UNIVERSIDAD DE LOS ANDES
FACULTAD DE CIENCIAS
DEPARTAMENTO DE FÍSICA
MÉRIDA, VENEZUELA

Modelo de portlet para estudiar la hidrodinámica Del colapso gravitacional en el Grid

Tesis que presenta

Reina Camacho

Para Obtener el Grado de

Licenciado en Física

Tutor de la Tesis: Dr. Luis Nuñez.
Cotutor de la Tesis: Ing. Vanessa Hamar.

Mérida, Venezuela — Marzo 2007

RESUMEN

Presentamos el ***portlet*** **CHOQUE** un ambiente integrado de cálculo y visualización centrado en interfases Web y con tecnología Grid, el cual permite estudiar la hidrodinámica radiativa de la evolución de una onda de choque durante el colapso gravitacional de esferas relativistas autogravitantes en Relatividad General. Esta herramienta permite estudiar cuantitativamente dicha situación, que implica una aproximación cuasiestática y mecanismo de transporte de radiación más realistas (factor variable de Eddignton), por lo cual puede convertirse en una herramienta importante para la comunidad de astrofísica relativista. Las técnicas de visualización empleadas muestran a través de la manipulación de la tabla de colores, la evolución de las variables físicas dentro de la distribución material.

Palabras claves: onda de choque, portlet, relatividad.

Agradecimientos

Quiero expresar mi agradecimiento

A mi Tutor de Tesis, Dr. Luis A. Nuñez por brindarme la oportunidad de recurrir a su capacidad y experiencia científica y aguantar mis crisis académicas.

A la Ing. Vanessa Hamar, al Dr. Francisco Hidrobo, al Dr. Kay Tucci, a Rafael León, a Alvaro Hernández y al Lic. Jorge Rueda, por sus valiosas sugerencias y acertados aportes durante el desarrollo de este trabajo.

A los profesores que he tenido a lo largo de mi carrera, que bien sea por sus buenas o malas acciones siempre dejan una enseñanza.

A mis padres y hermanos por brindarme un verdadero hogar y enseñarme que la perseverancia y el esfuerzo son el camino para lograr objetivos.

A Carlos Quintero por su apoyo incondicional y su cariño.

A la Facultad de Ciencias de esta ilustre Universidad de Los Andes.

Reina Camacho

... *L*evántate, pues, y vence tu flaqueza con el ánimo que triunfa en los combates [...] ... *y* al levantarse mira alrededor, desvanecido por la grande angustia por [la] *q*ue ha pasado...

Dante Alighieri

(1265-1321)

Índice

Introducción

La computación Grid es un sistema de *software* y *hardware* que permite compartir recursos computacionales, datos, espacio de almacenamiento y solucionar problemas en una o varias organizaciones a través de la red de una manera transparente al usuario. Esta tecnología pretende usar la Internet como una plataforma de servicios de computación y no sólo como fuente de información, como es en la actualidad. Para esto se utiliza un *software*, llamado *middleware*, encargado de coordinar todo este sistema.

El Grid define un nuevo campo para la comunidad científica en particular para la comunidad de astrofísica relativista cuya tendencia es el uso intensivo y necesario del computador, a fin de tratar problemas que toman horas de cálculo en sistemas de computación distribuidos hasta ahora empleados.

Con todo, a menudo es complicado utilizar el Grid , por la necesidad de aprender varias utilidades de líneas de comandos y nuevos conceptos. Por ello es necesario desarrollar entornos que permiten a un usuario utilizar fácilmente el Grid , presentando una vista atractiva, uniforme e intuitiva de los recursos que lo componen. Hoy en día la interfaz hombre-máquina natural para aplicaciones tipo Grid son los portales y *portlets*. Los *portlets* para aplicaciones o códigos científicos surgen como respuesta a la necesidad de simplificar el uso de dichas aplicaciones o códigos empleando interfases amigables además de aprovechar la seguridad y el beneficio de la tecnología de Grids.

En este escenario surge la idea del desarrollo del *portlet* CHOQUE el cual brinda acceso a un conjunto de subrutinas en fortran basadas en el modelo de Rueda y Núñez. Dicho modelo sigue un enfoque seminumérico comenzando con la solución interior estática con simetría esférica para la ecuación Tolman-Oppenheimer-Volkov a fin de determinar la influencia que ejerce el esquema de radiación y anisotropía local sobre la propagación de una superficie de discontinuidad en la aproximación cuasiestática. Este esquema se considera una extensión del método HJR.

El objetivo principal de este trabajo es desarrollar el *portlet* CHOQUE un ambiente integrado de cálculo y visualización, centrado en interfases WEB y con tecnología Grid, portátil, de fácil uso y bien documentado que permita simular de forma fácil y sistemática la propagación de discontinuidades hidrodinámicas en esferas radiantes.

Capítulo 1

Colapso gravitacional

1.1 Vision General

El colapso gravitacional de una concentración localizada de materia es uno de los temas centrales de la Relatividad General y ha atraído la atención de los investigadores desde los años 40 [48]. El colapso gravitacional tiene generalmente, cuatro tipos de posibles estados finales. El primero es la formación de objetos estelares autosustentados, tales como estrellas, enanas blancas o estrellas de neutrones. El segundo es simplemente la dispersión del objeto colapsado que finalmente deja un espacio-tiempo plano. El tercer tipo es la formación de agujeros negros con materia y radiación saliente, mientras que el cuarto es la formación de singularidades desnudas.

Estos escenarios han guiado a dos activas comunidades académicas a en la construcción de códigos numéricos para simular computacionalmente el colapso gravitacional: la comunidad de astrofísica numérica y la de relativistas numéricos. La comunidad de astrofísica numérica ha abordado el problema del colapso gravitacional buscando discernir mecanismos que conduzcan a explosiones Supernovas (en particular las tipo II). Estas simulaciones incorporan descripciones detalladas de la microfísica de medio material (ecuaciones de estado de la materia y transporte de radiación), sacrificando la descripción hidrodinámica y la injerencia del campo gravitacional con una descripción newtoniana en la cual materia y radiación vienen especificadas por teorías inequivalentes [49]. La comunidad de relativistas numéricos, movida por el interés de entender la radiación gravitacional ha construido importantes códigos con distintos esquema numéricos para resolver las ecuaciones de Einstein. Todo este muy reciente esfuerzo se centra, mayoritariamente en describir el colapso gravitacional no esférico y elude la simulación de las propiedades de la materia a tan altas densidades (para una mejor revisión [50] y las referencias allí citadas).

Es necesario entonces, estar en capacidad de resolver las Ecuaciones de Einstein para medios materiales radiantes en condiciones de simetría lo más generales posibles. Sin embargo, este problema es insoluble analíticamente para la mayoría de los casos de interés, aún para aquellos con simetría esférica. Esta dificultad sólo comienza a ser manejable cuando se hacen suposiciones adi-

cionales las cuales, pese a limitar los resultados, permiten intuir ciertas propiedades de la materia nuclear sometida a condiciones extremas. El tratamiento del problema en términos numéricos no presenta un panorama más alentador a pesar de los recientes avances en el tratamiento del colapso radiante esférico en Relatividad General [51] [52] [53].

En 1980 L. Herrera, J. Jiménez y G. Ruggeri (HJR) [54], inspirándose en un artículo de H. Bondi [55], presentan un tratamiento seminumérico del colapso gravitacional radiante en Relatividad General. En este enfoque se transita al máximo la posibilidad de una solución analítica, restando un mínimo de esfuerzo numérico para simular la evolución de las variables físicas en una esfera radiante de fluido perfecto. Es en este sentido que el algoritmo propuesto constituye un método semi-numérico general para resolver las ecuaciones de Einstein con simetría esférica. La limitación impuesta para este caso consiste en una hipótesis heurística sobre la forma funcional de variables auxiliares las cuales, en el caso estático, coinciden con la presión y la densidad físicas. Esta suposición, guiada por sólidos criterios físicos, permite trasladar el problema de resolver las ecuaciones de Einstein (un sistema de ecuaciones diferenciales no-lineales en derivadas parciales con condiciones de borde) a integrar (numéricamente) un sistema de ecuaciones diferenciales ordinarias para funciones evaluadas en las superficie. Al resolver este sistema en la superficie de la distribución, este método garantiza el acoplamiento con la solución exterior de Vaydia. El éxito del esquema seminumérico HJR puede medirse en la variedad de exitosas aplicaciones a los más diversos terrenos de la hidrodinámica relativista. Entre duchas aplicaciones se pueden mencionar: Fluidos cargado o neutros, fluidos isótropos o anisótropos, ondas de choques, tensión superficial, fluidos radiantes en escape libre o difusión [56].

Existe un consenso general referente a los ingredientes necesarios en un escenario de colapso gravitacional, estos son los siguientes [52]:

1. Como consecuencia de la microfísica del sistema una gran cantidad de radiación tiende a abandonar el sistema, pero la absorción y el scattering en el medio dificulta su escape libre.

2. Transiciones de fase pueden inducir una presión anisótropa local (es decir, $P_r \neq P_\perp$) [57].

3. La formación y propagación de una superficie de discontinuidad: una onda de choque, una detonación o un frente de combustión, cuyo ancho es pequeño en comparación con el tamaño del sistema.

La presencia y propagación de frentes de discontinuidades en las variables hidrodinámicas se observa cuando materia viajando a grandes velocidades y radiación interactúan. Un ejemplo son las explosiones de Supernovas (tipos I y II), las cuales representan dos de los escenarios más espectaculares [58] [59] [60]. Sin embargo otros fenómenos como transiciones de fase de materia nuclear ultradensa pueden también ser descritos a través de la propagación de discontinuidades [61].

Debido a la gran variedad de fenómenos en astrofísica relacionados con la formación y propagación de superficies de discontinuidades el modelo físico que sustenta al portlet CHOQUE es un modelo de Rueda y Núñez [62], el cual considera la evolución de una onda de choque radiante en esferas relativistas autogravitantes en el contexto de una aproximación post-cuasiestática, y

provee todos los ingredientes mencionados anteriormente requeridos para el estudio del colapso gravitacional. Este modelo será descrito a continuación.

1.2 Ondas de choque radiante en la aproximación post-cuasiestática

A fin de determinar la influencia que ejerce el esquema de radiación y anisotropía local sobre la propagación de una superficie de discontinuidad se sigue un enfoque seminumérico comenzando con la solución interior estática con simetría esférica para la ecuación Tolman-Oppenheimer-Volkov. Este esquema las ecuaciones diferenciales parciales de Einstein en un sistema de ecuaciones diferenciales ordinarias para cantidades evaluadas en las superficies de choque y de contorno, este esquema se considera una extensión del método HJR. Además se introduce el *factor de flujo* $f = \mathcal{F}/\rho_R$ (relación entre el flujo de radiación y la densidad de energía de radiación), el *factor variable de Eddington* $\chi = P_R/\rho_R$ (relación entre la presión de radiación y la densidad de energía de radiación) [63] [64] y la relación de clausura Lorentz-Eddington entre ellos:

$$\chi = \chi(f) = \frac{5}{3} - \frac{2}{3}\sqrt{4 - 3f^2} \tag{1.1}$$

La cual permite simular cualquier fase de radiación entre el límite de difusión y el límite de escape libre.

1.2.1 Configuración del sistema y ecuaciones de campo

La configuración puede dividirse en tres regiones: el núcleo que es la región más interna y es denotado por I, el manto en el medio denotado por II y el espacio exterior denotado por III como se observa en la Figura 1.1

El núcleo y el manto están separados por una onda de choque (hipersuperficie de discontinuidad) y una superficie de contorno (hipersuperficie con velocidad igual a la del fluido en ese punto). Tanto la onda de choque como la superficie de contorno son hipersuperficies temporales y sólo existe radiación no polarizada saliendo de la distribución, la cual es modelada por la métrica exterior de Vaidya. Los elementos de línea en cada región son:

$$ds^2_{I,II} = e^{\nu}dt^2 - e^{\lambda}dr^2 - r^2(d\theta^2 + sen^2\theta d\phi^2) \tag{1.2}$$

$$ds^2_{III} = \left[1 - \frac{2M(u)}{R}\right]du^2 + 2dudR - R^2(d\theta^2 + sen^2\theta d\phi^2) \tag{1.3}$$

donde ν y λ son funciones de t y r, u es el tiempo de retardo, R es una coordenada nula ($g_{RR} = 0$) y θ y ϕ son las coordenadas angulares usuales. Para un observador comóvil con velocidad radial ω relativa a las coordenadas locales de Minkowski, el contenido físico del fluido es representado por el flujo de radiación $\mathcal{F}$ en la dirección radial, la densidad de energía $\bar{\rho}$, la presión radial $\bar{P}_r$ y presión tangencial $\bar{P}_\perp$. La barra indica la contribución total, es decir, hidrodinámica + radiación. En estas coordenadas las componentes del tensor energía-impulso para cada región de un fluido

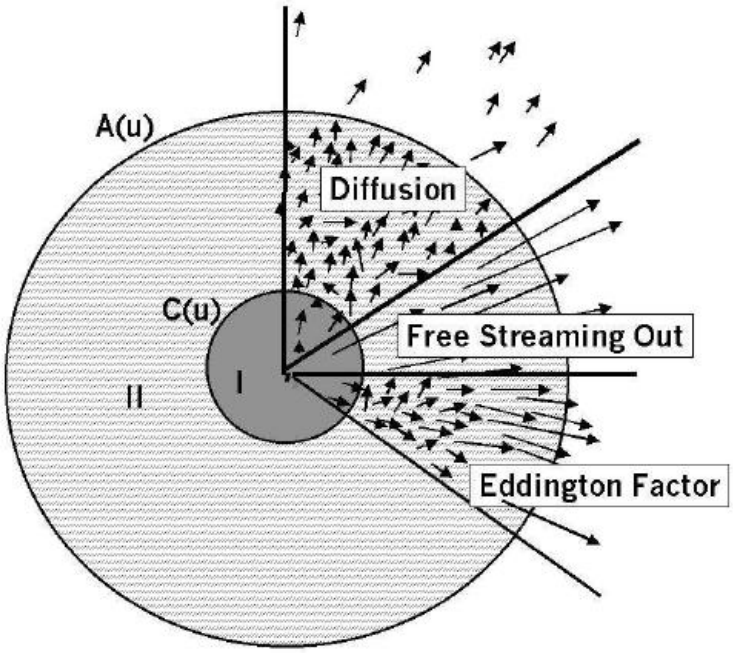


Figura 1.1: Regiones de la distribución de materia. El núcleo (I) es la región más compacta, el manto (II) situado en el medio y finalmente el espacio-tiempo exterior (III)

con simetría esférica no estático con materia y radiación puede ser escrito como:

$$\begin{aligned}(T_{\alpha\beta})_{I,II} &= (\bar{\rho}+\bar{P}_\perp)u_\alpha u_\beta - \bar{P}_\perp g_{\alpha\beta} + (\bar{P}-\bar{P}_\perp)\chi_\alpha\chi_\beta + 2\mathcal{F}_{(\alpha u_\beta)} \\ (T_{\alpha\beta})_{III} &= -\frac{1}{4\pi R^2}\frac{dM}{du}\delta^0_\alpha\delta^0_\beta \end{aligned} \tag{1.4}$$

donde [65]

$$\bar{\rho} = \rho + \rho_R \quad \bar{P} = P_r + P_R \quad \bar{P}_\perp = P_\perp + (P_\perp)_R \quad (P_\perp)_R = \frac{\rho_R - P_R}{2}$$
$$u_\alpha = \frac{e^{\nu/2}\delta^0_\alpha - \omega e^{\lambda/2}\delta^1_\alpha}{\sqrt{1-\omega^2}} \quad \chi_\alpha = \frac{-\omega e^{\nu/2}\delta^0_\alpha - e^{\lambda/2}\delta^1_\alpha}{\sqrt{1-\omega^2}} \quad \mathcal{F}_\alpha = -\mathcal{F}\chi_\alpha \tag{1.5}$$

donde el subíndice R denota la contribución de la radiación. Para dicho tensor de energía-momento las ecuaciones de Einstein $G^\alpha_\beta = -8\pi T^\alpha_\beta$ que relacionan la geometría del espacio-tiempo (tensor de Einstein) con la distribución de materia (tensor de energía impulso) son las siguientes:

$$-8\pi\left(\frac{\bar{\rho}+\omega^2\bar{P}+2\omega\mathcal{F}}{1-\omega^2}\right) = -\frac{1}{r^2} + e^{-\lambda}\left(\frac{1}{r^2} - \frac{\lambda'}{r}\right) \tag{1.6}$$

$$-8\pi\left(\frac{\bar{P}+\omega^2\bar{\rho}+2\omega\mathcal{F}}{1-\omega^2}\right) = -\frac{1}{r^2} + e^{-\lambda}\left(\frac{1}{r^2} + \frac{\nu'}{r}\right) \tag{1.7}$$

$$-8\pi\bar{P}_{\perp} = \frac{\mathrm{e}^{-\nu}}{4}\left[2\ddot{\lambda} + \dot{\lambda}\left(\dot{\lambda} - \dot{\nu}\right)\right] - \frac{\mathrm{e}^{-\lambda}}{4}\left[2\nu'' + \left(\nu' + \frac{2}{r}\right)(\nu' - \lambda')\right] \tag{1.8}$$

$$-8\pi\,\mathrm{e}^{\frac{\nu+\lambda}{2}}\left[\frac{\omega(\bar{\rho}+\bar{P}) + (1+\omega^2)\mathcal{F}}{1-\omega^2}\right] = \frac{\dot{\lambda}}{r}, \tag{1.9}$$

Donde los puntos indican derivadas con respecto al tiempo y las primas derivadas radiales.

1.2.2 Aproximación post-cuasiestática (PQA)

Herrera et al.[66] define el régimen post-cuasiestático como al correspondiente al sistema fuera del equilibrio (o cuasiequilibrio) pero cuyas variables efectivas comparten la misma dependencia radial que las variables físicas en el estado de equilibrio (o cuasiequilibrio), es decir, es la situación más cercana a la evolución cuasiestática. Matemáticamente esto es $\mathcal{O}(\omega^2) = \ddot{\lambda} = \ddot{\nu} = \dot{\lambda}\dot{\nu} = \dot{\lambda}^2 = 0$.

Las variables efectivas se definen como:

$$\tilde{\rho} = \frac{\bar{\rho} + \bar{P}\omega^2 + 2\omega\mathcal{F}}{1-\omega^2} \qquad \text{y} \qquad \tilde{P} = \frac{\bar{P} + \bar{\rho}\omega^2 + 2\omega\mathcal{F}}{1-\omega^2}, \tag{1.10}$$

A partir de estas variables efectivas, las ecuaciones de campo 1.6-1.9 pueden ser reeescritas como:

$$m = \int 4\pi r^2\tilde{\rho}dr, \tag{1.11}$$

$$\nu = \int \frac{2(4\pi r^3\tilde{P} + m)}{r(r-2m)}dr, \tag{1.12}$$

$$-8\pi\bar{P}_{\perp} = \frac{\mathrm{e}^{-\nu}}{4}\left[2\ddot{\lambda} + \dot{\lambda}\left(\dot{\lambda} - \dot{\nu}\right)\right] - \frac{\mathrm{e}^{-\lambda}}{4}\left[2\nu'' + \left(\nu' + \frac{2}{r}\right)(\nu' - \lambda')\right], \tag{1.13}$$

$$\dot{m} = -\frac{4\pi r^2\,\mathrm{e}^{\frac{\nu-\lambda}{2}}}{1+\omega^2}\left[\omega(\tilde{\rho}+\tilde{P}) + (1-\omega^2)\mathcal{F}\right], \tag{1.14}$$

donde la función masa es definida por $e^{-\lambda} = 1 - \frac{2m(t,r)}{r}$.

1.2.3 Modelo de colapso gravitacional con onda de choque en la PQA

Las regiones y ecuaciones de estado

Se emplea como ecuación de estado del núcleo la solución interior anisótropa de Schwarzchild [57],[67]:

$$\tilde{\rho}_I = f(t) \qquad \text{y} \qquad \tilde{P}_I = \tilde{\rho}_I\left\{\frac{3(1-8/3\pi r^2\tilde{\rho}_I)^{\xi_I/2}k(t) - 1}{3 - (1-8/3\pi r^2\tilde{\rho}_I)^{\xi_I/2}k(t)}\right\}, \tag{1.15}$$

donde $\xi_I = 1 - 2h_I$ es el parámetro de anisotropía, si $h_I = 0$ corresponde al modelo isotrópico. Para el manto se emplea la solución anisótropa Tolman VI:

$$\tilde{\rho}_{II} = \frac{3g(t)}{r^2} \qquad \text{y} \qquad \tilde{P}_{II} = \frac{\tilde{\rho}_{II}}{3}\left[\frac{1 - 9D(t)r^{\sqrt{4-3\xi_{II}}}}{1 - D(t)r^{\sqrt{4-3\xi_{II}}}}\right]. \tag{1.16}$$

de nuevo $\xi_{II} = 1 - 2h_{II}$ es el parámetro de anisotropía en el manto y las funciones $f(t)$, $k(t)$, $g(t)$ y $D(t)$ son obtenidas por medio de las condiciones de contorno.

Funciones métricas y variables efectivas

A partir de la introducción de las variables adimensionales en la superficie M, A, F y Ω y de las variables físicas $\hat{E}$ y L es posible obtener todas las variables físicas como función de las variables adimensionales en la superficie.

$$M = \frac{m_a}{m_a(0)}, \quad A = \frac{a}{m_a(0)}, \quad F = 1 - \frac{2M}{A}, \quad \Omega = \omega_a, \quad \text{y} \quad t \to \frac{t}{m_a(0)}, \tag{1.17}$$

$$\hat{E} = 4\pi a^2 \mathcal{F}_a, \qquad L = -\dot{M}, \quad \text{y} \qquad E = \frac{\dot{M}}{F}, \tag{1.18}$$

Donde $m_a(0)$ es la masa total inicial de la distribución, $\hat{E}$ es la luminosidad para un observador no comóvil y L es la luminosidad al infinito. Entonces para el manto se obtiene:

$$m_{II} = \frac{Mr}{A}, \qquad E = (1+\Omega)\hat{E}, \tag{1.19}$$

$$\tilde{\rho}_{II} = \frac{1-F}{8\pi r^2}, \qquad \tilde{P_{II}} = \frac{1-F}{24\pi r^2}\left[\frac{\psi - 9\chi(r/a)^{\sqrt{4-3\xi_{II}}}}{\psi - \chi(r/a)^{\sqrt{4-3\xi_{II}}}}\right] \tag{1.20}$$

$$\mathrm{e}^{-\lambda_{II}} = F, \qquad \mathrm{e}^{-\nu_{II}} = F\left\{\frac{r}{a}\left[\frac{\psi - \chi(r/a)^{\sqrt{4-3\xi_{II}}}}{\psi - \chi}\right]^{2/\sqrt{4-3\xi_{II}}}\right\}^{\frac{4(1-F)}{3F}} \tag{1.21}$$

Donde $\psi = 3(3+\Omega)(1-F)6(1+\Omega)E$ y $\chi = (1+3\Omega)(1-F) - 6(1+\Omega)E$

Mientras que para el núcleo mediante la introducción del parámetro de fuerza de choque $N(\tilde{P}_I)_c = N(\tilde{P_{II}})_c$ se obtiene:

$$m_I = \frac{Mc}{A}(r/c)^3, \qquad \tilde{\rho_I} = \frac{3M}{4\pi c^2 A}, \qquad \tilde{P}_I = \tilde{\rho_I}\,\frac{3u^{\xi_I/2}k(t) - 1}{3 - u^{\xi_I/2}k(t)} \tag{1.22}$$

$$\mathrm{e}^{-\lambda_I} = 1 - \frac{2Mc}{Ar}(r/c)^3, \qquad \mathrm{e}^{-\nu_I} = Hu^{\Phi}(3 - ku^{\xi_I/2}k(t))^{\frac{8}{\xi_I}} \tag{1.23}$$

Donde

$$k = \frac{24\pi c^2\tilde{\rho_I}\beta + 3N\alpha(1-F)}{F^{\xi_I/2}[72\pi c^2\tilde{\rho_I}\beta + N\alpha(1-F)]}, \qquad H = \frac{F^{1-\Phi}\ \frac{c}{a}\left[\frac{\beta}{8(F-1)}\right]^{2/\sqrt{4-3\xi_{II}}}{}^{\frac{4(1-F)}{3F}}}{(3 - kF^{\xi_I/2})^{8/\xi_I}} \tag{1.24}$$

$$u = 1 - \frac{8\pi r^2 \tilde{\rho}_I}{3}, \qquad \Phi = \frac{1}{2} - \frac{3(1-F)}{16\pi c^2 \tilde{\rho}_I}, \qquad \alpha = \psi - 9\chi(c/A)^{\sqrt{4-3\xi_{II}}}, \qquad \beta = \psi - \chi(c/A)^{\sqrt{4-3\xi_{II}}} \tag{1.25}$$

Todas las variables efectivas y las funciones métricas dependen de t a través de las variables de superficie A, F, Ω y L.

Ecuaciones de superficie

A fin de encontrar la evolución de las variables en la superficie, es necesario integrar un sistema de ecuaciones diferenciales ordinarias sobre A, F, Ω y L, llamadas ecuaciones de superficie. La primera ecuación se obtiene a partir de la relación entre la velocidad y la velocidad del observador comóvil $\dot{r} = e^{\frac{\nu-\lambda}{2}}\omega$, evaluada sobre la superficie de contorno $\sum r - a(t)$. La segunda ecuación surge de la derivada temporal de F y usando la definición de la luminosidad L.

$$\dot{A} = F\Omega, \qquad \dot{F} = \frac{(1-F)F\Omega + 2L}{A} \tag{1.26}$$

La ecuación diferencial para Ω se obtiene evaluando la ley de conservación $T^{\alpha}_{r;\alpha} = 0$ sobre $\sum$:

$$\tilde{P}' + \frac{(\tilde{\rho} + \tilde{P})(4\pi r^3 \tilde{P} + m)}{r(r-2m)} = \frac{2(\bar{P}_{\perp} - \tilde{P})}{r} + \frac{\mathrm{e}^{-\nu}}{4\pi r(r-2m)}\left(\ddot{m} + \frac{3\dot{m}^2}{r-2m} - \frac{\dot{m}\dot{\nu}}{2}\right) \tag{1.27}$$

Finalmente se introduce un pulso Gaussiano centrada a $t = t_0$ para describir la luminosidad:

$$-\dot{M} = L = \frac{\Delta M_{rad}}{s\sqrt{2\pi}} e^{\left[-\frac{1}{2}\left(\frac{t-t_0}{s}\right)^2\right]} \tag{1.28}$$

Donde s es el ancho del pulso y ΔM_{rad} es la masa total perdida en el proceso.

El conjunto de subrutinas desarrolladas en fortran permiten calcular a partir de las variables de radiación, las ecuaciones de campo 1.6-1.9, de la ecuación de estado anisótropa y de algunos parámetros de entrada, las variables físicas ρ, P_r, $P_\perp$, ω y $\mathcal{F}$, tanto las contribuciones hidrodinámicas como las contribuciones de la radiación para cada radio de la esfera y para un lapso de tiempo establecido. Dichos parámetros de entrada son los siguientes: $A(0)$ (radio inicial de la configuración), $\Omega(0)$ (velocidad inicial de la frontera), $c(0)$ (la posición inicial de la onda de choque), N (parámetro de fuerza del choque, es decir, relación entre la presión efectiva a ambos lados de la onda de choque), ξ_I (parámetro de anisotropía en el núcleo), ξ_{II} (parámetro de anisotropía en el manto), f_I (factor variable de Eddington en el núcleo), f_{II} (factor variable de Eddignton en el manto), $M(0)$ (masa inicial de la configuración), M_f (masa final de la configuración), τ, s (ancho de la función Gaussiana que se emplea para calcular la luminosidad), t_0 (inicio de la evolución) y t_f (fin de la evolución)

En resumen, la comprensión del proceso de colapso gravitacional dentro del marco de la Teoría General de Relatividad es indispensable para desentrañar las últimas etapas de la evolución estelar. Un caso muy interesante es la propagación de discontinuidades hidrodinámicas en esferas

radiantes. Existen un conjunto de subrutinas escritas en lenguaje fortran que permiten modelar la evolución de una onda de choque en una esfera relativista autogravitante en el contexto de una aproximación post-cuasiestática [62]. A partir de las condiciones iniciales es posible integrar numéricamente el sistema de ecuaciones diferenciales en la superficie de la distribución y luego evaluar las expresiones para las funciones métricas para determinar y validar el valor de las variables físicas (las contribuciones hidrodinámicas y de la radiación a la presión radial, presión tangencial, flujo de radiación, densidad y la velocidad radial del material). Estas subrutinas conforman el modelo físico que sustenta al portlet CHOQUE.

Capítulo 2

Visualización Científica

El pensamiento del ser humano esta lleno de evaluaciones cualitativas, lo cual implica un sentido aproximado de cantidad, tamaño y escala. En el mundo científico, el razonamiento a diferentes órdenes de magnitud y la precisión de las medidas son cuestiones dominantes. ¿Cómo pueden ser representadas las cantidades en expresiones visuales? ¿Cómo puede ser una imagen cuantitativamente elocuente? son cuestiones a las cuales investigadores de diferentes áreas tales como dinámica de fluidos, modelado molecular, ingeniería computacional, geofísica, matemáticas, relatividad numérica, tecnologías de la información entre otras se han avocado a responder, haciendo uso del creciente poder computacional y del desarrollo de diferentes sistemas gráficos.

