UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

JDataMotion: unha ferramenta para a visualización dinámica de diagramas de dispersión

Autor:
Pablo Pérez Romaní

Codirectores:
Paulo Félix Lamas
David González Márquez

Grao en Enxeñaría Informática

Febreiro 2015

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

D. Paulo Félix Lamas, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. David González Márquez**, bolseiro de FPU do Centro de Investigación en Tecnoloxías da Información da USC,

INFORMAN:

Que a presente memoria, titulada *JDataMotion: unha ferramenta para a visualización dinámica de diagramas de dispersión*, presentada por **D. Pablo Pérez Romaní** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa dirección no CiTIUS da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 08/09/2014:

Os codirectores, O alumno,

Paulo Félix Lamas David González Márquez Pablo Pérez Romaní

Índice xeral

Índice de figuras

Índice de cadros

Capítulo 1

Introdución

Na actual sociedade da información, onde a cantidade de datos que se manexan aumenta día a día de xeito exponencial, a minería de datos convértese nunha ferramenta fundamental para poder explotalos de maneira eficaz, co fin último de xerar coñecemento a partir dos mesmos.

Para visualizar estes datos unha das técnicas máis utilizadas son os diagramas de dispersión ou scatterplots. Estes permítennos analizar os datos e atopar con facilidade relacións entre as distintas variables, como a correlación entre elas, a distribución dos puntos no plano, a tendencia dos datos recollidos ou outras características que sería complicado extraer a partir dun simple listado, posiblemente desordenado, de vectores de datos. Non obstante, os diagramas de dispersión restrínxennos a unha perspectiva estática do problema. En moitos deses problemas imos encontrar datos cunha compoñente que os sitúa no tempo. Con este proxecto pretendemos dotar a esta representación da súa perspectiva dinámica, para amosar os datos engadindo outro punto de vista que enriqueza a información extraida.

Deséxase desenvolver unha ferramenta para etiquetar cada punto dun diagrama de dispersión cun valor de significado temporal, de tal xeito que este puidese ser empregado como índice nunha visualización dinámica. Este valor numérico podería referenciar dende o momento de captación da tupla que a contén, ata unha ordenación dos datos atendendo á súa prioridade ou relevancia.

1.1. Obxectivos xerais

A motivación principal deste proxecto é o desenvolvemento dunha ferramenta capaz de visualizar a evolución dun conxunto de datos ao longo dunha

magnitude como sería o tempo, ademais de permitir o preprocesado ou manipulación deses datos. Sendo máis específicos, este proxecto busca a realización da análise, deseño e implementación dunha aplicación que consiga:

- Facilitarlle ao usuario o procesado de volumes de datos dun tamaño significativo.
- Posibilitar o traballo con formatos de arquivo CSV ou ARFF.
- Dispor das funcionalidades necesarias para manipular os datos.
- Ser capaz de amosar os datos en forma de diagramas de dispersión, con funcións de reprodución básicas. Tamén se debe posibilitar a configuración desta reprodución por parte do usuario.
- Aplicar filtros nos datos cos que se traballa, de xeito que se poidan eliminar datos fora dun rango, normalizar os seus valores, etc.
- Interaccionar co usuario por medio dunha interface simple e amigable.
- Aplicar nun caso real a ferramenta JDataMotion, para apreciar a súa utilidade.
- Finalizar o desenvolvemento do proxecto antes do día 13 de Febreiro de 2015.

1.2. Relación da documentación

Esta memoria plasma o proceso de desenvolvemento do proxecto JDataMotion, que persegue os obxectivos citados no apartado anterior.

Os distintos capítulos repártense do modo que segue:

Capítulo 1. Introdución: composta por obxectivos xerais, relación da documentación que conforma a memoria, descrición do sistema (métodos, técnicas ou arquitecturas utilizadas e xustificación da súa elección).

Capítulo 2. Planificación e presupostos: inclúe a estimación dos recursos necesarios para desenvolver este proxecto, xunto co custo (presuposto) e planificación temporal do mesmo, así como a súa división en fases e tarefas.

Capítulo 3. Especificación de requisitos: inclúe a especificación do sistema, xunto coa información que este debe almacenar e as interfaces con outros sistemas, sexan hardware ou software, e outros requisitos (rendemento, seguridade, etc).

Capítulo 4. Deseño: rexistra como se realiza o sistema, a división deste en diferentes compoñentes e a comunicación entre eles. Así mesmo, neste apartado determínase o equipamento hardware e software necesario.

Capítulo 5. Exemplos: Avaliación do grao de cumprimento dos requisitos e tests os verifican.

Capítulo 6. Conclusións e posibles ampliacións.

Apéndice A. Manuais técnicos: incluirase toda a información precisa para aquelas persoas que se vaian a encargar do desenvolvemento e/ou modificación do sistema.

Apéndice B. Manuais de usuario: incluirán toda a información precisa para aquelas persoas que utilicen o sistema: instalación, utilización, configuración, mensaxes de erro, etc.

Apéndice C. Licenza.

Bibliografía

Capítulo 2

Xestión do proxecto

Neste capítulo comentaremos distintos aspectos relacionados coa planificación de como se vai xestionar este proxecto. Falaremos, por exemplo, da xestión de riscos que conleva o desenvolvemento do software, así coma os métodos de continxencia, prevención ou minimización que seguiremos en caso da incidencia dos mesmos. Cos riscos expostos, abordaremos a metodoloxía de desenvolvemento máis axeitada para o proxecto, de acordo tamén cos obxectivos anteriormente plasmados. Seguiremos coa planificación temporal do proxecto e finalizaremos coa estimación de custo e prazos, así como a xestión da configuración.

2.1. Xestión de riscos

Na fase de planificación dun proxecto hai que sopesar os distintos riscos aos que estará exposto o seu desenvolvemento, cuantificalos e deseñar estratexias para a súa aparición. Algúns dos riscos nun Traballo de Fin de Grao poden ter graves consecuencias na liña base do proxecto debido á inexperiencia do seu autor, polo que fronte á falta de experiencia hai que esforzarse en mellorar a planificación.

Na análise de riscos valoraremos a probabilidade de aparición e a súa gravidade, para a continuación deseñar unha medida de continxencia, prevención ou minimización. A escala de valoración da probabilidade e da gravidade vai ser:

- Moi baixa
- Baixa
- Media
- Alta
- Moi alta

Os riscos considerados son os seguintes:

- **Risco 01**

  Nome: Cambios no alcance durante o desenvolvemento

  Descrición: A lista inicial de requisitos funcionais que se captará nas primeiras reunións cos titores vai sufrir modificacións, incluso co proxecto en etapas avanzadas de desenvolvemento. A súa probabilidade duplícase pola existencia de dous clientes no Traballo de Fin de Grao.

  Probabilidade: Moi alta

  Gravidade: Alta

  Medidas de minimización: Botaremos man da folgura temporal do proxecto (marxe de tempo dispoñible para eventualidades). Os cambios razoaranse cos titores, presentando a lista de requisitos actual e valorando a parte da folgura que consumirían ditos cambios. Tamén se tratará de ter reunións de avaliación cos titores cunha alta frecuencia, para así detectar o antes posible calquera cambio nos requisitos, se ben pode acontecer que se propoñan cambios sobre as primeiras etapas cando o proxecto se atopa en etapas avanzadas.

- **Risco 02**

  Nome: Imprecisión á hora de fixar entregables

  Descrición: A inexperiencia do alumno manifestarase xa nas primeiras entregas programadas. Ao non ter traballado previamente en proxectos desta índole, resultará complicado estimar os prazos de entrega nas primeiras fases do proxecto, tanto por exceso como por defecto.

  Probabilidade: Alta

  Gravidade: Media

  Medidas de minimización: Intentaremos especificar entregas dun contido menor e máis frecuentes, sobre todo nas primeiras fases, para que sexa máis doado comezar a estimar correctamente os prazos de entrega.

- **Risco 03**

  Nome: Imposibilidade de reunirse cun dos titores

  Descrición: Un dos titores non pode acudir a algunha reunión proposta, nin estará nos seguintes 5 días.

  Probabilidade: Alta

Gravidade: Baixa

Medidas de minimización: Desenvolverase a reunión co titor dispoñible, sendo mester informar das conclusións sacadas ao outro titor por medio do correo electrónico en canto remate a reunión.

- **Risco 04**

  Nome: Descoñecemento ou inexperiencia coas solucións

  Descrición: O alumno non coñece as posibilidades que teñen as ferramentas das que dispón (librarías, módulos, solucións, etc.).

  Probabilidade: Alta

  Gravidade: Media

  Medidas de prevención: Adicarase un tempo prudencial, nas primeiras fases, a revisar as APIs e a documentación en xeral das librerías, proxectos de terceiros e demais ferramentas que se van empregar, para ser conscientes de como poden solucionar as nosas necesidades.

- **Risco 05**

  Nome: Imposibilidade de finalizar o proxecto en tempo

  Descrición: A folgura está esgotada, e o cumprimento do prazo de entrega vese ameazado.

  Probabilidade: Media

  Gravidade: Moi alta

  Medidas de prevención: Evitaremos na medida do posible recorrer á folgura, e trataremos de seguir a planificación do xeito máis estrito que podamos.

  Medidas de minimización: Poremos en coñecemento aos titores do estado do proxecto e dos seus prazos, para discutir a modificación ou eliminación dalgúns ítems da especificación.

- **Risco 05**

  Nome: Limitación das librarías gráficas

  Descrición: As APIs e librarías gráficas non dan solución todas as nosas necesidades

  Probabilidade: Media

  Gravidade: Alta

Medidas de prevención: Estudiar a especificación de cada solución e as funcionalidades que ofrece.

Medidas de minimización: Implementarase de xeito manual a solución que se necesita, podendo partir ou botar man do código da librería.

2.2. Metodoloxía de desenvolvemento

A elección da metodoloxía de traballo é un paso importante na planificación de calquera proxecto, xa que a posteriori influirá en varios aspectos deste: a xestión dos seus riscos, a súa tolerancia a cambios externos, a confianza na súa validez, etc. Hai dous enfoques fundamentais: as metodoloxías estritas e as metodoloxías áxiles. As primeiras esixen unha planificación estrita, practicamente inmutable e necesariamente realista de todo o plan de traballo, e son boas cando o conxunto de requisitos é fixo e moi concreto. As segundas, pola contra, son flexibles e adáptanse ben a cada situación, pois nelas asúmese que se van producir variacións nos requisitos.

