Proyecto Hevelius

Empresa DevNull

Plan de Proyecto

Carlos Guajardo Miranda
Jefe de Proyecto
cguajard@alumnos.inf.utfsm.cl
cel. 09-95046118

Marina Pilar Daza
Miembro del Equipo
mpilar@alumnos.inf.utfsm.cl
cel. 09-84085407

Esteban Espinoza Martínez
Miembro del Equipo
eespinoz@alumnos.inf.utfsm.cl
cel. 09-85596939

Tomás Staig Fernández
Miembro del Equipo
tstaig@alumnos.inf.utfsm.cl
cel. 09-97615666

25 de mayo de 2007

Índice

1. Introducción

En el presente documento se da a conocer el plan del Proyecto Hevelius, el cual tiene por objetivo mostrar el estudio realizado por la Empresa DevNull. En este estudio se contemplan las soluciones al desafío planteado por el grupo ACS-UTFSM, así como la concretitud de los requerimientos de éstos.

Se advierte que el carácter técnico, desarrollado en algunos items del documento, está dirigido a discusiones concretas y son comprensibles por el grupo ACS-UTFSM y por personas vinculadas con el tema.

El documento se estructura de la siguiente forma:

- **Solución conceptual:** En la cual se describe el problema actual, se bosquejan posibles soluciones y, finalmente, se escoge la mejor alternativa.
- **Técnicas y herramientas de desarrollo:** Esto es, definir los elementos técnicos con que se construirá la solución y la plantilla de trabajo.
- **Gestión de riesgos:** En esta sección se identificarán, clasificarán y se propondrán estrategias de mitigación y contingencia para los peligros ocurrentes del proyecto.
- **Implementación:** En la cual se explica la incorporación del nuevo sistema en las instalaciones del cliente.
- **Planificación de actividades:** Esto significa describir el proceso que se seguirá para llevar a cabo la solución propuesta.

En el contexto más general, el desafío planteado por el grupo ACS-UTFSM, es crear un sistema de control de telescipios capaz de poder manejar cualquier telescopio que se conecte a través de las diferentes coordenadas utilizadas en el mundo astronómico.

Lo que se espera crear consiste en una interfaz gráfica que permita operar al algún telescopio de manera remota, lograr un control en tiempo real y generar registros para posteriores análisis de los datos recibidos por el telescopio.

La mejor solución ideada, es el diseño y construcción de un producto de software diseñado para solventar los problemas actuales y cumplir con los requerimientos del cliente.

Los riesgos, que se detallan en el capítulo 4, corresponden a los peligros identificados que pueden aparecer

durante el desarrollo del proyecto, entre ellos se destaca: la poca escalabilidad del sistema de control y el no cumplimiento de los estándares ALMA.

2. Solución Conceptual

2.1. Diagnóstico de la situación actual

2.1.1. Situación Actual

En la actualidad cada telescopio se controla con un software diferente desde el lugar en que se encuentra el mismo y, en general, estos programas de control son bastante complejos de usar.

Esto genera un aumento en los costos de observación, puesto que es necesario utilizar buena parte del tiempo en aprender y no olvidar cómo se usan estos programas. Y si se cambia el telescopio, es necesario aprender cómo se utiliza, perdiendo gran parte de lo aprendido con el programa anterior. Otro factor que aumenta los costos de observación es que los programas actuales de control de telescopios carecen de funcionalidad de control a distancia, lo que obliga a traer astrónomos de todo el mundo para que realicen sus investigaciones.

2.1.2. Identificación de problemas y deficiencias

Unicidad de Software: En la actualidad existen diversos tipos de telescopios, los cuales están implementados de manera diferente dependiendo de su diseñador o de dónde fueron creados. Junto con esto, aparece el problema de que cada telescopio posee una aplicación diferente para su control, lo que obliga a los astrónomos, operadores de telescopios y aficionados a utilizar gran parte de su tiempo aprendiendo a ocupar los distintos softwares para cada uno de los equipos con los que van a trabajar.

Control: Actualmente el control de los telescopios se debe hacer de forma local, es decir, los operadores de telescopios y astrónomos deben estar en el observatorio para realizar sus investigaciones, pudiendo hacerse éste de forma remota, mejorando la situación para los astrónomos, especialmente para los que se encuentran lejos de los sitios de observación.

Dificultad de Uso: Muchos de los programas utilizados actualmente para control de telescopios son bastante complicados de usar, obligando a gastar una considerable cantidad de tiempo aprendiendo a usarlos y, también, a usarlos frecuentemente para no olvidar cómo es que se hace.

Seguridad del telescopio: Es importante que el telescopio tenga medios para protegerse de los distintos eventos que puedan ocurrir: luminosidad alta, clima inadecuado, entre otros.

2.2. Caracterización del cambio

2.2.1. Caracterícticas y Potencialidades deseadas

Características específicas deseadas para el producto.

