ARxCODE
Prototipo de software para el Análisis de Riesgo por Colisión con Desechos Espaciales.

Por ***M. Cecilia Valenti.***

Presentado ante la Universidad Nacional de La Matanza y la Unidad de Formación Superior de la CONAE como parte de los requerimientos para la obtención del grado de

MAGISTER EN DESARROLLOS INFORMÁTICOS DE APLICACIÓN ESPACIAL

UNIVERSIDAD NACIONAL DE LA MATANZA

DIRECTOR
Marcelo Colazo
Comisión Nacional de Actividades Espaciales (CONAE)

CO-DIRECTOR
Dra. Alicia Mon
Universidad Nacional de la Matanza (UNLaM)

Resumen

En este trabajo se presenta el prototipo de software ARxCODE: *Análisis de Riesgo por Colisión con Desechos Espaciales*. Un sistema diseñado para el estudio de acercamientos con riesgo de colisión, entre misiones satelitales operativas y desechos espaciales. ARxCODE tiene la capacidad de extraer la información que proviene de los mensajes de alerta de colisiones estandarizados, Conjunction Data Message (CDM), definidos por el Consultative Committee for Space Data Systems (CCSDS) y de procesar datos ingresados manualmente. Devuelve al operador parámetros para el análisis de riesgo como: la mínima distancia y la probabilidad de colisión o Probability of Collision (PoC) en una interfaz gráfica que facilita una clara caracterización y visualización de la situación.

Los estudios de acercamientos de riesgo que involucran desechos espaciales acarrean grandes incertezas respecto a la posición del desecho, en especial para aquellos organismos que no cuentan con instrumentos propios de rastreo. En la actualidad, además de la distancia mínima de acercamiento, debe considerarse la probabilidad de colisión. El cálculo de la probabilidad de colisión requiere tener conocimiento de los errores de las posiciones y esto no siempre es conocido, en particular para los desechos, cuyos Two-Line elements (TLE) son públicos, pero no sus errores asociados.

Para la construcción de la matriz de covarianza de la posición del desecho correspondiente al último TLE disponible más cercano al momento del máximo acercamiento Time of Closest Approach (TCA), se implementa el método desarrollado por Osweiler (Osweiler, 2006). Para la estimación de los errores que introducen las propagaciones de los TLE, con el modelo de propagación analítico Simplified General Perturbations model (SGP4) se desarrolló una metodología que incorpora el análisis histórico de los productos orbitales precisos de la misión SAC-D. Finalmente para el cálculo de la probabilidad de colisión, se implementa un método simplificado desarrollado por Lei-Chen (Lei et al., 2017).

ARxCODE es una herramienta que ofrece a los operadores de los centros de control de misión, una visión más clara de las situaciones de encuentro, para los momentos de diálogo e intercambio de información con los organismos internacionales de alerta.

Agradecimientos

Tabla de Contenidos

Índice de figuras

Índice de tablas

Lista de acrónimos

CDM	Conjunction Data Message
TLE	Two-Line elements
SGP4	Simplified General Perturbations model
PoC	Probability of Collision
TCA	Time of Closest Approach
CCSDS	Consultative Committee for Space Data Systems
IDE	Integrated Development Environment
CODS	CONAE Orbyt Dynamics Services

Capítulo 1

ARxCODE: Especificaciones

ARxCODE es un prototipo de software diseñado para el procesamiento y análisis de encuentros con riesgo de colisión, entre misiones operativas y desechos espaciales.

1.1. Especifiaciones Iniciales

En la Tabla 1.1, se listan los requerimientos iniciales del sistema.