Sopesando as circunstancias nas que se desenvolve un Traballo de Fin de Grao, onde a experiencia do alumno é practicamente nula no que respecta á xestión de proxectos, semella que deberíamos adoptar unha metodoloxía de traballo que se adapte ás necesidades de cambios que vaian xurdindo, e que consiga en cada entrega recibir certa retroalimentación por parte dos titores, de forma que tras cada iteración podamos ter a seguridade da correspondencia entre o proxecto e o modelo mental de quen o especificou. É dicir, necesitamos unha metodoloxía áxil.

Dentro do compendio de metodoloxías áxiles existentes, decantarémonos pola metodoloxía Scrum [7], pois enfoca todas as súas avaliacións sobre entregas parciais, pero funcionais, para facilitarlle ao receptor do proxecto a valoración do mesmo. Necesítase, polo tanto, unha gran implicación do cliente no proxecto, algo que se pode conseguir dada a dualidade da titoría (é máis doado que haxa un titor dispoñible para realizar a reunión). Por outra banda, os requisitos que constitúen as distintas entregas deben estar priorizados para que o proxecto poida avanzar cun carácter incremental, e ditas entregas deben resultar usables para o cliente.

Scrum define unha serie de ferramentas e de regras, idóneas para levar a cabo o desenvolvemento de proxectos que buscan unha metodoloxía áxil. Os preceptos básicos desta metodoloxía son:

- Adoptar unha estratexia de desenvolvemento incremental, no canto da planificación e execución completa do produto (é dicir, no canto dunha meto-

doloxía estrita).

- Basear a calidade do resultado máis no coñecemento das persoas que o especificaron ca na calidade dos procesos empregados.
- Solapamento das fases de desenvolvemento (análise, deseño, implementación e probas) no canto da sucesión secuencial que nos ofrecen metodoloxías como a fervenza.

A metodoloxía Scrum comeza coa adquisición de requisitos en reunión co cliente, da cal se extrae un Backlog, é dicir, unha lista ordenada por prioridade de requisitos funcionais (RFs). Ademais, Scrum define o sprint como unidade elemental de tempo de traballo. Un sprint dura entre 1 e 4 semanas, aínda que nós trataremos de manter a súa duración en 2 ou incluso 1 semanas para maximizar a supervisión e xestionar ben os riscos, sobre todo nas etapas iniciais.

Ao término de cada reunión co cliente, revísase o Backlog e incorpórase un certo número de RFs a un novo sprint, o cal dará comezo en canto remate a reunión. Para o remate dese novo sprint (dentro dunha semana no noso caso) terase programada a seguinte reunión, na que se valorará o sprint finalizado e se accederá ao Backlog para acordar o seguinte sprint, e así sucesivamente. A valoración do sprint en cada reunión realízase presentando a lista de RFs de dito sprint, e demostrando ante o receptor do proxecto que cada un dos ítems ou tarefas do sprint funciona correctamente.

Para regular o desenvolvemento desa metodoloxía pódese botar man de diversas ferramentas, de entre as cales nós escollemos Acunote [8] para o noso proxecto. Acunote é unha aplicación web especialmente deseñada para a xestión da metodoloxía Scrum. Ten varios plans de prezos, pero nós empregaremos o gratuíto porque as nosas necesidades restrínxense a un equipo de persoal pequeno (o alumno e os dous titores). Entre as prestacións desta ferramenta, sacarémoslle maior proveito ás seguintes:

Wiki: Empregarémola para engadir contido visible ao resto de membros do grupo.

Lista de sprints: Amosa os sprints en 3 grupos: sprints pasados, sprints presentes e sprints futuros. Ao abrir un sprint visualízanse os ítems ou tarefas (requisitos funcionais no noso caso) que o compoñen. Accederemos a este apartado na maioría dos casos para crear novos sprints.

Sprint actual: Visualiza os RFs do sprint actual. A medida que se vaian completando requisitos funcionais, accederase a esta lapela para cambiar o estado do requisito en cuestión. Os estados posibles son:

- Non comezado (por defecto)
- En progreso

- Reaberto
- Bloqueado
- Completado
- Verificado
- Duplicado
- Non se vai realizar

Backlog: Contén todos os ítems (requisitos funcionais) pendentes de ser asignados a un sprint. Este lista cumprimentarase ao principio, cos requisitos funcionais captados e ordenados por prioridade, e logo accederase a ela á hora de asignar RFs aos novos sprints. Na extracción de RFs débese respectar a orde dos mesmos dentro da lista, collendo sempre un número de ítems determinado da parte superior. Estes ítems desaparecerán do Backlog en canto sexan asignados.

Tarefas: Mostra todos os ítems especificados, independentemente de que fosen asignados a un sprint ou non.

2.3. Planificación temporal

A metodoloxía Scrum caracterízase como ben dixemos polo solapamento das fases que nun modelo en fervenza estarían ben separadas. As primeiras semanas de traballo estarán adicadas á análise para a captación inicial de requisitos, pero nas sucesivas iteracións ou sprints poderán realizarse en paralelo análise, deseño, implementación e probas. Esta é unha das licencias que outorga o emprego das metodoloxías áxiles. De todos xeitos, aínda dentro da variabilidade destas metodoloxías, podemos dividir a vida do proxecto en unha serie de fases fundamentais:

Inicio: Constitúe o primeiro sprint (Sprint 00) da planificación, e durará dúas semanas. Nesta fase programaranse reunións cos titores para realizar a captación de requisitos funcionais (análise de requisitos), e ordenaranse estes por prioridade, dando lugar ao Backlog. Tamén se definirá a especificación de cada requisito e se deseñarán as probas que os verifiquen.

Desenvolvemento: Abrangue dende o Sprint 01 ata o Sprint 11, ambos inclusive (20 semanas en total). Nesta fase elaborarase o produto de acordo cos requisitos.

Documentación: Abrangue 5 semanas de traballo. Nesta fase recompilarase toda a documentación xerada nas fases anteriores, e confeccionarase a me-

moria e máis a presentación, que constituirán os entregables do Traballo de Fin de Grao.

En total, o proxecto traballarase durante un período de 27 semanas (189 días, algo máis de 6 meses), co cal, para realizar as 401,25 horas de traballo necesarias teremos que levar un ritmo de traballo aproximado de 15,28 horas semanais (unha media de 2 horas e cuarto diarias). Non nos convén asumir un ritmo de traballo maior, pois durante ese período de tempo o alumno deberá repartir a súa axenda entre este proxecto, o resto de materias, as prácticas en empresa, etc.

2.4. Xestión da configuración

Todo proxecto ten elementos de interese para incluír na xestión da configuración. Estes elementos caracterízanse porque son candidatos a sufrir cambios que poden ameazar o correcto desenvolvemento do proxecto. A xestión da configuración trata de manter a integridade do proxecto perante a estes cambios. O noso deber é identificar que obxectos do proxecto (sexan entregables ou resultados parciais do mesmo) merecen a súa inclusión na xestión da configuración, e por outra parte, temos que especificar que ferramentas empregaremos para dar soporte a esta característica.

Para este caso consideraremos ao código fonte do proxecto (e máis das súas probas) e á documentación como elementos de configuración. O código fonte é o sustento do noso proxecto, e os cambios no seu contido veranse directamente reflexados no produto a entregar, polo que é mester incluír este elemento na xestión da configuración. Tamén incluiremos o código fonte das probas porque debemos respectar a integridade entre estas e o propio proxecto. Por outra parte, a documentación sufrirá cambios de xeito paralelo ao código, e evolucionará da man deste ao longo da vida do proxecto (rexistrará a súa especificación de requisitos, o seu deseño, etc.), así que tamén debe ser un elemento a considerar en aras de preservar a integridade do proxecto. En resumo, faremos seguimento de cambios dos tres directorios ('src', 'test' e 'doc'), e para iso botaremos man do software GitHub [9].

Github é unha plataforma para darlle aloxamento a distintos tipos de proxectos, por medio do sistema de control de versións Git. Para aloxar o noso proxecto crearemos un repositorio local e outro remoto, chamando a ambos 'JDataMotion', e outorgándolle ao remoto permisos de lectura e escritura para o alumno e permisos de lectura (ou de escritura tamén, opcionalmente) para os titores. Deste xeito estes poderán descargar a última versión do proxecto en calquera momento, mentres que o alumno poderá ir subindo as modificacións cos cambios implementados cada certo tempo.

O proxecto conterá moitos máis elementos, pero non podemos consideralos a todos aptos para a xestión de configuración por diversos motivos: as librarías empregadas non cambian (e no caso de querer actualizar algunha, asúmese que non deben xurdir problemas de integridade grazas á compatibilidade entre versións), os arquivos de configuración persoal non deben ser almacenados no repositorio, e os ficheiros de código obxecto e de distribución dependen directamente do código fonte (que xa é un elemento de configuración), pois xéranse como resultado da súa compilación.

2.5. Análise de custos

A estimación dos custos de desenvolvemento do proxecto amósase no Cadro 2.1. Para ela, consideramos a adquisición dun novo equipo informático. As horas de traballo neste caso non van ter un valor económico asociado, como consecuencia de que este proxecto pertenza a un Traballo de Fin de Grao, pois as horas do traballo do alumno correspóndense coas que este debe cumprimentar para a obtención do título. Para o consumo eléctrico tivemos en conta o prezo do kWh en España [5] e fixemos unha estimación [6] do consumo eléctrico dun equipo informático, que a plena potencia pode traballar a 120 W. Considerando que o desenvolvemento do traballo durará 401,25 horas, necesitaremos 120 W * 401,25 h = 48150 Wh = 48,15 kWh. Ademais, sabemos que o salario medio anual dun desenvolvedor de software en España é de 26740 /ano. Se un ano ten 249 días laborables, o custo por hora deste traballador sería de (26740 €/ano) / (249 días laborables/ano) / (8 horas/día laboral) = 13,4237 €/hora. Os custos inclúen o IVE, pero trataremos de diferencialos na seguinte táboa.