- **Control por internet de telescopios:** Se quiere que el sistema pueda funcionar situado en cualquier parte del mundo permitiendo controlar algún telescopio que se encuentre en otro lugar geográfico.
- **Interfaz Gráfica:** El software de control de telescopios debe tener una interfaz agradable a los usuarios y permitir el acceso eficiente a las funcionalidades que se requieran, además, debe mostrar siempre en pantalla la información de mayor importancia.
- **Reproducción de lo que ve la cámara:** El sistema debe mostrar a donde apunta el telescopio en todo momento de observación, por medio de la cámara CCD.
- **Interacción con ACS:** Es necesario que el sistema interactúe con los telescopios por medio de ACS, de manera que éste sea el que se conecte directamente con los observatorios y telescopios.
- **Ajustar posición del telescopio bajo sistema de coordenadas ecuatoriales:** El sistema debe poder recibir las coordenadas que se quiere observar y convertirlas a las coordenadas que utiliza el telescopio para poder moverlo a esa dirección.
- **Mover el telescopio a la hora sideral:** El sistema debe tener la funcionalidad de seguir la posición que se está observando, ya que si no se hace, parecería que lo observado se ve desplazando.
- **Impedir observaciones a lugares con luminosidad lunar:** El sistema debe evitar que el telescopio apunte a direcciones con notoria luminosidad lunar, debido a que esta luminosidad puede dañar severamente los lentes del telescopio.
- **Mostrar modelo visual del telescopio:** Debido a que el telescopio se quiere manipular de forma remota, es necesario otorgar alguna forma que permita ver a la persona que lo esté operando, en qué estado se encuentra. Para esto, el sistema debe tener un modelo visual que se comporte de la misma forma que lo hace el telescopio real.
- **Ajuste manual del telescopio:** El sistema debe permitir controlar el telescopio manualmente para permitir ajustes menores, que ayuden a corregir errores en la dirección que se observa, que pudieran ocurrir por factores externos, como es la deflexión por el peso propio del telescopio en algunas posiciones.
- **Detener de forma inmediata el telescopio en caso de emergencia:** El sistema tiene que tener una opción de emergencia para detener el telescopio de forma inmediata para evitar cualquier daño que se crea que pueda ocurrir. Por ejemplo, daño por alguna variación en las condiciones climáticas.
- **Controlar acceso a la aplicación (Sesiones):** El sistema debe tener acceso para los distintos usuarios, de manera que cada uno tenga su propia estadística de lo observado.
- **Guardar coordenadas de observación realizadas:** El sistema debe guardar registro de las coordenadas observadas por cada usuario del sistema. De esta forma ayuda a que se puedan repetir observaciones y a realizar estudios sobre éstas.

Relación de las características con los problemas identificados.

- El control por internet va a ayudar a solucionar el problema de tener que estar en el lugar de observación al momento de controlar al telescopio.
- La interfaz gráfica va a ayudar a disminuir la dificultad de uso, ocultando información que no sea requerida en todo momento, pero permitiendo verla de manera sencilla e intuitiva.
- La reproducción de lo que está viendo el telescopio es de gran utilidad para la experiencia remota, debido a que sino hiciera esto, no se podría ver lo que está viendo el telescopio, hasta que se enviara algún informe a quien controlaba el telescopio.
- La interacción con ACS es una de las características principales para el control genérico de telescopios y el control de telescopios por medio de internet, pues es esta plataforma la que permite la comunicación con los telescopios en los diferentes observatorios del mundo.
- Mover el telescopio a la hora sideral reduce la dificultad de uso para el seguimiento de la observación de algún objeto, puesto que nos permite ver en todo momento al objeto deseado, sin necesidad de realizar tareas adicionales.
- Al impedir que el telescopio apunte a lugares con luminosidad lunar se reduce la dificultad de uso, puesto que no es necesario estar preguntandose todo el tiempo si el telescopio va a apuntar a lugares potencialmente dañinos para el mismo. Además, aumenta la seguridad del telescopio puesto que lo proteje de la luz lunar, uno de los factores más comunes que dañan al telescopio.
- El modelo visual soluciona un aspecto muy importante de la dificultad de uso para el control a través de internet, ya que con este se puede saber en todo momento hacia dónde está apuntando físicamente el telescopio, dándonos un apoyo gráfico de lo que estamos haciendo. De la misma forma, también ayuda a los que operan el telescopio de forma local, aunqiue ellos podrían verlo directamente, puede ser más cómodo verlo en la misma pantalla que están trabajando.
- El ajuste manual ayuda a disminuir la dificultad de uso del sistema, puesto que con este, no es necesario intuir una dirección parecida a la que estamos observando de manera que se vea lo que debiera, sino que simplemente lo movemos manualmente hasta donde debiera estar.
- Al dar la posibilidad de detener manualmente al telescopio, aumentamos en gran medida su seguridad, puesto que mediante esta opción, podemos protegerlo de factores que no esperabamos, como son las variaciones inesperadas en el clima.
- El guardar coordenadas de observación realizadas por sesión facilita la dificultad de uso del sistema, ya que para gente no muy experimentada en el tema, permite repetir las observaciones hechas otros días.

2.2.2. Restricciones

- **Económicas:** El software no presenta restricciones económicas, puesto que tanto el sistema operativo, como las herramientas de desarrollo que se van a utilizar, son gratuitas. Por otro lado, los componentes de hardware como son el telescopio para pruebas y la cámara CCD si tienen un costo, pero en este caso serán facilitados por el cliente. Es por esto, que no vemos restricciones económicas peligrosas.

- **Sociales y Culturales:** Los usuarios actuales de los programas que controlan telescopios han tenido que usar diferentes aplicaciones para distintos telescopios a lo largo del tiempo que han dedicado a esto, prefiriendo quizás, el que usan actualmente, ya sea por costumbre o por gusto personal. Esto puede dificultar que se acostumbren a usar el sistema propuesto, pero se espera que el sistema final sea intuitivo y amigable, de manera que esto no debiera suceder.
- **Tecnológicas:** En el aspecto tecnológico es importante destacar que las pruebas iniciales no necesariamente se harán con un telescopio de observatorio, en estos casos se utilizará para las pruebas telescopios para aficionados, puesto que los costos de observación son elevados.

2.3. Análisis de las alternativas de solución

2.3.1. Alternativa 1: Desarrollo de Software basado en ACS

Desarrollo de un producto de software basado en la plataforma ACS que sea genérico, es decir, que nos permita controlar cualquier telescopio por medio del mismo programa, sin la necesidad de tener un programa diferente para cada telescopio.

Se utiliza un computador como estación de trabajo de quien opere el telescopio, en donde todo el control se realizará por medio de una interfaz gráfica. Este computador requerirá tener acceso a internet para poder obtener componentes desde ACS y para comunicarse con el telescopio que se quiera controlar.

La interfaz gráfica mostrará a quien opere el telescopio el estado actual del mismo, pudiendo verse lo que está observando el telescopio por medio de la cámara CCD y la disposición física en que se encuentra el telescopio, por medio del modelo hecho en OpenGL del mismo.

