilitzada dins l'empresa. A partir d'aquesta eina podem conèixer quines classes s'han modificat i podem generar una plantilla.

SuperDiccionari

Les eines Karat es basen en un contingut central d'informació, el SuperDiccionari, que manté una completa informació de les dades de l'aplicació, independentment de la base de dades on s'executi. S'hi troben components com formularis o objectes de negoci entre d'altres.

T

Tasca

És la representació gràfica de cada procés; és a dir, les diferents funcionalitats que podem executar dins de l'aplicació s'anomenen tasques.

Taula de components

És el nom que fem servir quan ens referim a la taula on els treballadors d'Unit4 que realitzen les distribucions guarden els elements que han anat modificant i que després s'hauran d'incloure en l'exportació.

U

UI .- Interfície d'usuari

UI .- User Interface, és l'abreviatura que utilitzem per parlar de la interfície de treball utilitzada per esmentar parts de l'aplicació gràfica. És molt utilitzada quan parlem dels processos de distribució.

X

XML

Extensible Markup Language, és un metallenguatge extensible d'etiquetes desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació del SGML que permet definir la gramàtica de llenguatges específica. És a dir, és una manera de definir llenguatges per diferents necessitats.

Annex B .- Contingut del CD

- Memòria del projecte en PDF.
- Vídeos de l'aplicació en funcionament.
- Exposicions i documents realitzats a Unit4.
- Jar de l'aplicació (codi de l'aplicació).
- Executable de l'aplicació.
- Fitxer de configuració XML.
- Exemples de fitxers XML per executar les tasques Del Jar i el Javadoc.
- Manual d'usuari de l'aplicació.
- Manual d'instal·lació de l'aplicació.
- Zip amb la documentació de l'aplicació (javadoc).

Josep Costa Sanmartí

Sabadell, ***Juny*** de ***2011***