Activo	**Cantidade**	**C.U. sen IVE**	**IVE**	**Custo total**
Ordenador portátil	1	570,00 €	21 %	689,70 €
Horas de traballo	401,25 horas	11,094 €/hora	21 %	5386,26 €
Consumo eléctrico	48,15 kWh	0,0663 €/kWh	21 %	3,86 €
			Total	6079,82 €

Cadro 2.1: Custos

A anterior estimación é válida só en caso de que o produto a desenvolver non teña unha finalidade comercial. No caso de que este se pretenda comercializar, certas ferramentas que imos utilizar poderían esixir o pago dunha licencia específica.

Capítulo 3

Análise

Na fase de análise do proxecto intentaremos discernir cales van ser as metas ou obxectivos do mesmo. Ao seu remate deberemos ter claro cales son os obxectivos que debe alcanzar o proxecto, así como ter definidos os pasos ou tarefas para logralos. Tras isto, nesta etapa tamén incorporaremos un estudo e valoración das tecnoloxías das que botaremos man.

3.1. Análise de requisitos

A extracción dos requisitos dun proxecto é unha fase fundamental na realización de calquera proxecto, pois inflúe non só nas propias tarefas a desenvolver para a súa implementación, se non tamén na valoración do produto final e da súa calidade. O proceso desta etapa pódese revisar na norma IEEE-STD-830-1998 [11]. A obtención de requisitos adóitase facer durante ou tras unha reunión cos clientes.

Durante a reunión cos clientes foron xurdindo requisitos ou condicións necesarias para o produto. Estes requisitos foron rexistrados para posteriormente seren ordenados e clasificados segundo certos criterios que se amosan a continuación.

Os casos de uso empréganse para modelar e representar cómo se vai realizar a interacción entre o sistema e os usuarios del, tamén coñecidos como actores. Os casos de uso constitúen as posibilidades das que dispón cada actor. Esta análise resulta especialmente útil en entornas orientadas a usuarios con distinta prioridade (un administrador, un usuario invitado, un usuario rexistrado, un usuario prémium, etc.) nas que cada un deses actores ten acceso a uns casos de uso específicos (por exemplo, moitas aplicacións web). Por tanto, a riqueza dos diagramas de casos de uso radica na variedade de tipos de usuario (actores). A nosa

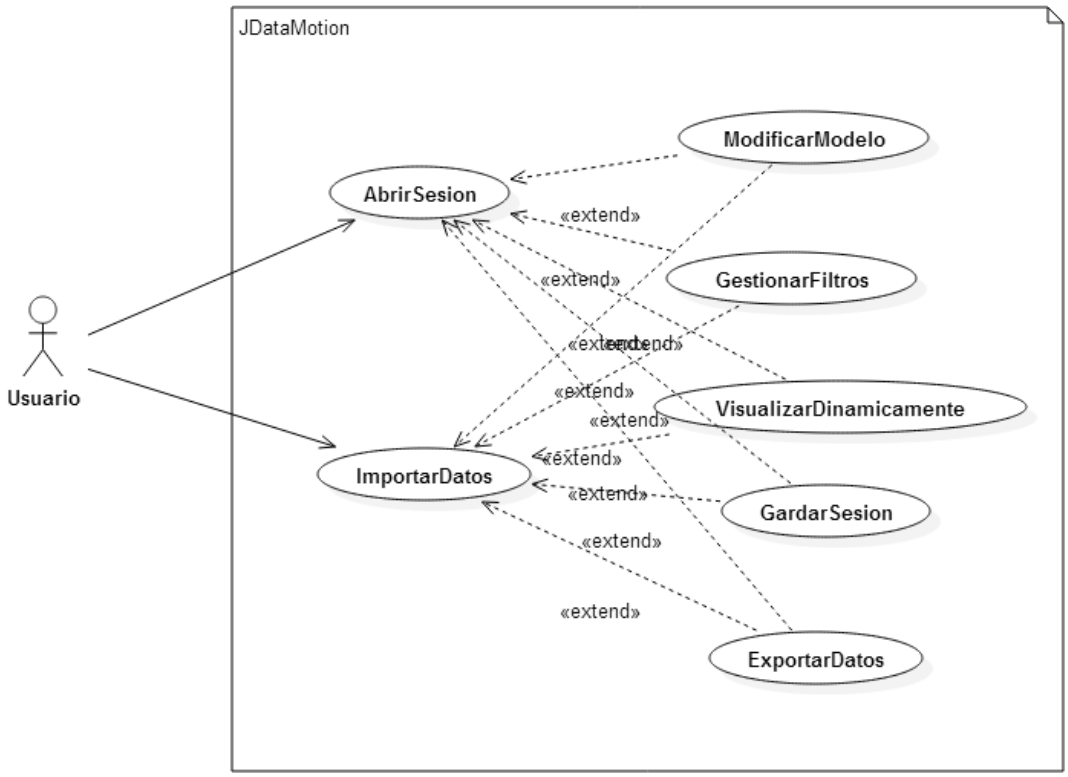

Figura 3.1: Diagrama de casos de uso

aplicación non necesita facer distinción algunha entre os tipos de usuario que poden facer uso dela. Todos van dispor das mesmas funcionalidades. De todas formas, o diagrama de casos de uso pódese apreciar na figura 3.1

Un escenario de caso de uso podería comezar cun traballador do ámbito médico-sanitario que dispón nun csv (exportado por outra aplicación, por exemplo) de certas medicións relacionadas cun paciente seu ao longo do seu seguimento. O traballador (usuario do sistema) podería importar o arquivo dentro do JDataMotion e obter unha táboa editable cos datos contidos, aplicar filtros para eliminar os datos atípicos das medicións do paciente e normalizar algunhas variables. Cos datos xa preprocesados, poderá reproducilos para ver o seu comportamento ao longo do tempo e a evolución do paciente. Finalmente, gardará os datos filtrados baixo un novo csv.

3.1.1. Requisitos funcionais

RF01

Título
Importar arquivos con datos para o experimento

Descrición
A aplicación debe permitir cargar do sistema de arquivos un ficheiro que conteña unha secuencia de datos (nun formato axeitado segundo o RNF01) para ser utilizados no experimento.

Importancia
Esencial

RF02

Título
Exportar datos

Descrición
A aplicación debe permitir almacenar nun arquivo o conxunto de datos do experimento actual (tendo en conta filtrados, modificacións, datos engadidos ou eliminados...). Os arquivos de saída deberán respectar o RNF01 en canto a formato de almacenamento. Importancia Esencial

RF03

Título
Gardar sesión

Descrición
A aplicación debe permitir gardar en disco a sesión (ou experimento) actual tal e como está no momento de executar esta acción.

Importancia
Esencial

RF04

Título
Abrir sesión

Descrición
A aplicación debe permitir restaurar unha sesión (ou experimento) gardada anteriormente, de xeito que se atope exactamente igual ca no momento en que se gardou.

Importancia
Esencial

RF05

Título
Representar os datos en forma de táboa

Descrición
A aplicación debe ser capaz de amosar os datos segundo unha táboa na que figuren cabeceiras, tipos, valores, etc.

Importancia
Esencial

RF06

Título
Insertar datos no experimento actual

Descrición
A aplicación debe permitir a inserción dinámica de datos no experimento actual.

Importancia
Esencial

RF07

Título
Modificar datos no experimento actual

Descrición
A aplicación debe permitir a modificación dinámica de datos no experimento actual.

Importancia
Esencial

RF08

Título
Eliminar datos no experimento actual

Descrición
A aplicación debe permitir a eliminación dinámica de datos no experimento actual.

Importancia
Esencial

RF09

Título
Asignar tipos aos atributos dun arquivo importado

Descrición
A aplicación debe permitir especificar os tipos de atributos presentes no arquivo importado. Por exemplo, os datos cuantitativos poderían ser enteiros ou reais, mentres que os cualitativos serían algo distinto (mesmamente strings).

Importancia
Esencial

RF10

Título
Sinalar identificación temporal

Descrición
A aplicación debe permitir sinalar unha columna que exprese o orde ou a temporalidade dunha tupla, ou ben definir esta columna manualmente.

Importancia
Esencial

RF11

Título
Representar os datos graficamente mediante diagrama de dispersión

Descrición
A aplicación debe ser capaz de representar graficamente (mediante diagrama de dispersión) o conxunto de parámetros de entrada. Concretamente, débense poder representar ata 3 parámetros por cada diagrama de dispersión (ordeadas, abscisas e cor e forma dos puntos). Todos os diagramas de dispersión estarán englobados dentro do "menú de visualización", que cumprirá co RNF06.

Importancia
Esencial

RF12

Título
Engadir diagramas de dispersión ao menú de visualización

Descrición
A aplicación debe permitir engadir dinámicamente novos diagramas de dispersión dentro do menú de visualización.

Importancia
Esencial

RF13

Título
Eliminar un diagrama de dispersión do menú de visualización

Descrición
A aplicación debe permitir eliminar un diagrama de dispersión do menú de visualización.

Importancia
Esencial

RF14

Título
Configurar diagramas de dispersión do menú de visualización

Descrición
A aplicación debe permitir especificar para os diagramas de dispersión do menú de visualización a súa configuración, respecto a que parámetros se

representarán en cada un dos eixos ou si as cores e formas dos puntos se desexan usar para representar algún atributo nominal.

Importancia
Esencial

RF15

Título
Detallar punto seleccionado dentro do diagrama de dispersión

Descrición
Cada punto dos diagramas de dispersión pode ser seleccionado para ver nun apartado os seus detalles (todos os seus atributos).

Importancia
Esencial

RF16

Título
Resaltar punto en diagramas de dispersión

Descrición
Cada punto seleccionalo dentro dun diagrama de dispersión resaltarase tanto nel coma en todos os demais diagramas de dispersión (que plasmarán outras proxeccións do mesmo punto).

Importancia
Esencial

RF17

Título
Desprazar a ventá de visualización por arrastre de cada diagrama de dispersión

Descrición
Para cada diagrama de dispersión poderemos usar unha ferramenta “man” para desprazar a ventá polo diagrama de dispersión.