2.1 Definiciones y motivaciones

Visualización se define como la generación de una imagen mental o una imagen real de algo abstracto o que no pueda observarse por métodos comunes, para hacerlo visible a la mente o a la imaginación. La Visualización Científica envuelve las técnicas concernientes a la presentación de datos científicos a los usuarios por medio de imágenes empleando un proceso comprensible y reproducible. Una visualización exitosa provee una representación que permite al usuario conocer y entender la estructura de los datos, o comunicar aspectos de su estructura en forma efectiva [1] [2] [3].

La Visualización Científica permite comprimir una gran cantidad de datos en una imagen, revelar cosas que no podemos captar al ver una tabla de datos como correlaciones entre diferentes cantidades en el espacio y/o en el tiempo o patrones de comportamiento, la comprensión de procesos y conceptos abstractos, simplifica la comunicación entre científicos ya que es independiente del lenguaje y existen técnicas que permiten ver la representación de los datos en forma selectiva e interactiva en tiempo real.

El uso de técnicas de visualización para representar información no es un fenómeno nuevo, éstas han sido empleadas en mapas y gráficos de datos por cientos de años. Pero el aumento

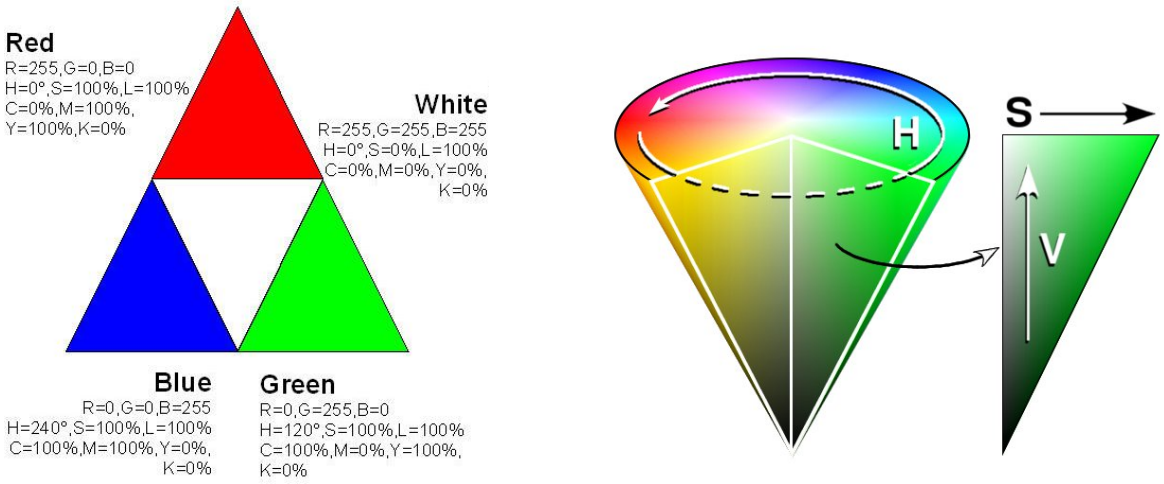
del poder de procesamiento y de almacenamiento de las computadoras que ha permitido simular fenómenos naturales con una precisión muy alta, generando diariamente bases de datos numéricas del orden de Terabytes (10^{12} bytes); el incremento de las cantidades de datos generados por telescopios, satélites, aceleradores de partículas, aparatos médicos entre otros; y el inicio de ciencias computacionales dieron origen a la visualización científica como creciente disciplina a finales de los años 80. Desde entonces han habido muchas conferencias y congresos patrocinados por la Sociedad de Computación del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y por la SIGGRAPH (*Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques*) destinados al desarrollo de nuevas técnicas y nuevos sistemas de visualización . Entre los sistemas de visualización de propósito general actualmente empleados se encuentran IRIS Explorer, Stardent's AVS, Khoros (de la Universidad de Nuevo Mexico), PV-WAVE (Precision Visuals' Workstation Analysis and Visualization Environment), the Visualization Toolkit (VTK) y IBM Visualization Data Explorer precursor de IBM OpenDX [4].

2.2 Procesos de la Visualización Científica

Para generar imágenes entendibles y confiables a partir de datos abstractos es necesario definir procesos estándares que satisfagan criterios establecidos para la representación. Robertson y De Ferrari (1994) describen un modelo de sistema de visualización de seis componentes: manipulación de la data, especificaciones de la visualización, establecimiento del tipo de representación visual a ser utilizada, renderización, visualización final e interacción. El proceso de visualización incluye varias etapas, *preprocesamiento de la data*, *mapeo de la data* y *renderización*. La data es obtenida bien sea a través de mediciones o de la ejecución de modelos computacionales, su preprocesamiento por lo general incluye operaciones como la realización de interpolaciones y la reducción o filtrado de la data basados en las especificaciones de la visualización. El mapeo es la etapa más importante del proceso, envuelve el diseño de una representación adecuada de la data preprocesada, por lo general esta representación es un objeto geométrico bidimensional o tridimensional con atributos como color y opacidad. Finalmente estos resultados son renderizados, es decir, se genera la imagen para comunicar la información al usuario. Existen diferentes métodos para procesar la data, mapearla y renderizarla, algunos de ellos serán discutidos a continuación [5] [3].

2.3 Técnicas de la Visualización Científica

La clasificación de las técnicas de visualización se basa en diferentes criterios: objetivo de la visualización (obtener una visión general de los datos, comparar los datos, hallar patrones de distribución, realizar análisis de correspondencia y/o correlación), del tipo (escalares, vectoriales o tensoriales) y/o de la dimensionalidad de la data (1D, 2D, 3D, multidimensional), del modo de visualización (pixel orientadas, proyecciones geométricas, basadas en íconos, orientadas a establecer jerarquías), del estilo de interacción, entre otros [6].



(a) Modelo de color aditivo RGB

(b) Cono de color del modelo HSV

Figura 2.1: Códigos de colores empleados usualmente en el mapeo por color

La manipulación por color (o **mapeo por color**) es una de las técnicas de visualización empleadas con mayor frecuencia, por eso, es preciso conocerla con mayor profundidad.

2.3.1 Uso del color para representar datos. Mapeo por color

Una manera de emplear el color para representar cantidades es mediante el uso de un *mapa de color*. Un mapa de color es un arreglo de colores usado para mapear los valores de la data en colores, a fin de que sea posible conocer el rango de la data durante la visualización. Ya que el mapeo de datos a colores envuelve los códigos de colores o sistemas de organización de colores, se considera necesario describirlos con algún detalle. En general en los códigos de colores cada color es localizado en un espacio tridimensional (formado por tres dimensiones perceptibles del color), los más comunes son el sistema RGB y el HSV [7] [8].

La descripción **RGB** (del inglés *Red*, *Green*, *Blue*; Rojo, Verde, Azul) de un color hace referencia a la composición del color en términos de la intensidad de los colores primarios con que se forma: el rojo, el verde y el azul. La aplicación más común del sistema RGB es la visualización de colores en un tubo de rayos catódicos, una pantalla de cristal líquido o de plasma. Es frecuente que en las representaciones del computador se codifique cada color primario con un byte, aunque el intervalo pudiera ser cualquiera. Así, de una manera estándar, la intensidad de cada una de las componentes se mide según una escala que va del 0 al 255, lo cual implica que es posible obtener 16777216 colores diferentes.

El modelo **HSV**(del inglés *Hue*, *Saturation*, *Value*; Tonalidad, Saturación, Valor), también llamado HSB (*Hue*, *Saturation*, *Brightness*; Tonalidad, Saturación, Brillo), define un modelo de color en términos de sus componentes constituyentes en coordenadas cilíndricas:

- Tonalidad, es la dimensión perceptual del color asociada con los nombres de los colores:

amarillo, naranja, rojo, azul y verde. Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100%).

- Saturación, es la dimensión asociada con la viveza del color, lo que en arte se conoce como tonos. Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100%.

- Valor del color, el brillo del color, es la dimensión que describe la cantidad de luz que parece ser reflejada por un objeto. Representa la altura en el eje blanco-negro. Los valores posibles van del 0 al 100%.

El modelo HSV se trata de una transformación no lineal del espacio de color RGB y en general es el modelo empleado por diversos sistemas como Munsell, CIELAB y PRAVDA para producir las producciones de color lógicas para la representación de datos.

La interpretación de resultados producidos por esta técnica depende de colocar el color apropiado en el lugar adecuado, lo cual es un asunto complejo ya que el ojo humano es más sensible a algunas partes del espectro visible de la luz que a otras y el cerebro es capaz de interpretar diferentes secuencias de colores en forma diferente, tal que el impacto perceptual del color es difícil de predecir. Interpretaciones erradas de la data debido a la elección equivocada de secuencias de colores y a la sensibilidad de los códigos de colores a efectos contextuales interactivos son comunes en muchas presentaciones visuales de datos, lo cual hace que la visualización sea menos convincente.

Por ejemplo, la Figura 2.2 muestra la Imagen de Resonancia Magnética (IRM) de una cabeza humana, la única diferencia entre ellas son los mapas de color empleados para representar la data, aun así las cuatro representaciones son diferentes. El *mapa de color arcoiris*, utilizado comúnmente en la mayoría de las visualizaciones científicas (superior izquierdo), crea contornos percibidos que no refleja en este caso transiciones discretas en la data. El *mapa de color isomorfo* (superior derecho) produce una representación confiable de la estructura de la data y su diseño depende de la frecuencia espacial de la data, en este caso muestra las características de frecuencias espaciales bajas (como el tumor cerca del centro de la imagen) a través de la variación de la tonalidad. El *mapa de color segmentado* (inferior izquierdo) delínea las regiones de la imagen mediante el uso de segmentos de color que pueden ser percibidos visualmente, en este caso demuestra características de frecuencias espaciales altas. El *mapa de color de realce* es diseñado para realzar rangos particulares de la data, en este caso el mapa de color destaca valores de datos cercanos a la mitad del rango.

En general una buena estrategia es emplear secuencias de colores encontradas en la naturaleza de intensidad suave ya que son familiares, coherentes y brindan armonía al ojo humano, mientras que colores de intensidades altas impresionan demasiado al ojo y causan pérdida de información al momento de realizar la interpretación de la data [9] [10] [11] [12].

[1] NASA http://www.nasa.gov/

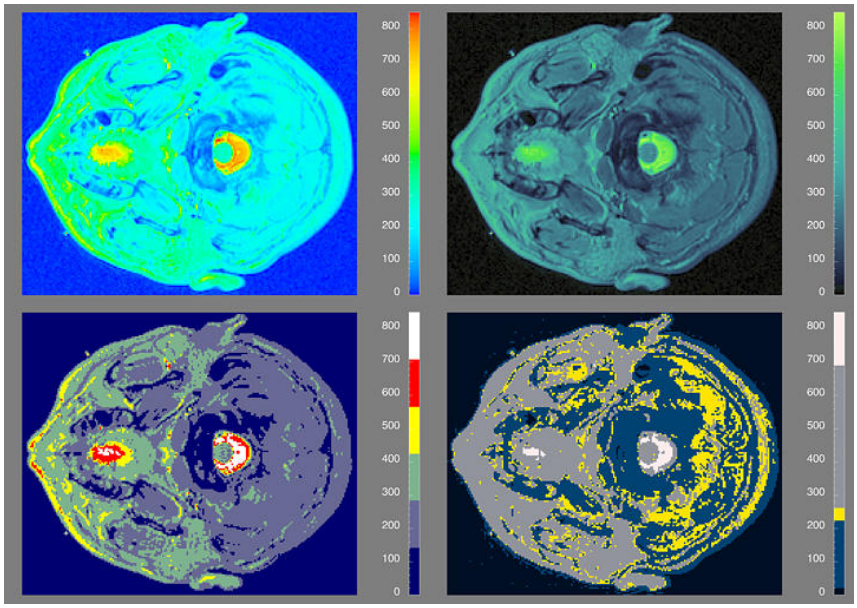


Figura 2.2: Datos de MRI visualizados empleando mapa de color de arcoiris,isomorfo, segmentado y de realce. Tomada de la página de NASA/GSFS. [1]

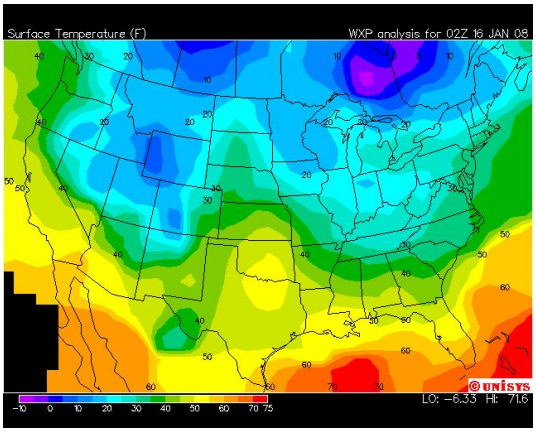
2.3.2 Técnicas de visualización de campos escalares

Una de las operaciones más comunes es la visualización de una variable expresada en función de su posición en el espacio y el tiempo (cantidad escalar) en un campo tridimensional. Cantidades tales como esfuerzo, temperatura, presión, densidad, rapidez o errores estimados son generalmente representados como campos bidimensionales o tridimensionales de una sola variable. En general las técnicas empleadas para visualizar variables escalares unidimensionales y bidimensionales son el *mapeo por color* y *gráficos de contorno* donde los resultados son representados mediante líneas de colores o regiones coloreadas sobre las superficies visibles exteriores de una estructura. Técnicas más recientes permiten mostrar la variación tridimensional completa de una variable escalar tridimensional en un campo volumétrico, estas técnicas incluyen *isosuperficies* y *volume slicing* también conocida como *planos cortantes*. La combinación de estos métodos permiten una evaluación completa del comportamiento tridimensional del campo escalar, ya que permiten ver información que no están en la superficie exterior visible del campo.

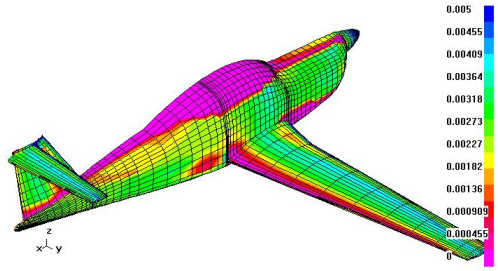
Isosuperficies

Una isosuperficie es la superficie 3D que representa la localización del valor de un resultado escalar constante en un volumen. Existen varios métodos para generar isosuperficies a partir de un conjunto discreto de datos, la mayoria se basa en la interpolación para construir una función continua que represente dichos datos. Uno de los desarrollos claves en visualización volumétrica

[2] UNYSYS http://weather.unisys.com/

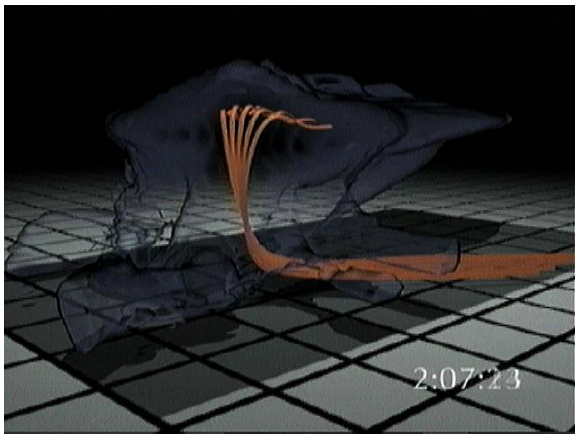


(a) Temperaturas superficiales en Estados Unidos[2] (imagen tomada de UNISYS Weather)

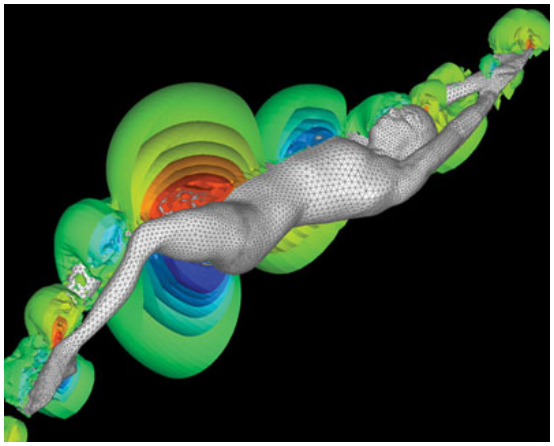


(b) Coeficiente de fricción mapeado sobre el modelo empleando Postmarc

Figura 2.3: Imágenes generadas empleando gráficos de contorno



(a) Modelo de una tormenta realizada por Wilhelmson et al. en 1990 en la Universidad de Illinois. Esta tormenta ocurrió el 3 de abril de 1964, cruzó Oklahoma y Texas por 2 $\frac{1}{2}$ horas.



(b) Isosuperficies de presión, alrededor del cuerpo del nadador. El rojo indica presiones altas, mientras que el azul las bajas (Flow Simulations and Analysis Group at George Washington University (GWU)).

Figura 2.4: Imágenes generadas empleando isosuperficies

de datos escalares fue el algoritmo *the Marching Cubes* de Bill Lorensen y Harvey Cline [13], el cual fue patentado por la Compañía General Electric en 1987. Uno de los ejemplos más famosos de isosuperficies fue hecho por Wilhelmson et al. en la Universidad de Illinois en 1990, el cual presenta una animación de una tormenta, aparte de isosuperficies empleo otras tecnicas como streamlines, ribbons y sombreado [14] [15].

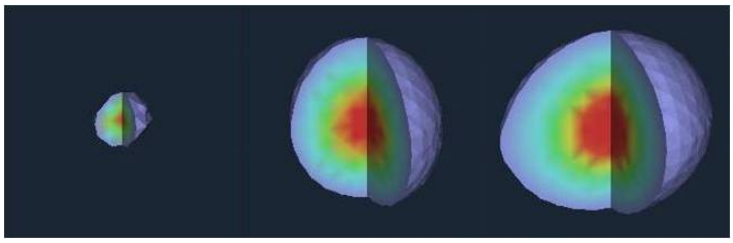


Figura 2.5: Probabilidad de distribución del par quark-antiquark dentro de un mesón en el tiempo empleando planos cortantes (Babich R. et al., Lattice QCD 2006, Universidad de Boston).

Planos cortantes de volumen

Planos cortantes es una técnica que permite remover parte de un volumen empleando un plano cortante para observar los elementos individuales o interiores. La importancia de esta técnica reside en que permite visualizar información crítica que tal vez empleando los otros métodos no pueda ser observada, ya que los gráficos de contorno solo muestran las superficies exteriores visibles y las isosuperficies pueden ocultar otras isosuperficies.

2.3.3 Visualización de flujo de datos

Visualización de flujo de datos (*flow data*) juega un rol importante tanto en ciencia como en ingeniería en áreas como dinámica de fluidos, simulaciones atmosféricas y aerodinámica, sus aplicaciones van desde estudio de turbulencias de plasmas hasta el diseño de jets. La data representada en un espacio n-dimensional incluye campos escalares n-dimensionales (univariados), campos vectoriales (n-variados) y campos tensoriales de segundo orden (n^2-variados). En general datos multivariados son más difíciles de visualizar cuando el número de variables incrementa.

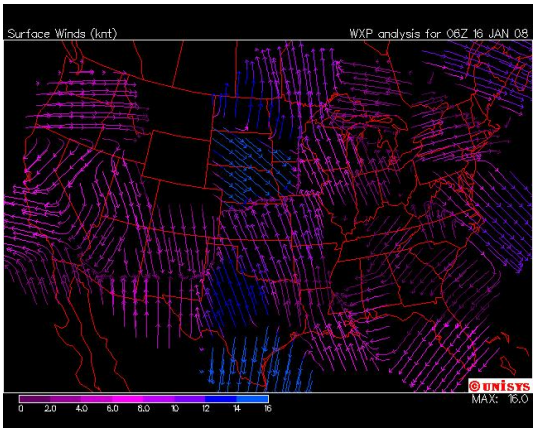
Íconos

En visualización científica los íconos son definidos como objetos geométricos que codifican la data bien sea a través de características geométricas como longitudes y ángulos o de atributos visibles como color u opacidad. Un ícono está basado en la semejanza entre la data y su representación.

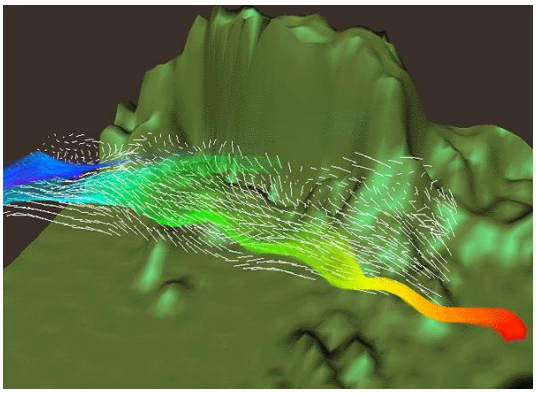
a) Íconos puntuales.

El mapeo de vectores más sencillo consiste en dibujar íconos puntuales tal como flechas en puntos seleccionados en el flujo o corriente. Por otra parte el mapeo de tensores más sencillo consiste en emplear elipsoides en vez de flechas. Sin embargo, no es posible comprender la estructura subyacente de un campo vectorial o tensorial mediante la interpolación mental de íconos adyacentes, excepto para objetos simples [16].

[3] UNISYS http://weather.unisys.com/
[4] NASA http://www.nasa.gov/



(a) Campo de velocidad del viento en Estados Unidos (imagen tomada de UNISYS Weather) [3]



(b) Corriente dinámica simulada empleando datos de la velocidad del viento en Indonesia (imagen tomada de la página de la NASA) [4]

Figura 2.6: Imágenes de campos vectoriales generadas empleando íconos puntuales

b) Trazas, curvas de flujo y curvas de velocidad

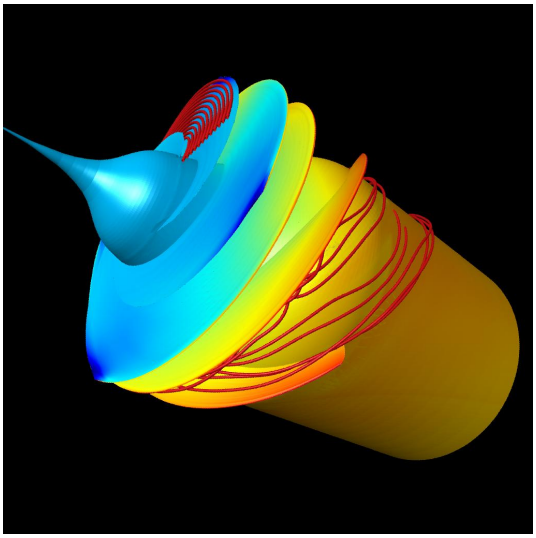
El uso de líneas como íconos es más eficiente en el sentido que proveen una representación continua de la data, entonces evitan realizar interpolaciones mentales, mejorando la percepción del fluido y enfatizando la continuidad del campo vectorial. Las trazas representan el camino de una partícula. Las curvas de flujo o *streaklines* son las líneas que pasan a través de un punto por el cual todas las partículas han pasado. Las curvas de velocidad o *streamlines* son curvas instantáneamente tangentes a la traza de la partícula. Es posible generalizar *streamlines* a *hyperstreamlines*, las cuales representan toda la información tensorial a lo largo de la trayectoria.

c) Superficies de velocidad (*streamsurfaces*) y *stream ribbons* Las *streamsurfaces* son superficies definidas por un conjunto de *streamlines* adyacentes. Mientras que un *stream ribbon* es una superficie entre dos *streamlines* adyacentes, lo cual puede apreciarse en las Figura 2.8 y en la Figura 2.9.

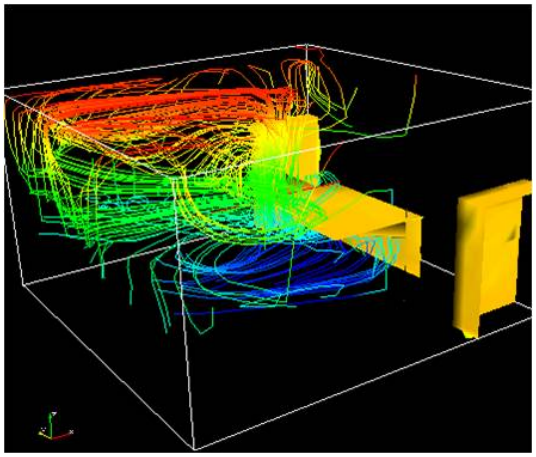
2.3.4 Visualización de Volúmenes continuos

La visualización volumétrica de data tridimensional se ha convertido en una necesidad en diferentes áreas de la ciencia, ingeniería y medicina. Por ejemplo la construcción de un modelo tridimensional a partir de la data bidimensional obtenida a partir de Imágenes por Resonancia Magnética (IRM) o Tomografías Computarizadas (TC) constituye data volumétrica, la cual necesita ser visualizada para diagnósticos médicos. TC es también empleada en las industrias para

[5] IBM Open Visualization Data Explorer http://www.research.ibm.com/dx/
[6] http://www.nasa.gov/centers/ames/home/index.html
[7] http://www.ccs.lanl.gov/source/orgs/ccs/ccs1/acl/index.shtml
[8] GWU http://project.seas.gwu.edu/~fsagmae/

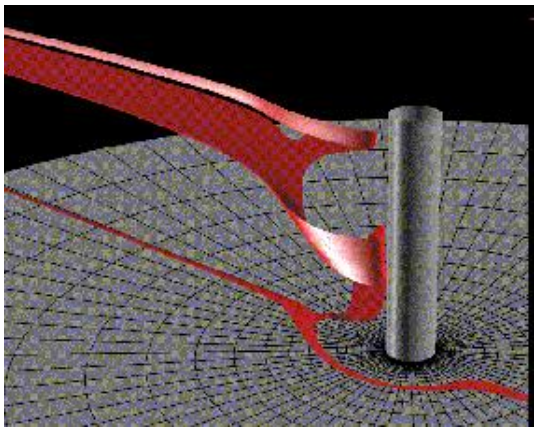


(a) Las curvas de flujo o *streaklines* permiten visualizar el vórtice del cohete (imagen generada empleando IBM Open Visualization Data Explorer)[5]

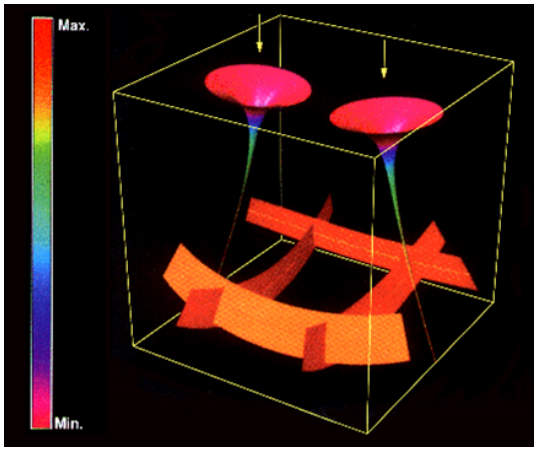


(b) Corrientes de aire que fluyen por una oficina. Los *streamlines* fueron coloreados de acuerdo a la presión de aire.

Figura 2.7: Curvas de velocidad del flujo de aire en una oficina, los muebles dentro de la oficina son isosuperficies. Las *streamlines* fueron coloreadas de acuerdo a la presión del aire.



(a) Superficies de velocidades alrededor de un poste (NASA Ames Research Center)[6]



(b) *Hyperstreamlines* que representan el tensor de stress inducido por dos fuerzas compresivas (Los Alamos Advanced Computer lab)[7]

Figura 2.8: *Hyperstreamlines* y superficies de velocidades.

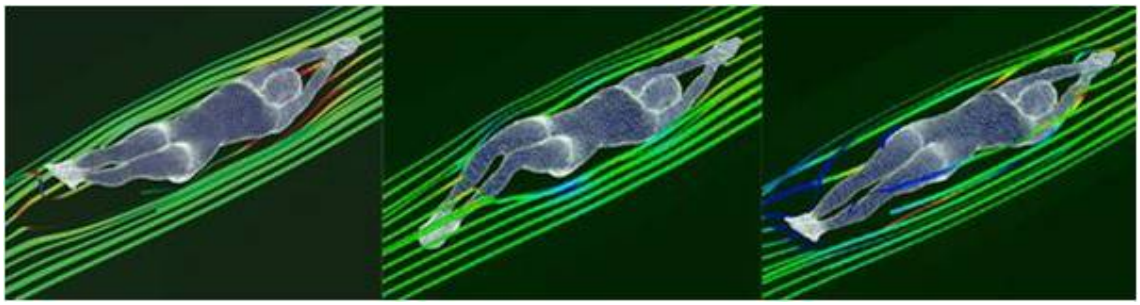
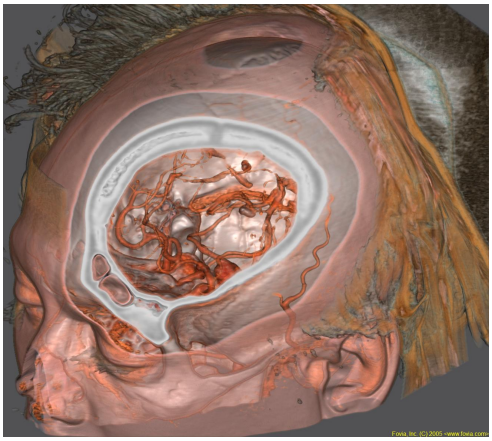


Figura 2.9: *Stream ribbons* que indican la dirección del agua en tres instantes de tiempo (Flow Simulations and Analysis Group at George Washington University (GWU)). [8]



(a) Renderización de volúmenes de la cabeza

(b) Renderización de superficie de un feto de tres meses

realizar una inspección no destructiva de materiales compuestos o de partes mecánicas. Esto ha impulsado el desarrollo de muchas técnicas para la visualización de data tridimensional.