Tabla 1.1

Lista de requerimientos inciales del sistema

Req. ID	Descripción
1	Requerimientos FUNCIONALES
ARR-001	El sistema deberá interpretar mensajes estandarizados de alerta de colisiones.
ARR-002	A partir del identificador de ambos objetos y una época estimada para el TCA, el sistema deberá calcular la mínima distancia entre los objetos involucrados.
ARR-003	El sistema deberá estimar los errores en la posición de los objetos al momento del TCA.
ARR-004	El sistema deberá estimar la PoC para la situación de encuentro que se decida analizar.
ARR-005	El sistema deberá ofrecer la información en un entorno gráfico al operador.
2	Requerimientos de INTERFACES
ARR-006	El sistema debe interactuar con el operador para que sea este quien ingrese/elija los datos del encuentro; ya sea cargando los objetos y el TCA o seleccionando un mensaje de alerta.
3	Requerimientos de PERFORMANCE
ARR-007	El sistema devolverá los resultados al operador en menos de un minuto.
4	Requerimientos de DISEÑO
ARR-008	El sistema deberá tener una estructura modular,implementado en un lenguaje orientado a objetos.
ARR-009	El sistema contará con un módulo principal que administrará la interfaz gráfica y hará uso de toda la librería.

1.1.1. Casos de Uso

El sistema se piensa como un sistema acoplado al software principal del departamento de Dinámica Orbital. En este sentido, no existe gran complejidad en la estructura del prototipo, ya que su valor, radica en la correcta implementación de los algoritmos que procesan la información del encuentro. A tal fin, se identifican dos casos de uso:
Fig. 1.1

- *Procesar Encuentro*: que nuclea el procesamiento vertebral de ARxCODE
- *Ver informes de encuentros anteriores*: ofrece encuentros anteriores.

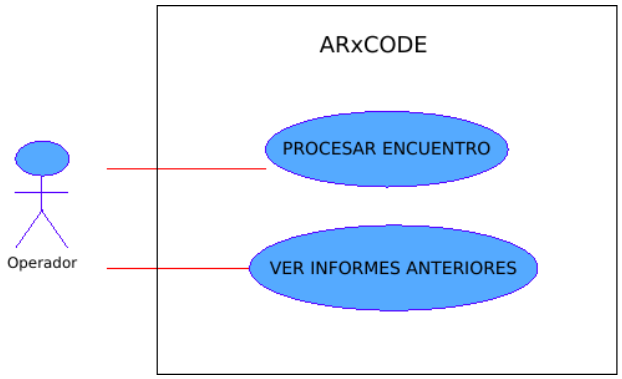


Figura 1.1. Casos de Uso de ARxCODE

Tabla 1.2

Caso de uso: Procesar Encuentro

Nombre	**Procesar Encuentro**
Actor	Operador de Dinámica Orbital con Autorización
Propósito	Calcular la probabilidad de colisión, la mínima distancia total y mínima distancia en la coordenada radial, para poder hacer un análisis de la situación de encuentro.
Resumen	Procesa la ingesta de datos de un encuentro. y calcula los parámetros de la situación de riesgo: mínima distancia total, mínima distancia en la coordenada radial y probabilidad de colisión. Realiza gráficos e informes.
Precondiciones	El operador debe estar registrado en la página space-track de NORAD. Los archivos CDM deben estar previamente cargados en el Directorio de búsqueda. El operador debe conocer el encuentro que desea analizar y sus datos en caso del ingreso manual.
Flujo Principal	1 - El operador selecciona un archivo con un mensaje de alerta. 2 - El operador oprime el botón *Track* para visualizar el encuentro proyectado en la superficie terrestre (opcional) 3 - El operador oprime el botón para generar un informe (opcional)
Flujo Alternativo	1 - El operador ingresa los números de identificación de los objetos (NORAD_ID) 2 - El operador ingresa la fecha y hora del máximo acercamiento (TCA) 3 - El operador oprime el botón *Procesar* para procesar el encuentro
Postcondiciones	El informe de análisis de riesgo fue generado y almacenado.

1.1.2. Ámbito del sistema

Se evaluó el concepto del ARxCODE en el contexto de la Unidad de Desarrollo de Desechos Espaciales de la CONAE. Fundamentalmente la vinculación con el departamento de Dinámica Orbital y los procedimientos actuales que se realizan en relación a los riesgos de colisión con desechos.

Se hizo un estudio de las estructuras orgánicas existentes y los sistemas asociados. Los distintos tipos de productos y usuarios, las interfaces que existen y el acceso a los datos reales con los que se podría contar.