Importancia
Esencial

RF18

Título
Escalar a ventá de visualización de cada diagrama de dispersión

Descrición
Para cada diagrama de dispersión poderemos usar unha ferramenta de escalado da ventá para facer zoom no diagrama de dispersión.

Importancia
Esencial

RF19

Título
Escalar e reposicionar dinamicamente

Descrición
Para cada diagrama de dispersión permitirase que a ventá de visualización que o enfoca se adapte dinamicamente ao conxunto de datos representados (movéndose, afastándose e aproximándose para englobar todos os datos).

Importancia
Esencial

RF20

Título
Reproducir a secuencia de datos

Descrición
A aplicación debe de permitir que a visualización dos diagramas de dispersión poida basearse na variable temporal (ou de orde) para reproducir a secuencia de datos, amosando os datos de cada diagrama de dispersión baixo unha secuencia de vídeo. Nesta secuencia engadiríase á visualización en cada instante a tupla de atributos asociada a esa marca temporal.

Importancia
Esencial

RF21

Título
Representar estela

Descrición
A aplicación debe de permitir que cada novo punto pintado se ligue ao último representado no diagrama de dispersión por medio dunha liña recta.

Importancia
Esencial

RF22

Título
Difuminar estela ao longo da reprodución

Descrición
A aplicación debe permitir difuminar as estelas xa representadas a través do avance temporal.

Importancia
Esencial

RF23

Título
Configurar a reprodución da secuencia de datos

Descrición
A aplicación debe de permitir que a visualización dos diagramas de dispersión sexa configurable en canto a tempo transcorrido entre marcas temporais. Para a reprodución usando marcas temporais ponderadas, este tempo representará a separación entre as dúas marcas temporais mais próximas (tempo mínimo). Ademáis débese poder especificar o número de marcas temporais que durará o difuminado dos puntos que se ploteen, de xeito que durante ese intervalo cada punto se vaia difuminando ata desaparecer. Pode ser igual a 0 para que os puntos non se difuminen.

Importancia
Esencial

RF24

Título
Pausar a reprodución

Descrición
A aplicación debe permitir parar a reprodución na marca de tempo na

que se atope ao executar esta acción, mantendo as visualizacións para ese momento.

Importancia
Esencial

RF25

Título
Ir a un determinado instante dentro do intervalo temporal da reprodución

Descrición
A aplicación debe permitir situarse directamente sobre un instante de tempo, mantendo a reprodución pausada sobre esa marca temporal, e visualizando os diagramas de dispersión tal e como deben estar nese momento.

Importancia
Esencial

RF26

Título
Insertar filtros para os datos do experimento

Descrición
A aplicación debe permitir engadir unha serie de filtros que se aplicarán de xeito secuencial sobre a secuencia de datos coa que se esté a traballar. Chamarémoslle "secuencia de filtros" a esta secuencia.

Importancia
Esencial

RF27

Título
Eliminar un filtro para os datos do experimento

Descrición
A aplicación debe permitir eliminar un determinado filtro dentro da secuencia de filtros.

Importancia
Esencial

RF28

Título
Configurar filtros para os datos do experimento

Descrición
A aplicación debe permitir seleccionar un determinado filtro dentro da secuencia de filtros para modificar a regla de filtrado implícita.

Importancia
Esencial

RF29

Título
Gardar unha secuencia de filtros do experimento

Descrición
A aplicación debe permitir gardar unha secuencia de filtros, non necesariamente correlativos, dentro dos que se estean aplicando sobre o experimento. Esta secuencia pode comprender tanto un só filtro como a secuencia de filtros enteira.

Importancia
Esencial

RF30

Título
Cargar unha secuencia de filtros para o experimento

Descrición
A aplicación debe permitir cargar do sistema de arquivos unha secuencia de filtros que se engadirá á cabeza da secuencia de filtros (a cal pode estar baleira). Esta secuencia tamén pode estar composta por un só filtro.

Importancia
Esencial

RF31

Título
Mover os filtros dentro da secuencia de filtros

Descrición
A aplicación debe permitir desprazar un filtro dentro da secuencia de filtros do experimento, de xeito que o orde de aplicación dos filtros varíe. O desprazamento realizarase inserindo o filtro en cuestión nunha nova posición.

Importancia
Esencial

RF32

Título
Configurar o menú de visualización

Descrición
A aplicación debe permitir cambiar os parámetros de visualización dos diagramas de dispersión que compoñen o menú de visualización, por exemplo, a cor das etiquetas e lendas, do fondo, dos eixos... ou a fonte, tamaño de letra...

Importancia
Optativa

3.1.2. Requisitos de calidade

RC01

Título
Latencia mínima para o procesamento

Descrición
A aplicación debe responder nun tempo razoable ás operacións executadas polo usuario, e intentar que esa latencia escale de xeito controlado ao aumentar a talla dos parámetros.

Importancia
Esencial

3.1.3. Requisitos de deseño

RD01

Título
Modularidade no deseño dos filtros

Descrición
A aplicación debe facilitar unha interface para a inclusión e uso de filtros personalizados por parte de calquera desenvolvedor de software que a implemente dentro do proxecto.

Importancia
Esencial

3.1.4. Requisitos non funcionais

RNF01

Título
Formatos de arquivo admitidos ao importar e exportar arquivos

Descrición
A aplicación debe estar preparada para importar e exportar arquivos en distintos formatos, como son o CSV e ARFF.

Importancia
Esencial

RNF02

Título
Relación programa-sesión

Descrición
Cada instancia do programa debe traballar cunha única sesión (experimento).

Importancia
Esencial

RNF03

Título
Implementación en Java

Descrición
O software tense que desenvolver na linguaxe de programación Java.

Importancia
Esencial

RNF04

Título
Representación matricial dos diagramas de dispersión

Descrición
Os diagramas de dispersión represéntanse de xeito matricial, facendo que cada parámetro dentro dun eixo sexa enfrontado a cada un dos demais do outro eixo, e en cada punto desa dupla se sitúe o diagrama de dispersión que compara ambos parámetros. Deste xeito, os diagramas de dispersión non son acumulables: se temos un que representa X (abscisas) fronte a Y (ordenadas), non podemos engadir outro que represente X (abscisas) fronte a Y (ordenadas), pois ocuparían ambos a mesma cela dentro da matriz de diagramas de dispersión.

Importancia
Esencial

RNF05

Título
Entrega dentro de prazo

Descrición
Débese entregar unha versión funcional e documentada antes do día 13 de Febreiro de 2014, ás 14:00 horas, pois é o momento no que remata o prazo de entrega.

Importancia
Esencial

3.1.5. RFs dos sprints

Imos a detallar a asignación de requisitos funcionais (RFs) aos distintos sprints ao longo da fase de Desenvolvemento, asignando uns prazos aproximados de traballo sobre uns conxunto de RFs relacionados entre si.

Sprint 01

Nome: Interacción co sistema de ficheiros

Fase: Desenvolvemento

Comezo: 17/02/2014

Finalización: 24/02/2014

RFs a implementar: RF01, RF02, RF03, RF04

Sprint 02

Nome: Manipulación de datos

Fase: Desenvolvemento

Comezo: 24/02/2014

Finalización: 10/03/2014

RFs a implementar: RF05, RF06, RF07, RF08

Sprint 03

Nome: Preprocesado

Fase: Desenvolvemento

Comezo: 10/03/2014

Finalización: 24/03/2014

RFs a implementar: RF09, RF10

Sprint 04

Nome: Visualización dos datos

Fase: Desenvolvemento

Comezo: 24/03/2014

Finalización: 14/04/2014

RFs a implementar: RF11, RF12, RF13, RF14

Sprint 05

Nome: Ferramentas de visualización

Fase: Desenvolvemento

Comezo: 14/04/2014

Finalización: 28/04/2014

RFs a implementar: RF15, RF16, RF17, RF18, RF19

Sprint 06

Nome: Reprodución

Fase: Desenvolvemento

Comezo: 28/04/2014

Finalización: 05/05/2014

RFs a implementar: RF20

Sprint 07

Nome: Configuración da reprodución

Fase: Desenvolvemento

Comezo: 05/05/2014

Finalización: 19/05/2014

RFs a implementar: RF21, RF22, RF23

Sprint 08

Nome: Funcións de reprodución

Fase: Desenvolvemento

Comezo: 19/05/2014

Finalización: 02/06/2014

RFs a implementar: RF24, RF25

Sprint 09

Nome: Filtros

Fase: Desenvolvemento

Comezo: 02/06/2014

Finalización: 16/06/2014

RFs a implementar: RF26, RF27, RF28

Sprint 10

Nome: Xestionar filtros

Fase: Desenvolvemento

Comezo: 16/06/2014

Finalización: 30/06/2014

RFs a implementar: RF29, RF30, RF31

Sprint 11

Nome: Outras funcións de visualización

Fase: Desenvolvemento

Comezo: 30/06/2014

Finalización: 07/07/2014

RFs a implementar: RF32

A partir da planificación temporal de grupos de RFs en común podemos estimar con certa confianza a duración total do proxecto, polo menos na súa fase de desenvolvemento. Faltarían dúas fases máis por considerar:

- Fase de inicio, previa á execución dos sprints, duraría un prazo de 2 semanas e constaría das seguintes tarefas:
  - Lectura da especificación e documentación
  - Reunión cos directores do proxecto
  - Redacción do anteproxecto
- Fase de documentación, posterior á execución dos sprints, duraría un prazo de 5 semanas e constaría das seguintes tarefas:
  - Revisión e documentación do código
  - Compilación de documentos cos sprints
  - Redacción da memoria