Estas técnicas de visualización se clasifican en *técnicas de renderización de superficies* y *técnicas de renderización de volúmenes*. La renderización de superficies consiste en realizar representaciones bidimensionales de campos escalares o vectoriales tridimensionales, lo cual implica que una dimensión de información es desechada. Entre estas técnicas se encuentran isosuperficies y la utilización de íconos.

Las técnicas de renderización de volúmenes permiten la visualización de data tridimensional sin la conversión a representaciones bidimensionales, por lo tanto, son técnicas que transmiten más información que las de renderización de superficies, pero esto implica algoritmos más complejos y por lo general mayor tiempo. En general los métodos de implementación de renderización de volúmenes difieren en la forma de realizar la proyección de los datos en el plano de visualización [17] [14].

2.3.5 Animación

La animación provee la importante habilidad de visualizar el comportamiento de un sistema determinado en el espacio y el tiempo. La animación es la presentación de una secuencia de imágenes, llamados cuadros, a gran velocidad a fin de crear la ilusión de movimiento, es decir, a fin de que el ojo sea capaz de integrarlas en un movimiento continuo. Las animaciones tradicionales fueron inicialmente desarrolladas por artistas para crear la ilusión de vida y hoy en día se ha convertido en una forma de arte. El uso de animaciones como técnica de visualización científica es fundamental en el investigación y en los procesos de diseño ya que permite entender cada paso del fenómeno o sistema en estudio. Por ejemplo detectar conflictos en el movimiento de las partes de una pieza es más fácil en una animación que a través de visualizaciones estáticas. Las animaciones científicas requieren un diseño más sencillo que las animaciones tradicionales a fin de evitar distracciones de la data y preservar la fidelidad y precisión científica.

2.4 Retos en la Visualización Científica

- Mejorar la calidad de las imágenes para lograr una mayor semejanza con la realidad.
- Integrar la realidad virtual con la realidad física, es decir, eliminar los lentes, guantes y cascos a fin de que la visualización sea más natural.
- Integrar la visualización con redes de computación distribuida, voz y visión artificial.
- Encontrar maneras efectivas para visualizar la información no numérica, como secuencias genéticas.
- Integrar el conocimiento sobre la percepción en el ser humano en el proceso de diseño de las interfases de visualización para lograr una mayor comprensión de la información que se desea transmitir.
- Desarrollar representaciones para datos de dimensiones muy altas y para el análisis de errores [18].

Diferentes técnicas de visualización científica han sido desarrolladas a fin de revelar la estructura completa de los campos escalares, vectoriales o tensoriales en estudio, solo es necesario recordar que la elección de la(s) técnica(s) apropiada(s) es crítica para comprender los datos, lo cual beneficiará la percepción de la data, el subsecuente análisis, procesamiento y la toma de decisiones. Tradicionalmente, la visualización de datos de estuvo restringida a unas estaciones de trabajo especializadas en el tratamiento gráfico, conectadas a supercomputadores de cálculo que abastecen de resultados a la estación gráfica en tiempo real. Hoy en día con el desarrollo de la Web y de sistemas de computación distribuidos es posible crear un medio para presentar las visualizaciones científicas con una mayor nivel de interacción y con mayor rapidez.

Capítulo 3

Computación Grid

Del mismo modo que la disponibilidad de computadores potentes y económicos, junto con redes de alta velocidad, permitieron la popularización de la computación distribuida, la disponibilidad y popularidad de la Internet permiten unir recursos computacionales distantes geográficamente para utilizarlos como si fueran un recurso computacional único y unificado; esto último se conoce como computación Grid. Igual que la red eléctrica permite obtener energía en cualquier lugar y de forma transparente al usuario, se pretende que los Grids computacionales provean "energía de computación" (ciclos de CPU, espacio en disco, ...) desde cualquier ubicación, utilizando para ello la Internet. Para esto, es necesario utilizar ciertas herramientas que cohesionen los recursos pertenecientes a uno de estos sitios de Grid y proporcionen los mecanismos necesarios para que los usuarios puedan acceder al mismo. Estas herramientas, en muchas ocasiones, se complementan con interfases de usuario conocidas como "portales Grid" que permiten al usuario una fácil interacción con el Grid.

3.1 Definición de la computación Grid y motivaciones

Es prácticamente imposible hoy en día hacer ciencia sin computadores. Con frecuencia una computadora, un cluster de computadores o inclusive una supercomputadora de propósito definido no es suficiente para realizar los cálculos que los científicos desean. Como resultado los científicos enfrentan situaciones en las cuales el logro de ciertas metas es muy difícil, costoso o incluso imposible con la tecnología computacional disponible. Una de esas situaciones se presenta en el campo de la física de altas energías, específicamente en el 2008 cuando el acelerador de partículas de más alta energía del mundo el *Large Hadron Collider* (LHC) comience a funcionar generará alrededor de 10 Petabytes de datos por año, y miles de físicos alrededor del mundo querrán tener acceso a ellos para analizarlos diariamente. La solución es conseguir una mayor potencia de cálculo, almacenamiento, aprovechamiento de recursos, etc. combinando los recursos computacionales

en forma segura de varias organizaciones como una grande, única y poderosa computadora a través de la Internet. Esto, en esencia, se conoce como *computación Grid*, un nuevo paradigma de computación distribuida propuesto por Ian Foster y Carl Kesselman a mediados de los 90. El término Grid se acuñó por analogía con las redes de distribución eléctricas (*power grids*), gracias a las que se puede consumir energía eléctrica de forma confiable, transparente y en cualquier lugar y en cualquier momento, independientemente de su origen o de las necesidades de otros centros de consumo [19] [20].

Varios conceptos coexisten acerca de qué es un Grid. Uno de ellos, elaborado por el *Grid Computing Information Centre* [21], una de las asociaciones dedicada exclusivamente al desarrollo de esta tecnología, llama Grid a un "tipo de sistema paralelo y distribuido que permite compartir, seleccionar y reunir recursos 'autónomos' geográficamente distribuidos en forma dinámica y en tiempo de ejecución, dependiendo de su disponibilidad, capacidad, desempeño, costo y calidad de servicio requerida por sus usuarios". Según esta definición, se busca aprovechar la sinergia que surge de la cooperación entre recursos computacionales y proveerlos como servicios. La infraestructura Grid, conecta diversos recursos de *hardware*, *software* y datos a través de una red, asegurando un acceso seguro y flexible para aplicaciones, datos, poder de procesamiento y capacidad de almacenamiento, de la misma manera en que la Web permite acceso seguro y flexible a la información [22], esta idea es ilustrada en la figura 3.1

Otra definición más estructurada expuesta por Foster, Kesselman y Tuecke [23], precursores de la computación Grid, plantea la existencia de *organizaciones virtuales* (VO por sus siglas en inglés) como puntos de partida de este enfoque. Una organización virtual es "un conjunto de individuos y/o instituciones definida por reglas que controlan el modo en que comparten sus recursos". Básicamente, son organizaciones unidas para lograr objetivos comunes. Las VOs varían enormemente en cuanto a sus objetivos, alcance, tamaño, duración, estructura, comunidad y sociología. Sin embargo, existen varios requerimientos y problemas subyacentes tales como la necesidad de relaciones flexibles para compartir recursos, niveles de control complejos y precisos, variedad de recursos compartidos (programas, archivos, datos, sensores y redes, entre otros), modos de funcionamiento (individual, multiusuario), calidad de servicio, etc. Las tecnologías actuales o bien no proveen espacio para la variedad de recursos involucrados o no aportan la flexibilidad y control de las relaciones cooperativas necesarias para establecer las VOs. Como solución, se propone el Grid como un modelo de trabajo para "compartir recursos en forma coordinada y resolver problemas en organizaciones virtuales multi-institucionales de forma dinámica". De esta manera, varias instituciones pueden formar distintas VOs e incluso formar parte de más de una al mismo tiempo, realizando diferentes roles e integrando distintos recursos.

La computación Grid va más allá de una simple comunicación entre computadoras, y su objetivo final es convertirse en la red global dentro de un enorme recurso computacional. Esto es aún es un sueño, ya que la realidad es que el Grid es un trabajo en progreso y la tecnología subyacente aún es un prototipo que está siendo desarrollado por cientos de investigadores e ingenieros de *software* alrededor del mundo. En la actualidad no existe un único Grid (como existe una única Internet o una única Web), por el contrario existen varios Grids en desarrollo, algunos públicos, otros privados, localizados en diferentes regiones o países, algunos dedicados a un problema en particular y otros de propósitos generales. Aún falta que los recursos estén disponibles

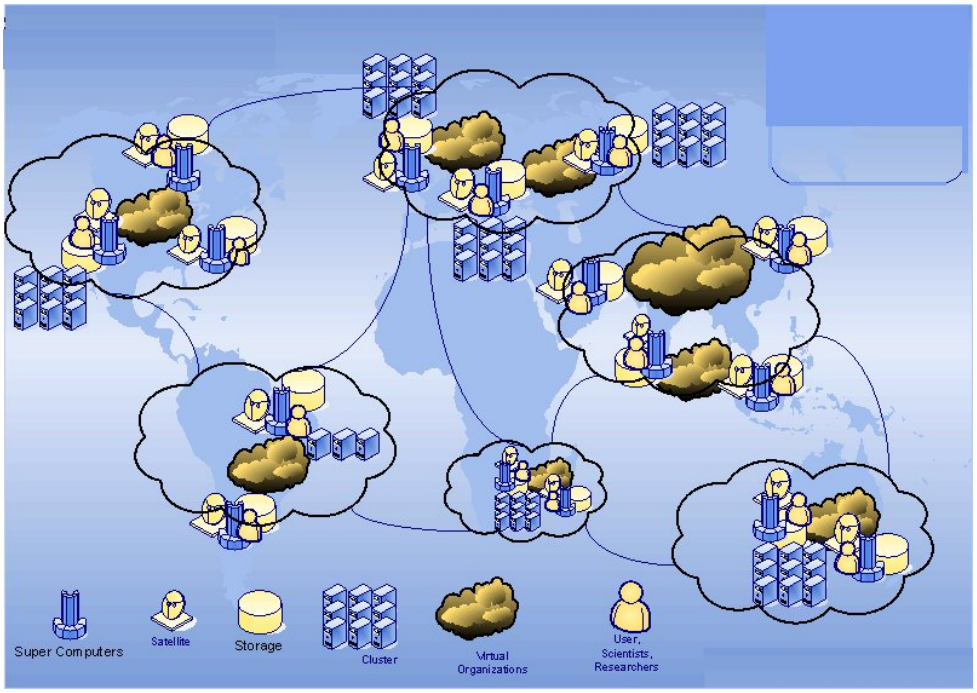


Figura 3.1: Los Grids Computacionales ofrecen a las organizaciones grandes beneficios como el aumentar su capacidad de cómputo y almacenamiento al compartir los recursos existentes y de esta manera ahorrar dinero evitando la adquisición de nuevos equipos, compartir datos almacenados y aumentar la velocidad de cálculo.

en cualquier lugar, en cualquier momento, en forma confiable y segura, que la ubicación de los procesos y datos sea transparente al usuario, y que el Grid sea fácil de usar, pero se va en camino.

Sin embargo, la infraestructura del Grid en desarrollo puede beneficiar muchas aplicaciones incluyendo aquellas con grandes necesidades computacionales (simulaciones, predicciones, monitoreo de procesos,...), con grandes necesidades de almacenamiento o procesamiento de datos (LHC, análisis distribuido de mamografías,...) y aplicaciones colaborativas (teleconferencias, reuniones virtuales,...). Es por ello que es considerada como el futuro de la computación que permitirá a las personas acceder a un enorme recurso computacional sólo con tener un navegador Web con conexión a Internet [24].

3.2 Computación Grid vs. computación distribuida

Sistemas de computación distribuida ya existen pero tienden a ser sistemas con un único propósito o empleados por un único grupo de usuarios. La computación Grid va más alla, ya que toma en cuenta: diferentes tipos de recursos, diferentes tipos de interacciones y su naturaleza es dinámica, ya que es posible agregar, remover y cambiar recursos y usuarios frecuentemente. Una definición de computación Grid, por Ian Foster, permite distinguirla de otras formas de computación. De

acuerdo a esta definición la computación Grid debe satisfacer tres criterios:

- **La coordinación de los recursos no debe estar sujeta a un control centralizado**, el Grid coordina e integra recursos y usuarios que viven bajo diferentes dominios de control, si ellos estuvieran bajo un control centralizado estaríamos lidiando con un sistema de manejo local como es el caso de los clusters.
- **Uso de protocolos e interfases de propósito general, abiertos y estándares**, para facilitar la interoperabilidad entre los componentes del Grid; en caso contrario el Grid sería sólo un sistema de aplicación específica como es el caso de SETI@home.
- **Alta calidad de servicio**, lo cual hace al Grid diferente de la computación punto a punto. La calidad de servicio se relaciona con el tiempo de respuesta, el rendimiento, la disponibilidad y la seguridad de los recursos para cumplir con las demandas de los usuarios.

Además, el Grid estará conformado por computadores en paralelo, clusters, entre otras y asignaría el trabajo del usuario al recurso más apropiado para resolverlo. En este sentido, la computación Grid es la forma más general de computación distribuida que pueda imaginarse [25].

3.3 Bases del funcionamiento de la computación Grid

Los aspectos más importantes para el funcionamiento de la computación Grid [26] [27] son los siguientes:

3.3.1 Compartir recursos

Una de las características del Grid es la de proporcionar un ambiente de contribución entre una comunidad amplia de usuarios. La computación Grid debe establecer mecanismos que permitan obtener beneficios al usuario que coloque a disposición sus recursos, ésta persona debe establecer qué recursos desea compartir, cuándo y para qué pueden ser utilizados. El Grid ofrecerá acceso remoto a recursos que en general no le pertenecen al usuario, los mismos deben ser virtualizados para proporcionar una interoperabilidad más uniforme entre participantes heterogéneos del Grid. Según los roles de los usuarios se asignarán prioridades a los recursos compartidos.

3.3.2 Seguridad de acceso

Este es el aspecto más crítico dentro de la computación Grid, ya que cuando se comparten recursos y se maneja información en forma remota debe existir un alto nivel de seguridad. Los componentes de seguridad básica son:

- **Política de acceso**: se encarga de definir en forma transparente y cuidadosa que recurso puede ser compartido y bajo que condiciones.

- **Autenticación**: este mecanismo informa sobre la identidad del usuario o recurso dentro del Grid.
- **Autorización**: se encarga de mapear un usuario o recurso con algún conjunto de privilegios.
- **Confidencialidad e integridad**: la confidencialidad implica la encriptación de la información de tal manera que solo el receptor pueda entenderlo, mientras que la integridad asegura que dicha información no haya sido alterada en la transmisión.

El Grid debe ser extremadamente flexible en el sentido de que toda la información correspondiente a los usuarios y recursos puede cambiar diariamente.

3.3.3 Uso eficiente de los recursos

A fin de reducir el tiempo de espera, se deben establecer mecanismos para distribuir los trabajos de los usuarios en forma eficiente y automática entre varios recursos. Desde el punto de vista del usuario, el manejo de recursos y la programación de las tareas debe ser transparente, el usuario debe poder conocer el estado de su trabajo desde el momento en que lo envía la Grid. La distribución óptima de los trabajos no funciona completamente hoy en día, pero muchos proyectos de Grid están desarrollando el *software* necesario para realizar esta tarea.

3.3.4 Uso de las tecnologías de redes

El Grid es posible por la disponibilidad y confiabilidad en el ancho de banda y debido a que la velocidad de las redes se duplica cada nueve meses, lo cual permite el uso de recursos distribuidos globalmente en una manera integrada. Estas redes se consideran el sistema nervioso que envía mensajes a distintas partes del Grid. Los servicios de red usados proveen al Grid de parámetros QoS tales como latencia, ancho de banda, confiabilidad y tolerancia.

3.3.5 Estándares abiertos

La estandarización asegura la interoperabilidad entre diferentes productos e implementaciones. Entre los cuerpos encargados de desarrollar estándares relevantes para distintos aspectos de la computación Grid tenemos: el *Foro Grid Global* (GGF [1], por sus siglas en inglés), *World Wide Web Consortium* (W3C [2]), y la *Organización para el Desarrollo de Estándares de Información Estructurados* (OASIS [3], por sus siglas en inglés).

3.3.6 Jerarquía Administrativa

Los recursos del Grid están distribuidos geográficamente bajo diferentes dominios administrativos y pertenecen a diferentes organizaciones, tal que cada ambiente Grid debe implementar jerarquías

[1] GGF http://www.gridforum.org/
[2] W3C http://www.w3c.es/
[3] OASIS http://www.oasis-open.org/home/index.php

administrativas para determinar como se distribuirá la información a través del Grid. El Grid ofrece el servicio de administración de prioridades entre diferentes organizaciones virtuales.

3.4 Arquitectura de la computación Grid

En general, en un ambiente Grid las capacidades de la infraestructura incluyen:

- Almacenamiento remoto y/o duplicado de datos
- Seguridad: autorización y políticas de autenticación
- Acceso uniforme a recursos distribuidos (datos y recursos computacionales)
- Manejo de cuentas y pagos, en caso de ser comercial o no ser institucional
- Manejo de recursos computacionales
- Garantizar el rendimiento
- Descubrimiento de recursos computacionales
- Envío y monitoreo de la ejecución de trabajos
- Alta velocidad de transferencia de datos
- Transferencia de datos y/o código entre la máquina del usuario y los recursos computacionales donde correrá el trabajo
- Manejo de fallas
- Reforzamiento de los requerimientos de calidad de servicios (QoS)

Todo esto hace de la computación Grid un sistema escalable, disponible todo el tiempo, confiable, fácil de usar, extenso, transparente, heterogéneo, dinámico y económico, en el sentido de que aprovecha y utiliza los recursos actuales de la organización o institución [21]. Los diferentes componentes del Grid que proveen estas capacidades son ordenados en capas, que definen mecanismos básicos que permiten a los usuarios gestionar los recursos compartidos.

3.4.1 Infraestructura

La capa de *Infraestructura* (*Fabric*) consiste de todos los recursos distribuidos globalmente que son accesibles desde cualquier sitio a través de la Internet, se encuentran los recursos computacionales que serán compartidos por las organizaciones virtuales como bases de datos, sistemas de almacenamiento, computadoras con distintos sistemas operativos, instrumentos científicos tales como telescopios o sensores; junto con la infraestructura de red y sus mecanismos de gestión y control. Un recurso puede ser una entidad lógica (como un sistema de archivos distribuido, o un cluster de computadoras) en ese caso la implementación del recurso requiere el uso de protocolos internos (e.g., protocolo de acceso y manejo NFS).

3.4.2 Conectividad

Dentro de la capa de *Conectividad* (*Connectivity*) se encuentran protocolos estándar de seguridad y comunicación para transacciones de red. Los **protocolos de comunicación** permiten el intercambio de datos entre la capa más inferior y los recursos mientras que los **protocolos de autorización y/o de autenticación** brindan mecanismos de criptografía para identificar usuarios y recursos. Dentro de los protocolos de capa de conectividad se sitúan algunos protocolos correspondientes al conjunto TCP/IP dado que la comunicación implica por ejemplo enrutamiento y transporte, se utilizan protocolos IP, ICMP, TCP - así como también el protocolo TLS (*Transport Layer Security*) que proporciona autenticación única, delegación e integración con soluciones de seguridad local; y certificados de identidad X.509 como protocolos estándar que brindan seguridad para acceder a los recursos definidos en la capa de Infraestructura.

3.4.3 Recurso

En el nivel correspondiente a la capa de *Recurso* (*Resource*) es donde se encuentran los protocolos que permiten obtener la información de un recurso en particular y gestionarlo controlando el acceso, arranque de procesos, gestión, monitoreo y auditoría. Se distinguen dos clases principales de protocolos: los **protocolos de información** que brindan información sobre el estado y la estructura del recurso y los **protocolos que permiten el manejo de los recursos**, utilizados para negociar el acceso al recurso compartido remotos en forma uniforme. En esta capa existen protocolos para el manejo de recursos de almacenamiento (e.g. SRM/DRM/HRM del Laboratorio de Lawrence Berkeley), servicios para filtrar o transformar los datos (e.g., *DataCutter* de la Universidad de Ohio), protocolos o servicios de acceso a los datos (e.g., Globus GridFTP), servicios para el manejo de bases de datos (e.g.,RDBMS local) y servicios para el manejo de recursos computacionales.

3.4.4 Recursos Colectivos

Al contrario de la capa de recurso, destinada a gestionar un recurso específico, la siguiente capa denominada *Recursos Colectivos* (*Collective services*), contiene los protocolos y servicios que permiten gestionar la interacción de un conjunto de recursos. Algunos ejemplos son los servicios de directorios, que permiten a las organizaciones virtuales descubrir y ubicar recursos compartidos (e.g., *Globus Replica Location Service* y *Globus Metadata Catalog Service*); *schedulers* o *brokers* distribuidos que permiten asignar tareas a cada recurso (e.g., Condor); servicios de monitoreo y diagnóstico de recursos ante fallas y servicios de replicación de datos. Esta capa se encarga además del descubrimiento de nuevos recursos y de aspectos de Calidad de Servicios (QoS, por sus siglas en inglés) como reservación de recursos e intercambio de información entre recursos.

3.4.5 Aplicación

La última capa definida es denominada *Aplicación* (*Application*). En esta se encuentran definidos los protocolos que permiten acceso a la estructura Grid. Incluye aplicaciones científicas, de in-

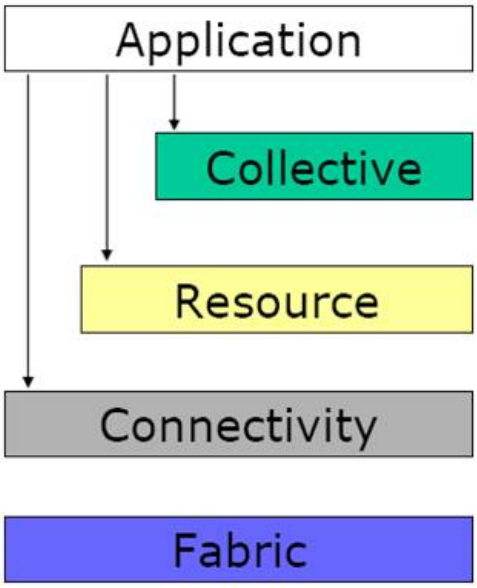


Figura 3.2: Arquitectura en capas del Grid [23].

geniería, financieras, entre otras, así como portales y *kits* de desarrollo para soportar las aplicaciones, los portales ofrecen aplicaciones de servicios Web donde los usuarios pueden enviar y recoger los resultados de sus trabajos a través de la Web. Esta es la capa del Grid con la cual es usuario interactúa.

La unión de las capas de Conectividad, Recurso y Recursos Colectivos se conoce como el *Middleware*, que puede compararse con el "cerebro" del Grid, ya que en general el *Middleware* es el *software* que organiza e integra los recursos en el Grid; su rol principal es automatizar todas las negociaciones máquina-máquina (M2M) requeridas para crear una infraestructura unificada única.

3.5 Servicios Web

Los servicios Web constituyen una tecnología de computación distribuida que permite la creación de aplicaciones cliente-servidor usando como plataforma de comunicación protocolos abiertos y altamente conocidos como HTTP o XML. De esta manera es posible ejecutar un servicio a través de la Internet, ocultándole al cliente detalles relativos a la implementación (servicios desarrollados en Java, C++, perl, etc) y plataforma (servidor Linux, Windows, Unix), lo único que le interesa la cliente es conocer la URL a través de la cual el servicio puede ser accesado. Los servicios Web son definidos por W3C (*World Wide Web Consortium*) empleando el WSDL (*Web Service Description Language* Lenguaje de Descripción para Servicios Web), que es un lenguaje basado en XML independiente de la plataforma. Básicamente un archivo WSDL de servicio Web define los métodos disponibles, parámetros, tipos de datos, protocolos de transporte, la sintáxis de

transferencia de datos y URI (*Uniform Resource Identifier* Identificador Uniforme de Recurso) de un servicio.

Para ubicar y usar un servicio Web W3C creó la especificación UDDI (*Universal Description, Discover and Integration* Descripción, Descubrimiento e Integración Universal). UDDI especifica el formato XML en el cual los datos son almacenados y un API para buscar datos existentes usando SOAP. Un protocolo estándar para comunicación también fue definido llamado SOAP (*Simple Object Access Protocol* Protocolo Simple de Acceso a Objeto), el cual permite intercambiar mensajes basados en XML a través de la Internet usando HTTP como protocolo de transporte; implementaciones de SOAP están disponibles para Java, Perl, Python y otros lenguajes. SOAP constituye la base para las tecnologías de servicios Web.

Los servicios Web es la tecnología ideal para aplicaciones distribuidas cuyos clientes y servidores están acoplados, tal como aplicaciones orientadas a Grid. Sin embargo, los servicios Web carecen de algunas características básicas que pueden causar problemas en aplicaciones orientadas a Grid: estado, transitoriedad, servicio de notificaciones, entre otros. Es por ello que nacen los llamados servicios Grid [28].

3.6 Servicios Grid

Un servicio Grid es definido como un servicio Web que proporciona un conjunto de interfases responsables del descubrimiento, creación, monitoreo y notificación de servicios dinámicos. Esta tecnología se beneficia de los formatos de mensajes estándar y de los mecanismos de comunicación como HTTP y XML para lograr la comunicación entre componentes y arquitecturas heterogéneas. La cualidad dominante de los servicios Grid es que sea *stateful*(es decir, tienen estado, guardan en memoria información), mientras que un servicio Web tradicional es *stateless* (es decir, no tienen estado y no guardan en memoria información durante invocaciones).

Ahora se está realizando un gran esfuerzo de estandarización, para definir comportamientos e interfases estándares para todos los servicios que podemos encontrar en una Grid: gestión de recursos gestión de trabajos, seguridad, cobro por uso de recursos, etc. Entre los esfuerzos realizados por el GGF por estandarizar la computación Grid se encuentra la *Open Grid Services Architecture* (OGSA), esta arquitectura fue inicialmente introducida por Foster et al.[29] y la *Open Grid Services Infrastructure* (OGSI). Es importante tener en cuenta que el único contacto entre los Servicios Grid y sus usuarios es la interfaz de servicios. Esas interfases de servicios son definidas por el Lenguaje de Descripción de Servicios Web (WSDL) existente. Varias mejoras a WSDL han sido identificadas para requerimientos de OGSI y actualmente están siendo agregadas al estándar WSDL.

3.6.1 OGSA

OGSA [30] inició en el 2003 y define una arquitectura estándar y abierta común para las aplicaciones basadas en la Grid. OGSA es una especificación que trata de estandarizar el acceso a los servicios presentes en una infraestructura Grid. Define un conjunto de interfases que deben cumplir los servicios Grid más comunes como: servicios de manejo de trabajos, de manejo de re-

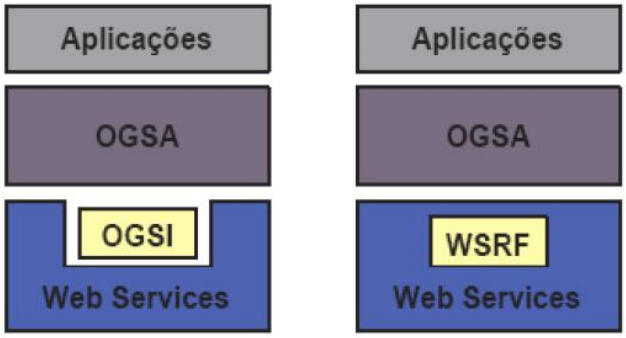


Figura 3.3: Estructura de la plataforma de servicios Grid en *Globus Toolkit 3.0* a la [4] izquierda y en *Globus Toolkit 4.0* a la derecha

cursos, de seguridad, de gerencia de OV, de información, de infraestrutura y de manejo de datos; las aplicaciones Grid están basadas en estos servicios definidos por OGSA. Brevemente OGSA es una arquitectura de computación e interacción distribuida basada en tecnologías de servicios Web (como WSDL y SOAP), que asegura la interoperabilidad entre sistemas heterogéneos a fin de que diferentes tipos de recursos puedan comunicarse y compartir información. OGSA ha sido adoptado como una arquitectura Grid por un gran número de proyectos Grid como la *Globus Alliance*.

3.6.2 OGSI

Creada en agosto del 2004, OGSI [31] se refiere a la infraestructura base sobre la cual se construye la OGSA. En su núcleo se encuentran las especificaciones de Servicios Grid, que definen la interfaz estándar y conductas de un Servicio Grid, armando una base de Web service [32]. OGSI define detalles tales como estabilidad de Servicios Web, la herencia de interfases de Servicios Web, notificación asíncrona, referencias a instancias de servicios, colección de instancias de servicios y datos de estados de servicios. Este estándar fue implementado por *Globus Toolkit 3.0* (ver Figura **??**)

3.6.3 Web Services Resource Framework (WSRF)

Uno de los objetivos de OGSI era conseguir la convergencia con los los servicios Web, este objetivo no se logró [33]. Para resolver los problemas de OGSI e iniciar un proceso de convergencia entre Grid y servicios Web, OASIS (*Organization for the Advancement of Structured Information Standards*) desarrollo WSRF para substituir a OGSI. WSRF provee un conjunto de operaciones que pueden implementar los servicios Web para convertirse en stateful, los clientes de servicios Web se comunican con servicios de recursos que permiten almacenar y retornar datos [32]. WSRF es una extensión que permite incorporar a los servicios Web las funcionalidades que aportaba un servicio Grid. Los mayores contribuyentes en este proyecto incluyen la *Globus Alliance* e IBM,

[4] *Globus Alliance* http://www.globus.org/

en la actualidad es implementado por *Globus Toolkit 4*, *WebSphere Application Server* versión 6.1, UNICORE versión 6.0, entre otros. WSRF es la base para WSDM (*Web Services Distributed Management*) que es un servicio Web estándar para el manejo y monitoreo del estado de otros servicios [34].