Se analizó cómo trabajan otras agencias espaciales el problema de los desechos espaciales y se sacaron conclusiones respecto de qué es lo que podría ofrecerse y bajo qué premisas.

De las consideraciones más importantes que se desprendieron de esta etapa, cabe destacar que se decidió un prototipo para funcionar montado sobre el software principal de Dinámica Orbital, como un anexo que no interfiere de ninguna manera con los procesos actuales. Así mismo, sus productos finales no serán considerados en la toma de decisiones hasta tanto sus resultados no hayan sido validados durante un periodo suficiente, que permita verificar y mejorar su funcionamiento, contrastándolo con un acumulado de situaciones reales.

1.1.3. Características de los Usuarios

Debido a la complejidad del problema y sus consecuencias, será un software diseñado para ser utilizado por un analista experto, con conocimientos de Dinámica Orbital.

1.1.4. Restricciones

Se contó con el asesoramiento y el intercambio de información con personas del área de Dinámica Orbital y otros departamentos de la CONAE. Se realizaron algunas reuniones e intercambio de correos electrónicos, aunque por ser una temática que se aborda bajo regímenes especiales de acuerdos de confidencialidad, no fue posible contar con la totalidad de la información.

1.1.5. Interfaces

ARxCODE fue pensado para ser un sistema anexo a las estructuras del departamento de Dinámica Orbital. El diseño completo, contempla un acceso directo al servidor de la base de datos de los productos de Dinámica Orbital para la obtención de: las efemérides propagadas de la misión operativa, y los mensajes de alerta (CDM). Estos últimos también podrán ser recibidos por correos electrónicos de los organismos internacionales de alerta, como por ejemplo JSpOC o mediante una solicitud a la

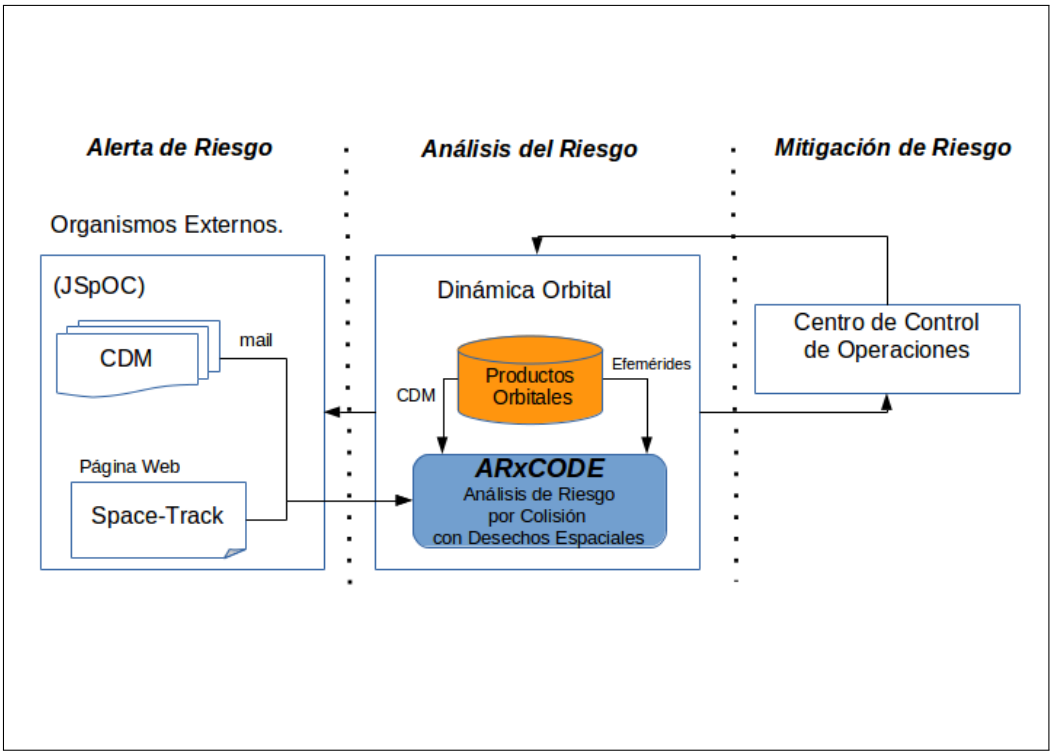


Figura 1.2. Diagrama de Interfaces del Sistema

página Space-track, previa notificación y registro autorizado del operador a cargo, (Fig. 1.2).