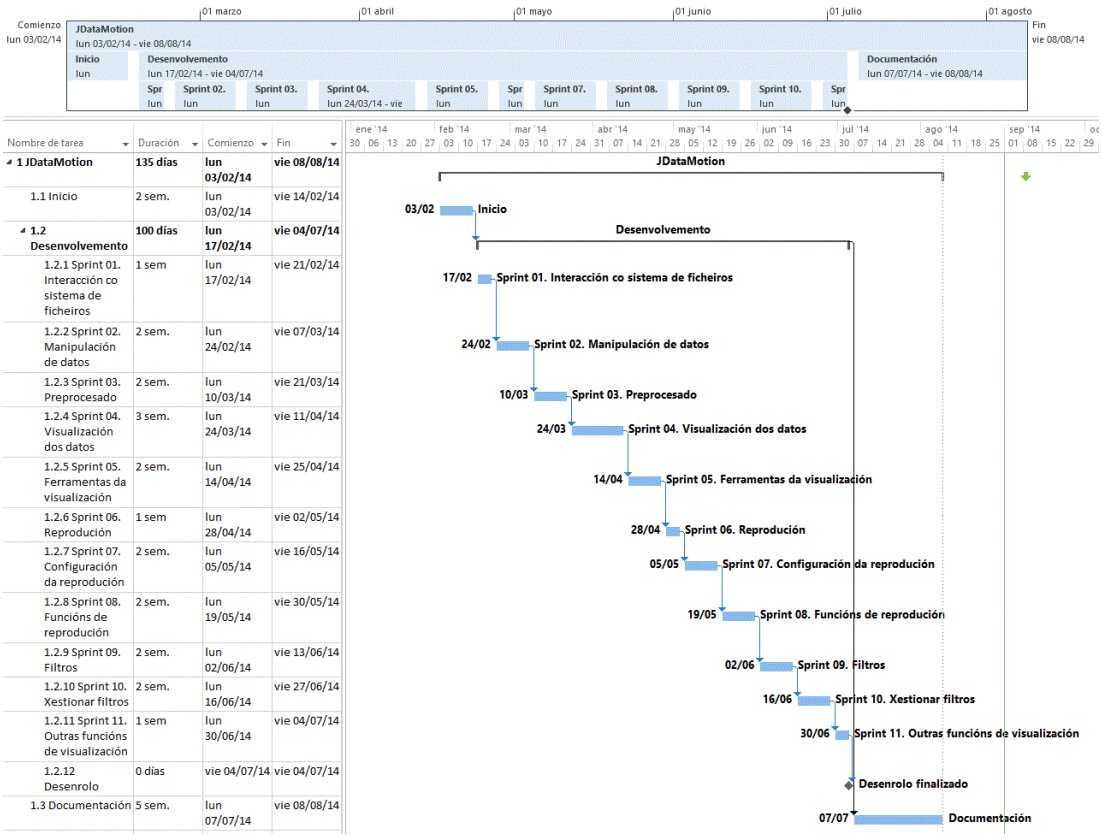

Figura 3.2: Diagrama de Gantt

- Redacción dun manual de usuario

- Reunión co director para revisión

Agora que temos as estimacións da fase de inicio, da fase de documentación e da fase de desenvolvemento (cos seus sprints estimados) podemos realizar un diagrama de Gantt do proxecto (figura 3.2) para establecer a liña base do mesmo:

Considerando que coñecemos as tarefas da fase de inicio e da fase de documentación, así como os sprints da fase de desenvolvemento e incluso os RFs a desenvolver dentro de cada un deles, podemos plasmar todas estas tarefas nun Esquema de Descomposición do Traballo ou EDT (figura 3.3). Isto implica que os RFs (ou máis ben o seu desenvolvemento) van ter a consideración de tarefas a partir de agora no noso proxecto.

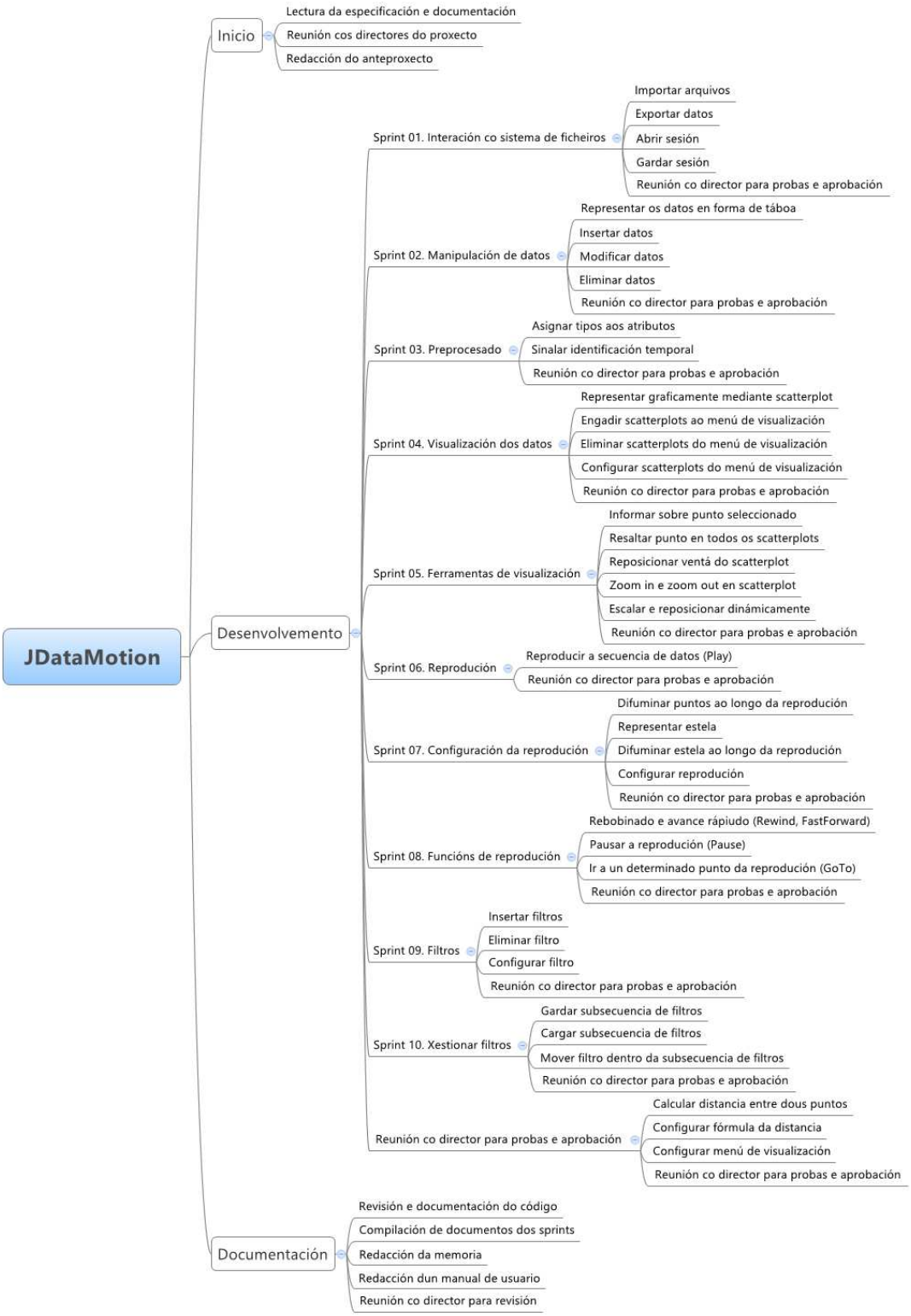

Figura 3.3: Esquema de Descomposición do Traballo (EDT)

3.2. Análise de tecnoloxías

Chegados a este punto, temos un conxunto de tarefas especificadas e delimitadas no tempo que se van realizar, así que agora abordaremos as tecnoloxías (ferramentas e técnicas) que imos utilizar para levalas a cabo.

3.2.1. Arquitectura

Nesta sección imos comentar a arquitectura sobre a que estará cimentada o noso proxecto. É importante considerar á hora de elixila que a nosa aplicación non só debe funcionar correctamente con ela, se non que ademais debe facilitar a súa mellora e evolución ao longo do tempo, ou o que é o mesmo, debe permitir a súa escalabilidade.

Na especificación deste proxecto e na captura de requisitos falouse do JDataMotion como unha ferramenta que permite visualizar dinamicamente conxuntos de datos que o usuario aporta. Esta premisa xa nos permite entrever que o modelo de datos non ten necesidade de ser extraido da máquina do cliente. Pódese traballar sobre ela de xeito local e obter resultados (visuais, ou un novo ficheiro de información procesada) sen necesidade de interactuar con sistemas externos (o que sería a computación distribuída). A arquitectura elixida será, segundo estas premisas, a dunha aplicación de escritorio.

3.2.2. Tecnoloxías

A partir do paradigma de computación que escollemos hai que elixir un conxunto de tecnoloxías que non só sexan compatibles con ela, se non que ademais permitan desenvolver satisfactoriamente os RFs pactados.

Ferramentas de deseño

UML

UML (Unified Modeling Language) é un estándar de modelado perfectamente aplicable ao un entorno de desenvolvemento de software. Baséase na especificación e documentación dos compoñentes do sistema a diferentes niveis, dentro da fase de deseño do sistema. Nós empregaremos este estándar mediante 3 tipos de diagramas:

Diagramas de casos de uso: Na figura 3.1 xa presentamos o diagrama de casos de uso do sistema.

Diagrama de clases: Este tipo de diagrama presenta as clases que subxacen baixo o sistema, xunto cos seus métodos, atributos e interrelacións.

Diagramas de secuencia: Estes diagramas plasman a interacción entre os compoñentes (ou obxectos) do sistema durante escenarios concretos da execución da aplicación. Neles amósase o intercambio de información e o fluxo de control entre compoñentes.

Ferramentas de desenvolvemento

A única imposición a nivel de desenvolvemento dada pola especificación do anteproxecto é perfectamente válida coas premisas tomadas ata agora: o sistema debe ser implementado na linguaxe de programación Java.

Java

Java é unha linguaxe de programación de propósito xeral, é dicir, contén librarías e soporte para diversos propósitos: acceso a base de datos, comunicación entre máquinas, cálculo matemático, procesamento de imaxes, etc. Tamén é unha linguaxe concorrente, pois contén fornecemento para fíos de execución que van dan lugar, entre outras moitas vantaxes, á posibilidade de crear unha interface gráfica como a que imos necesitar.

Outras vantaxes que Java nos ofrece implicitamente son as seguintes:

- Java é multiplataforma, é dicir, o código que se programa compílase unha soa vez para ser empregado en calquera sistema, sen importar a súa especificación (sistema operativo, arquitectura do computador, etc.).
- O colector de lixo, que nos permite abstraernos do proceso de liberación de recursos na memoria. Este sistema localiza e libera automaticamente na memoria aqueles obxectos que deixaron de ser referenciados (e polo tanto xa non vai volver a empregar o programa).