3.7 Principales Herramientas de computación Grid

3.7.1 Planificadores de tareas (Resource Brokers)

Un planificador es un sistema que es capaz de asignar recursos a las tareas que los usuarios desean ejecutar. Los planificadores más empleados funcionan en un cluster de computadores o en una red local; de muchos de ellos existen versiones Grid, capaces de planificar recursos situados incluso en distintos continentes.

Condor-G

Condor, originalmente, era un *software* de código abierto diseñado para utilizar ciclos de CPU de estaciones de trabajo ociosas, para aprovechar potencia de cálculo que, de otro modo, no se utilizaría; desarrollado por el Proyecto de Investigación Condor de la Universidad de Wisconsin-Madison (UW-Madison), a partir de 1988. Sin embargo, hoy en día es un planificador de tareas por lotes dotado con un mecanismo de colas de trabajos, políticas de planificación, esquemas de prioridades, monitoreo y gestión de recursos. Condor cuenta con un mecanismo flexible para los trabajos y recursos, en el cual los trabajos pueden indicar requisitos para su ejecución y también los recursos pueden especificar requisitos sobre los trabajos que se van a ejecutar. Condor-G es la versión Grid de Condor, el cual se encarga solo de la gestión de trabajos (colocar los trabajos en cola, verificar el estado del trabajo y manejar los archivos de entrada y salida) y emplea *Globus Toolkit* para empezar a correr el trabajo en recursos distantes y la gestión de recursos remotos en general [35].

Portable Batch System (PBS)

Es una herramienta de *software* que se creó con el objetivo fundamental de solucionar el manejo de carga de trabajo para sistemas de computación de alto rendimiento y para manejar clusters de computadoras Linux y Unix. PBS fue diseñado originalmente por la NASA, a mediados de los noventa, porque requería un *software* para el manejo de recursos.

PBS permite definir e implementar una política para la ejecución de los trabajos, como definir que tipos de recursos y cuánto de cada recurso puede ser usado por los trabajos. Además, proporciona un mecanismo que el usuario puede usar para asegurar que un trabajo tenga acceso a los recursos requeridos para ejecutarse. PBS está formado por varios componentes como el servidor y el cliente, que funcionan estableciendo una interacción entre ellos. El cliente realiza una solicitud a un servidor y el servidor se encarga de realizar el trabajo del cliente. El servidor de PBS proporciona servicios de crear, ejecutar, modificar o eliminar trabajos del cliente, dependiendo de su solicitud.

Sun Grid Engine (SGE)

Es un proyecto desarrollado por la comunidad de código abierto. El principal objetivo de este proyecto es desarrollar y proporcionar una herramienta de *software* de código abierto, que facilite la adopción de soluciones de computación distribuida y promueva el desarrollo de estándares abiertos para el manejo de recursos distribuidos. Es patrocinado por la compañía Sun Microsystems. *Sun Grid Engine* es una herramienta de *software* para el manejo de recursos distribuidos en ambientes de red heterogéneos que se encarga de agregar poder de cómputo y lo entrega como un servicio de red. Provee las funciones de un manejador de recursos distribuidos como mecanismos para despachar trabajos, balanceo de cargas, estadísticas de trabajos despachados, manejo dinámico de los recursos, protocolos de seguridad, administración de los recursos y monitoreo de los procesos [36].

3.7.2 *Middleware*

En un entorno de computación distribuida, el *middleware* se define como la capa de *software* que se encuentra entre el sistema operativo y las aplicaciones en cada *host* que participa en el sistema. En la actualidad, el Grid *middleware* ha evolucionado hacia los llamados servicios Grid (es decir, está compuesto por servicios Grid), a su vez basados en la tecnología de servicios Web.

Globus Toolkit

Este *software* de arquitectura abierta es un conjunto de servicios y librerías de código abierto que soportan Grids y sus aplicaciones y permite direccionar asuntos de seguridad, hallazgos de información, de gestión de recursos y datos, comunicación, fallas y transportabilidad. Globus [37] es el *toolkit* estándar de facto para la construcción de Grids. Es un proyecto desarrollado por la *Globus Alliance* constituida por el Laboratorio Nacional Argonne, el Instituto de Ciencias de la Información de la Universidad del Sureste de California, la Universidad de Chicago, la Universidad de Edimburgo, el Centro Sueco de Computación Paralela y el Centro Nacional para Aplicaciones en Supercomputación (*National Center for Supercomputing Applications* (NCSA)), además existen otras universidades, empresas e instituciones afiliados a la Alianza que también contribuyen en este proyecto.

Globus Toolkit incluye una serie de componentes que proporcionan servicios básicos, como seguridad, información y gestión de los recursos, monitoreo, descubrimiento, manejo de datos y comunicaciones. El *Globus Toolkit* está diseñado especialmente para ser usado por arquitecturas Grid, integración de sistemas y diseños de aplicación Grid, para reducir la cantidad de trabajo requerida para lograr los objetivos y asegurar un nivel básico de interoperabilidad entre sistemas y aplicaciones Grid. Globus se concibe como un conjunto de servicios, que se pueden utilizar de forma independiente, si fuera necesario:

- *Globus Resource Allocation Manager* (GRAM) - Manejador de Asignación de Recursos: Es el componente que proporciona servicios de gestión de recursos y de creación, monitoreo y gestión de procesos. GRAM convierte las peticiones, expresadas en un lenguaje llamado

RSL (*Resource Specification Language*), en órdenes para los planificadores y computadores locales.

- *Grid Security Infrastructure* (GSI) - Infraestructura de Seguridad Grid: Es un servicio de autenticación y cifrado, que permite a la vez la gestión local de los permisos de acceso de los usuarios y la autenticación única.

- *Monitoring and Discovery Service* (MDS) - Sistema de Descubrimiento y Monitoreo: Es un servicio de información del Grid , que proporciona los datos utilizando un directorio LDAP (*Lightweight Directory Access Protocol*). MDS proporciona un mecanismo uniforme para acceder a información sobre la configuración de los servidores, la red, los recursos disponibles y ocupados, etc.

- *Grid File Transfer Protocol* (GridFTP) - Protocolo de Transferencia de Archivo Grid: Es un protocolo de transferencia de datos confiable, seguro y de alto rendimiento, desarrollado para ser utilizado en ambientes de computación con un gran ancho de banda en redes de área amplia. Brinda acceso parcial a archivos, notificación del estado actual de la transferencia, transferencia de terceros, soporte para trasferencia de archivos grandes y paralelas y control del tamaño del buffer TCP (Protocolo de Control de Transmisiones).

- *Global Access to Secondary Storage* (GASS) - Acceso Global a Almacenamiento Secundario: Este componente implementa un conjunto de estrategias de movimiento de datos, que permiten que los programas que se ejecutan en lugares remotos puedan acceder a datos locales.

- Nexus y Globus-IO: Proporcionan servicios de comunicaciones para entornos heterogéneos.

- *Heartbeat Monitor* (HBM): Es un componente que permite a los administradores del sistema y/o los usuarios detectar los fallos de componentes del sistema o procesos en ejecución.

Existe un API (*Application Programming Interface*) desarrollado en C o Java publicado para cada componente, lo que permite realizar aplicaciones que utilicen estos servicios. También se proporcionan herramientas de línea de comandos para uso por parte de los usuarios. Gracias a todo esto, una gran cantidad de personas y entidades han desarrollado servicios y aplicaciones que utilizan Globus, como MPICH-G2 o Condor-G. Aunque *Globus Toolkit* es una herramienta de *software* de amplios servicios y funcionalidad para ambientes de computación Grid, de código abierto, portable, se integra con sistemas por lotes o sistemas de manejo de colas de trabajos existentes, por ejemplo *Sun Grid Engine*, PBS, LSF; se integra con servicios de bajo nivel como MPI, la seguridad está presente en todos sus componentes y posee soporte para un gran número de tecnologías estándares; también tiene sus desventajas, ya que requiere realizar algunas configuraciones difíciles a sus componentes, se requiere un cierto grado de conocimientos en administración, instalación y configuración de sistemas de computación para poder manejarlo y no posee un contenedor Web propio lo que implica la utilización de una herramienta externa como contenedor Web, en general emplea Tomcat por defecto.

GT ha evolucionado bastante desde V2.x. GT2 incluye un conjunto de *scripts* para ejecutar tareas como GridFTP basada en la transferencia de archivos. GT3 es una puesta en práctica de OGSA. Aunque GT3 trae una nueva arquitectura e intenta utilizar las ventajas de los estándares industriales de los servicios Web, no ha tenido mucho éxito debido a su complejidad. Sin embargo, algunos países han desarrollado proyectos basados en GT3. La *Globus Alliance* pronto respondió a las ediciones planteadas por GT3 y lanzó su última versión, GT4, la cual incluye componentes y herramientas de desarrollo de *software* para construir sistemas bajo el estándar OGSA y además cumplen con el WSRF.

gLite

gLite [5] (*Lightweight Middleware for Grid Computing*) es un sistema complejo, compuesto por varios paquetes instalados en diferentes máquinas, interactuando entre ellos y cada uno de ellos juega un rol diferente dentro de las actividades del Grid. gLite puede ser configurado de muchas maneras, debido a su modularidad y escalabilidad y depende finalmente de *Local Bash System* (también llamados Local *Resource Management System* (LRMS)) tales como PBS/TORQUE-MAUI, LSF y/o Condor. El *Globus Toolkit* (GT) está embebido en gLite, en el caso de gLite 3.0.1 contiene el GT2 [38].

gLite nace como parte del proyecto EGEE (*Enabling Grids for E-sciencE*). gLite facilita la interoperabilidad entre servicios Grid en conformidad con los nuevos estándares. La estructura de sevicios de gLite ha sido influenciada enormemente por los requerimientos del proyecto LCG. Los componentes de la estructura de gLite son los siguientes:

- ***Grid Security Infrastructure*-Infraestructura de Seguridad del Grid (GSI)**: Permite la autenticación y comunicación segura a través del Grid. La autenticación se basa en el uso de certificados X.509. A fin de reducir la vulnerabilidad la identificación del usuario es hecha usando *proxies* de los certificados que identifican al individuo (*Certificate Authorities*- Certificados de Autoridad (CA)).

- ***User Interface*-Interfaz de usuario (UI)**: Puede ser cualquier máquina donde el usuario tiene una cuenta personal y su certificado de usurio este instalado. Permite al usuario acceder a la Grid, desde este equipo el usuario se autentica y envía los trabajos al Grid, bien sea utilizando líneas de comando o a través de los portales.

- ***Computing Element*-Elemento de Cómputo (CE)**: Es un conjunto de recursos computacionales ubicados en un determinado lugar que se encarga de manejar los nodos de cálculo, tiene instalado un planificador y un manejador de colas (PBS ó Condor) el cual le permite aceptar y enviar esto trabajos a ejecutarse en los nodos de trabajo o *worker nodes*(WN).

- ***Worker Node*-Nodo de trabajo (WN)**: Uno o más nodos encargados de ejecutar o realizar los cálculos de los trabajos enviados al Grid. Están conectados al CE a través del

[5] gLite http://www.eu-egee.org/glite

manejador de colas (*Local Resource Management System* (LRMS) también conocido como *batch system*), al cual los trabajos son enviados.

- ***Storage Element*-Elemento de Almacenamiento (SE)**: Provee acceso uniforme a recursos de almacenamiento de datos. Emplea los protocolos GSIFTP y/o GridFTP para transferencia completa de archivos, y RFIO (*Remote File Input/Output Protocolo*) o gsidcap (*GSI dCache Access Protocol*) para el acceso a archivos locales o remotos. GSIFTP es un protocolo que ofrece las funcionalidades del FTP pero con soporte para el GSI.

- ***LCG File Catalog*-Catálogo de Archivos desarrollado por el equipo del LHC(LFC)**: Es un catálogo que permite almacenar la información acerca de la data almacenada en los diferentes SEs, permite también hacer replicas de los datos.

- ***Workload Manager System*-Sistema de Manejo de Cargas (WMS)**: Es un conjunto de componentes responsables de cotejar, aceptar los trabajos de los usuarios, asignarles el CE apropiado, conocer su estado en cada momento y visualizar las salidas de los trabajos enviados al Grid. El *Resource Broker-Manejador de Recursos* (RB) es la máquina donde se ejecutan los servicios del WMS. El *Logging and bookkeeping service*-Servicio de rastreo (LB) sigue el rastro de los eventos que le suceden a los trabajos manejados por el WMS, a fin de grabar el estado del trabajo.

- ***Information Service*-Servicio de Información (IS)**: se encargan de mantener información disponible al usuario sobre los recursos disponibles y el estado del sistema. Para ello emplea el *Berkeley Database Information Index-Catálogo de Base de Datos de Berkeley* (BDII), el cual se encarga de recolectar la información que producen los recursos y responder a los usuarios, acerca del estado del Grid.

- ***Proxy Server* (PX)**: Para trabajos largos, se usa un Sistema de Renovación de Proxy que consiste de: *Proxy Renewal Service* (PRS) sobre el WMS y LB y *Proxy Server* (PS) sobre una maquina dedicada.

- **VOMS:** Autenticación basada en una base de datos (BD) central, una por VO. Estas BD son accedidas por: WMS, LB, CE y SE para construir localmente una lista de usuarios autorizados. En el caso del Grid de la ULA este posee una organización virtual que provee servicio a los usuarios locales.

En resumen, el término computación Grid define un nuevo campo para las ciencias computacionales que difiere de la computación distribuida tradicional al focalizarse en compartir grandes recursos computacionales heterogéneos, aplicaciones innovadoras y computación de alto rendimiento.

Importantes aplicaciones del área científica y en particular de astrofísica (*International Virtual Observatory Alliance* [6]) y relatividad general numérica (CACTUS [7]) comienzan a migrar a este

[6] IVOA http://www.ivoa.net/
[7] CACTUS http://www.cactuscode.org

esquema, dando lugar a la e-ciencia. Es importante mencionar dos esfuerzos continentales para desarrollar ambientes y facilidades de cálculo científico distribuido: *TeraGrid* [8] en los Estados Unidos y *Enabling Grids for E-sciencE* (EGEE) de la Unión Europea [9] y más recientemente *E-infrastructure shared between Latin America and Europe* [10]. El impacto de las tecnologías Grids se ha percibido principalmente en diferentes áreas como el almacenamiento de datos, la visualización científica y la colaboración a distancia.

[8] TeraGrid http://www.teragrid.org/
[9] EGEE http://public.eu-egee.org/
[10] EELA http://www.eu-eela.org/

Capítulo 4

Portales y portlets

Los códigos científicos son desarrollados por los usuarios para resolver determinado tipo de problemas. Algunos de ellos evolucionan y se convierten en aplicaciones utilizadas por extendidas comunidades académicas. Unas posteriormente dan origen a esfuerzos comerciales que garantizan su desarrollo y evolución, y otras se convierten en proyectos de código abierto, colectivos y voluntarios. Por lo general su utilización es complicada y requieren un importante esfuerzo de aprendizaje. Desde hace unos años, la tendencia de las comunidades de investigadores es a minimizar ese esfuerzo a través de la implementación de "portales" los cuales permiten el acceso de los usuarios a través de interfases Web.

4.1 Portal Grid

Para facilitar la adopción y utilización de Grids por parte de las distintas comunidades de usuarios, es necesario presentar el Grid de una forma visual, atractiva e intuitiva. Una solución común a esta necesidad consiste en la elaboración de portales Web, a los que se puede acceder desde cualquier computador dotado de un navegador. Este enfoque evita que los usuarios tengan que instalar y administrar aplicaciones en sus estaciones de trabajo, y permite que puedan acceder al Grid desde cualquier lugar. Un portal es una aplicación Web que provee los siguientes servicios: personalización, autenticación única y agregación de contenido desde diferentes fuentes además de albergar la capa de presentación de los sistemas de información. Un portal puede tener características de personalización sofisticadas para proveer contenidos adaptados a diferentes tipos de usuarios.

Un portal es una entrada a un conjunto de servicios de red distribuidos que pueden ser accedidos desde un navegador. Y provee una interfaz común para estos servicios de tal manera que sus usuarios se sientan en un mismo ambiente cuando realmente están accediendo a diferentes tipos de servicios distribuidos. Los portales están constituidos por *portlets*. Los siguientes portales

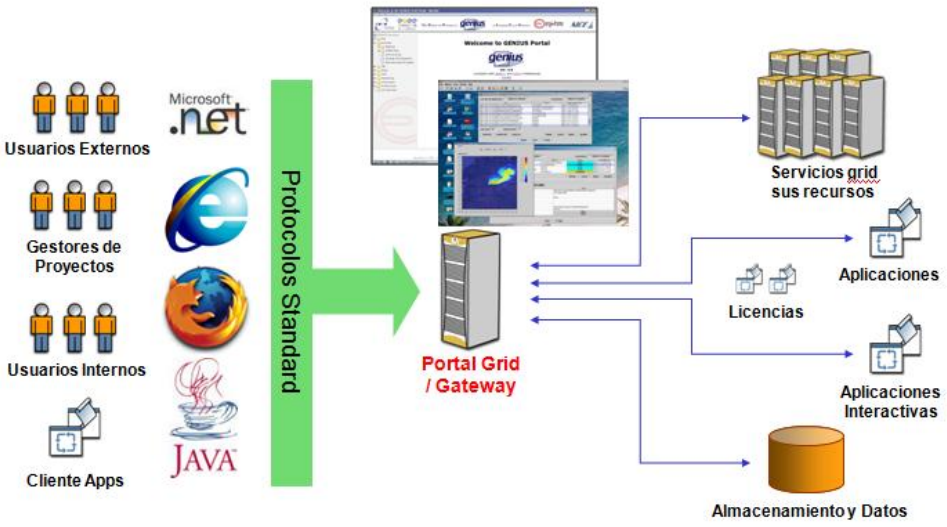


Figura 4.1: Un portal es un ambiente Web seguro a través del cual una OV puede compartir contenido, accesar a los servicios Grid y a aplicaciones.

Grid : OGCE Release 2, GridPort V4, NGS Portal release 2, contituyen los más importantes y grandes proyectos en la actualidad [39] [40].

4.1.1 Conceptos claves

Portlet

Un *portlet* es un componente Web hecho en Java, portátil, basado en estándares y manejado a través de un contenedor de *portlets* que procesa las peticiones de los clientes y produce contenido dinámico. El contenido generado por un *portlet* es llamado fragmento, una pieza de código (HTML, XHTML, WML) adherida a ciertas reglas. Un fragmento puede ser agregado a otros fragmentos a fin de formar un documento completo, por ejemplo un portal es un conjunto de fragmentos generados por diversos *portlets*.

El contenido generado por un *portlet* puede variar de un usuario a otro dependiendo de cómo haya configurado el usuario el *portlet*. A diferencia de los *servlets*, los *portlets* no tienen interacción directa con los clientes Web. En su lugar, los clientes Web interactúan con el portal a través de un mecanismo de solicitud/entrega aplicado por un contenedor de *portlet* el cual también maneja el ciclo de vida de los *portlets*.

Los *portlets* son similares a los *servlets* en que:

- Los *portlets* son manejados por un contenedor especializado
- Los *portlets* generan contenido dinámicamente
- El ciclo de vida de los *portlets* es controlado por el contenedor

- Los *portlets* interactúan con el cliente Web mediante el uso del paradigma *request/response*

Los *portlets* son diferentes a los *servlets* en que:

- Los *portlets* son únicamente generados como fragmento de etiquetado y no como documentos completos
- Los *portlets* son accesibles directamente a una URL
- Los *portlets* no pueden generar contenido arbitrario, ya que el contenido de los *portlets* va a estar incluido la página del portal
- Si un servidor de un portal esta solicitando html/text, entonces todos los *portlets* deben ser generados en text/html. Por otro lado si el servidor del portal esta solicitando por WML, entonces cada portal deberá ser generado en contenido WML.

Generalmente, los *portlets* tienen una clara separación entre el contenido y la presentación la cual es manejada por una o más clases de Java que contienen la aplicación lógica. Los portales usan a los *portlets* como componentes modulares para interfaz de usuario. Los *portlets* para aplicaciones científicas surgen como respuesta a la necesidad de simplificar el uso de aplicaciones haciendo uso de interfases intuitivas además de aprovechar la seguridad y el beneficio de la tecnología de Grids.

Contenedor Web

Un contenedor Web es un servidor que ejecuta *servlets* y permite el acceso a dichos *servlets* vía HTTP. El contenedor Web más usado en la actualidad es Apache Tomcat (versión actual 6.0.16). Otros contenedores Web incluyen Borland Enterprise Server, JBOSS, IBM Websphere, Resin, BEA WebLogic, Apple WebObjects, etc.

Contenedor de *portlets*

Un contenedor de *portlets* es un *software* corriendo como un App Java dentro de un contenedor Web como Apache Tomcat, el cual les provee con el ambiente de ejecución adecuado, albergándolos y gestionando sus ciclos de vida. También provee mecanismos de almacenamiento persistentes para las preferencias de los *portlets*. Un contenedor de *portlets* recibe peticiones del portal para ejecutarlas en los *portlets* albergados por él, pero no es responsable por agregar el contenido producido por los *portlets*, ya que esto es tarea del portal. Sin embargo, los *portlets* tienen una manera muy limitada de interactuar con el contenedor, el Portlet API es prácticamente unidireccional. El contenedor de *portlets* puede operar como parte de o en forma independiente de un *marco de portal* (*portal framework*, también conocido como *servidor del portal*), el cual es responsable por los servicios de portales comunes.

Un *framework* típico de un portal generalmente proporciona funcionalidades como administración de cuentas de usuario y el despliegue de los *portlets* . Por lo tanto, la carga en los desarrolladores de portales se disminuye, y los desarrolladores pueden centrarse en el desarrollo

del *portlet*, particularmente en la capa de la lógica. Ejemplos de contenedores de *portlets* de código abierto empleados comúnmente son: GridSphere, Apache Pluto, jPortlet, JBOSS Portal, Jetspeed2, Liferay, Sun Java System Portal Server 7.0 etc. Contenedores de *portlets* comerciales incluyen: Oracle AS Java Portlet Container, IBM WebSphere, BEA WebLogic, etc.

Service Oriented Architecture (SOA)

SOA [1] es una arquitectura que permite la creación de aplicaciones construidas combinando acoplamiento e interoperabilidad de servicios; SOA es independiente de la tecnología de desarrollo (como Java, .NET, etc.) y por ende del distribuidor. Según se muestra en la Figura 4.2 SOA se expresa como una arquitectura de la tres niveles en la cual las tres capas: portal, servicio, y recursos, son iguales que la arquitectura de GridPort V4. Aquí la capa del servicio se ha ampliado de modo que la lógica de la presentación pueda ser incluida. Además, la capa de portal no es más que un contenedor de clientes del servicio, pero podría sí mismo ser abastecedor de servicio. Esto destaca a los servicios orientados a la presentación propuestos en WSRP (*Web Service Remote Portlet*, el cual se explicara mas adelante) y hace que la capa del portal sea capaz de proporcionar componentes Web como servicios.

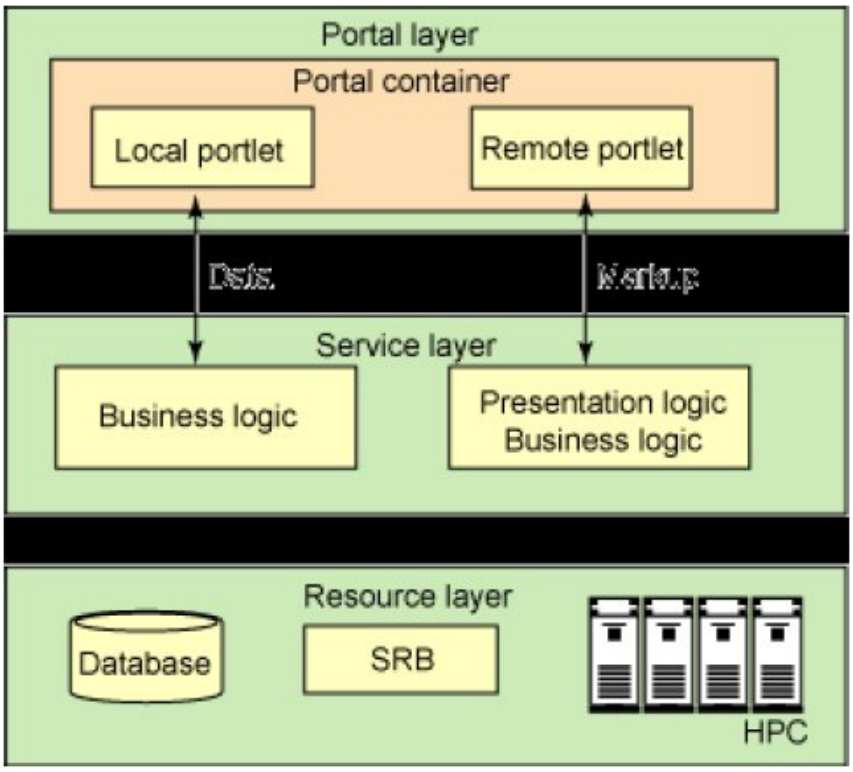


Figura 4.2: SOA para portales Grid [41].

[1] Guía SOA http://dev2dev.bea.com/pub/a/2006/09/soa-practitioners-guide.html

Los portales Grid se dividen, generalmente, en portales orientados al usuario y portales orientados a una aplicación. Los primeros son portales genéricos, diseñados para que los usuarios puedan utilizar los recursos del Grid . Los portales del segundo tipo son portales diseñados para que los usuarios utilicen una determinada aplicación.

4.1.2 Estándares portlets

JSR 168

La especificación de *portlet* de Java (originalmente creada a través del JSR-168) provee un estándar para el desarrollo de componentes de portal con el lenguaje de programación Java. Esta especificación fue desarrollada bajo la *Java Community Process*[2] como *Java Specification Request* (*JSR*) 168.

La meta principal del JSR 168, es habilitar la interoperabilidad entre *portlets* y portales. Esta especificación define el contrato entre el *portlet* y el contenedor de *portlets*, y coloca un conjunto de APIs de *portlets* que se encargan del manejo del ciclo de vida del *portlet*, de la personalización (detección y establecimiento de estados de ventanas y modos del *portlet*), presentación (acceso a preferencias de usuario) y seguridad (acceso del *portlet* a la autenticación del usuario y a información de la sesión). La especificación también define como empaquetar *portlets* en aplicaciones de *portlets*. Además esta especificación hace posible que desarrolladores de *portlets* intercambien componentes Web. La Figura 4.3 muestra la Arquitectura de Referencia de un Portal JSR 168, presentada por Alejandro Abnelnur [42].

La industria de TI ha aceptado ampliamente al JSR 168. Todas las grandes compañías en el espacio de portales son parte del grupo de expertos del JSR 168: Apache, ATG, BEA, Boeing, Borland, Broadvision, Citrix, EDS, Fujitsu, Hitachi, IBM, Novell, Oracle, SAP, SAS Institute, Sun Microsystems, Sybase, TIBCO, y Vignette. La lista de patrocinantes es mucho mayor.