Es importante notar que al utilizar la plataforma ACS para la distribución de componentes de software, se podrían usar componentes realizados por otras personas, asi como realizar cambios en componentes que utiliza nuestro software, obteniendo un producto de alta modularidad, enfocado al trabajo por componentes.

2.3.2. Alternativa 2: YYY+1

2.4. Solución recomendada

La mejor solución que encontramos es la alternativa 1, en la cual se piensa construir un producto de software genérico basado en ACS, el cual se acopla bastante bien con los requerimientos del cliente.

Este producto deberá ser modular basado en componentes con el estilo que ACS nos impone, por lo que se tendrán algunos componentes sobre esta plataforma, mientras que otros irán junto con el software principal. Es importante hacer la separación de las capas de comunicación, interfaz gráfica y software, para simplificar la mantención del producto al tener las funcionalidades separadas.

Se eligió esta alternativa, porque en la actualidad no hay productos que controlen telescopios de forma genérica, y esto es algo muy importante para enfocar los esfuerzos de observación en las observaciones mismas y no en aprender a utilizar los programas asociados al control de telescopios.

3. Técnicas y Herramientas de desarrollo

3.1. Modelo de desarrollo

Las características más importantes del proyecto Hevelius con respecto a la elección de un modelo de desarrollo se expone en lo siguiente. Primero se consideran las propiedades del producto:

- **Complejidad del Proyecto:** una estimación informal del proyecto muestra una complejidad media, la cual permite la finalización del proyecto dentro del plazo predeterminado;
- **Solidez de los Requerimientos:** cómo el producto a desarrollar está inserto en un proyecto de investigación, es posible que los requerimientos capturados durante el análisis sean completos en su generalidad, sujetos a pocos cambios.
- **Innovación del Producto:** en vista de que la principal característica del proyecto es la búsqueda de la generalidad en el control de telescopios, existen errores que durante el desarrollo se descubrirán, lo que nesecitará cambios en el código implementados o en el diseño del software.

Además, el ambiente del desarrollo tiene las siguientes características:

- **Tamaño del Equipo:** cuatro personas durante la planificación y el desarrollo del proyecto.
- **Recursos Disponibles:** nuestro cliente nos provee de un lugar físico, computadores y un telescopio para el desarrollo de nuestro proyecto.
- **Tratamiento de los miembros del equipo entre sí:** informal, ya que existen lazos de amistad anteriores a la formación del grupo de trabajo.
- **Proyección en el tiempo:** se cuenta con un plazo determinado, debido a que el proyecto pretende ser parte de la Feria de Software 2007, de aproximadamente cinco meses a partir de la fecha de entrega de este informe.

Teniendo en cuenta el conjunto de características del proyecto, se decide usar un método ágil de desarrollo para la mayoría de las tareas; específicamente se prevee el uso del *Modelo Evolutivo* (también llamado *diseño continuo* o *diseño incremental*) en el trabajo. Sin embargo, no es posible usar un método ágil de forma exclusiva, porque es necesario cumplir con las fechas de entregas intermedias predeterminadas por los ramos "Ingeniería de Software" y "Taller de desarrollo de Software". También será necesario contar al final del desarrollo con una documentación comprensible por el cliente[1] del diseño y del código del software, cosa que

[1] que en el caso del proyecto presente está compuesto de ingenieros informáticos y electrónicos

se considera más fácil usando algunos elementos de un proceso planificado. Por tanto, el uso parcial de un proceso planificado será inevitable.

Habiendo elegido este método de desarrollo, es necesario realizar las siguientes actividades durante el desarrollo para el modelo evolutivo:

1. **Pruebas:** los requerimientos (funcionales o no-funcionales) deben ser transformados en pruebas automatizables.
2. **Construir:** el código producido debe ser siempre en un estado que permite su compilación.
3. **Escuchar:** La comunicación a lo largo del desarrollo del proyecto con el cliente, debe ser fluida, esto para atender a las verdaderas necesidades y para solicitar sus comentarios sobre el estado del desarrollo, y posibles cambios en los requerimientos.
4. **Diseñar:** para facilitar el proceso de codificación y permitir el trabajo paralelo de grupos independientes, hay que contar con un diseño del sistema. Esta actividad hay que hacerla durante todo el tiempo del desarrollo, adaptándose a funcionalidad agregada.
5. **Comunicación dentro del equipo:** es importante para el uso de un método ágil que la comunicación entre los integrantes funcione muy bien, es decir, todos deben saber la mayoría del tiempo en qué están trabajando los demás y cuáles son los cambios realizados por ellos.

Para los puntos 3 y 5 se necesitan además muchas versiones intermedias.

Para el componente planificado, hay que hacer un plan inicial de recursos y tiempo necesitado, para lo que sirve este documento, y controlar el progreso actual del proyecto, para lo que sirven los *fichas de estado*. Además es necesario documentar el diseño actual del programa.

3.2. Herramientas y técnicas de soporte para el desarrollo

3.2.1. Técnicas a utilizar en el desarrollo del proyecto

Durante el desarrollo del proyecto se utilizará el método evolutivo ya que es necesario contar con una planificación a seguir en los meses de desarrollo. Además, es necesario una documentación del desarrollo del proyecto para futuras modificaciones.

Se optó por este método debido a la seguridad que da su planificación al cliente, ya que él puede *ver* en

que se está trabajando y cuales serán los próximos pasos a seguir.

Dentro de los lenguajes evaluados, se optó por la programación en C++ al ser este un potente lenguaje con una orientación a objetos que facilita el diseño gracias a las herramientas de planificación y modelamiento de UML. Todos los integrantes del grupo, cuentan con conocimientos en C y otros en C++. Para quienes no cuenta con los conocimientos en C, el paso a C++ es diminuto en comparación con otros lenguajes de programación.

Dentro de los lenguajes descartados, se encuentra Java. Esto se debe principalmente a que los requerimientos de hardware de su máquina virtual, en muchos cosas sobrepasa las características de los equipos en los que podremos desarrollar el proyecto. Esta situación a provocado que no exista una empatía por parte del grupo hacia Java, debido a la ralentización que produce en los equipos personales la realización de otras operaciones.