1.2. Especificaciones Finales

1.2.1. Requerimientos Finales

El sistema debe tener la capacidad de interpretar los mensajes estandarizados CDM y presentar la información que allí se registra, en forma clara al operador. Por otro lado, debe tener la capacidad de colectar datos ingresados manualmente por el operador y ofrecer los parámetros que resulten del procesamiento propio del ARxCODE, como mínima distancia, TCA calculado y PoC.

Esta última funcionalidad implica que ARxCODE debe poder solicitar a la página Space-Track los TLEs correspondientes a los objetos involucrados, debe poder estimar las matrices de covarianza de ambos objetos, ya sea mediante el método de Osweiler o incorporando efemérides predichas del departamento de Dinámica Orbital; debe poder propagar esos errores al momento del TCA y finalmente calcular la PoC. En la Tabla 1.3, se listan todos los requerimientos del sistema.

Tabla 1.3

Tabla de especificación de requerimientos del ARxCODE

Req. ID	Descripción
1	Requerimientos FUNCIONALES
ARR-010	ARxCODE debe calcular la probabilidad de colisión de un acercamiento de riesgo.
ARR-020	ARxCODE debe aceptar como inputs: un mensaje de alerta (CDM), o los identificadores de NORAD de ambos objetos y el tiempo de máximo acercamiento (TCA).
ARR-030	ARxCODE debe utilizar los productos orbitales de la misión o realizar el mismo procedimiento que se aplica al desecho, a la misión.
ARR-040	ARxCODE debe calcular la mínima distancia, total y en la coordenada radial.
ARR-050	ARxCODE debe manipular los sistemas de referencia: TEME, TOD y RTN.
ARR-060	ARxCODE debe permitir al operador analista experto visualizar el encuentro, generar reportes y notificaciones.
ARR-070	ARxCODE debe extraer el conjunto de TLEs de los objetos involucrados de los últimos 15 días anteriores al TCA.
ARR-080	ARxCODE debe estimar los errores en la posición inicial del desecho y de la misión operativa.
ARR-090	ARxCODE debe propagar los errores de la posición inicial al TCA.
2	Requerimientos de INTERFACES
ARR-100	ARxCODE deberá permitir la carga manual de la situación de encuentro.
ARR-110	ARxCODE deberá descargar los TLE de Space-Track.
ARR-120	ARxCODE deberá manipular los CDM con formato xml.
3	Requerimientos de RENDIMIENTO y/o PERFORMANCE
ARR-210	ARxCODE deberá ofrecer el reporte de la situación en no más de 1 minuto.
4	Requerimientos de VALIDACIÓN
ARR-300	Los módulos de implementación de metodologías de ARxCODE serán validados con los resultados de las publicaciones pertinentes y de la bibliografía.
ARR-310	Las propagaciones realizadas con el SGP4 serán validadas con el software STK.
5	Requerimientos de DISEÑO
ARR-400	ARxCODE tendrá un diseño modular.
ARR-410	ARxCODE se desarrollará como una librería.
ARR-420	ARxCODE contará con una interfaz gráfica.
6	Requerimientos de IMPLEMENTACIÓN
ARR-500	ARxCODE será implementado en python 2.7.
ARR-510	ARxCODE será implementado en el entorno de desarrollo Eclipse.
ARR-520	El control de versiones se realizará con el gestor Git.
7	Requerimientos de REUSABILIDAD
ARR-600	ARxCODE utilizará la librería de SGP4 en Python.
ARR-610	ARxCODE utilizará la librería de Element Tree para el parseo del CDM.
ARR-620	ARxCODE utilizará la librería de requests de python para la conexión con Space-Track.
ARR-630	ARxCODE utilizará la librería de numpy y scipy de python para los cálculos estadísticos y de integración.