Java é unha das linguaxes máis populares para practicar a programación orientada a obxectos. Este paradigma, que traballa con clases e obxectos para modelar entidades reais como van ser as táboas, os filtros e por suposto os diagramas de dispersión, está particularmente ben adaptado para traballar coas interfaces gráficas. Outras vantaxes da programación orientada a obxectos das que nos imos beneficiar son a herdanza, o encapsulamento e o polimorfismo.

Para o noso proxecto, a versión de Java que imos empregar vai ser a 8, en calquera das súas actualizacións (1.8.X, sendo X calquer valor). Para a execución do JDataMotion deberemos ter instalada unha versión igual ou superior da máquina virtual de Java, que podemos descargar dende o sitio web oficial [4].

JFreeChart

Na especificación do proxecto tamén se deixa caer unha decisión que afectará significativamente ao desenvolvemento do proxecto cerca das últimas tarefas deste (sprints asociados a visualización dos datos) e é o de elixir unha libraría gráfica axeitada para as nosas necesidades á hora de representar os diagramas de dispersión.

Facendo un balance das principais solucións que existen actualmente na web, obtivemos:

- JFreeChart
- JZY3D
- Project Waterloo
- GRAL
- JChart2D
- JenSoft Java Chart API
- JRobin

Deste conxunto saleu como mellor posicionado o proxecto JFreeChart [2] de acordo ás seguintes características:

- É a solución que conta cun soporte máis activo. JFreeChart parecía ser o único proxecto no momento de tomar esta decisión que seguía sacando novas versións, actualizacións e parches. A solución JZY3D, sendo a segunda máis recente, non sacou unha nova versión dende o 2013, e no caso de JRobin a última versión é do 2011. Tendo en conta que Java 8 foi liberado en Marzo do 2014, parece ser que só JFreeChart se fixo eco da aparición da nova versión da máquina virtual, e polo tanto é a única librería que lle estará sacando partido a estas melloras.
- Non só é unha solución de software libre, senón que ademais é de código aberto. Isto será determinante á hora de acceder aos métodos e clases da librería, pois deberemos sobrescribir algúns para que se adecúen ao noso proxecto. De todos xeitos, as demais solucións presentadas tamén debían reunir esta característica á hora de ser elixidas.
- A documentación de JFreeChart é moi completa, os Javadocs da súa interface de programación de aplicacións (API) explican perfectamente a funcionalidade da librería, co que se reducen os custes temporais de aprendizaxe.
- O seu deseño favorece a herdanza de clases en beneficio das necesidades do noso proxecto.

- Incorpora as funcións de configuración gráfica necesarias no que atinxe aos diagramas de dispersión. Por medio de menús contextuais, cada diagrama de dispersión pode fixar a cor de fondo, títulos e eixos, ou a fonte e tamaño de letra das etiquetas por exemplo. Tamén pode exportar en distintos formatos de imaxe o diagrama en cuestión, copialo ao portapapeis.
- Permite a adición dinámica de puntos ao diagrama de dispersión. Isto será clave no proceso de reprodución dos datos.
- Incorpora as funcións de axuste da ventá de visualización para os diagramas de dispersión. Cada vez que creemos un diagrama de dispersión, este xa contará coa posibilidade de ser desprazado ou escalado dentro da súa ventá de visualización. Ademais tamén mellora a inserción dinámica de puntos no diagrama, pois cada vez que un punto se engade ao diagrama, a ventá reposiciónase para seguir abarcando todos os puntos.

JCommon

JCommon é unha librería da cal depende JFreeChart. Temos que acoplala ao proxecto, pero non imos ter necesidade de utilizala directamente.

Formatos de arquivo

No RNF01 fálase de dous tipos de arquivo de entrada e saída: “csv” e “arff”.

O formato csv (comma-separated values) de arquivos almacena os datos dun xeito tabular simple. A primeira liña dun arquivo csv contén separados por N-1 comas os N nomes dos atributos ou columnas da táboa (se segue o estándar). A continuación, cada nova liña conterá tamén N-1 comas para separar os N valores que toma esa entrada para cada atributo. Un valor nulo represéntase deixando baleiro o oco correspondente.

O formato arff

é un formato de arquivo que contén a mesma información ca o csv, pero a maiores especifica o tipo de valores que se admiten para cada atributo, e tamén o nome da relación. A estrutura desta información é a seguinte:

@RELATION da relación> só unha vez en cada arquivo, sinala o nome da relación contida no ficheiro.

@ATTRIBUTE dun atributo>do atributo> incluirase unha entrada coma esta por cada atributo da relacion. O tipo do atributo pode ser un dos seguintes:

- numeric (atributo numérico)
- <valor nominal 1, valor nominal 2, valor nominal 3, ...>(atributo nominal, é dicir, que só pode tomar un dos valores nominais especificados

na lista)

- string (calquera cadea de caracteres)
- date [<formato de data>] (data no formato dado)

@DATA \n <entrada 1 \n entrada 2 \n entrada 3 \n ...> despois de poñer a cláusula @DATA, engadirase unha nova liña para cada entrada. En cada entrada figurarán separados por comas os valores que esta toma para cada atributo (igual ca nos csv). Cada un desos valores asóciase ao atributo declarado na mesma posición.

% <calquera liña de texto > os comentarios comezan co símbolo '%' e abranguen a liña enteira. Poden ir intercalados en calquera zona do ficheiro.

Weka

A raíz da documentación atopada sobre o formato arff vimos mencionado varias veces o término Weka. Weka (Waikato Environment for Knowledge Analysis) é unha plataforma de software libre para o aprendizaxe automático e a minería de datos, desenvolvido en Java pola Universidade de Waikato. Esta universidade precisamente creou o formato arff para Weka.

O software de Weka pódese obterse a través do seu sitio web [1]. A apariencia do programa tras importar un arquivo arff pódese observar na figura 3.4, e na figura 3.5 podemos observar como Weka plasma os datos do arquivo baixo unha matriz de diagramas de dispersión. Como se pode ver, Weka xa realiza as funcións de importación e visualización dos datos.

Podemos extraer de Weka non só ideas para a interface do JDataMotion e das súas funcionalidades, se non que ademais, gracias a que Weka é un proxecto de código aberto, poderemos reutilizar o proxecto como unha libraría mais, utilizando os métodos e clases públicas que ofrece a súa interface de programación (API) para implementar facilmente a importación dos datos desde arquivo. A visualización dos datos tamén podería ser reutilizada, pero os diagramas de dispersión de Weka son estáticos, e a súa solución de visualización non vai servir para o JDataMotion. Weka non implementa a reprodución da visualización que JDataMotion busca.

Swing

Swing é unha libraría gráfica expresamente deseñada para Java. Inclúe unha serie de elementos (coñecidos como "widgets") para facilitar a confección da interface gráfica a calquera desenvolvedor. Ao igual que Java, Swing é independente da plataforma, e ademáis posibilita en gran medida e extensión de clases, ou widgets, para adaptalos ás necesidades de calquera interface gráfica.