Es posible que se utilice alguna herramienta de diseño vectorial para la creación de las gráficas a utilizar en el software, pero aún no se ha descartado ni confirmado ninguna de ellas.

Para asegurar la calidad de software, se realizarán constantes revisiones del producto con el cliente, de forma que esto permita obtener una retroalimentación y así mejorar o cambiar pequeños defectos en corto tiempo.

Dentro de este mismo marco, utilizaremos la herramienta de control de versiones SVN para mantener un desarrollo ordenado, una coordinación entre los desarrolladores, saber cual de los desarrolladores realizó que cambio, etc. Además, se planea la utilización la herramienta bugtracker[2] Trac[3].

3.2.2. Herramientas o plataformas específicas a utilizar

La plataforma operacional de Hevelius está constituida por el Sistema Operativo Linux Fedora Core, debido a que es esta la distribución utilizada por nuestro cliente para desarrollar software.

Tal como se mencionaba antes, se utilizará UML como plataforma de modelamiento. De esta forma, se está evaluando que herramienta CASE nos será de mayor utilidad. Entre estas están Umbrello[4], Rational

[2] Sistema diseñado especialmente para manejar reportes de errores
[3] `http://www.edgewall.com/trac/`
[4] `http://uml.sourceforge.net/`

Rose[5] y DBDesigner 4[6]

3.3. Personal y capacitación del grupo de desarrollo

- Personal del equipo del proyecto.

  La empresa DevNull cuenta actualmente con cuatro miembros, de los cuales se hará una breve descripción de sus perfiles personales, paso primordial para conocer al equipo de trabajo:

  Marina Alejandra Pilar Daza .

  Alumna de cuarto año de ingeniería civil informática de la UTFSM, se caracteriza por ser una persona responsable y comprometida con los trabajos asignados. Tiene capacidad de liderazgo, a pesar que no siente mucha motivación por utilizarlo. Además, podemos rescatar su manera directa de decir las cosas, que hacen que exista una buena relación en el equipo de trabajo y, no existan malas interpretaciones de las cosas y el buen trabajo en equipo que realiza.

  Dentro de sus areas de interés , se encuentra la informática, la historia universal y lectura de cualquier tipo de libros; a pesar que aún no tiene un área especifica a la cual se va a dedicar, actualmente se centra en la área de Sistemas Computacionales, lo cual se puede aprecia al ver que trabaja en el laboratorio de Computación en el departamento de informática de la UTFSM, del cual ha adquirido experiencia tanto en el ámbito técnico como en el humano, al tener contacto con otras personas, ya sean con usuarios o el mismo trabajo en equipo que se tiene que llevar en un trabajo como ese.

  - Conocimientos Técnicos:
    - Administrador de Sistemas: Cuentas, Bases de Datos.
    - Sistemas Operativos: Microsoft Windows, Linux.
    - Lenguajes de Programación: C, C++, Basic, Visual Basic, Java, PHP, ASP, Javascript, Scheme, Prolog, Bash, Perl.

  Tomás Ignacio Staig Fernández .

  Persona metódica y analítica. Es una persona que se esfuerza por lograr que las cosas funcionen bien, para lo cual utiliza tanto los conocimientos que ha obtenido como soluciones que el mismo pueda ingeniar.

  Muestra interés principalmente por las áreas de Modelos y Métodos cuantitativos y por la de Sistemas Computacionales. Esto no quita que le guste desarrollar software, sino que remarca el hecho de que le gusta desarrollar para solucionar problemas.

  Posee un buen uso del lenguaje español oral y escrito, pero destaca por tener buen manejo del idioma inglés, tanto oral como escrito, estando certificado por la evaluación First Certificate in English de la Universidad de Cambridge.

  Sus hobbies son el volleyball, tennis y juegos de computador. Dentro de éstos destaca que se encuentra en la rama de Volleyball de la Universidad Técnica Federico Santa María.

  - Conocimientos Técnicos:

[5] `http://www-306.ibm.com/software/rational/`
[6] `http://fabforce.net/dbdesigner4/`

- Sistemas Operativos: Microsoft Windows, Linux.
- Lenguajes de Programación: C, C++, Visual Basic, Java, PHP, ASP, Javascript, Scheme, Prolog, Bash, Perl, HTML.
- Librerías: OpenGL.

Esteban Ignacio Espinoza Martínez .

Persona metódica y analítica. Actualmente se encuentra finalizando su formación académica, de la cual valora sobre todo, los conocimientos técnicos que le ha entregado su estudio.

Destaca en él su alto grado de responsabilidad y su facilidad de aprendizaje. Es capaz de aceptar desafíos y concretarlos en el tiempo acordado, demostrando su capacidad y compromiso al trabajar en equipo e individualmente.

Se identifica con el área de Desarrollo de Software, aunque su interés es la amplia área de la Computación e Informática.

Es fuertemente valorable sus conocimientos sobre variados lenguajes de programación y su habilidad para programar sobre estos, con los cuales ha llevado a la práctica algunos proyectos de desarrollo personal.

También destacan sus habilidades deportivas en atletismo y fútbol. Sus hobbies son tocar armónica, jugar rol y entretenerse con juegos de PC.

Posee dominio del idioma Inglés a nivel intermedio (técnico y escrito).

- Conocimientos Técnicos:
  - Administrador de Sistemas: Cuentas, Mail, Web, SVN, Bases de Datos.
  - Sistemas Operativos: Microsoft Windows, Linux (Fedora, Ubuntu, CentOS, Gentoo).
  - Lenguajes de Programación: C, C++, Basic, Visual Basic, Java, PHP, ASP, Javascript, Scheme, Prolog, Bash, Perl.
  - Librerías: SDL, GTK, QT, CUPS, STL.

Carlos Alberto Guajardo Miranda .