1.3. Flujo dinámico de ARxCODE

ARxCODE se inicializa cuando un operador lo ejecuta. Una vez iniciado el programa, el operador debe seleccionar alguna de las dos opciones de procesamiento: *Cargar CDM* o *Carga Manual*. Una vez cargados los datos se inicia la etapa del PROCESAMIENTO.

Procesamiento a partir de un CDM

Si el procesamiento se realiza a partir de la carga de un CDM, el código se ocupa de la extracción y la publicación de los datos del CDM.

Procesamiento a partir de una carga manual

En el caso de la carga manual, el procesamiento implica distintos pasos (Fig. 1.3):

- Solicita los TLE a NORAD.
- Propaga los TLE y calcula el TCA y la mínima distancia.
- Calcula la covarianza del desecho, y de la misión si la misma no fue provista por el departamento de Dinámica Orbital.
- Calcula la matriz de covarianza combinada.
- Propaga la matriz de covarianza combinada hasta el TCA.
- Calcula la probabilidad de colisión.
- Publica los datos en la pantalla para el operador.

Finalizada esta etapa de procesamiento y publicación de los datos por pantalla, el operador puede solicitar la visualización de las trayectorias en el intervalo analizado, para ello presiona el botón *Track* y el gráfico se agrega a la pantalla.

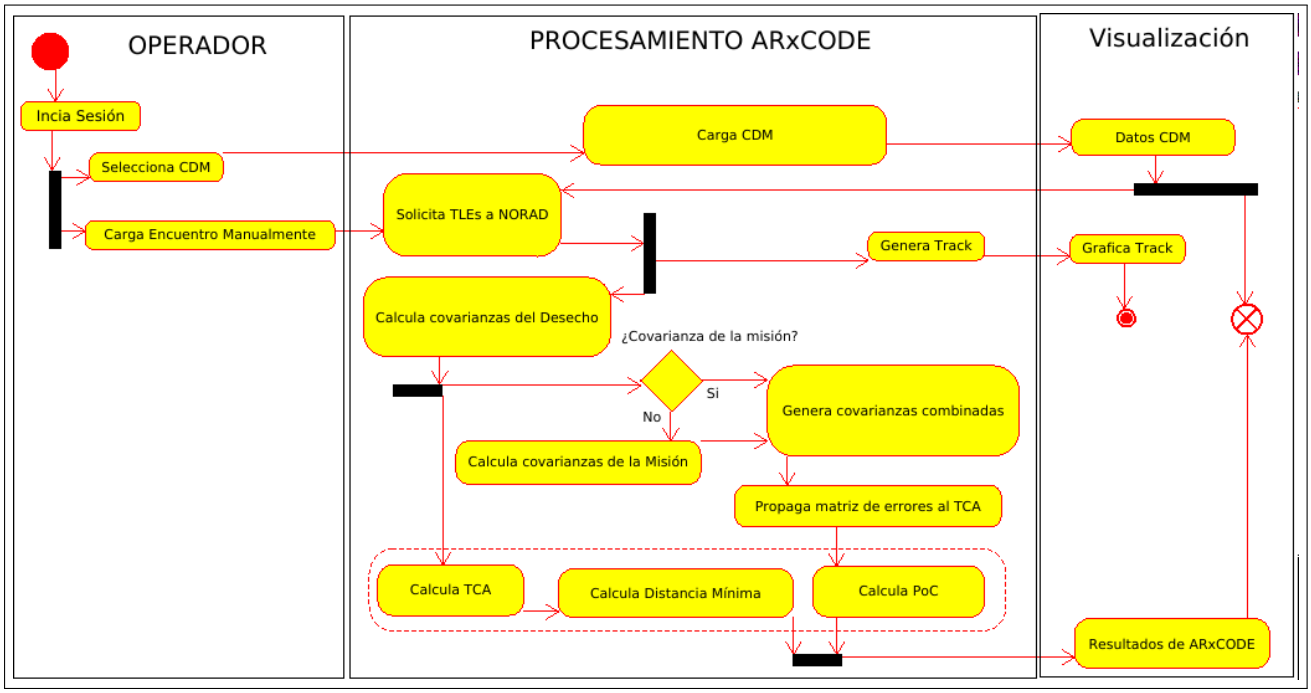


Figura 1.3. Diagrama de Actividades de ARxCODE

Capítulo 2

Metodología de Desarrollo

Para el desarrollo de este trabajo se optó por un modelo de desarrollo de tipo incremental (que resulta iterativo por naturaleza). En cada iteración, se optimizó el diseño y se fueron agregando nuevas funcionalidades y capacidades al sistema.