NetBeans IDE

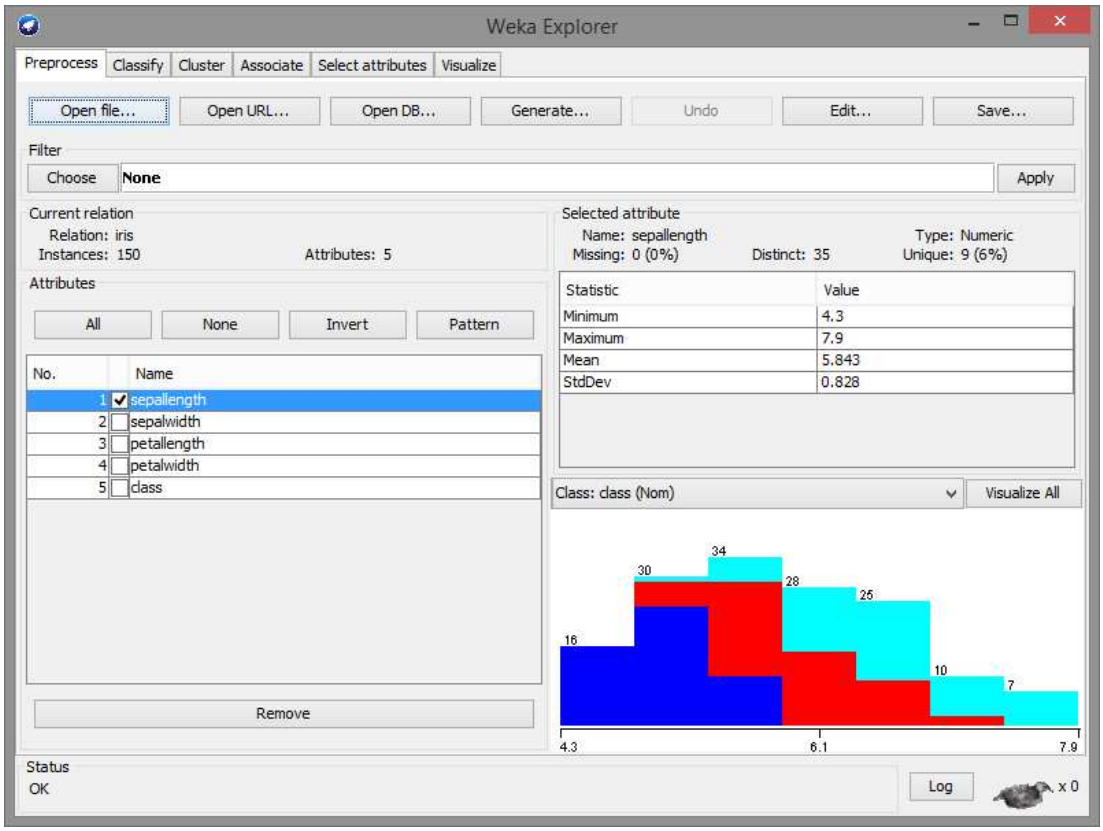

Figura 3.4: Software Weka

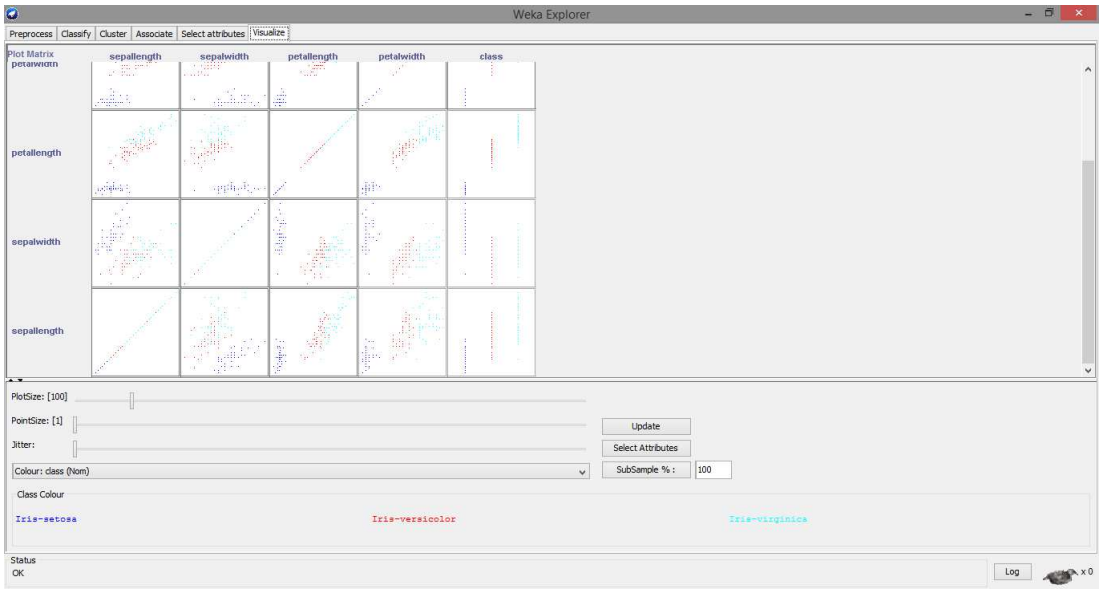

Figura 3.5: Diagramas de dispersión en Weka

NetBeans IDE é un entorno de desenvolvemento integrado, especialmente deseñado para programar sistemas en linguaxe Java, aínda que tamén da soporte a moitos outros linguaxes de programación. En NetBeans poderemos programar con facilidade as clases que logo o IDE compilará e executará. Tamén facilitará as labores de adxuntar librarías e dependencias, e dará fornecemento ao deseño da interface por medio dun lenzo e unha paleta de "widgets" propios de Swing, o cal reducirá o tempo que tardemos en aprender a manexar a libraría Swing.

Ferramentas de validación

A validación é un proceso que permite contrastar os resultados da implementación do sistema fronte aos requisitos (funcionais, non funcionais, de deseño, de calidade, etc) que inicialmente se esperaban dela.

JUnit

JUnit [10] é un compendio de librarías para aplicacións Java destinadas a realizar probas unitarias do sistema no que se inclúen. Contén un conxunto de clases que actúan sobre as clases e métodos do sistema orixinal, validando a súa resposta ante certos parámetros de entrada.

Capítulo 4

Deseño

Neste capítulo documentarase o proceso de deseño que se irá formando de xeito incremental nas sucesivas iteracións da metodoloxía Scrum. Comentarase a arquitectura do modelo de datos que vai manexar a aplicación, así como a distribución deses datos na aplicación, o que nos obrigará a falar da interface gráfica de usuario. Despois disto, mergullarémonos no que é o deseño e máis a implementación dos compoñentes da aplicación.

4.0.3. Estrutura do deseño

Os datos son o punto de partida de todas as funcionalidades deste proxecto. Todo xira arredor do arquivo con información que o usuario importa tras abrir por primeira vez o JDataMotion. O dato debe ser almacenado de xeito adecuado, e accedido só a través dos métodos e clases necesarios, para manter o fluxo de información controlado. A clase que captará, almacenará e distribuirá os datos de cara ás demais clases da aplicación recibirá o nome de Modelo, posto que realmente alberga o modelo da aplicación.

JDataMotion vai ser un programa moi dependente da súa interface de usuario. Os diagramas de dispersión non poden ser visualizados a través dunha simple terminal, e o constante fluxo de datos entre o usuario e o sistema non se pode activar a través dun menú de opcións en termos de usabilidade. Si traballamos co JDataMotion a través dunha interface gráfica multifío como as que Java permite deseñar, a aplicación pode procesar a información ao mesmo tempo que o usuario realiza outras interaccións (ver o modelo, configurar un filtro ou mesmo cancelar a visualización dinámica). Por isto, a clase Vista será un dos artefactos que máis tempo adicaremos a implementar. Vista conterá os "widgets" da libraría Swing necesarios para presentar a información ao usuario, e recibir del as novas ordes.

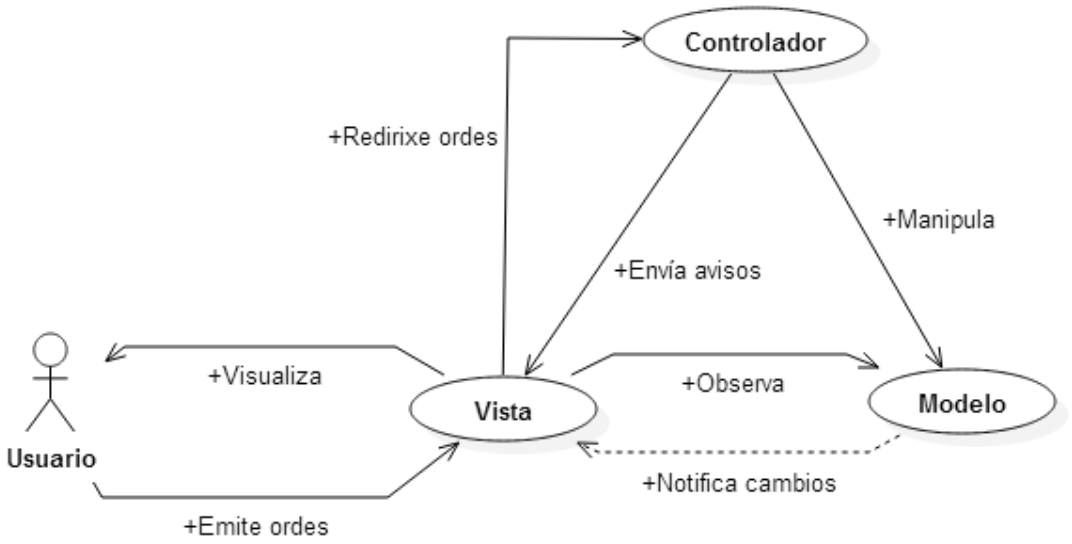

Figura 4.1: Modelo-Vista-Controlador para JDataMotion

Será unha clase complexa e de bastante peso que delegará certas funcións en clases internas ou asociadas.

Só falta unha clase que reciba da Vista as funcionalidades activadas polo usuario e cree un comando que se encargue de realizar o traballo. Esta clase deberá non só disparar o comando, se non que tamén terá que xestionalo, e reaccionar dun xeito específico en caso de que o comando chegue a unha situación de erro. A maioría de comandos influirá directa ou indirectamente sobre o Modelo. A esta clase chamarémoslle Controlador porque a súa responsabilidade será esencialmente esa, a de recibir eventos da Vista e xestionar comandos que actúen de cara ao Modelo. Deste xeito a Vista utilizará a API do Controlador, e quedará exenta de responsabilidade no caso de que se detecten fallas no Modelo.

Acabamos de definir dun xeito explícito cal é o patrón de deseño sobre o que vai xirar toda esta fase do proxecto: o Modelo-Vista-Controlador (MVC). Os patróns de deseño son solucións prácticas a problemas de deseño comúns. En concreto, o Modelo-Vista-Controlador divide o sistema en tres compoñentes coas responsabilidades definidas que xa comentamos anteriormente. A mellor forma de expoñer a aplicación exacta do patrón MVC no noso proxecto é o diagrama da figura 4.1.

As relacións marcadas cunha liña sólida son asociacións directas (unha clase que actúa de xeito explícito sobre os atributos ou métodos doutra), mentres que a relación marcada cunha liña descontinua representa unha relación indirecta (por exemplo, unha clase que resulta afectada pola actividade doutra). As relacións entre compoñentes detallaranse a continuación.

- O usuario emite ordes en forma de eventos cara a Vista ao interactuar cos seus botóns e menús.
- A Vista identifica os eventos que recibe e redirixe cara o Controlador a orde asociada ao evento en cuestión.
- O Controlador envía un aviso directo á Vista en caso de que a orde que recibe dela levase ao sistema a un estado de erro.
- O Controlador manipula o Modelo segundo o contido da orde que recibe.
- O Modelo actúa como entidade observada de cara á Vista. A Vista ten acceso ao Modelo para actualizar a súa aparencia.
- Do mesmo xeito, que o Modelo sexa observado pola Vista implica que os cambios no modelo serán notificados á Vista, para que esta se actualice no momento do cambio.

Imos documentar máis en profundidade a relación entre a Vista e o Modelo. Comentábamos que a Vista accede ao Modelo para actualizarse cando o desexe, pero ademais o Modelo, cando cambia, notifica á Vista o suceso deste cambio. Temos entón que a Vista actúa como entidade "observadora" dun Modelo que é a entidade "observada". En esencia, estamos definindo o patrón de deseño Observer.

O patrón de deseño Observer é a mellor resposta que podemos atopar a nivel de deseño para solucionar o problema de manter actualizados ao mesmo nivel o Modelo e a Vista. Se non botásemos man desta solución, teríamos dúas alternativas:

- Refrescar a vista cada certo intervalo tempo, o cal é ineficiente e seguiría permitindo a posibilidade de que se dese unha incoherencia Vista-Modelo entre refrescos.
- Permitir ao Modelo que acceda directamente á Vista para invocar un método de actualización, é dicir, asociar directamente a Vista ao Modelo. Isto atenta contra a natureza do Modelo-Vista-Controlador, xa que o Modelo ten que almacenar a información do sistema e abstraerse por completo da Vista ou do módulo que se encargue de representar os seus contidos. Témonos que manter na premisa de que a Vista pode observar ao Modelo e acceder a el para ler datos, pero o Modelo non debe ser consciente da existencia de ningunha Vista.