Es un alumno proveniente de la RWTH Aachen, Alemania, que este año ha llegado a la UTFSM a través de una beca de intercambio obtenida en el país del cual es originario, el cual, en sus ganas de trabajar e interactuar con sus nuevos compañeros chilenos, eligió la asignatura de Ingeniería de Software, en donde se integra a la empresa Taranis, equipo el cual lo acoge desde el primer momento.

Se acaracteriza por ser una persona que siempre trata de entender las cosas a través de la comunicación con otras personas, especialmente potenciado por su necesidad de adaptación con la cultura y el idioma de Chile, y su deseo de transmitir su forma de vida en su país de procedencia.

Es capaz de identificar problemas con rapidez y proponer soluciones, ya sean soluciones propias o buscando a la gente que más adecuada para resolverlos; además, suele hacer planes factibles y realizarlos, de mostrando efectividad en el trabajo. Por lo mismo es que también cumple con el trabajo prometido, por su consecuencia a la hora de adquirir compromisis, y conocer sus límites de trabajo.

Estas características personales, así como su desempeño académico le permitieron lograr una beca del servicio alemán de intercambio académico (DAAD) para sustentar su estadía de intercambio en la UTFSM. Sus áreas de interés en el campo informático son la fiabilidad y seguridad de sistemas, comunicación de datos, y el modelamiento matemático.

Fuera del campo informático, ha sido tenismesista por los últimos 17 años de su vida. Posee dominio de los idiomas Alemán, el cual es su lengua materna, Inglés a nivel avanzado y posee buen manejo del idioma Español.

- Conocimientos Técnicos:
  - Administrador de Sistemas: Cuentas, Mail, Web, SVN, Bases de Datos.
  - Sistemas Operativos: Microsoft Windows, Linux (Fedora, Ubuntu, CentOS, Gentoo).
  - Lenguajes de Programación: C, C++, Basic, Visual Basic, Java, PHP, ASP, Javascript, Scheme, Prolog, Bash, Perl.
  - Librerías: SDL, GTK, QT, CUPS, STL.

- Perfil del equipo ideal para el proyecto.

  Para la empresa DevNull, el equipo ideal requerido para este proyecto requiere que cumpla con las siguientes características:

  - Conocimientos de Astronomía: Esencial es el conocimiento del funcionamiento de robots, como también conocer la lógica de su funcionamiento interno de las piezas electrónicas y/o Hardware que posee y conocer en específico con qué tipo de robot el equipo se encuentra trabajando para el desarrollo del software.
  - Conocimientos sobre Sistemas: Se requiere conocer tanto el sistema desde el diseño interno del robot, como su funcionamiento autónomo. Conocer de su arquitectura también es fundamental para lograr entender su funcionamiento o nuevas funcionalidades que pueda ser agregadas en el robot.
  - Conocimientos de Lenguajes de Programación:
    - C++: En este lenguaje de programación orientado a objetos se basará la funcionalidad de nuestro software.
    - LaTeX: Este lenguaje será usado para generar la documentación que surge a través de los entregables o a partir de la propia investigación que surge durante el avance del trabajo.
  - Conocimientos de Software:
  - Conocimientos de Sistemas Operativos: Se requiere un conocimiento en específico del sistema operativo Ubuntu, el cual se encuentra actualmente en la version 5.10 y próximo a cambiar a la versión 6.06, debido a que es el sistema sobre el cual funciona el robot.

- Capacidad, experiencia y disponibilidad de cada miembro del equipo (disponible o buscado).

  Marina Pilar
  - Administrador de Sistemas: Cuentas, Mail, Web, SVN, Bases de Datos.
  - Sistemas Operativos: Microsoft Windows, Linux (Fedora, Ubuntu, CentOS, Gentoo).
  - Lenguajes de Programación: C, C++, Basic, Visual Basic, Java, PHP, ASP, Javascript, Scheme, Prolog, Bash, Perl.
  - Librerías: SDL, GTK, QT, CUPS, STL.
  - Otros conocimientos relacionados: Arquitectura de computadores, Redes de computadores, Física (Dinámica y Electromagnetismo), Robótica a nivel básico.

  Tomás Staig
  - Administración de Sistemas: Correo Electrónico, Servicios Web, Cuentas de Usuarios, Bases de Datos, DHCP, Firewall, LDAP.
  - Sistemas Operativos: Linux (Fedora Core, CentOS, Debian, Ubuntu, ArchLinux), Microsoft Windows 98/2000/Me/XP
  - Lenguajes de Programación: C, Java, Visual Basic, HTML, Javascript, ASP, PHP, SQL, Bash, Latex, Perl, Scheme, Prolog, Tck, Tk.

- Librerías: GTK, QT.
- Otros conocimientos relacionados: Arquitectura de computadores, Redes de computadores, Física (Dinámica), Robótica a nivel básico.

Esteban Espinoza
- Software: MS Office, OpenOffice, Rational Rose, SQL Server, Visual Studio 6.0, PostgreSQL, Eclipse, Klogic.
- Sistemas Operativos: MS Windows 95, 98, XP (Nivel avanzado), RedHat Linux, Fedora Core (Nivel Medio).
- Lenguajes de Programacion: C, C++, Pascal, Visual Basic, ASP, PHP, SQL, JavaScript, Cobol, Dbase, Clipper, Java, Perl, Scheme, Prolog.
- Otros conocimientos relacionados: Arquitectura de computadores, Redes de computadores, Física (Dinámica y Electromagnetismo), Robótica a nivel básico.

Carlos Guajardo
- Sistemas Operativos: MS Windows 95, 98, XP (Nivel Medio), RedHat Linux, Fedora Core, Ubuntu (Nivel Medio).
- Lenguajes de Programacion: C, C++, Pascal, Visual Basic, ASP, PHP, SQL, JavaScript, Java, Prolog, entre otros.
- Otros conocimientos relacionados: Arquitectura de computadores, Redes de computadores, Física (Dinámica y Electromagnetismo), Robótica a nivel básico.

A partir de los datos tabulados, se puede decir que la mayoría de los recursos técnicos requeridos por el equipo son cubiertos.