El problema que se propone resolver, requiere de la implementación de distintos algoritmos en una estructura en donde los resultados de un módulo se utilizan en el siguiente. Esto implica que para un análisis completo cada una de las instancias debió estar previamente validada.

No obstante, las salidas de los módulos que eran ingesta de otros módulos podían reemplazarse por datos ya conocidos, y así desarrollar y validar el módulo siguiente, mientras se analizaba en paralelo cómo mejorar o corregir los resultados no satisfactorios. De esta manera se fue armando el cuerpo general del software a gran escala, y luego se fue revisando y afinando cada uno de los paquetes.

Esta forma de trabajo permitió dividir la complejidad del proyecto, y a su vez, desarrollar un conjunto de bibliotecas fácilmente modificables, sin alterar la estructura central.

2.1. Plan de Desarrollo

Dado que el trabajo se realizaba en el contexto de la maestría en general y que las tareas y los cronogramas se modificaban frecuentemente, ajustándose al dictado de cursos y tareas de un proyecto integrador que también formaba parte del ciclo lectivo; en un principio no era posible armar un cronograma fijo de las tareas.

Se planificó una distribución horaria (Fig. 2.1) y una planificación general (Fig. 2.2) para los semestres I, III y IV del año 2016.

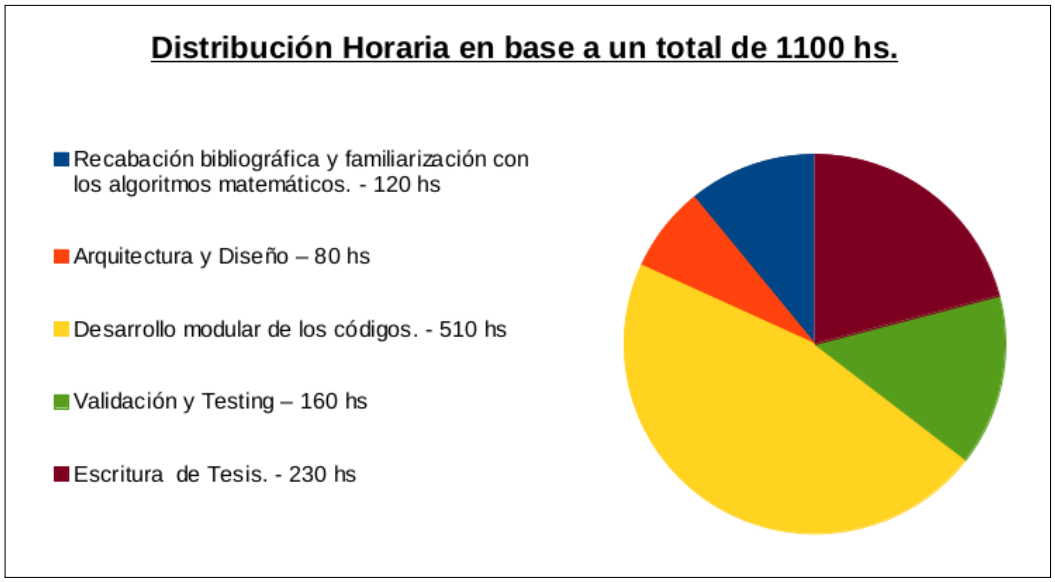


Figura 2.1. Planificación de la distribución del total de horas asignadas dentro de la maestría al desarrollo y la escritura del trabajo de tesis.