O patrón Observer pódese aplicar de forma moi sinxela ao noso proxecto. Basta con facer que a Vista implemente a interface *Observer*, de xeito que a obrigue a implementar un método de actualización chamado *update()*, que se disparará cando un obxecto observado cambie o seu estado. Por outra parte, o Modelo só necesita estender a clase *Observable* para ser susceptible de ser observado por un *Observer*. Ao estender esta clase, a Vista xa pode chamar

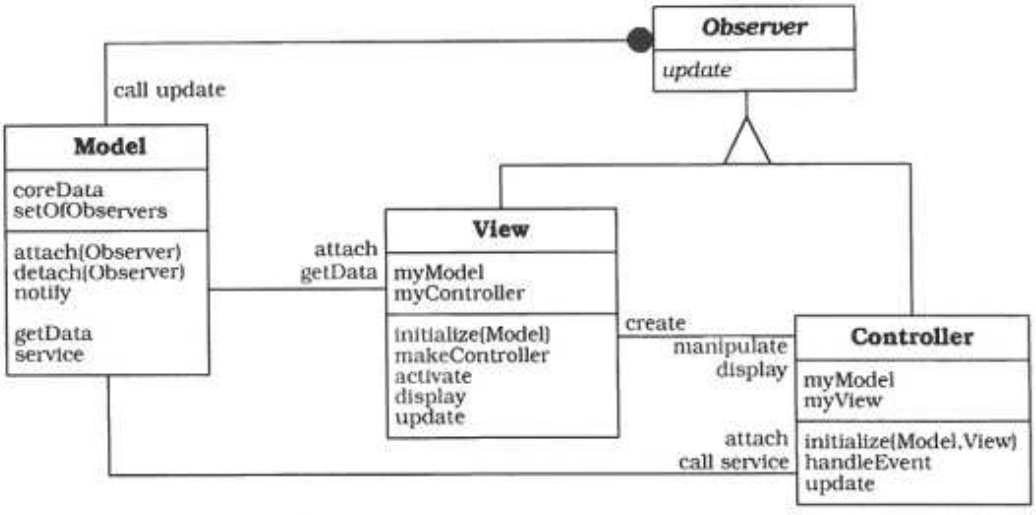

Figura 4.2: Modelo-Vista-Controlador con Observer

ao método *addObserver(Observer o)* do Modelo, pasándose a ela mesma como parámetro para así incluírse como observadora e ser notificada dos seus cambios.

A asociación dos patróns Modelo-Vista-Controlador e Observer baixo esta forma é bastante común. O libro Pattern-Oriented Software Architecture [12] fai unha boa exposición do que acabamos de mencionar, e aporta o diagrama da figura 4.2 para sintetizar ambos patróns.

Probablemente a diferencia máis destacable que podemos atopar entre este diagrama e o diagrama que correspondería coa nosa aplicación sería que no noso caso só a Vista implementará a interface *Observer*. O Controlador no noso proxecto non actuará de observador, pois non ten que recibir notificacións ante os cambios do Modelo, xa que de feito el é o responsable directo deses cambios, e non necesita ser notificado das súas propias accións.

4.0.4. Interface gráfica

Nesta sección configuraremos o deseño da interface gráfica. A Vista é a clase encargada de darlle soporte, delegando certas funcións en clases propias asociadas ou internas.

Capítulo 5

Exemplos

5.1. Un exemplo de sección

Esta é *letra cursiva*, esta é **letra negrilla**, esta é letra subrallada, e esta é `letra curier`. Letra tiny, scriptsize, small, large, Large, LARGE e moitas más. Exemplo de fórmula: $a = \int_o^\infty f(t)dt$. E agora unha ecuación aparte:

$$S = \sum_{i=0}^{N-1} a_i^2. \tag{5.1}$$

As ecuaciones se poden referenciar: ecuación (5.1).

5.1.1. Un exemplo de subsección

O texto vai aquí.

5.1.2. Otro exemplo de subsección

O texto vai aquí.

Un exemplo de subsubsección

O texto vai aquí.

Izquierda	Derecha	Centrado
ll	r	cccc
llll	rrr	c

Cadro 5.1: Esta é a táboa de tal e cal.

Un exemplo de subsubsección

O texto vai aquí.

Un exemplo de subsubsección

O texto vai aquí.

5.2. Exemplos de figuras e cadros

A figura número 5.1.

O cadro (taboa) número 5.1.

5.3. Exemplos de referencias á bibliografía

Este é un exemplo de referencia a un documento descargado da web [**?**]. E este é un exemplo de referencia a unha páxina da wikipedia [**?**]. Agora un libro [**?**] e agora unha referencia a un artigo dunha revista [**?**]. Tamén se poden pór varias referencias á vez [**?**, **?**].

5.4. Exemplos de enumeracións

Con puntos:

- Un.
- Dous.
- Tres.

Con números:

1. Catro.

Figura 5.1: Esta é a figura de tal e cal.

2. Cinco.

3. Seis.

Exemplo de texto verbatim:

```
O texto        verbatim
     se visualiza tal
            como se escribe
```

Exemplo de código C:

```c
#include <math.h>
main()
{   int i, j, a[10];
    for(i=0;i<=10;i++) a[i]=i; // comentario 1
    if(a[1]==0) j=1; /* comentario 2 */
    else j=2;
}
```

Exemplo de código Java:

```java
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Capítulo 6

Conclusións e posibles ampliacións

Conclusións e posibles ampliacións

Apéndice A

Manuais técnicos

Manuais técnicos: en función do tipo de Traballo e metodoloxía empregada, o contido poderase dividir en varios documentos. En todo caso, neles incluirase toda a información precisa para aquelas persoas que se vaian a encargar do desenvolvemento e/ou modificación do Sistema (por exemplo código fonte, recursos necesarios, operacións necesarias para modificacións e probas, posibles problemas, etc.). O código fonte poderase entregar en soporte informático en formatos PDF ou postscript.

Apéndice B

Manuais de usuario

Manuais de usuario: incluirán toda a información precisa para aquelas persoas que utilicen o Sistema: instalación, utilización, configuración, mensaxes de erro, etc. A documentación do usuario debe ser autocontida, é dicir, para o seu entendemento o usuario final non debe precisar da lectura de outro manual técnico.

Apéndice C

Licenza

Se se quere pór unha licenza (GNU GPL, Creative Commons, etc), o texto da licenza vai aquí.

Bibliografía

[1] Weka 3 - Data Mining with Open Source Machine Learning Software in Java. Sitio web *http://www.cs.waikato.ac.nz/ml/weka/*.

[2] Proxecto JFreeChart. Sitio web *http://www.jfree.org/jfreechart*.

[3] Formato de Archivo Atributo-Relación (ARFF). Información dispoñible en *http://www.cs.waikato.ac.nz/ml/weka/arff.html* [Última consulta: 02/09/2014].

[4] Java. Sitio web *https://www.java.com/en/*.

[5] Tarifasgasluz. Precio del kWh en España. Sitio web *http://www.endesaonline.com/es/empresas/dual/empresas/4/precios/index.asp* [Última consulta: 02/09/2014].

[6] Electricity usage of a Laptop or Notebook. Sitio web *http://energyusecalculator.com/electricity_laptop.htm* [Última consulta: 02/09/2014].

[7] Scrum. Definicións extraídas de *http://es.wikipedia.org/wiki/Scrum* [Última consulta: 20/12/2014].

[8] Acunote. Sitio web *http://www.acunote.com/*.

[9] GitHub. Sitio web *https://github.com/*.

[10] JUnit. Sitio web *http://junit.org/*.

[11] IEEE-STD-830-1998: Especificaciones de los requisitos del software. Leer más *http://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf* [Última consulta: 25/03/2015].

[12] Buschmann, Frank. 1996. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley.

Glosario

Sesión
: Interacción co sistema e as súas funcionalidades. Inclúe a importación duns datos, o seu procesado, filtrado e visualización.

Experimento
: Ver sesión.

Diagrama de dispersión
: Diagrama matemático que fai uso das coordenadas cartesianas para amosar os valores de dúas variables para un conxunto de datos. Cada punto no diagrama referencia un valor ao longo do eixo de ordenadas e outro ao longo do eixo de abscisas.

Modelo
: Estrutura e contido dos datos cos que se traballa no experimento. Abrangue tanto os tipos dos atributos coma os seus valores dentro de cada entrada, así coma o nome do conxunto de datos, ou a marca do atributo que actúa de índice temporal.

Filtro
: Ferramenta configurable que en función dos seus parámetros pode converter, a partir dun modelo A de entrada, un modelo B de saída.

Visualización
: Presentación gráfica da relación, neste caso baixo a forma de diagramas de dispersión.

Reprodución
: Activación dunha visualización para que cambie o seu estado ao longo do tempo, neste caso en función dun índice ou variable temporal.

Índice temporal
: Atributo do modelo que representa a orde na que os datos foron tomados, foron detectados, queren ser priorizados ou simplemente se desexan amosar. Se non se define, asúmese como índice temporal a orde das entradas do modelo.

Entrada
: Ver instancia.

Instancia
: Vector de datos que contén un valor para cada atributo do modelo. Pode conter valores nulos.

Atributo
: Cada unha das variables coas que traballa o modelo.

Atributo numérico
: Atributo que só pode tomar valores numéricos.

Atributo de tipo data
: Atributo que só pode tomar valores de tipo data.

Atributo de tipo string
: Atributo que pode tomar calquera valor en forma de cadea de caracteres.

Atributo nominal
: Atributo que só pode tomar unha serie de valores concretos.

Relación
: Ver modelo.

Reprodución
: Visualización dinámica da relación, é dicir, presentación dunha relación na que cada entrada se visualiza nun momento do tempo determinado.

Comando
: Entidade que representa unha orde para o sistema. Contén información sobre o artefacto ao que afecta, o tipo de orde que se expresa e os parámetros necesarios para a súa execución.