- Análisis de insuficiencias de los miembros del equipo, si las hubiere.

  A partir de lo anterior, las insufiencias que se deberían cubrir son los siguientes:

  - Falta de capacitación en astronomía y acs: En conjunto los miembros del equipo sólo mantienen nociones básicas del funcionamiento de un robot, por lo que limita el trabajo por el poco entendimiento técnico de éste, lo que también trae problemas de comunicación con el cliente.

- Programa de mejoramiento (capacitación, mentoría, incorporación de nuevos miembros...), indicando específicamente acciones a realizar y plazos.

  Para el mejoramiento de las insuficiencias detectadas, el equipo recurrirá al siguiente plan de accion:

  - Falta de capacitación en robótica: Para mejorar esta insuficiencia, el equipo ya tomó su primer plan de acción, que fue la asistencia a un Seminario de Robótica dictado por la empresa Austral technologies (Austec), con la cual trabajan en el proyecto de Software, para conocer más de cerca el mundo de la robótica, con un punto a favor que al tratarse de los mismos clientes con que el equipo trabaja, el enfoque del Seminario se basó en gran parte al tipo de trabajo que ellos realizan. El siguiente plan de acción es asistir con el cliente en un plazo regular de una vez a la semana para recibir mayores instrucciones respecto a robótica, para poder así cada vez más entender las propias necesidades que requieren.
  - Retiro de un miembro del equipo. Para la mejora de esta insuficiencia, se proponen dos planes de acción:
    - Redistribución del trabajo inicial de cinco personas en cuatro partes
    - Buscar una persona externa que cumpla con el rol del integrante

Para llevar a cabo alguno de estos planes, dependemos del trabajo que se prosiga con el tiempo, ya que a partir de este veremos en que temas aquel integrante a resultado especialista e imprescindible, para la búsqueda de un nuevo integrante durante el período de vacaciones, luego de un análisis de trabajo del integrante retirado, como también puede darse el caso de que su trabajo sea perfectamente repartible entre los integrantes que se quedan. Como la segunda opción resulta ser, a vista del equipo, la que menos compromete factores externos o que escapen de su alcance, la lógica de trabajo será repartir tareas de distinto tipo a distintos integrantes del grupo, para que así se especialice en varias áreas de trabajo y así evitar problemas de desconocimiento de los temas por falta de uno de los integrantes faltantes.

4. Gestión de Riesgos

4.1. Análisis de riesgos

- Riesgos de Negocio:
  - Poca escalabilidad de Hevelius con respecto a telescopios profesionales.

    Un gran riesgo es que una vez terminado el proyecto, al intentar ser probado en un telescopio profesional, no sea lo suficientemente escalable y termine por no realizar un funcionamiento adecuado.
- Riesgos del Proyecto:
  - Ausencia de algún integrante del proyecto

    La ausencia de algún integrante del equipo de trabajo implicaría un mayos esfuerzo por parte del resto del equipo a demás de incrementar el tiempo de desarrollo del proyecto, lo cual podría traer serios problemas.
  - Incumplimiento con fechas.

    Todos los proyectos tienen una fecha limite para cada etapa y sus respectivas entregas. Un riesgo importante es el incumplimiento de las fechas finales, las cuales corresponden a la presentación del proyecto en La Feria de Software.
  - Telescopio Amateur no disponible

    Para la presentación de la feria, Hevelius será probado en un telescopio amateur el cual es propiedad de ACS-UTFSM Group, por lo que la disponibilidad de este se puede ver afectada para dicha presentación.
  - Estimación del proyecto errónea

    Una mala estimación del proyecto, en cuanto a complejidad y desarrollo implicaría un gran riesgo, puesto que podría darse el caso en que no se logre finalizar el proyecto dentro de la fecha estimada.
  - Falta de conocimiento en cuanto al área de desarrollo

    Hevelius al tratarse de un proyecto que esta en el área de la astronomía, obliga necesariamente al equipo de trabajo familiarizarse con términos e información propia de dicha área, lo cual implica una capacitación para el equipo de manera de poder visualizar de mejor manera el problema y poder desarrollar una mejor solución, a demás de permitir una mejor obtención de información de parte del cliente.
  - No cumplimiento con los estándares del proyecto ALMA-CONICYT

    Hevelius al formar parte del proyecto ALMA-CONICYT debe cumplir ciertos estándares, por lo que algún tipo de incumplimiento produciría conflictos en cuanto a utilización y una mala evaluación del proyecto por parte de nuestro cliente.
- Riesgos Técnicos:
  - Mala elección en cuanto a lenguajes de programación.

    Una mala elección del lenguaje de programación implicaría un gran riesgo, puesto que produciria un mayor esfuerzo en la creación de funciones propias del proyecto, lo cual puede retrasar el desarrollo de manera exponencial si no se hace adecuadamente.

- Compleja interfaz gráfica para el usuario.
  Hevelius entre otros objetivos busca ser una herramienta amistosa para los operadores de telescopio, por lo tanto el tener una interfaz muy engorrosa volvería la utilización del proyecto muy compleja y poco entendible, para esto se deben utilizar herramientas y técnicas que permitan un desarrollo de interfaz amigable.
- Pocas pruebas realizadas al proyecto.
  Una herramienta importante en el desarrollo de software es la fase de pruebas, una vaga fase de pruebas al proyecto puede significar errores que no son identificados en la etapa correcta y que conlleva a errores en futuras etapas lo que implica un cambio en la estructura del proyecto, los cuales son mas costosos a medida de que el proyecto avanza.

Nombre del Riesgo	Ocurrencia	Impacto
Poca escalabilidad de Hevelius con respecto a telescopios profesionales.	Muy Alto	Muy Alto
Ausencia de algún integrante del proyecto	Baja	Muy Alto
Incumplimiento con fechas	Media	Alto
Telescopio amateur no disponible	Baja	Muy Alto
Estimación errónea del proyecto	Media	Muy Alto
Falta de conocimiento en cuanto al área de desarrollo	Media	Alto
No cumplimiento con los estándares del proyecto ALMA-CONICYT	Baja	Alto
Mala elección en cuanto a lenguajes de programación	Baja	Alto
Compleja interfaz grafica para el usuario	Alta	Muy Alto
Pocas pruebas realizadas al proyecto	Media	Alto

Para priorizar los riesgos es necesario evaluar tanto su ocurrencia como el impacto que provocaría en el proyecto. Se debe crear un consenso en el equipo de trabajo para poder tomar la decisión correcta.