Cuatrimetres 2016	Tareas
I	Diseño y Elaboración de Requerimientos
	Recabación Bibliográfica.
	Investigación de procedimientos en otras agencias/organismos
III	Investigación de Metodologías y Algoritmos para los distintos módulos.
	Generación del entorno de desarrollo
	Desarrollo del módulo para la administaración y propagación de TLEs
IV	Pruebas de verificación sobre el módulo de TLEs
	Elaboración del módulo para la generación de matrices de covarianza
	Integración y pruebas del módulo para la generación de las matrices de covarianza.

Figura 2.2. Planificación de las tareas agrupadas por semestres para el año 2016

Ya a comienzos del año 2017 se contaba con un cronograma fijo que pautaba tres semanas por mes de dedicación exclusiva a la elaboración de la tesis. Eso permitió realizar una planificación semanal (Fig. 2.3) de las tareas.

Meses	Semanas			
	I	II	III	IV
Enero			Elaboración del módulo que genera la matriz de propagación de covarianzas.	
Febrero		Pruebas sobre los datos de Misión de CODS	Pruebas sobre el módulo que genera la matriz de propagación de covarianzas	
Marzo		Pruebas de integración del módulo para la propagación de la matriz de covarianzas	Implementación de los algoritmos para el cálculo de PoC	
Abril		Desarrollo del módulo Encuentro que integra todo el flujo y calcula la PoC		
Mayo		Pruebas del módulo Encuentro y pruebas de integración	Desarrollo de la interfaz gráfica	
Junio		Desarrollo de la interfaz gráfica	Desarrollo del módulo de administración del CDM	Pruebas de integración y de visualización de la interfaz gráfica
Julio		Escritura de Tesis	Escritura de Tesis	Escritura de Tesis
Agosto		Escritura de Tesis	Depuración del systema e incorporación de las correcciones de los directores al texto.	

Figura 2.3. Planificación de las tareas agrupadas por semanas para el año 2017

2.2. Entorno de Desarrollo

Para el desarrollo se utilizó:

- Plataforma de Desarrollo (Integrated Development Environment (IDE)): Eclipse Ver. 3.8.1.
- Lenguaje de Programación: Python 2.7
- Biblioteca de Interfaz gráfica: QT por medio del enlace PyQT.
- Gestor de Configuración: Git.

Eclipse

Eclipse (Ver. 3.8.1) es una plataforma de desarrollo multiplataforma ampliamente utilizada y ya muy madura, cuya estructura de perspectivas, editores y vistas, facilita el desarrollo en distintos lenguajes de programación. En este trabajo se incorporó el IDE para python, *Pydev*.

Eclipse ofrece excelentes capacidades para la gestión de proyectos, permitiendo incorporar en un mismo proyecto distintos archivos y documentación, que, en esta tesis, agrupó no sólo los datos de entrada y salida, como los TLE, los CDM o los productos orbitales; sino que también incluyó todos los ploteos y gráficos que resultaban de los procesamientos y la propia documentación referida a la

escritura de este documento. Esto fue muy productivo en lo que respecta al control de versiones, ya que se aprovechó el hecho de que Eclipse ya tiene incorporado el gestor Git. Cabe destacar también, que cuenta con una excelente herramientas de depuración.

Python

El lenguaje de programación Python se destaca en sus capacidades tanto de cálculo como de manejo de texto. Esto agiliza mucho los procesos que involucran el manejo de tablas de datos plasmadas en texto plano, como son por ejemplo los datos TLE y las efemérides orbitales que se generan como productos del departamento de Dinámica Orbital. Así mismo facilita el manejo de las nomenclaturas de los distintos archivos de datos o imágenes generadas.

Existen numerosas, potentes y optimizadas bibliotecas para la realización de cálculos, y el tratamiento vectorial. En nuestro caso se aprovechó particularmente la biblioteca *numpy*, y muy poco de *scipy* específicamente, para interpolar datos. Finalmente su utilización masiva permite tener acceso rápido a sus potencialidades.

QT

Para el desarrollo de la interfaz gráfica se utilizó QT, a través del enlace PyQT.
QT es un framework ampliamente utilizado para el desarrollo de aplicaciones multiplataforma. Cuenta con mucha contribución de la comunidad y está soportado por Nokia.