WSRP

A fin de ejecutar *portlets* JSR 168 como *portlets* remotos se propuso la especificación del OASIS WSRP V1.0 . WSRP es un estándar para agregación de contenido, para acceder y mostrar contenido como *portlets* que están en un servidor remoto. En la Figura 4.4 se muestra la Arquitectura de Referencia de un Portal WSRP que soporta tanto *portlets* remotos como *portlets* locales. *Portlets* locales corren sobre el servidor del portal (*framework*) e interactúan con él a través del Portal API (JSR 168 para *frameworks* basados en J2EE). *Portlets* remotos corren sobre otro servidor e interactúan con el servidor del portal a través de los *Generic Portlet Proxies* y WSRP. *Generic Portlet Proxies* permiten al servidor del portal interactuar con los *portlets* remotos de la misma manera en que interactúa con sus *portlets* locales, a través del Portal API. Mientras que WSRP provee el protocolo para que el servidor del portal interactúe con el servidor remoto

WSRP separa *portlets* de portales, este introduce los conceptos del productor y del consumidor. Un productor son los portal frameworks o aplicaciones de servicios Web que proveen

[2] JCP http://www.jcp.org/en/jsr/detail?id=168

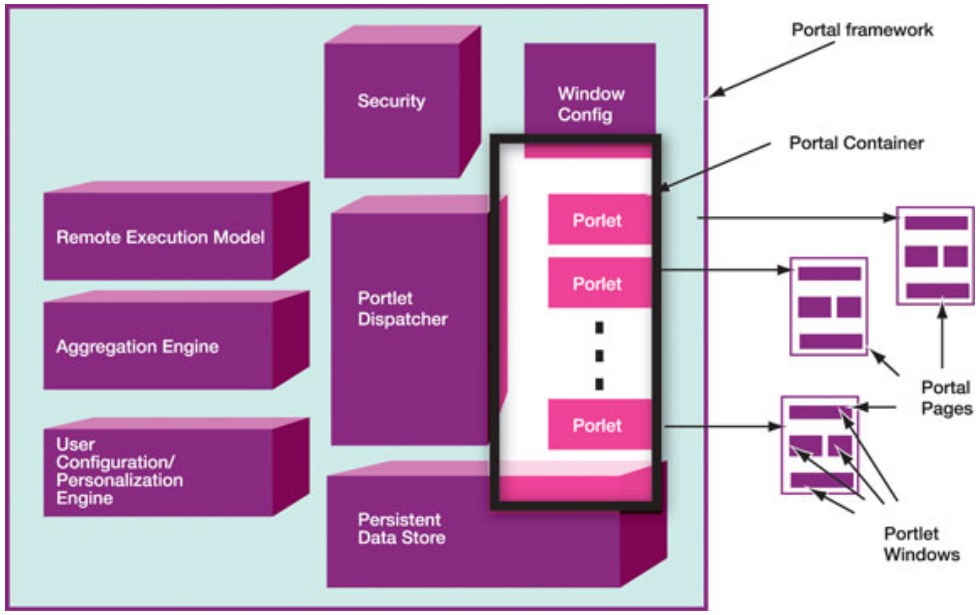


Figura 4.3: En esta arquitectura el *portlet* genera fragmentos que el contenedor de *portlets* envía al *portal framework*. El *portal framework* agrega un marco, un título y botones de control para crear la ventana del portal, la cual se convierte en parte de la página del portal [42].

servicios de *portlets* y un consumidor son las aplicaciones y portales que "consumen" estos servicios. WSRP define un *portlet* como un servicio Web que genera fragmentos y permite a los consumidores interactuar con ellos. Desafortunadamente, la puesta en práctica actual de WSRP para código abierto sigue siendo inmadura.

JSR 286 y WSRP V2.0, sucesores de JSR 168 y WSRP V1.0, están en forma de borrador aún y deben proporcionar mejoras en la comunicación entre *portlets*, y soportar tecnologías de Web 2.0.

4.1.3 Ciclo de vida de un *portlet*

Como se expuso anteriormente, es la función del contenedor de *portlets* manejar el ciclo de vida de un *portlet*. Cada *portlet* experimenta cuatro métodos en su ciclo de vida [39] [44]:

init(PortletConfig config)

Es llamado una vez, inmediatamente después una nueva instancia del *portlet* es creada. Puede ser usada para ejecutar tareas de arranque y es similar al método init de un *servlet*. *PortletConfig* representa datos de configuración de sólo lectura, especificados en el archivo descriptor del *portlet*, portlet.xml. Por ejemplo, *PortletConfig* provee acceso a los parámetros de inicialización.

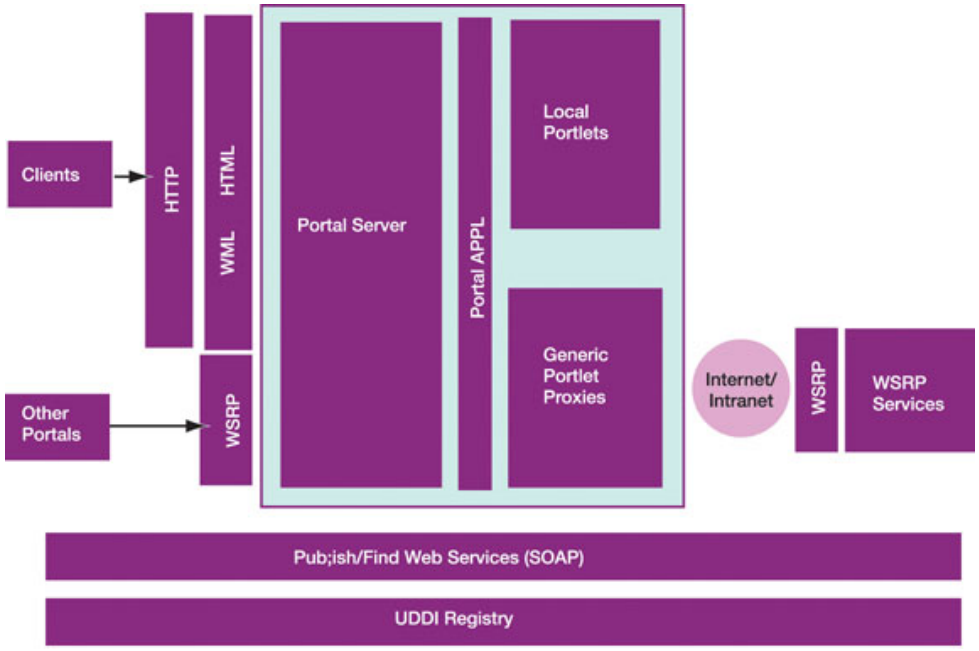


Figura 4.4: WSRP permite que los *portlets* locales pertenecientes al portal framework están disponibles para otros portales [43].

processAction(ActionRequest request, ActionResponse response)

Es llamado en respuesta a la acción de un usuario como hacer click en un enlace o enviar una planilla, es decir, cuando el usuario envía cambios a un *portlet*. En este método, un *portlet* puede invocar componentes lógicos como JavaBeans para lograr este objetivo. Las interfases *ActionRequest* y *ActionResponse* son subinterfases de *PortletRequest* y *PortletResponse*. La interfase *Action Request* representa el envío de la solicitud del usuario al portal para que el mismo maneje la acción, mientras que la interfase *ActionResponse* representa la respuesta de un *portlet* a una acción de un requerimiento del usuario. El contenedor de *portlet* crea un objeto *ActionResponse* y otro *ActionRequest* y los pasa como argumento a este método. En processAction, un *portlet* puede modificar su propio estado así como su información persistente.

render(RenderRequest request, RenderResponse response)

Sigue al processAction en la cadena de métodos del ciclo de vida, es decir, es llamado cuando el *portlet* es redibujado por el desktop. Render genera el etiquetado que será accesible al usuario del portal. Los métodos *RenderRequest* y *RenderResponse*, también son subinterfases de *PortletRequest* y *PortletResponse*, y están disponibles durante la visualización del *portlet*. La forma en la cual el método render genera la salida puede depender del estado actual del *portlet*.

destroy()

Es el último en el ciclo de vida, llamado justo antes que la basura del *portlet* sea recogida y provee una última oportunidad de liberar los recursos del *portlet*.

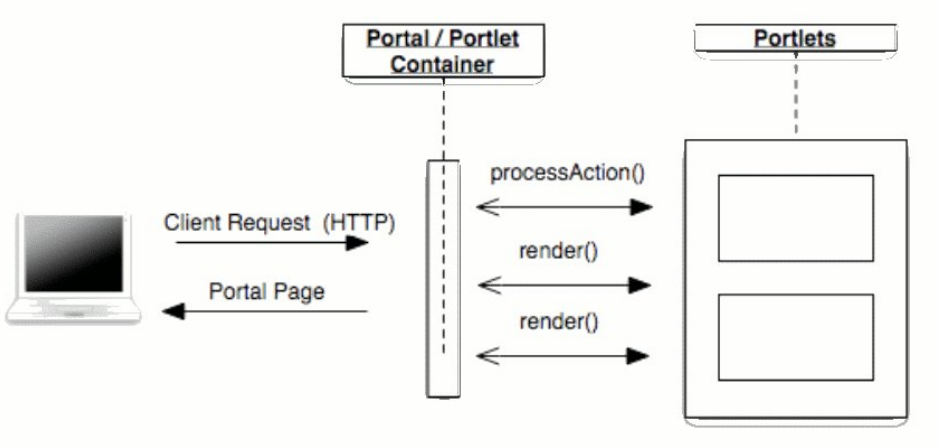


Figura 4.5: Flujo de datos dentro de un *portlet* [44].

4.1.4 Características de los *portlets*

Modos del *portlet*

Los *portlets* desempeñan diferentes tareas y crean contenidos de acuerdo a su función actual. Un modo de *portlet* indica la función que un *portlet* está desempeñando en cierto momento. Un modo de *portlet* especifica el tipo de tarea que el *portlet* debería desempeñar y que contenido debería generar. Cuando se invoca a un *portlet*, el contenedor de *portlets* provee el modo para el actual requerimiento al *portlet*. Los *portlets* pueden programáticamente cambiar su modo mientras procesan una petición de acción [45].

JSR 168 define 3 categorías de modos de *portlet*:

- ***Edit*** Muestra una o más vistas que permiten al usuario personalizar los parámetros del *portlet*.
- ***Help*** Muestra pantallas de ayuda.
- ***View*** Muestra la salida del *portlet*.

Además de los métodos mencionados en la sección anterior los cuales son llamados directamente por el contenedor, la clase *GenericPortlet* puede implementar el método render() y delegar el llamado a métodos más específicos para mostrar el *portlet* basado en su modo. Estos métodos son:

- ***doEdit()*** Llamado por render() cuando el *portlet* esta en modo *Edit*. Contiene la lógica que visualiza la página *Edit* para el *portlet*.
- ***doHelp()*** Llamado por render() cuando el *portlet* esta en modo *Help*. Contiene la lógica que visualiza la página *Help* para el *portlet*.
- ***doView()*** Llamado por render() cuando el *portlet* esta en modo *View*. Contiene la lógica que visualiza la página *View* para el *portlet*.

Estado de ventana

Un estado de ventana es un indicador de la cantidad del espacio de portal asignado al contenido generado por un *portlet*. El contenedor de *portlets* provee el estado de ventana inicial al *portlet*, y el *portlet* usa este estado de ventana para decidir cuanta información debería mostrar. No obstante, Los *portlets* pueden programáticamente cambiar su estado de ventana mientras procesan una petición de acción.

JSR 168 define los siguientes estados de ventana:

- ***Normal*** El *portlet* comparte el espacio con otros *portlets* y debería tomar esto en cuenta cuando produzca su salida.
- ***Maximized*** Una ventana tiene mayor espacio para colocar su salida más que en su estado de ventana normal.
- ***Minimized*** El *portlet* debería producir una salida mínima o nula.

Aparte de estos estados de ventana, JSR 168 permite al portal definir estados de ventana personalizados.

Modelo de datos

JSR 168 define diferentes mecanismos para que el *portlet* acceda a datos transitorios y persistentes. El *portlet* puede colocar y obtener datos transitorios en los siguientes escenarios:

- ***Request*** La petición tiene datos incluidos, como los parámetros y atributos de la petición, similar a la petición del *servlet*. La petición puede contener propiedades para permitir que la extensión y los encabezados del cliente sean transportados del portal al *portlet* y viceversa.
- ***Session*** El *portlet* puede guardar datos en la sesión con alcance global, para dejar que otros componentes de la aplicación Web tengan acceso a los datos, o en el alcance del *portlet*, el cual es de acceso restringido al *portlet*.
- ***Context*** El *portlet* puede guardar datos en el contexto de la aplicación Web, así como lo hacen los *servlets*.

El *portlet* puede acceder a datos persistentes con estos alcances:

- ***Por portlet*** El *portlet* puede guardar datos de configuración y personalización en las preferencias del *portlet* para habilitar al *portlet* crear salidas personalizadas. El *portlet* puede definir que datos el usuario puede cambiar en el modo de edición (por ejemplo, cuenta de correo), y que datos son parámetros de configuración que solo pueden ser cambiados por un administrador en el modo de configuración (por ejemplo, el servidor de correos).
- ***Por usuario*** La información del perfil del usuario puede ser leída por el *portlet* para confeccionar su salida en función al usuario (por ejemplo, mostrar el clima de la ciudad donde el usuario vive).

Todos los recursos, *portlets*, descriptores de despliegue son empaquetados juntos en un archivo de aplicación Web (WAR) que se conoce como *portlet application*. Existen 2 descriptores de despliegue:

- Todos los recursos de aplicación que no son *portlets* deben ser especificados en el descriptor de despliegue *web.xml*.
- Todos los *portlets* y las configuraciones de *portlets* deben ser especificados en el descriptor de despliegue *portlet.xml*.

4.2 Herramientas para Portales Grid

Actualmente existen varias herramientas para el desarrollo de portales que se integran con sus implementaciones Grid. Entre ellas podemos citar: Grid Portal Development Kit (GPDK) [3], Legion Grid Portal [4], Grid Portal Toolkit (GridPort) [5] y GridSphere [6].

4.2.1 GridSphere

El marco de portal GridSphere provee un portal Web de código abierto. GridSphere permite a los desarrolladores desplegar *portlets* de aplicaciones Web de terceros que pueden ser corridos y administrados a través del contenedor de *portlet* del GridSphere. El marco de portal GridSphere ofrece las siguientes características:

- Implementación del API de *portlet* 100
- Desarrollo de *portlets* usando el estándar *Java Server Faces* (JSF).
- Implementación de API de *portlet* adicional casi completamente compatible con WebSphere 4.2 [7] de IBM (otro contenedor de *portlets*).

[3] GPDK http://doesciencegrid.org/projects/GPDK/
[4] Legion http://legion.virginia.edu/
[5] GridPort http://gridport.net/
[6] GridSphere http://www.gridsphere.org/
[7] WebSphere http://www-306.ibm.com/software/websphere/

- Soporte para el fácil desarrollo e integración de nuevos *portlets* de aplicaciones.
- Modelo de alto nivel para construir *portlets* complejos usando "beans" visuales y la librería de etiquetas de la interfaz de usuario del GridSphere.
- Presentación flexible basada en XML que puede ser fácilmente modificada para crear disposiciones del portal personalizadas.
- Modelo de servicio de *portlets* sofisticado que puede encapsular lógica de *portlet* reutilizable en servicios que pueden ser compartidos entre muchos *portlets*.
- La persistencia de la data para soporte de bases de datos a través del uso de Hibernate JDO/OQL
- Soporte para múltiples idiomas. Es de código abierto y 100% gratuito.

4.2.2 GridPort

El conjunto de herramientas GridPort permite el rápido desarrollo de portales Grid altamente funcionales que simplifican el uso de los servicios Grid subyacentes al usuario final. Comprende un juego de *portlets* y servicios en la capa de portal que proveen acceso a un amplio rango de servicios Grid y de información provistos por tecnología Grid de bajo nivel como Globus, el Repositorio de Información del Portal Grid (GPIR), y Condor. Los *portlets* estos servicios a través de interfases Web configurables a fin de permitir la personalización de las interfases de usuario del portal Grid. GridPort está diseñado para ser usado por desarrolladores de portales Grid, *portlets* y aplicaciones.

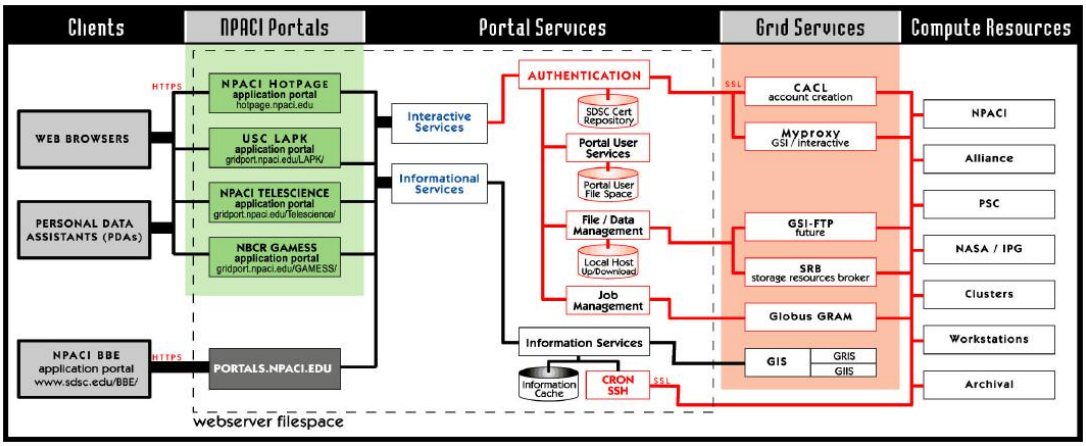


Figura 4.6: El diagrama indica los servicios de portal y los servicios a los que puede acceder en general a través de un portal (en este caso se hace referencia al portal NPACI, basado en GridPort)[46].

GridPort facilita y mejora la velocidad con la que los desarrolladores de portales pueden superar la brecha entre los usuarios finales y el Grid. GridPort es un proyecto del *Texas Advanced Computing Center* (TACC) que empezó en 1997 y fue creado inicialmente para crear el *NPACI HotPage Portal* [8], pero recientemente se ha convertido en parte del proyecto de *software* para portales *Open Grid Computing Environments* (OGCE) [9], el cual provee herramientas de *software* adicionales como *Extreme Lab* [10], *Community Grids Lab* (CGL) [11], y el proyecto Sakai [12]. Los componentes de GridPort 4.0 que constituyen la última versión forman una arquitectura tipo SOA y son los siguientes:

- Autenticación (*Proxymanager Portlet*)
  - Usando el repositorio del GridPort
  - Usando *MyProxy*
- Manejo de archivos (*File Management Portlet*)
  - Listado de archivos
  - Transferencia de archivos
- Manejo de recursos
  - Visualización de estado de recursos (*GPIR Browser Portlet*, el cual soporta servicios Web)
  - Envío de trabajos simples (*GRAM Job Submission Portlet*)
  - Envío de trabajos para Condor (*Condor Job Submission Portlet*)
  - Los servicios de información sobre el estado de los recursos (*Grid Resource Information System* (GIS/GRIS) de Globus)
- Servicios independientes
  - Los servicios Web GPIR y CFT (*Comprehensive File Transfer Service*) fueron extraídos del GridPort, tal que pueden ser instalados por separados. GridPort emplea GridFTP y/o el servicio Web CFT para transferencia de archivos.
  - Estos servicios emplean una base de datos liviana Hypersonic SQL de uso sencillo.

GridPort emplea Tomcat como contenedor Web y GridSphere como contenedor de *portlets*.

[8] NPACI http://www.npaci.edu/online/
[9] OGSE http://www.collab-ogce.org/ogce/index.php/Main_Page
[10] Extreme Lab http://www.extreme.indiana.edu/xgws/index.html
[11] CGL http://communitygrids.iu.edu/index.php
[12] Sakai http://sakaiproject.org/

4.2.3 GPDK

GPDK [47] (iniciales de *Grid Portal Development Kit*) proporciona tanto un entorno de desarrollo para crear nuevos portales como una colección de componentes Java para realizar operaciones básicas sobre el Grid , tales como envío de tareas, transferencias de ficheros, obtención de información, etc. GPDK propone una arquitectura de tres capas: servidor Web, Globus y Grid. Primero, el navegador se comunica con el servidor Web utilizando una conexión segura. El servidor Web puede acceder a los servicios del Grid utilizando la infraestructura de Globus. Un portal de GPDK es una aplicación Web construida utilizando tecnologías JSP (Java Servlet Pages) y *Java Beans*, que se ejecutan sobre Tomcat, el servidor de aplicaciones Java de referencia.

Tomcat se puede configurar para funcionar tras cualquier servidor Web del mercado. Los componentes Java proporcionados por GPDK proporcionan un interfaz de alto nivel que engloba las funcionalidades de "Java CoG kit" [13] y de otras bibliotecas de clases commodity que proporcionan servicios de gestión de certificados y consulta de información. "Java CoG kit" proporciona acceso a servicios del Grid a través del entorno Java. Con "Java CoG kit" se proporcionan bibliotecas de clases para incorporar en aplicaciones Java, así como utilidades de línea de comandos implementadas con estas clases.

En resumen, aún cuando el *middleware* se encarga de exponer los recursos distribuidos que conforman un Grid de manera transparente, la utilización de estos recursos no resulta del todo sencilla. Es así que surge la creación de Portales Grid, que es una aplicación basada en Web que actúa como interfaz entre el usuario y el entorno Grid. La función principal del portal es ocultar la complejidad de la infraestructura Grid para acercarla a usuarios no especializados. Un portal está compuesto por un conjunto de interfases, donde cada una facilita la interacción con las aplicaciones y/o funcionalidades del Grid, dichas interfases se denominan *portlets*.

En la actualidad existen muchos portales y *portlets* con diversos propósitos, entre ellos tenemos Portal NASA QuakeSim [14], el cual realiza simulaciones geofísicas; el GridChem referente a propiedades de diversos compuestos químicos [15], BIRN un portal de biomedicina [16], y los *portlets* Gaussian y MPQC [17], estos últimos proveen interfases para los programas de química cuántica GAUSSIAN[18] y MPQC[19]. Los servicios de portales incluyen típicamente personalización, seguridad, manejo de datos, envío de trabajos, servicios de información, servicios de colaboración y puede proveer herramientas de visualización de datos.

[13] Java CoG Kit http://www.globus.org/cog/java/
[14] NASA JPL QuakeSim Portal http://complexity.ucs.indiana.edu:8282
[15] GridChem Project http://lqcd.jlab.org/
[16] BIRN Portal https://portal.nbirn.net/gridsphere/gridsphere
[17] Computacional Chemistry CeCalcula http://quantum.cecalc.ula.ve/
[18] Gaussian Portlet http://www.gaussian.com
[19] MPQC Portlet http://www.mpqc.org/

Capítulo 5

Portlet CHOQUE

Este *portlet* ha sido diseñado para ser usado por la comunidad de astrofísica relativista (tanto docentes como investigadores) quienes encontrarán una herramienta para simular la propagación de discontinuidades hidrodinámicas en esferas radiantes. En este caso se trabaja con un código numérico desarrollado en fortran cuyas bases físicas fueron establecidas en el Capítulo 1. El *portlet* proporciona una forma sencilla de generar archivos de entrada para las subrutinas en fortran que ejecutan los cómputos y luego mandar a procesar estos mismos en un GRID para luego mostrar los resultados obtenidos.

5.1 Recursos computacionales

El *portlet* objeto del presente proyecto se desarrolló en un Grid instalado utilizando gLite 3.10.01 en la Universidad de Los Andes (ULA). El Grid de la ULA el cual se encuentra instalado en CeCalCULA, está compuesto por una serie de servicios que utilizan *hardware* heterogéneo, la descripción de este *hardware* es mostrado en la siguiente Tabla 5.1

Tabla 5.1: hardware de CeCalCULA utilizado en los proyectos de Grid.

Cantidad	Marca	Modelo	Memoria	DD (GB)
4	Sun	V20z	4 GB	147
4	Sun	X4100	8 GB	147
8	Clon	Pentium IV	512 MB	80

Además, se cuenta con un equipo de almacenamiento de datos de la marca SUN modelo 3300 con una capacidad de 1 TB. El sistema operativo instalado actualmente en el Grid de la ULA es

el *Scientific Linux* versión 3.0.8 [1]. La instalación se hizo utilizando el repositorio dispuesto por EELA que se encuentra en la *Universidade Federal do Río de Janeiro* (UFRJ) [2] en Brasil. Para la implementación de *portlets* en el Grid se instaló en una misma máquina GridPort 4.0 y una Interfaz de Usuario (UI, por sus siglas en inglés *User Interface*) utilizando el *middleware* gLite 3.0.1.

5.2 Arquitectura de *portlet* CHOQUE

Para el desarrollo del *portlet* CHOQUE se utilizo el GridPort, un conjunto de herramientas de *software* que facilita de construcción de portales Grid. En este caso se hizo uso de su módulo de autenticación compuesto por la Infraestructura de Seguridad Grid de Globus (*Globus Grid Security Infrastructure* (GSI)) y *Myproxy*, este último es un repositorio de credenciales en línea. GridPort está diseñado para enviar y monitorear trabajos a GT, sin embargo el *middleware* empleando en el Grid de la ULA es gLite, es por ello que fue necesario emplear en la parte lógica del *portlet* los comandos propios del *middleware* gLite a fin de enviar, monitorear y recuperar los trabajos.

El marco de portal empleado es GridPort 4.0 y el contenedor Web empleado es Apache Tomcat 5.0 [3] en combinación con Apache Maven 1.0.2 [4] . GridPort permite crear *portlets* de aplicación bajo la especificación JSR 168, que pueden correr y ser administrados en el contenedor de *portlets* GridSphere 2.2. Maven es una herramienta de *software* usada para la construcción y manejo de cualquier proyecto basado en Java, que proporciona una estructura de proyecto bien definida, unos procesos de desarrollo bien definidos a seguir, y una documentación coherente que mantiene a los desarrolladores y clientes informados sobre lo que ocurre en el proyecto. Maven es capaz de construir el proyecto en diferentes tipos de salidas como JAR o WAR con facilidad.

Por otra parte, los recursos utilizados incluyen el *hardware* descrito en la sección anterior y el compilador de fortran G95. La arquitectura es tipo SOA y puede observarse en la Figura 5.1.

Además se empleó el entorno de programación Eclipse [5] como herramienta de desarrollo IDE (*Integrated Development Environment*).

5.3 Estructura de directorios del *portlet* CHOQUE

La especificación para *portlets* JSR 168 especifica que los componentes del *portlet* debe ser empaquetados y desplegados como parte de un archivo de aplicación Web (WAR) estándar; es por ello que se utiliza Apache Maven. La Figura 5.2 se muestra la estructura de directorios del *portlet* CHOQUE, la es muy útil en el desarrollo y empaquetamiento del *portlet*. Esta estructura contiene lo siguiente:

[1] Scientific Linux https://www.scientificlinux.org/
[2] UFRJ http://www.ufrj.br/
[3] Apache Tomcat http://tomcat.apache.org/
[4] Maven http://maven.apache.org/
[5] Eclipse http://www.eclipse.org

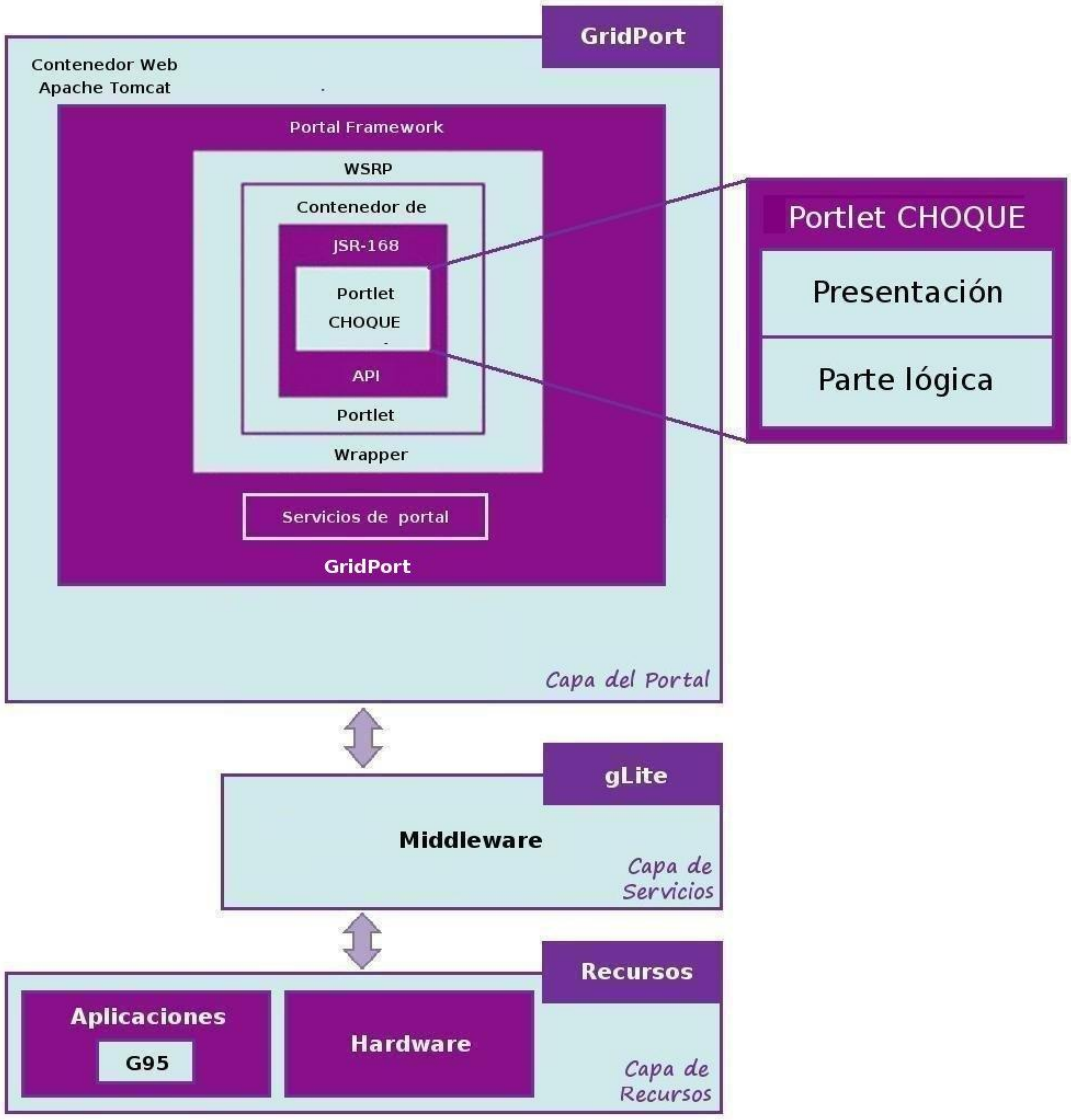


Figura 5.1: Arquitectura del *portlet* CHOQUE

1. **compilar**: con el comando ./compilar se despliega el *portlet* en el portal (**maven-deploywar**) y de reiniciar el servidor Web Tomcat (**$CATALINA_HOME/bin/shutdown.sh**; **$CATALINA_HOME/bin/startup.sh**).