1. Poca escalabilidad de Hevelius con respecto a telescopios profesionales.
2. Compleja interfaz grafica para el usuario.
3. Estimación del proyecto errónea.
4. Falta de conocimiento en cuanto al área de desarrollo.
5. Pocas pruebas realizadas al proyecto.
6. No cumplimiento con las fechas.
7. Mala elección en cuanto a lenguajes de programación.
8. Telescopio amateur no disponible.
9. Ausencia de algún integrante del proyecto.
10. No cumplimiento con los estándares del proyecto ALMA-CONICYT.

4.2. Preparación para control de riesgos

ID 6	No cumplimiento con las fechas
Prioridad: 6	Clase: Riesgos del Proyecto
Probabilidad: Media	Impacto: Alto
Descripción: Incumplimiento de las fechas de entrega que están descritas en el plan de proyecto y las dispuestas en la feria de software.	
Período: Todo el proyecto	Estado: Latente
Contexto: Durante el desarrollo del proyecto, en el plan de proyecto se establecieron ciertas fechas, en las cuales cada persona se comprometió a cumplir con cierta partes del proyecto,las cuales son especificadas en la carta Gantt.	
Plan de Mitigación: Para poder mitigar este riesgo se realizará: ▪ Cumplir con las planificaciones establecidas. ▪ Análisis periódico del trabajo de cada integrante, para dimensionar bien el trabajo e involucrar a más personas si es necesario.	
Plan de contingencia: Si el plazo para entregar una tarea esta por cumplirse y con las horas que quedan según la planificación no alcanzan, entonces existen dos formas de abarcarlo, primero es injectando horas de trabajo para esa tarea. Y una segunda opción es hacer un replanteamento de la planificación.	
Resolución: Cuando la tarea es terminada en el plazo establecido.	

Cuadro 1: Hoja Control de Riesgo: ID6

ID 7	Mala elección en cuanto a lenguajes de programación
Prioridad: 7	Clase: Riesgos Técnicos
Probabilidad: Bajo	Impacto: Alto
Descripción: El problema de escoger un lenguaje de programación que provea las características necesarias para abordar el proyecto en su cabalidad, que sea del agrado de la plantilla de trabajo, que exista una documentación y plataforma asequible para su desarrollo es un aspecto crucial para concretar la solución.	
Período: Etapa de construcciíon	Estado: Latente
Contexto: La amplia gama de lenguajes de programación que existen, para distintas plataformas y modelamientos, con distintos paradigmas y características, proveen muchas alternativas tentadoras en el momento de concretar una solución. Ahora bien, escoger el o los lenguajes que provean todas las características y funcionalidades que se requieren para desarrollar el proyecto es un tema que requiere un estudio detallado.	
Plan de Mitigación: Para poder mitigar este riesgo se realizará una previa investigación que permita definir el o los lenguajes adecuados para la construcción. Dentro de los requerimientos de nuestro cliente, los lenguajes que podemos utilizar para desarrollar nuestro software son C++, Java y Phyton. Lo que significa evaluar las característica de ellos contrastandos con la funcionalidad requerida por desarrollar, la que sea de mayor manejo para los integrantes del proyecto y exista mayor documentación.	
Plan de contingencia: El plan de contingencia corresponde a la utilización de librerías y/o otros lenguajes, dentro de los establecidos por el cliente, que pueda suplir y complementar las carencias de el o los lenguajes escogidos anteriormente. Una solución muy extrema, en la construcción, será la regeneración de código en otro lenguaje de los mencionados.	
Resolución: Elegir el lenguaje que provea todas las funcionalidades necesarias para el cumplimiento de los requerimientos, independiente de que exista menos documentación y no sea del total agrado de los integrantes del proyecto.	

Cuadro 2: Hoja Control de Riesgo: ID7

5. Implementación (Entrega y Operación)

5.1. Plan de operación del sistema

Los componentes computacionales mínimos requeridos por Hevelius para su operación consisten en un computador con Sistema Operativo Linux Fedora Core y Software ACS 6.0, no se restringe sólo a la utilización de esta versión, puede utilizar otras, pero con las distintas versiones puede ser que existan variaciones que impliquen modificaciones al código fuente, pero se deja establecido que en la versión 6.0 funcionará correctamente, de acuerdo a los requerimientos del cliente.

El equipo en el cual se implemente Hevelius también debe poseer acceso a Internet y sin olvidar el acceso al telescopio que se desea operar. Sobre los requerimientos mínimos de hardware aún no están definidos.

ID 8	Telescopio amateur no disponible
Prioridad: 8	Clase: Riesgos del Proyecto
Probabilidad: Bajo	Impacto:Muy Alto
Descripción: El telescopio amateur es una pieza fundamental dentro de nuestro proyecto, ya que trabajamos en base a él, por lo que la ausencia de éste retrasaría y detendria el proyecto totalmente.	
Período: Todo el Proyecto	Estado: Latente
Contexto: Actualmente el telescopio se encuentra en camino, puesto que ya fue comprado, lo que puede influir es sólo el tiempo de espera, el cual puede alargarse en gran cantidad, provocando retraso en el desarrollo del software.	
Plan de Mitigación: No es algo que dependa directamente de nosotros, si no de los proveedores.	
Plan de contingencia: La primera alternativa será de tratar de conseguirse otro telescopio para poder desarrollar nuestro software. UNa segunda alternativa, parálela a la ya mencionada sería tratar de continuar con el desarrollo por vías que no impliquen, por el momento, pruebas físicas con el telescopio.	
Resolución: Consiste en tener un telescopio de remplazo.	