2. **maven.xml**: contiene objetivos adicionales del proyecto, este archivo se encuentra en el mismo directorio que project.xml a fin de que sea cargado y sus objetivos sean procesados cuando el *portlet* sea desplegado.

3. **project.properties**: este archivo define los repositorios de Maven y la localización del

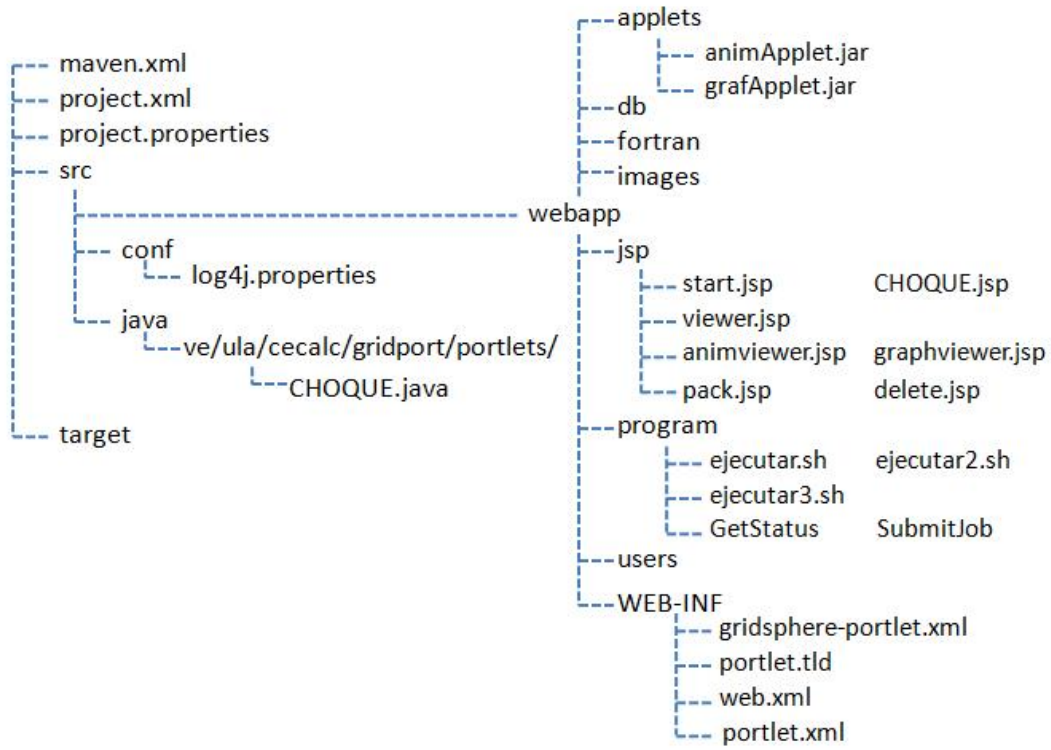


Figura 5.2: Estructura de directorios y archivos del *portlet* CHOQUE

Tomcat.

4. **project.xml**: es el archivo descriptor del proyecto. Contiene dos secciones principales: las dependencias del proyecto y los recursos del proyecto (como los archivos properties). Por ejemplo, la dependencia jglobus.jar provee las clases necesarias para obtener una credencial Grid y portlet-api-1.0.1.jar provee las clases e interfases de la especificación JSR 168.

5. **src/java**: en este directorio se encuentra el archivo CHOQUE.java que extiende de *GenericPortlet*, este archivo corresponde a la parte lógica del *portlet*.

6. **src/conf**: aquí está ubicado log4j.properties en el cual se define la configuración Log4j.

7. **src/webapp**:

   7.a. **applets**: es la ubicación de los archivos circuloApplet.jar y grafApplet.jar.

   7.b. **db**: contiene archivos HSQL [6]. HSQL es un sistema gestor de bases de datos escrito en Java.

[6] HSQL http://www.hsqldb.org/

7.c. **fortran**: en este directorio se ubican las subrutinas de fortran que realizan los cálculos.

7.d. **images**: ubicación de las imágenes empleadas en la interfaz gráfica del *portlet*.

7.e. **jsp**: ubicación de los archivos de interfase Web, es decir, de los archivos JSP.

7.f. **program**: en este directorio se ubican los *script* ejecutar.sh ejecutar2.sh y ejecutar3.sh son los encargados de enviar, consultar y traer los trabajos del Grid empleando comandos propios de gLite.

7.g. **users**: es el directorio de los usuarios.

7.h. **WEB_INF**:

7.h.a. **gridsphere-portlet.xml**: especifica un *portlet* de GridSphere que se encarga de cargar el *portlet*.

7.h.b. **portlet.tld**: en este archivo se incluye una librería JSP tag para ayudar a desplegar páginas del *portlet* con tecnología JSP. Por ejemplo un JSP tag declara automáticamente los objetos *request* y *response* a fin de que ellos puedan ser empleados en los archivos JSP.

7.h.c. **portlet.xml**: es el descriptor de despliegue del *portlet* el cual define la meta información que necesita el contenedor de *portlets* para correrlo. Para cada definición en portlet.xml el contenedor de *portlet* instancia un objeto único de esa clase para despachar todas las solicitudes realizadas al *portlet*. En resumen, es el archivo de configuración del *portlet*.
portlet-name, *portlet-class*, *init-param* y *user-attribute* definen el nombre, la clase, los parámetros iniciales y atributos para el *portlet* como la localización del *proxy*, el puerto, el nombre del usuario y el *password* en la base de datos. Los elementos *supports* definen que tipo de contenido soporta el *portlet* y los modos del *portlet* disponibles, en este caso solo está definido el modo VIEW y el contenido es text/html. el elemento *portlet-info* es utilizado para establacer el título por defecto del *portlet* y las palabras claves (*keywords*).

7.h.d. **web.xml**: como una aplicación *portlet* es una aplicación Web extendida, se debe incluir un archivo web.xml, el cual es el descriptor de despliegue Web que define las clases usadas por el *portlet* y el nombre de la aplicación *portlet*. Cada *portlet* es mapeado a un servidor definido en web.xml, en este caso se especifica el servidor de GridSphere que carga el *portlet*.

8. **target**: éste subdirectorio es creado por Maven cuando se compila y empaqueta el *portlet* (**./compilar**) para ser utilizado como un *workspace*. La estructura de subdirectorios dentro de target es la estructura de directorios descrita anteriormente, además es aquí donde se guarda el archivo WAR creado.

El archivo WAR creado incluye las clases de java compiladas, todos los archivos jar requeridos como gridsphere-ui-tags.jar el cual es un archivo específico del contenedor, el archivo WEB-INF/web.mxl, el archivo WEB-INF/portlet.xml y la librería JSP tag.

5.4 Diseño de la interfaz de usuario

5.4.1 Casos de uso

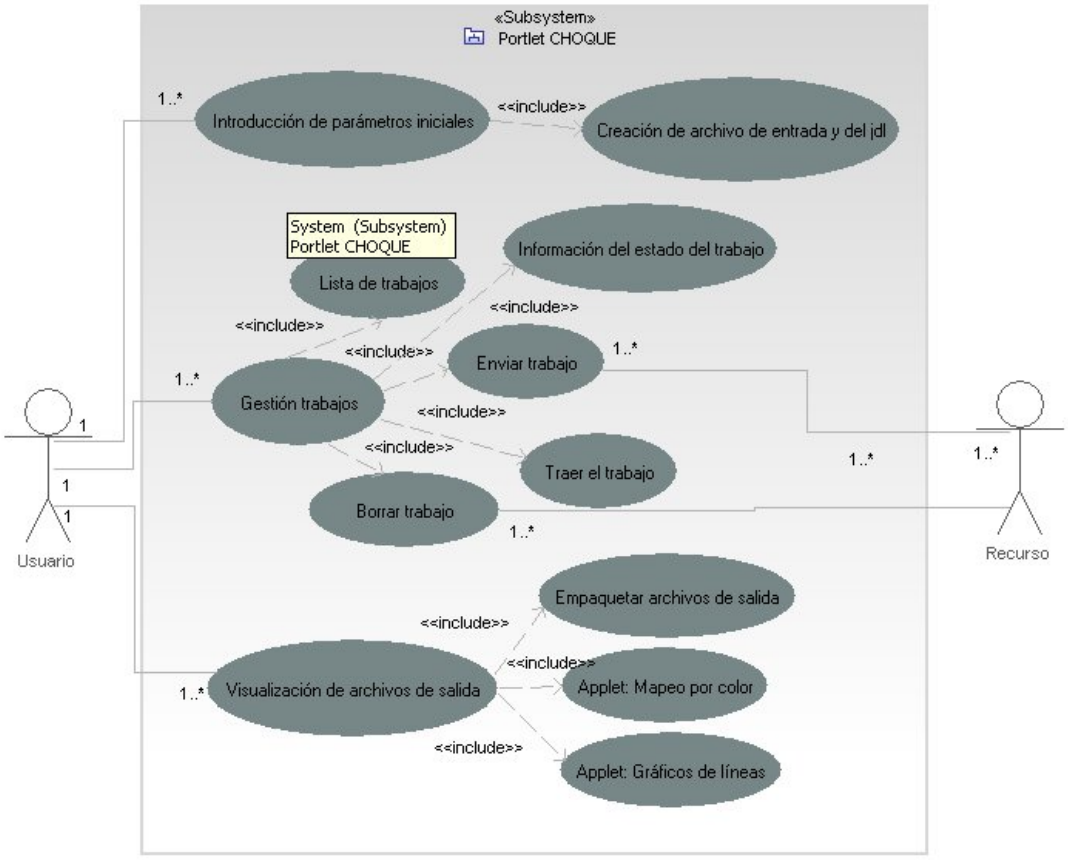


Figura 5.3: Casos de uso del *portlet* CHOQUE

El *portlet* CHOQUE permite realizar las operaciones que se muestran en el diagrama de la Figura 5.3, y que son descritos a continuación:

- **Introducción de parámetros iniciales**: El usuario puede introducir los valores de los parámetros iniciales necesario para la ejecución del modelo de una onda de choque en esferas relativistas autogravitantes.
- **Gestión de trabajos**: El usuario puede enviar un trabajo al Grid, así como obtener una lista de trabajos enviados con su respectivo estado, para poder traer los archivos de salida desde el recurso remoto o eliminarlos una vez que ha finalizado su ejecución.

- **Visualización de los archivos de salida**: El usuario puede obtener un comprimido con los archivos de salida del *portlet*, puede visualizarlos mediante gráficos de líneas o mediante mapeos por color de los datos.

5.4.2 Navegación del portlet

La estructura de páginas y enlaces entre páginas propia de un sitio Web se puede modelar mediante un diagrama de navegación. En la Figura 5.4 se muestra la descomposición del diagrama de navegación del portlet en páginas jsp.

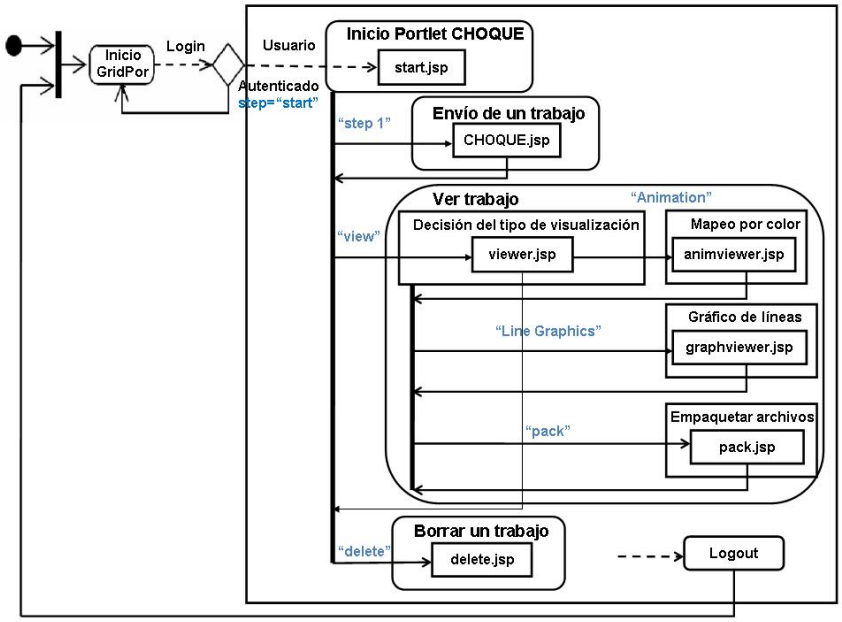


Figura 5.4: Descomposición del diagrama de navegación del *portlet* CHOQUE en páginas jsp

Cada evento (por ejemplo presionar un determinado botón o enlace) en el *portlet* CHOQUE es detectado mediante un cambio en una variable llamada *step*, el valor de esta variable define que jsp se va desplegar. Los posibles valores de dicha variable son los siguientes:

a. **step="start"**: es el valor por defecto de step, en ese caso se actualiza la base de datos y el doView despliega el archivo start.jsp, en el cual se muestran el nombre de los trabajos que han sido enviados, la fecha y la hora en la que fueron enviados, su estado, existe un botón para eliminarlos, un botón para visualizar y/u obtener los archivos de salida y otro para crear un nuevo trabajo.

b. **step="step 1"**: doView despliega el archivo CHOQUE.jsp, en el cual se encuentra un formulario donde el usuario introduce los parámetros de entrada del sistema, el nombre del trabajo y una breve descripción. La validación de los datos se realiza a través de funciones hechas en JavaScript.

c. **step="step 2"**: processAction almacena los valores de los parámetros introducidos por el usuario en la *hashtable*, crea el archivo de entrada para las subrutinas de fortran, crea el archivo del trabajo a enviar al Grid en formato jdl[7] (*Job Description Language*), crea el ejecutable incluido en el jdl y luego envía estos archivos al Grid.

   Para realizar el envío de este trabajo al Grid se debe realizar 2 operaciones: Petición de procesamiento al Grid, para ello para ello se ejecuta el *script* ejecutar.sh, el cual envía el jdl empleando el comando de gLite **glite-job-submit**. Transferencia del archivo desde el servidor del portal a los recursos remotos (a través del uso del GridFTP) donde será procesado por las subrutinas de fortran que integran numéricamente el sistema de ecuaciones diferenciales en la superficie de la distribución y luego evalúan las expresiones para las funciones métricas para determinar y validar el valor de las variables físicas.

   Finalmente se actualiza la base de datos y se muestra start.jsp. A fin de actualizar el estado del trabajo en la base de datos se ejecuta el *script* ejecutar2.sh, el cual emplea el comando **glite-job-status**.

d. **step="view"**: si el estado del trabajo es DONE o CLEARED, se recuperan los archivos de salida del trabajo enviado ejecutando el *script* ejecutar3.sh el cual emplea el comando **glite-job-output**. doView despliega el archivo viewer.jsp este posee dos botones que permiten seleccionar el tipo de visualización y un botón para obtener un comprimido con los archivos de salida del trabajo.

e. **step="Line Graphics"**: el processAction() instancia el método viewer() cuya salida es la dirección del archivo de salida del trabajo que se pasará como parámetro al applet grafApplet, el archivo dependerá de que variable física (velocidad radial, presión radial hidrodinámica, presión radial de radiación, presión tangencial hidrodinámica, presión tangencial de radiación, presión total, flujo de radiación, densidad de energía hidrodinámica, densidad de energía de radiación y/o densidad de energía total) seleccione el usuario en el comboBox de graphviewer.jsp (ésta es la página que contiene el applet grafApplet). Por omisión el archivo a visualizar es el correspondiente a la velocidad radial de la esfera.

f. **step="Animation"**: el processAction() instancia el método viewer() cuya salida es la dirección del archivo que se pasará como parámetro al applet animApplet y el método selectMap() cuya salida es el tipo de mapa de color que se utilizará para representar los datos. El archivo de la salida del trabajo que se visualizará dependerá de que variable física seleccione el usuario en el comboBox de animviewer.jsp (ésta es la página que contiene el applet animApplet). De nuevo, el archivo por omisión a visualizar es el correspondiente a la velocidad radial de la esfera. El tipo de mapa de color empleado depende de la opción que seleccione el usuario en un combobox ubicado en animviewer.jsp

g. **step="pack"**: processAction empaca el contenido de la sesión en un archivo .tgz y doView despliega el archivo pack.jsp, en el cual se encuentra una confirmación de que el contenido de la sesión fue empaquetado.

[7] JDL http://www.grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf

h. **step="delete"**: en este caso processAction invoca el método deleteDir() que elimina el proyecto completo en cuestión, archivos y datos en la BD. doView muestra el archivo delete.jsp en el cual se encuentra una confirmación de que el contenido de la sesión fue borrado.

En la Figura 5.5 se muestra el diagrama de secuencia para la variable *step*.

5.5 Métodos del *portlet* CHOQUE

El archivo CHOQUE.java contiene principalmente tres métodos, init(), processAction() y doView() definidos la especificación JSR 168. Cuando el *portlet* es instanciado por primera vez, el método init() es invocado y provee al *portlet* con las inicializaciones e información de configuración necesarias para manejar las solicitudes realizadas al *portlet*, además crea la base de datos de CHOQUE y proporciona permisos de ejecución a los *script* ejecutar.sh, ejecutar2.sh y ejecutar3.sh. En la base de datos se almacenan el nombre del usuario, su *password*, el estado del trabajo y el identificador asignado al trabajo enviado por el usuario. Los valores de los parámetros iniciales se obtienen empleando *getInitParameter()* para obtener la información del archivo portlet.xml.

El modelo de un *portlet* en general separa la presentación (rendering del *portlet*) de la parte lógica (operaciones que se llevan a cabo en respuesta a una acción ocurrida) del *portlet* en distintos métodos (ver Figura 5.1). En el caso del *portlet* CHOQUE la presentación está a cargo del método doView() y la parte lógica del método processAction().

doView(RenderRequest request, RenderResponse response) es el método que selecciona que respuesta mostrar por pantalla, es decir, que jsp desplegar y también actualiza la base de datos. Los datos que este método recibe vienen del processAction(), y envía datos a los archivos jsp. Para obtener un dato que proviene del processAction() se utiliza *getParameter*("nombre del parámetro a capturar"), mientras que para enviar un dato (cadena de caracteres) a un archivo jsp se utiliza *setAttribute*("nombre variable en el jsp",variable que se envia).

processAction(ActionRequest request, ActionResponse response) es el método encargado de procesar los datos enviados por algun jsp, ejecutar ciertos procesos y luego enviar datos al método doView. Para capturar y almacenar las variables y el contenido del archivo jsp se hace uso de una *hashtable*, mientras que para enviar un dato al doView() se utiliza *setRenderParameter*("nombre variable en el en el doView",variable que se envía). El flujo de los datos en el *portlet* CHOQUE pueden ser visualizados en la Figura 5.6.

El contenedor de *portlets* es el encargado de invocar al método doView() para desplegar una determinada página jsp en el *portlet* y de invocar al processAction() cuando el *portlet* recibe un evento, por ejemplo el usuario hizo click sobre un botón o envió un formulario.

5.6 Archivo jdl y envío de un trabajo en gLite

El *Job Description Language* (JDL) es usado para describir los trabajos a ejecutar en el Grid; para especificar las características del trabajo: el ejecutable que será usado, los archivos de entrada (*InputSandbox*), de salida (*OutputSandbox*) y los requerimientos para que el trabajo

sea ejecutado utilizando el *middleware* gLite; dichas características serán usadas por el WMS para seleccionar el mejor recurso que satisfaga los requerimientos del trabajo. A continuación se muestra un ejemplo de un archivo jdl que describe un trabajo enviado empleando el *portlet* CHOQUE.

```
Type="Job";
JobType="Normal";
Executable="trabajo1.sh";

StdOutput="trabajo1.out"; StdError="trabajo1.err";

InputSandbox={"trabajo1.sh","makefile","acepta.f","main.f","derivs.f",
"evalvarfis.f","imprime.f","integrador.f","metdermanto.f","metdernucleo.f",
"varfis.f","varfisenCmanto.f","varfisenCnucleo.f","datos.dat","global.inc"};

OutputSandbox={"trabajo1.out","trabajo1.err","supmaschoque.res","lum.res",
"vfcmas.res","vfcmenos.res","omega.res","Pr.res","flujorad.res","rho.res",
"rhoR.res","PreR.res","rhobarra.res","presbarra.res","pretanR.res",
"pretan.res","ptbarra.res","radio.res"};

Requirements =
Member("G95-3.5.0",other.GlueHostApplicationSoftwareRunTimeEnvironment);

RetryCount = 7;
```

GridPort accesa al Grid desde una máquina Interfaz de Usuario (UI) con la Autoridad de Certificación (CA) del usuario para crear su certificado *proxy* y autenticarlo empleando MyProxy (parte superior izquierda de la Figura 5.7); una vez creado el archivo jdl por el *portlet* CHOQUE y que se ha ejecutado el *script* ejecutar.sh el trabajo es enviado desde la UI hacia el Manejador de Recursos (RB) (parte superior central de la figura). En ese momento un evento es registrado por el Servicio de Rastreo (LB) y el estado del trabajo es SUBMITTED.

El RB solicita al Sistema de Información del Grid (GIS, por sus siglas en inglés *Grid Information System*) (parte derecha de la figura) el estado actual de los recursos computacionales y de almacenamiento y al Catálogo de Archivos (LFC) (parte superior derecha) la localización de cualquier archivo externo que necesite el trabajo y no este incluido en *InputSandbox*, de ésta manera puede seleccionar cual es el mejor Elemento de Cómputo (CE) y de Almacenamiento (SE) para correr el trabajo (parte inferior derecha). El GIS es una cache interna de información leída desde el BDII. Otro evento es registrado en el LB y el estado del trabajo pasa a WAITING.

Posteriormente el RB prepara el trabajo para enviarlo creando un *script wrapper* el cual es transferido en conjunto con otros parámetros al CE seleccionado, en ese momento el estado del trabajo pasa a ser READY.

El CE recibe la solicitud y envía el trabajo a ejecutar al Sistema de Manejo de Recursos Locales (LRMS). Un evento es registrado en el LB y el estado del trabajo cambia a SCHEDULED.

El LRMS maneja la ejecución de los trabajos en los WNs locales . El *InputSandbox* es copiado desde el RB a un WN disponible, en el cual es trabajo es ejecutado. En ese momento el estado del trabajo cambia a RUNNING. El estado del trabajo puede solicitado por el usuario todo el tiempo.

Si el trabajo culmina sin errores, los archivos especificados en el *OutputSandbox* son transferidos al RB y el estado del trabajo pasa a ser DONE. En este punto, el usuario puede presionar el icono de archivos en start.jsp, la variable step adquirirá el valor "step 2" y será ejecutado el *script* ejecutar3.sh, el cual recupera los archivos de salida del trabajo y los envía a la UI. Nuevamente un evento es registrado en el LB y el estado del trabajo cambia a CLEARED.

Si el sitio donde se envió el trabajo no puede aceptarlo, el mismo será automáticamente redirigido a otro CE que satisfaga los requerimientos. Luego de un máximo número de reeenvíos el estado del trabajo será establecido como ABORTED.

5.7 Visualización científica en el *portlet* CHOQUE

Las técnicas de visualización empleadas en el *portlet* CHOQUE son implementadas en applets que permiten observar y analizar la evolución de la variable física con respecto al radio de la configuración esférica en el tiempo y posibilitan la comunicación entre el usuario y el programa por medio de los controles que se disponen en sus áreas de trabajo. Estos applets se encargan de la visualización de los archivos de salida de las subrutinas de fortran.

La clase grafApplet derivada de *Applet* que implementa el interfase *Runnable* se encarga de leer el archivo de radios de la configuración y el archivo de una de las variables físicas (para lo cual instancia la clase *readFile*) cuyos nombres y ubicación son pasados como parámetros al applet desde graphviewer.jsp y de proveer una visualización tradicional de la variable física en estudio $F(r)$ mediante gráficos de líneas $F(r)$ vs. r (para ello instancia la clase *DrawingArea*).

Mientras que la clase animApplet recibe los nombres, la ubicación de los archivos a visualizar y el tipo de mapeo por color (mapeo por color arcoiris, isomorfo o segmentado) que el usuario desea, desde animviewer.jsp como parámetros y realiza una asociación entre un arreglo de colores y la distribución de los valores de la variable física en el interior de la configuración material y mediante la presentación de ésta, cuadro a cuadro, la evolución del modelo es evidente. El código de colores empleado para crear los arreglos de colores empleados fue RGB, esto se realiza en la clase *circuloCanvas*. Para leer los archivos instancia la clase *readFile*.

En las Figuras 5.8 y 5.9 se muestran los diagramas de clase para los applets empleados.

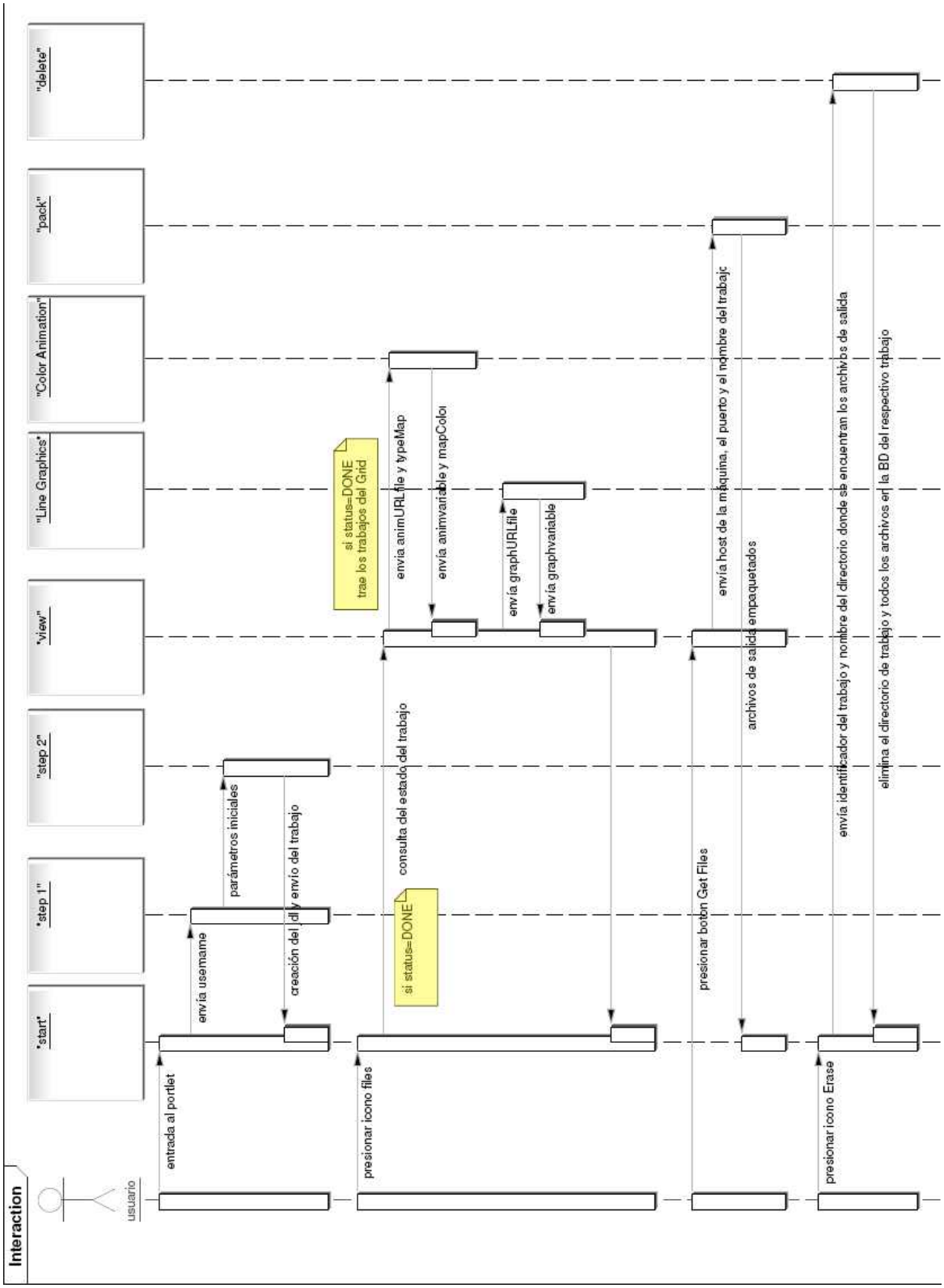


Figura 5.5: Diagrama de secuencia de la variable *step*.

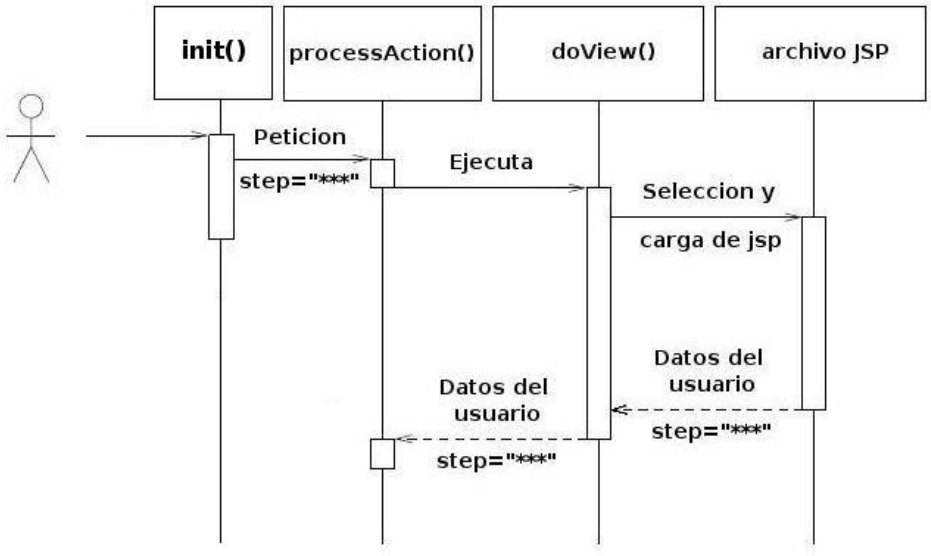


Figura 5.6: Interacciones entre los métodos del *portlet* CHOQUE tras una petición del usuario.

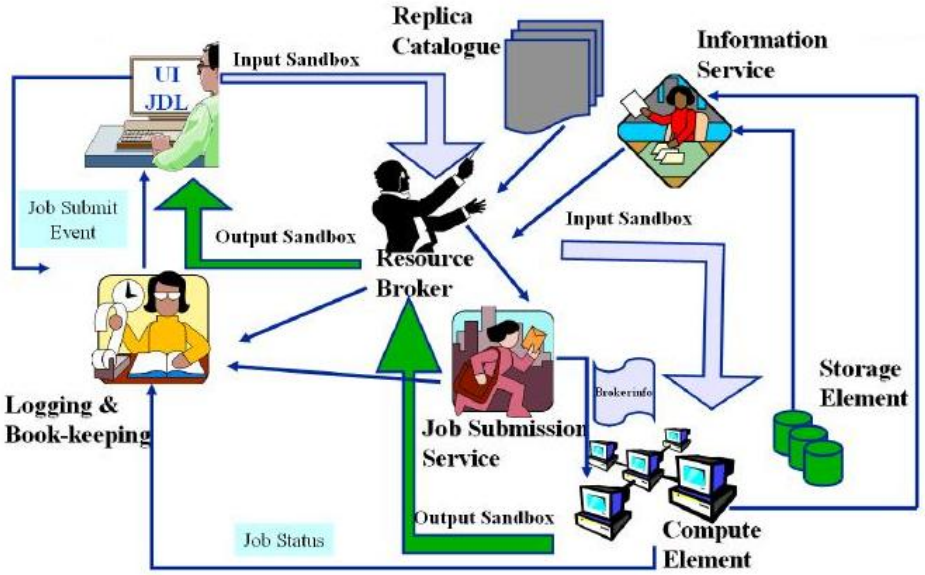


Figura 5.7: Secuencia de ejecución de un trabajo empleando gLite.

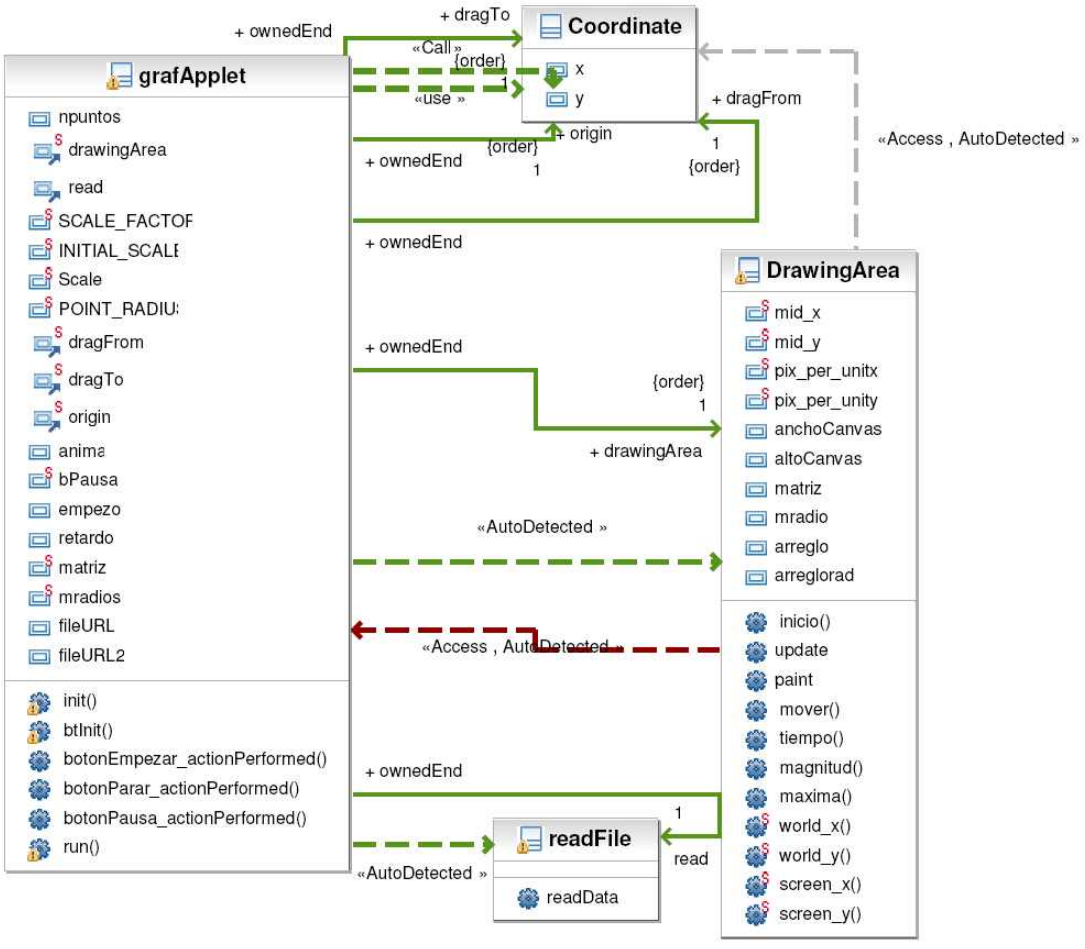


Figura 5.8: Diagrama de clases del applet encargado de la visualización mediante gráficos de líneas.

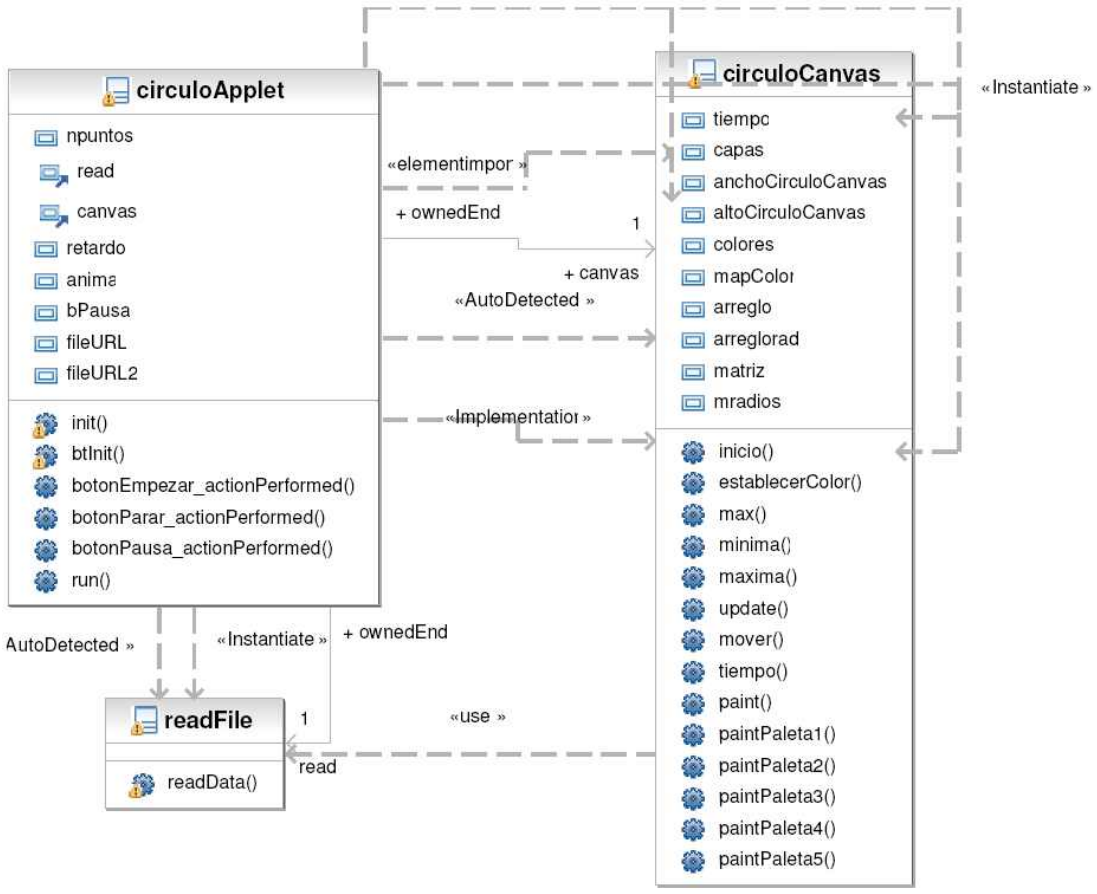


Figura 5.9: Diagrama de clases del applet encargado de la visualización mediante mapeo por color de los datos.

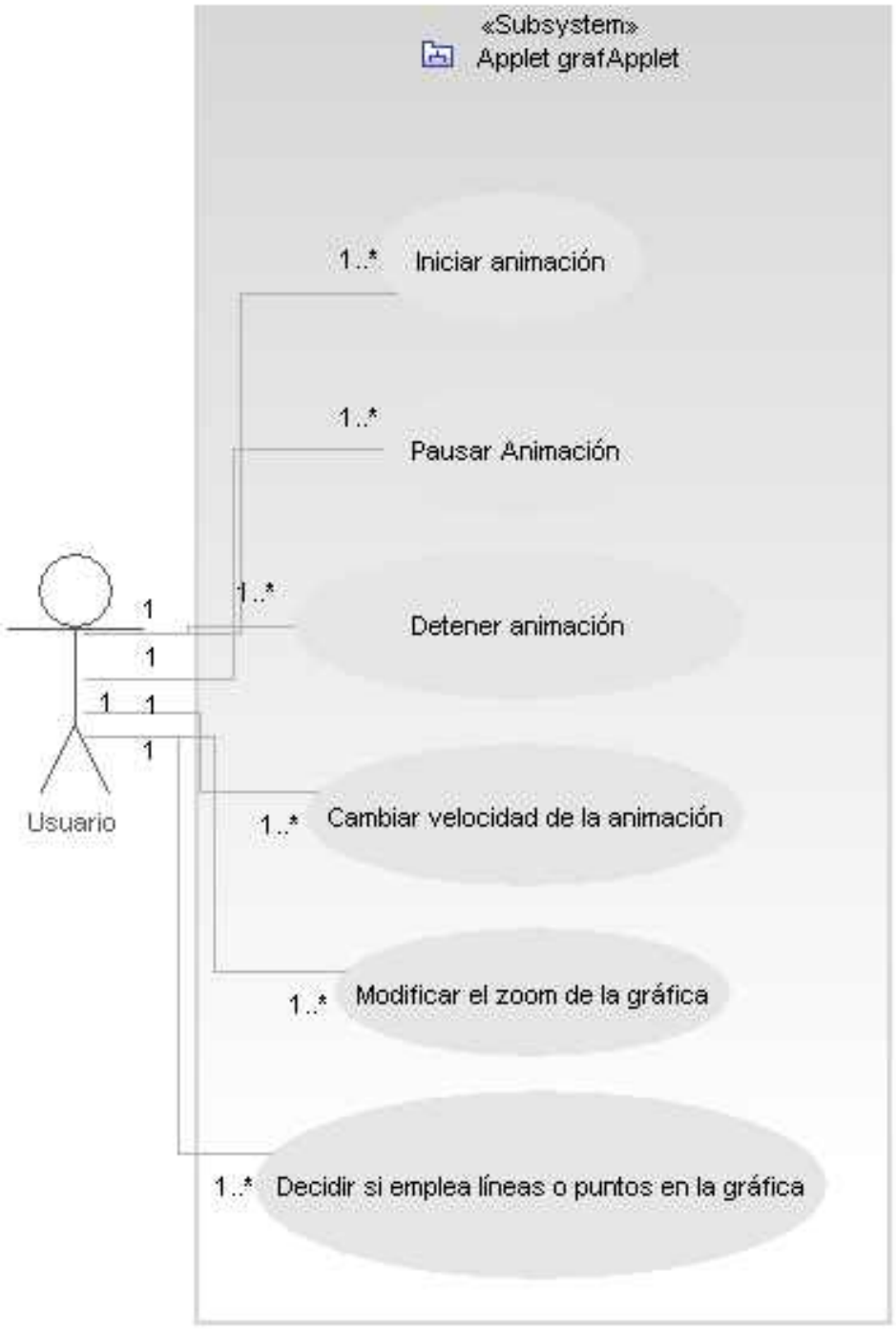


Figura 5.10: Casos de uso del applet grafApplet.

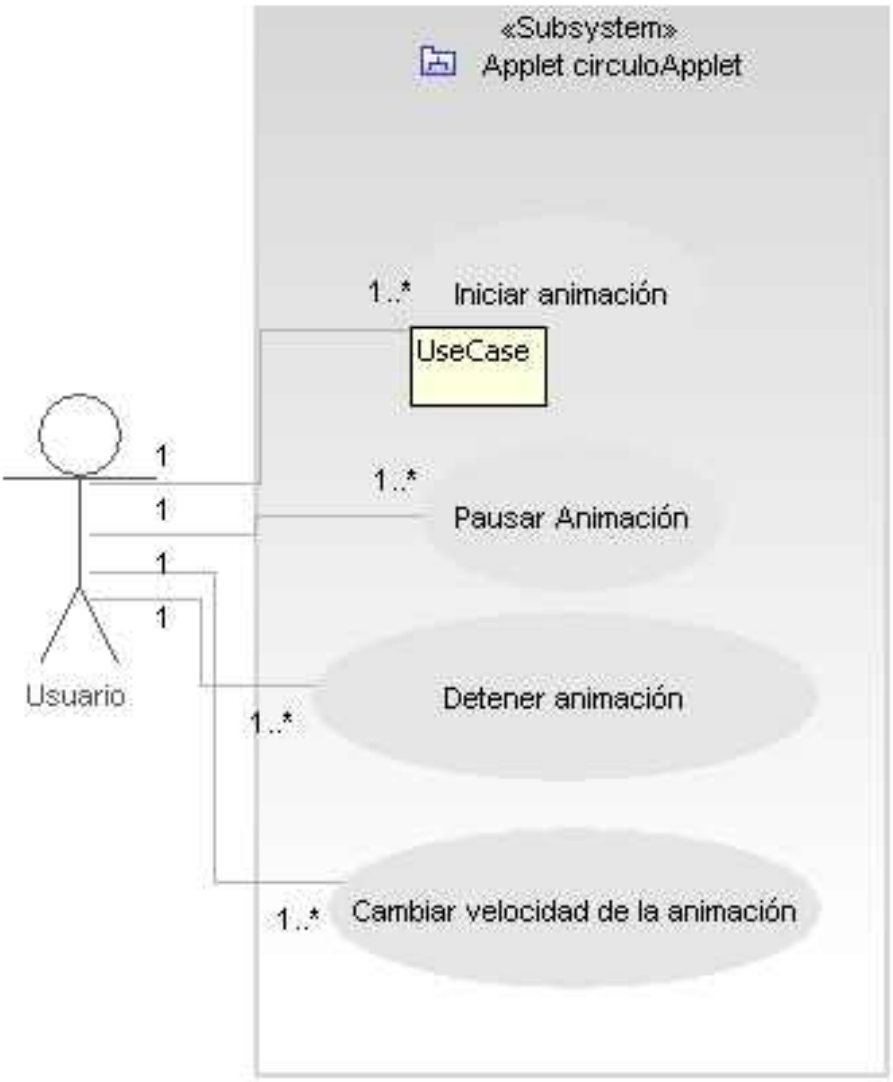


Figura 5.11: Casos de uso del applet animApplet.

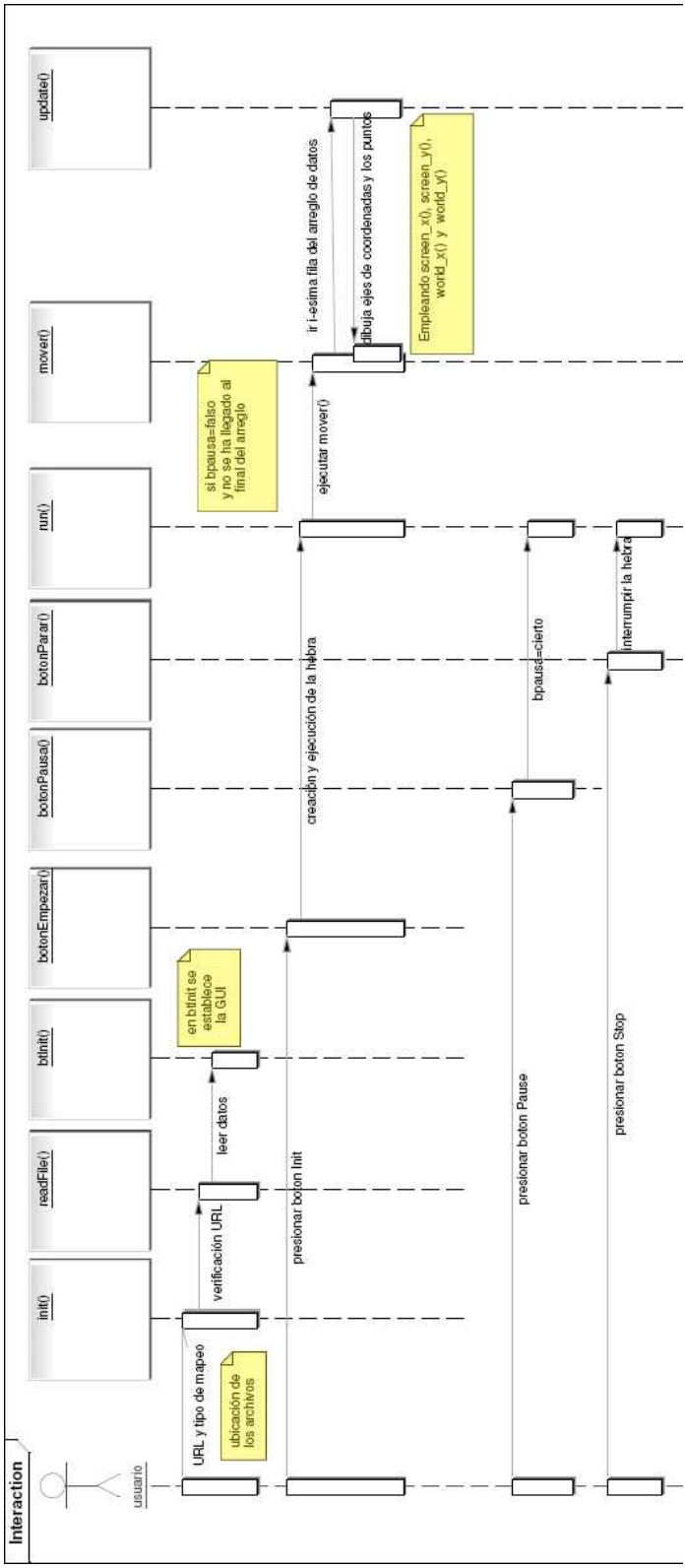


Figura 5.12: Diagrama de secuencia en el applet grafApplet.

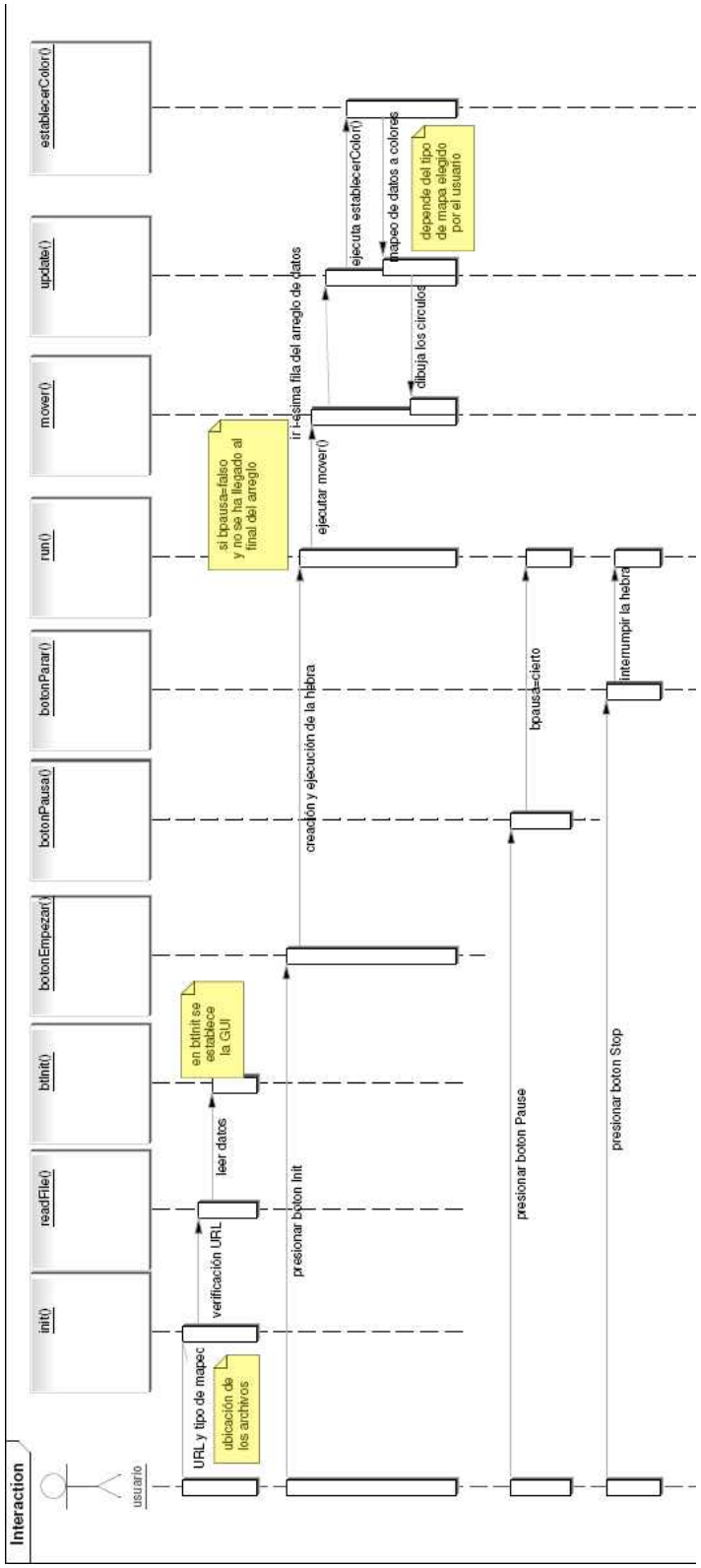


Figura 5.13: Diagrama de secuencia en el applet animApplet.

Capítulo 6

Conclusiones

Los Grids se han convertido en un tema de alto interés en los últimos años, gracias a todas las ventajas a nivel computacional y de recursos que proporcionan. Sin embargo, es necesario que los usuarios puedan utilizar los Grids y las aplicaciones que funcionan en estos de una forma sencilla y transparente. Esto se puede lograr mediante los portales y portlets, que presentan un interfaz sencilla y atractiva a los recursos hardware y software que componen el Grid.

En nuestro caso se desarrollo un portlet llamado CHOQUE basado en la especificación JSR 168, el cual cumple con las siguientes tareas:

- Construcción del archivo de entrada para las subrutinas en fortran.
- Construcción del archivo jdl necesario para especificar las características y necesidades del trabajo.
- Uso del middleware gLite para: la transferencia de archivos de entrada y de salida, el envío y el monitoreo del trabajo.
- Visualización de los archivos de salida, bien sea a través de animaciones de gráficos de líneas o mediante animaciones de mapeos por color, logrando un efecto de movimiento de la esfera radiante a través de los cambios cromáticos en el tiempo.

El portlet CHOQUE puede convertirse en una herramienta para la comunidad de astrofísica relativista (tanto docentes, como investigadores) para simular la influencia que ejerce un esquema de radiación y anisotropía local sobre la propagación de una superficie de discontinuidad en una esfera autogravitante.

Bibliografía

[1] E. Tufte. *The Visual Display of Cuantitative Information.* Connectitud: Graphics Press, 1983. [citado en p. 10]

[2] W. Cleveland. *The Elements of Graphing Data.* Wadsworth, Inc., 1991. [citado en p. 10]

[3] R. Haber y D. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *Proceedings of IEEE Visualization '90, IEEE Computer Society Press*, 1990. [citado en p. 10, 11]

[4] B. McCormick et al. Special issue on visualization in scientific computing. *Computer Graphics*, 21(6), Noviembre 1987. [citado en p. 11]

[5] P. Robertson y L. De Ferrari. Systematic approaches to visualization: Is a reference model needed? *Scientific Visualization Advances and Challenges (L.Rosenblum, R.A.Earnshaw, J.Encarnacao, H.Hagen, A.Kaufman, S.Klimenko, G.Neilson, F.Post y D.Thalmann, eds). IEEE Computer Society and Academic Press*, pages 287–305, 1994. [citado en p. 11]

[6] C. Qin y C. Zhou y T. Pei. Taxonomy of visualization techniques and systems - concerns between users and developers are different. In *Asia GIS Conference 2003*, 2003. URL http://www.hku.hk/cupem/asiagis/fall03/Full_Paper/Qin_Chengzhi.pdf. [citado en p. 11]

[7] E. Tufte. *Envisioning Information.* Connectitud: Graphics Press, 1990. [citado en p. 12]

[8] E. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative.* Connectitud: Graphics Press, 1997. [citado en p. 12]

[9] G. Abram y L. Treinish. An extended data-flow architecture for data analysis and visualization. *Computer Graphics*, 29(2):17–21, 1995. URL http://www.research.ibm.com/dx/proceedings/dx_paper/index.htm. [citado en p. 13]

[10] H. Lefkowitz y G. Herman. Color scales for image data. *IEEE Computer Graphics and Applications*, 12(1):72–80, Enero 1992. [citado en p. 13]

[11] B. E. Rogowitz. Human vision and display design. *Proceedings of Japan Display '92*, pages 335–339, 1992. [citado en p. 13]

[12] P. K. Robertson. Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *IEEE Computer Graphics and Applications*, 8(5):50–63, Septiembre 1988. [citado en p. 13]

[13] W. Lorensen y H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987. [citado en p. 15]

[14] R. S. Gallagher. *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis*. CRC Press, Inc., 1995. [citado en p. 15, 19]

[15] R. B. Wilhelmson et al. A study of the evolution of a numerically modeled severe storm. *International Journal of High Performance Computing Applications*, 4(2):20–36, 1990. [citado en p. 15]

[16] T. Delmarcelle y L. Hesselink. Visualization of second-order tensor fields and matrix data. In *IEEE Visualization*, pages 316–323, 1992. [citado en p. 16]

[17] R. A. Drebin y L. Carpenter y P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, Agosto 1988. [citado en p. 19]

[18] C. R. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, Julio 2004. [citado en p. 20]

[19] M. Chetty y R. Buyya. Weaving computational grids: How analogous are they with electrical grids? *Computing in Science and Engineering*, 4(4):61–71, Julio-Agosto 2002. [citado en p. 22]

[20] I. Foster y C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*. 1999. [citado en p. 22]

[21] M. Baker y R. Buyya y D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Software -Practice and Experience*, 2002. [citado en p. 22, 26]

[22] S. Malaika y A. Eisenberg y J. Melton. Standards for databases on the grid. *Sigmod Record*, 32(3):92–100, 2003. [citado en p. 22]

[23] I. Foster y C. Kesselman y S. Tuecke. The anatomy of the grid. enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3), 2001. URL http://www.globus.org/alliance/publications/papers/anatomy.pdf. [citado en p. 22, 28, 87]

[24] CERN. Gridcafe. URL http://gridcafe.web.cern.ch/gridcafe/. [citado en p. 23]

[25] I. Foster. What is the grid? a three point checklist. *GRIDToday*, Julio 2002. URL http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf. [citado en p. 24]

[26] V. Berstis. Ibm fundamentals of grid computing. URL http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf. [citado en p. 24]

[27] B. Jacob et al. Ibm introduction to grid computing. URL http://www.redbooks.ibm.com/redpapers/pdfs/redp6778.pdf. [citado en p. 24]

[28] B. Sotomayor. The globus toolkit 3 programmers tutorial, Diciembre 2003. URL http://gdp.globus.org/gt3-tutorial. [citado en p. 29]

[29] I. Foster y C. Kesselman y J. Nick y S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. Wiley Interscience, 2003. URL http://www.globus.org/alliance/publications/papers/ogsa.pdf. [citado en p. 29]

[30] GGF. Open grid services architectur (OGSA). URL http://forge.gridforum.org/projects/ggf-editor/ogsa. [citado en p. 29]

[31] GGF. Open grid services infrastructure (OGSI). URL http://forge.gridforum.org/projects/ggf-editor/document/draftogsi-service-1/en/1. [citado en p. 30]

[32] K. Czajkowski et al. From open grid services infrastructure to wsresource framework: Refactoring & evolution. Mayo 2004. [citado en p. 30]