Cuadro 3: Hoja Control de Riesgo: ID8

Hevelius se desarrollará sobre la plataforma Linux Fedora Core y Software ACS 6.0 como ya se había especificado y con el telescopio NEXSTAR 4 SE y añadido a éste una cámara CCD para la obtención de imágenes.

Como Hevelius es sólo el primer paso para el desarrollo completo de un software de control genérico para telescopios, es muy importante la comprensión del código entregado, debidamente comentado, como requerimiento del cliente en inglés y, de la misma forma, informar los avances en twiki de ACS-UTFSM Group, para que posteriormente pueda ser modificado de acuerdo a requerimientos futuros.

5.2. Plan de implementación (entrega)

Una vez finalizado el desarrollo del software, el proceso de entrega debe consistir de dos fases.

1. **Entrega del programa y código.**
   Como ya se ha mencionado anteriormente Hevelius es un paso a la construcción de un software genérico, es por ello la importancia del código, puesto que es la base para que posteriormente se siga desarrollando en este tema, por estas razones se entrega el código debidamente ordenado, organizado y comentado en inglés, por ser nuestro cliente de carácter internacional.

   En lo que se refiere al programa en sí, no se puede hacer una capacitación a quienes usarán este software, ya que no son personas específicas. Pero al finalizar el desarrollo de Hevelius se tratará que astrónomos prueben el funcionamiento del software. Es por esto, que para aquellos que deban tratar con Hevelius, existe una documentación en la cual se detalla los componentes y la utilización de ellos,

ID 9	Ausencia de algún integrante del proyecto.
Prioridad: 9	Clase: Riesgos del Proyecyo
Probabilidad: Bajo	Impacto:Muy Alto
Descripción: Que algún integrante del grupo decida salirse de él por alguna razón.	
Período: Todo el proyecto	Estado: Latente
Contexto: Que un integrante del grupo se retire de él es un riesgo que siempre está presente, pero es más probable que suceda en períodos de alta tensión. En nuestro caso ya sucedió una vez y es por eso que estamos muy concientes de lo complicado que es.	
Plan de Mitigación: Organizarnos de buena forma para trabajar minimizando la tensión y el estrés generados por el proyecto.	
Plan de contingencia: A esta altura del proyecto ya no tenemos la posibilidad de pasar sin notar la ausencia de algún integrante, pero lo que si podríamos hacer, es tener bien organizadas las tareas de cada uno, para luego distribuir sus tareas entre los integrantes restantes.	
Resolución: Este dejará de ser un riesgo cerca del final del proyecto, ya que a pesar de que podría quedar un poco por hacer, a esa altura no debiera ser mucho, por lo que el impacto ya no sería mayor.	

Cuadro 4: Hoja Control de Riesgo: ID9

esta documentación será especificada en la siguiente fase.

2. **Documentación.**
   Como Hevelius está siendo creado para personas especializadas en el tema de la astronomía, se les entrega documentación detallada del software, ya que no existe una instancia directa en donde se pueda preguntar acerca de su funcionamiento, donde el único contacto podría ser mediante correo electrónico, puesto que el ambiente en que trabaja Hevelius es el de los observatorios y, por lo tanto, el trato directo se hace más complicado.

   La especificación de la documentación consiste en las siguientes partes:

   - **Explicación de la interfaz:** Esta consiste en la explicación de dónde se encuentra ubicado cada uno de los componentes que tiene implementado Hevelius.
   - **Componentes Implementados:** En una sección se especifica qué hace cada uno de sus componentes y cómo es el funcionamiento de ellos, qué parámetros recibe, etc.

     Toda la documentación debe ser desarrollada en inglés.

5.3. Plan de mantención

Como ya se ha mencionado anteriormente Hevelius esta implementado mayormente para observatorios, los cuales se encuentran en distintas partes del mundo y los usuarios del programa pueden acceder desde donde prefieran para manipular los telescopios, por lo que nos es imposible dar mantenimiento presencial a

ID 10	No cumplimiento con los estándares del proyecto ALMA-CONICYT	
Prioridad: 10	Clase: Riesgos del Proyecyo	
Probabilidad: Bajo		Impacto:Alto
Descripción: El proyecto debe cumplir con los estándares que proponer ALMA.		
Período: Etapa de construcciíon		Estado: Latente
Contexto: Hevelius forma parte del proyecto de ALMA, por lo que debe cumplir con los estándares que ellos exigen, entre los que destacan el uso de la plataforma ACS para la distribución de componentes.		
Plan de Mitigación: Trabajar con un esquema modular, obligando a hacer la separación por componentes, para luego distribuir entre componentes del programa y aquellos que serán distribuidos por ACS. Acostumbrarnos desde temprano a comentar el código en inglés y a usar el twiki para informar de lo hecho en cada sesión de trabajo.		
Plan de contingencia: En el caso de los comentarios, lo que se debiera hacer es revisar el código e ir comentándo. Si el problema fuera por falta de información sobre las sesiones de trabajo, se debiera agregar una entrada con lo hecho en las fechas en que no se informó, para luego hacerlo periodicamente de manera correcta. En el caso de haber problemas en la modularidad, o en la separación por componentes del sistema. Se debería revisar el código y reestructurarlo, de manera que se separe en los componentes adecuados, enviando los correspondientes a ACS.		
Resolución: Acostumbrarnos a realizar de forma metódica el proceso de comentarios y de informar lo hecho diariamente por medio del twiki. Y la modularidad y separación por componentes, dejará de ser un riesgo, en cuanto tengamos en cuenta las diferencias funcionales entre secciones de código, y lo separemos acorde a ello.		

Cuadro 5: Hoja Control de Riesgo: ID10

todos los usuarios.

Puede existir una asistencia remota, principalmente a través de correo electrónico para tratar de resolver cualquier tipo de problema que pueda existir.

6. Planificación de Actividades

6.1. Work Breakdown Structure (WBS)

6.2. Carta Gantt

6.3. Resumen de Compromisos