[33] T. Banks. Web services resource framework (WSRF) - primer v1.2. URL http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf. [citado en p. 30]

[34] B. Sotomayor. Globus toolkit 4 programmer's tutorial, 2005. URL http://gdp.globus.org/gt4-tutorial/multiplehtml/index.html. [citado en p. 31]

[35] D. Thain y T. Tannenbaum y M. Livny. Distributed computing in practice: The condor experience. concurrency and computation: Practice and experience, 2004. URL http://www.cs.wisc.edu/condor/doc/condor-practice.pdf. [citado en p. 31]

[36] V. Hamar. Grid Engine. In *Primer Taller de Adiestramiento de Grid en Venezuela*, 2005. [citado en p. 32]

[37] I. Foster y C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997. URL ftp://ftp.globus.org/pub/globus/papers/globus.pdf. [citado en p. 32]

[38] V. Hamar. gLite Overview. In *Segundo Taller Latinoamericano de Computación Grid*, 2006. [citado en p. 34]

[39] F. Rojas. Gridport 4. In *Latin American Workshop for Grid Administrators*, 2005. URL http://gridport.net/main/presentations.html. [citado en p. 38, 42]

[40] A. Shankar. A general (gentle?) introduction to (grid) portals/gateways, Noviembre 2006. URL http://dhruv.uits.indiana.edu/portals/portals-101.pdf. [citado en p. 38]

[41] I. Foster et al. Open grid services architecture use cases, Octubre 2004. URL http://ogf.org/documents/GFD.29.pdf. [citado en p. 40, 87]

[42] A. Abdelnur et al. Portlet specification proposed final draft 2, version 1.0, Agosto 2003. URL www.oasis-open.org. [citado en p. 41, 42, 87]

[43] T. Schaeck. Web services for remote portals (wsrp) note 21, 2002. URL http://www-900.ibm.com/developerWorks/cn/webservices/ws-wsrp/index_eng.shtml. [citado en p. 43, 87]

[44] J. Novotny y M. Russell y O. Wehrens. The core portal architecture: Jsr-168 based portals. URL http://www.extreme.indiana.edu/~gannon/b649/Chapter11.pdf. [citado en p. 42, 44, 87]

[45] SUN MicroSystems. Introduction to jsr 168the java portlet specification. URL http://developers.sun.com/portalserver/reference/techart/jsr168/pb_whitepaper.pdf. [citado en p. 44]

[46] M. P. Thomas et al. The gridport toolkit: a system for building grid portals. *HPDC*, pages 216–227, Agosto 2001. [citado en p. 47, 87]

[47] J. Novotny. The grid portal development kit. *Grid Computing environments: Special Issue of Concurrency and Computation: Practice and Experience*, 14(13-15):11291144, 2002. [citado en p. 49]

[48] J. R. Oppenheimer y H. Snyder. On continued gravitational contraction. *Phys. Rev.*, 56(5):455–459, Sept 1939. [citado en p. 2]

[49] E. Müller. *Computational Methods for Astrophysical Fluid Flow*, chapter Simulation of astrophysical fluid flow, pages 343–494. Springer-Verlag, Berlin, 1988. [citado en p. 2]

[50] J. A. Font. Numerical hydrodynamics in general relativity. *Living Reviews in Relativity*, 6(4), 2003. [citado en p. 2]

[51] M. Liebendörfer et al. Probing the gravitational well: No supernova explosion in spherical symmetry with general relativistic Boltzmann neutrino transport. *Phys. Rev. D*, 63(10):103004–+, May 2001. [citado en p. 3]

[52] S. W. Bruenn y K. R. De Nisco y Mezzacappa. General Relativistic Effects in the Core Collapse Supernova Mechanism. *Astrophys. J.*, 560:326–338, Octubre 2001. [citado en p. 3]

[53] L. Dessart et al. Multi-dimensional radiation/hydrodynamic simulations of protoneutron star convection. *The Astrophysical Journal*, 645:534550, Julio 2006. [citado en p. 3]

[54] L. Herrera y J. Jiménez y G. J. Ruggeri. Evolution of radiating fluid spheres in general relativity. *Phys. Rev. D*, 22(10):2305–2316, Noviembre 1980. [citado en p. 3]

[55] H. Bondi. The Contraction of Gravitating Spheres. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 281(1384):39–48, 1964. [citado en p. 3]

[56] L. Herrera y L. A. Núñez. Evolution of Radiating Spheres in General Relativity: A Seminumerical Approach. *Fundam. of Cosmic Physics*, 14:235–319, 1990. [citado en p. 3]

[57] R. L. Bowers y E. P. Liang. Anisotropic spheres in general relativity. *Astrophysical J.*, 188:657–665, 1974. [citado en p. 3, 6]

[58] L. Herrera y L. A. Núñez. Propagation of a Shock Wave in a Radiating Spherically Symmetric Distribution of Matter. *Astrophys. J.*, 319:868–884, 1987. [citado en p. 3]

[59] L. Herrera y L. A. Núñez. Luminosity Profiles and the Evolution of a Shock Wave in General Relativistic Radianting Spheres. *Astrophys. J.*, 364:212–221, 1990. [citado en p. 3]

[60] W. Barreto y L. Herrera y L. A. Núñez. The Evolution of Discontinuities in Radiating Spheres in the Diffusion Approximation. *Astrophys. J.*, 375:663–673, 1991. [citado en p. 3]

[61] L. Herrera y L. A. Núñez. Hydrodynamics Phase Transitions in a Radiating Spherically Symmetric Distribution of Matter. *Astrophys. J.*, 339:339–353, 1989. [citado en p. 3]

[62] J. A. Rueda and L. A. Núñez. General Relativistic Radiant Shock Waves in the Post-Quasistatic Approximation. In *XXIXth Spanish Relativity Meeting (ERE 2006). Journal of Physics*, Conference Series 66, 2007. [citado en p. 3, 9, 79]

[63] J. A. Pons et al. Hyperbolic character of the angular moment equations of radiative transfer and numerical methods. *Monthly Notices of the Royal Astronomical Society*, 317:550–562, 2000. [citado en p. 4]

[64] J. M. Smit y L. J. van den Horn y S. A. Bludman. Closure in flux-limited neutrino diffusion and two-moment transport. *Astronomy and Astrophysics*, 356:559–569, April 2000. [citado en p. 4]

[65] B. W. Mihalas y D. Mihalas. *Foundations of Radiation Hydrodynamics*. Courier Dover Publications, 1984. [citado en p. 5]

[66] L. Herrera et al. Relativistic gravitational collapse in noncomoving coordinates: The post-quasistatic approximation. *Phys. Rev. D*, 65(10):104004, Apr 2002. [citado en p. 6]

[67] M. Cosenza et al. Some models of anisotropic spheres in general relativity. *J. Math. Phys*, 22:118–125, 1980. [citado en p. 6]

BiblioWeb

- Apache Maven: http://maven.apache.org/
- Apache Tomcat: http://tomcat.apache.org/
- Condor: http://www.cs.wisc.edu/condor/
- EGEE: http://public.eu-egee.org/
- GGF: http://www.ggf.org
- GPIR: http://www.tacc.utexas.edu/projects/gpir.php
- GLite: http://glite.web.cern.ch/glite/
- Globus Toolkit 2 (GT2): http://www.globus.org/toolkit/downloads/2.4.3/
- Globus Toolkit 3 (GT3): http://www.globus.org/toolkit/downloads/3.0.2/
- Globus Toolkit 4 (GT4): http://www.globus.org/toolkit/downloads/4.0.2/
- GridFTP: http://www.globus.org/grid_software/data/gridftp.php
- GridPort: http://www.gridport.net
- Gridsphere: http://www.gridsphere.org/
- GPDK: http://doesciencegrid.org/projects/GPDK/
- IBM Websphere: http://www.ibm.com/websphere
- J2EE: http://java.sun.com/javaee/
- JDL: http://www.ogf.org/documents/GFD.56.pdf
- JSRs: http://jcp.org/en/jsr/all
- JSR 168: http://jcp.org/en/jsr/detail?id=168
- JSR 286: http://jcp.org/en/jsr/detail?id=286
- LSF: http://www.platform.com/Products/platform-lsf-family

- MPICH-G2: http://www3.niu.edu/mpi/
- MyProxy: http://grid.ncsa.uiuc.edu/myproxy/
- .NET: http://www.microsoft.com/net/default.mspx
- Nimrod-G http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm
- OASIS: http://www.oasis-open.org/
- OGCE: http://www.collab-ogce.org/ogce/index.php/Main_Page
- OGSA: http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf
- OGSA-DAI: http://www.ogsadai.org.uk/
- OGSI: http://www-128.ibm.com/developerworks/grid/library/gr-ogsi/ http://xml.coverpages.org/OGSI-SpecificationV110.pdf
- Portable Batch System (PBS): http://www.pbsgridworks.com/Default.aspx
- Proyecto Sakai: http://www.proyectosakai.org/
- SOA: http://en.wikipedia.org/wiki/Service-Oriented_Architecture
- SOAP: http://en.wikipedia.org/wiki/SOAP
- Storage Resource Broker (SRB): http://www.sdsc.edu/srb/index.php/Main_Page
- Sun Grid Engine (SGE): http://gridengine.sunsource.net/
- UDDI: http://www.uddi.org/
- Unicore: http://www.unicore.eu/
- WS: http://en.wikipedia.org/wiki/Web_services
- WS-Addressing: http://www-128.ibm.com/developerworks/library/specification/ws-add/
- WS-I: http://www.ws-i.org/
- WS-Notification: http://www-128.ibm.com/developerworks/library/specification/ws-notification/
- WS-Reliability: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm
- WS-Security: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- WSDL: http://www.w3.org/TR/wsdl
- WSRF: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
- WSRP: http://en.wikipedia.org/wiki/WSRP
- WSRP 1.0: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
- WSRP 2.0: http://www.oasis-open.org/committees/download.php/18617/

Φ

Anexos

Anexo A

Manual de usuario del *portlet* CHOQUE

Este portlet pertenece fue diseñado para ser usado en el área de la astrofísica relativista. En este caso se trabaja con un conjunto de subrutinas hechas en fortran siguiendo el modelo de Rueda y Núñez [62]. Estas subrutinas permiten modelar la evolución de una onda de choque en una esfera relativista autogravitante en el contexto de una aproximación post-cuasiestática. El portlet proporciona una forma sencilla de generar archivos de entrada para las subrutinas y luego mandar a procesar estos mismos en un Grid para luego visualizar sus resultados.

A.0.1 *Software* necesario para la instalación del *portlet* CHOQUE

a) Navegador Web (*Web Browser*) que soporte JAVA y JavaScript.

b) GridPort.

c) Maven 1.0.2.

A.0.2 *Software* necesario para los recursos remotos

a) gLite.

b) Java j2SDK [1] instalado en todos los nodos.

c) Compilador fortran g95.

A.0.3 Instalación

a) Antes de compilar el portlet es necesario ajustar las variables del entorno:

a.1) CATALINA_HOME: Directorio donde se encuentra instalado tomcat.

a.2) MAVEN_HOME: Directorio donde se encuentra maven (colocarlo en el PATH).

a.3) project.properties: Ajustar los valores de catalina.home en este archivo y de los repositorios de Maven.

a.4) portlet.xml: Ajustar las direcciones absolutas para las variables proxy.location, nsHost.location, lbHost.location, resources, db.user, dp.password (src/webapp/WEB-INF) dependiendo del lugar de instalación del *portlet*.

b) Luego se debe instalar el *portlet* en el portal (*./compilar*).

c) Abrir el navegador Web y colocar la dirección donde esté corriendo GridPort a fin de agregar el *portlet* a un grupo o crear un grupo para el *portlet*.

A.0.4 Uso del *portlet* CHOQUE

Actualmente es posible conectarse al *portlet* CHOQUE, accediendo a la dirección https://grid012.cecalc.ula.ve:8080/gridsphere/.

Luego de acceder al portal se hace click en la pestaña de CHOQUE, y se verá la página de trabajos (start.jsp). En esta página se puede crear nuevos trabajos o ver los trabajos creados por el usuario anteriormente, comentarios y estado del trabajo. Además tiene la opción de ver los archivos creados en trabajos anteriores y borrarlos.

Una vez que se presione el botón *Create New Session* aparecerá la página de creación de trabajos (CHOQUE.jsp) ver Figura A.1. En esta página se ingresan los datos para un trabajo nuevo,

[1] Java http://developers.sun.com/downloads/

datos como: Nombre del trabajo, algún comentario del trabajo, y los valores de los parámetros iniciales: $A(0)$ (radio inicial de la configuración), $\Omega(0)$ (velocidad inicial de la frontera), $c(0)$ (la posición inicial de la onda de choque), N (parámetro de fuerza del choque), ξ_I (parámetro de anisotropía en el núcleo), ξ_{II} (parámetro de anisotropía en el manto), f_I (factor variable de Eddington en el núcleo), f_{II} (factor variable de Eddignton en el manto), $M(0)$ (masa inicial de la configuración), M_f (masa final de la configuración), τ, s (ancho de la función Gaussiana que se emplea para calcular la luminosidad), t_0 (inicio de la evolución) y t_f (fin de la evolución).

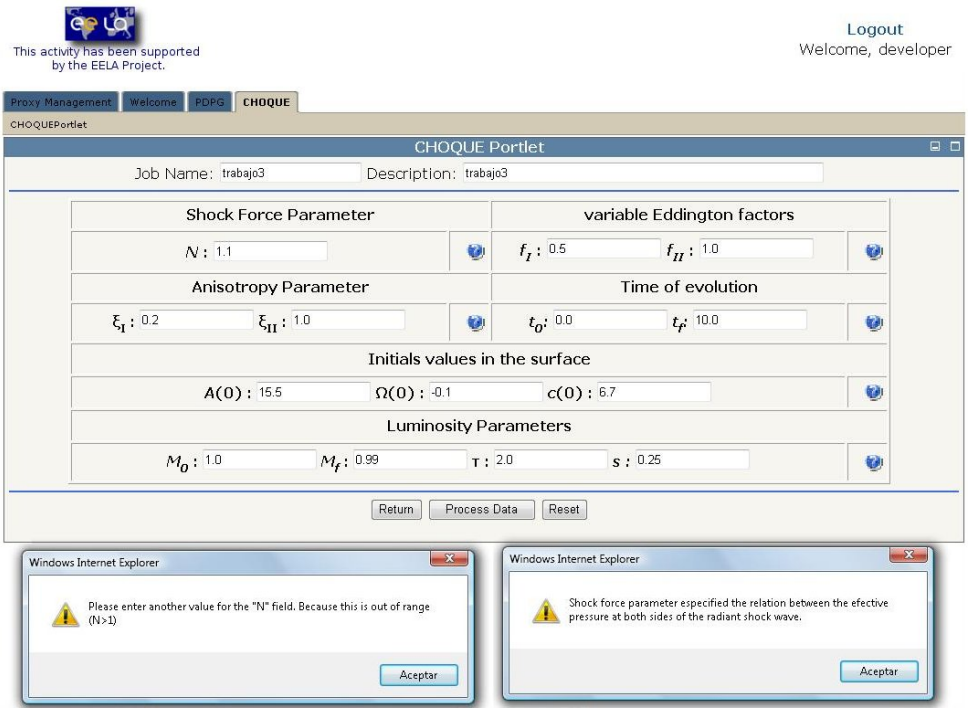


Figura A.1: Pantalla de captura de datos.

Si se desea enviar el trabajo se debe presionar el botón *Process Data*. Luego de mandar a procesar nuestro trabajo, volvemos a la página inicial (ver Figura A.2), pero con la diferencia que ahora aparece el trabajo que se acaba de enviar. La columna "status" se refiere al estado del proceso. Si el usuario desea actualizar el estado debe hacer click en el link *CHOQUEPortlet*, de lo contrario el estado se actualizará cada 180 segundos.

Una vez que el estado del trabajo sea DONE se puede proceder a realizar la visualización

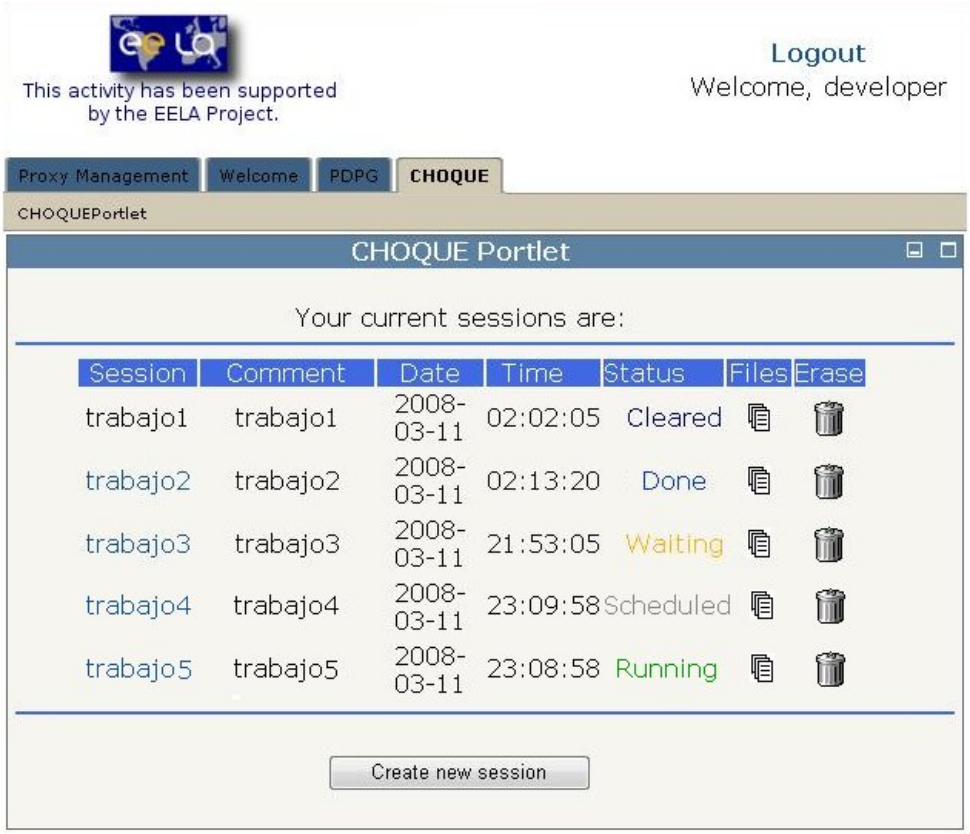


Figura A.2: Pantalla de trabajo en proceso.

de los datos de salida originados, para ello se debe hacer click en el ícono *Files*. Aparecerá la página de archivos (viewer.jsp), aquí se puede seleccionar si se desea realizar la visualización de los archivos de salida mediante gráficos de línea (opción *Line Graphics*) o animación por color (opción *Color Animation*). Además, se pueden descargar los archivos de salida del *portlet* desde el servidor al computador oprimiendo el botón *Get these files* (ver Figura A.3).

Al oprimir el botón *View* si la opción seleccionada fue *graphics* aparecerá la página de visualización de datos mediante gráficos de líneas (graphviewer.jsp), en ella el usuario puede elegir que variable desea visualizar. En caso de que la opción fuese *animation* (animviewer.jsp) aparecerá la página de visualización mediante mapeo por color, en la cual el usuario puede seleccionar la variable física que desee visualizar y que tipo de mapa de color. Ver Figuras **??** y **??**

Al presionar el botón *Get these files* aparecerá la pantalla de descarga de archivos (pack.jsp).

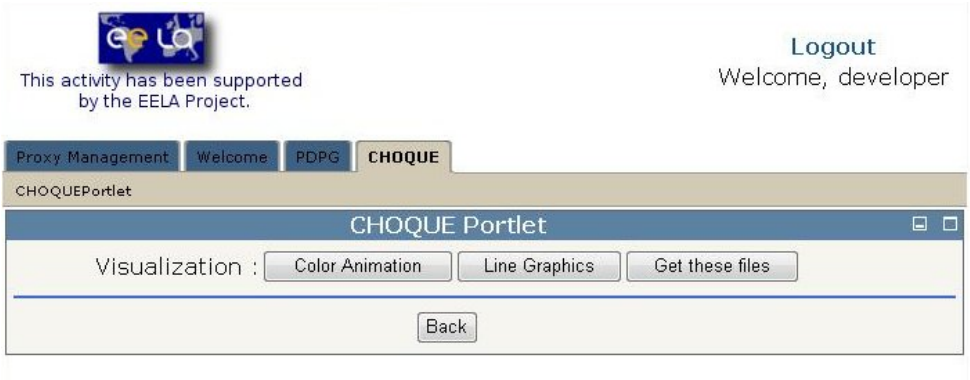
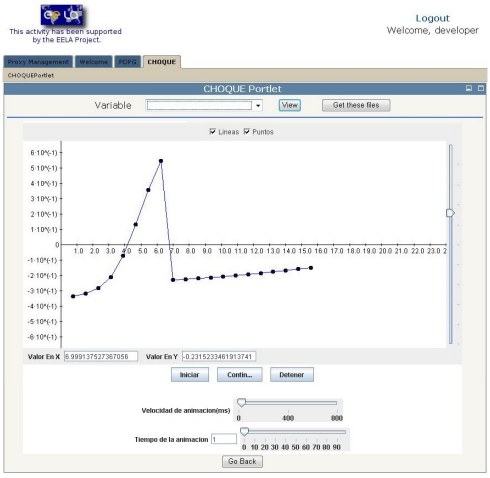
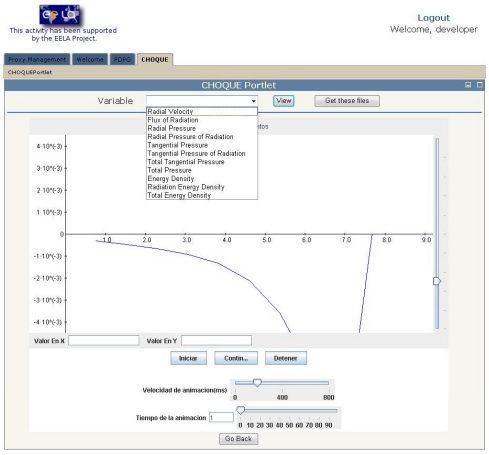


Figura A.3: Pantalla de archivos de salida.



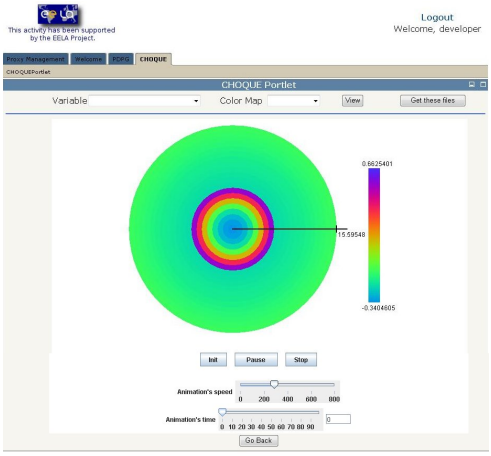
(a) Visualización mediante gráfico de líneas del flujo de radiación



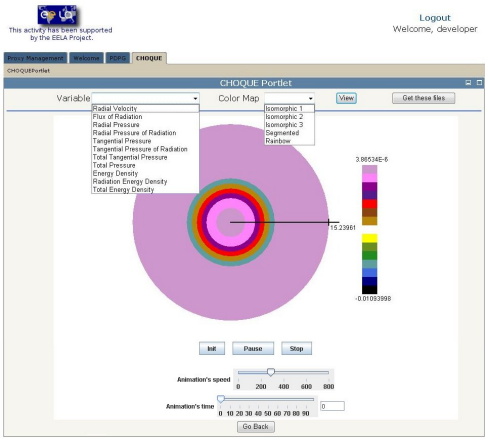
(b) Visualización mediante gráfico de líneas de la velocidad radial

Se descarga a nuestro computador un archivo (tar) que contiene los archivos de entrada y salida de CHOQUE.

Finalmente si se desea borrar cualquier trabajo solo se debe presionar el ícono de la papelera correspondiente al trabajo que se desea suprimir. Un mensaje de confirmación aparecerá, (delete.jsp).



(a) Visualización empleando mapa de color arcoiris de la velocidad radial.



(b) Visualización empleando mapa de color segmentado de la velocidad radial.

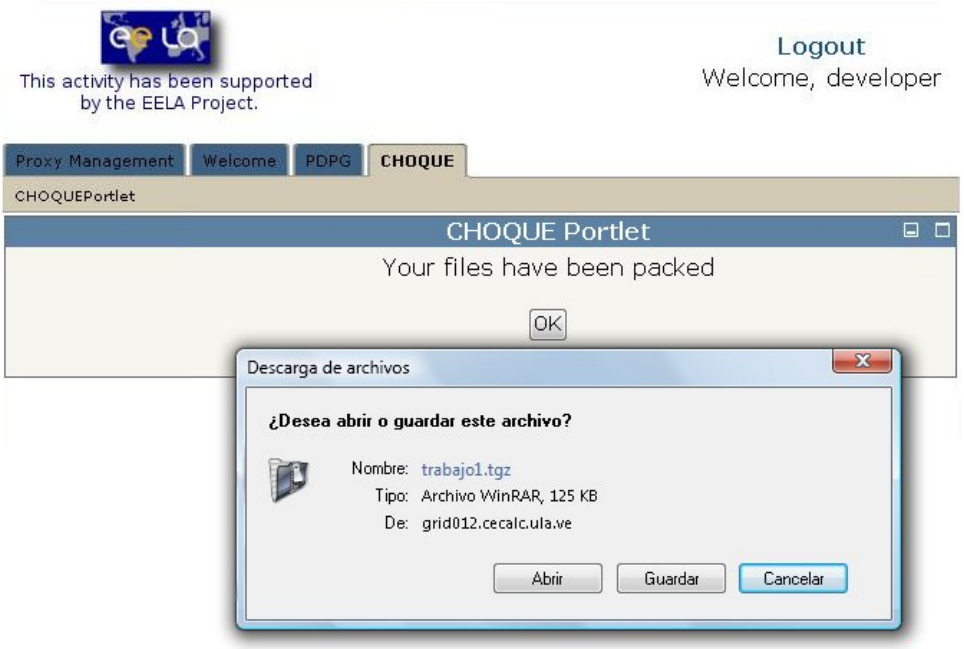
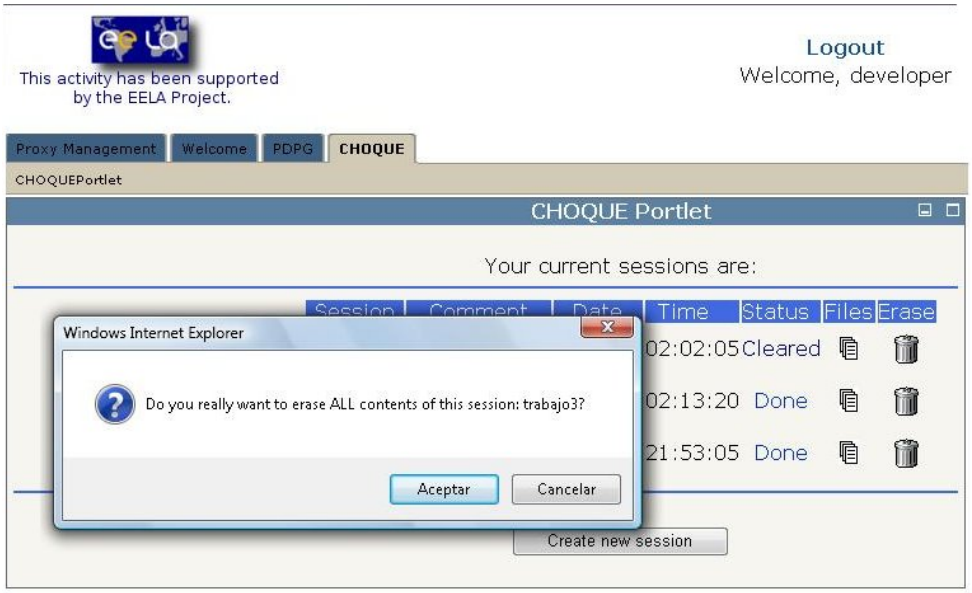
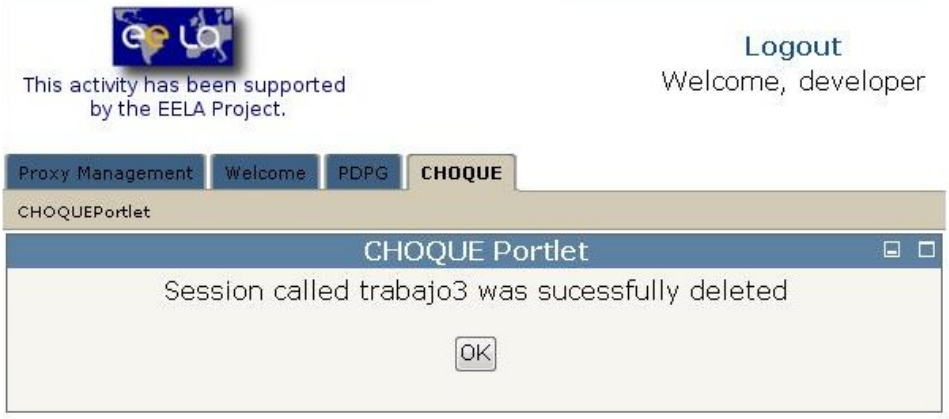


Figura A.4: Pantalla de descarga de archivos.



(a) Confirmación de eliminación de archivos.



(b) Mensaje de confirmación.

Lista de Figuras