Su mecanismo de conexión de señales y eventos es simple, esto permite definir los eventos sencillos en la estructura del GUI, y luego invocar el código python con las acciones más avanzadas. Subyace su implementación en C++ que muchas veces dificulta la comprensión para los que estamos familiarizados con la lógica del python, y lo mismo ocurre con la documentación y prevalecen los ejemplos para C++.

Git

Si bien, esta herramienta no fue aprovechada en todo su potencial en este trabajo, por tratarse de un proyecto sencillo y desarrollada por dos personas, fue fundamental para agilizar la posibilidad de trabajar desde cualquier computadora, siempre en la última versión del proyecto. Así mismo, el trabajo con control de versiones, permitió realizar distintas pruebas e implementaciones que luego se descartaron o se quitaron del producto final, pero que pueden ser reutilizados en futuros proyectos. En particular, en la utilización de la interfaz intermedia que fue generada para una ágil evaluación de los resultados parciales.

Capítulo 3

Diseño, Seguimiento y Control

ARxCODE resulta una aplicación de escritorio, de estructura modular, reusable y modificable. Pensada con una filosofía de expansión y perfeccionamiento, su arquitectura permite la adición de funcionalidades sin mayores inconvenientes.

3.1. Arquitectura

Componentes

En un planteo conceptual, con alto grado de abstracción, los distintos paquetes de ARxCODE pueden agruparse en cinco componentes (Fig. 3.1): PROCESAMIENTO, ADMINISTRACIÓN DE DATOS, INTERFAZ GRÁFICA Y VISUALIZACIÓN, Sistemas de Referencia y VALIDACIONES.

- PROCESAMIENTO: Involucra los cuatro paquetes que, no sólo operan con los datos ingresados, sino que también los manipulan y procesan para generar nuevos productos. Estos módulos son, de alguna manera, los distintos núcleos del código.

  - *AjustarTle*
  - *Comparar*
  - *Estadistica*
  - *Encuentro*

- ADMINISTRACIÓN DE DATOS: Son aquellos paquetes que se encargan de la obtención, el desglose y el preprocesamiento de los datos que serán utilizados por el resto de los módulos.

  - *TleAdmin*
  - *CodsAdmin*
  - *CDM*

- INTERFAZ Y VISUALIZACIÓN: Agrupa el paquete que genera la interfaz gráfica y el paquete que contiene todos los módulos que generan representaciones visuales, como los gráficos o las proyecciones de las trayectorias sobre el mapa.

  - *Aplicacion*
  - *visual*

- Sistemas de Referencia: *SistReferencia*, es el paquete que contiene todo lo referente a las transformaciones entre los distintos sistemas de referencia, ya sean espaciales o de tiempo.

- VALIDACIONES: Agrupa todos los módulos desarrollados para validar los resultados.

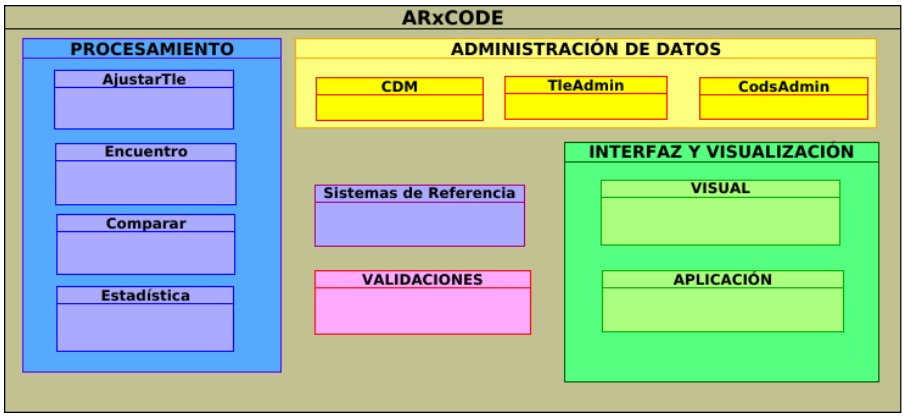


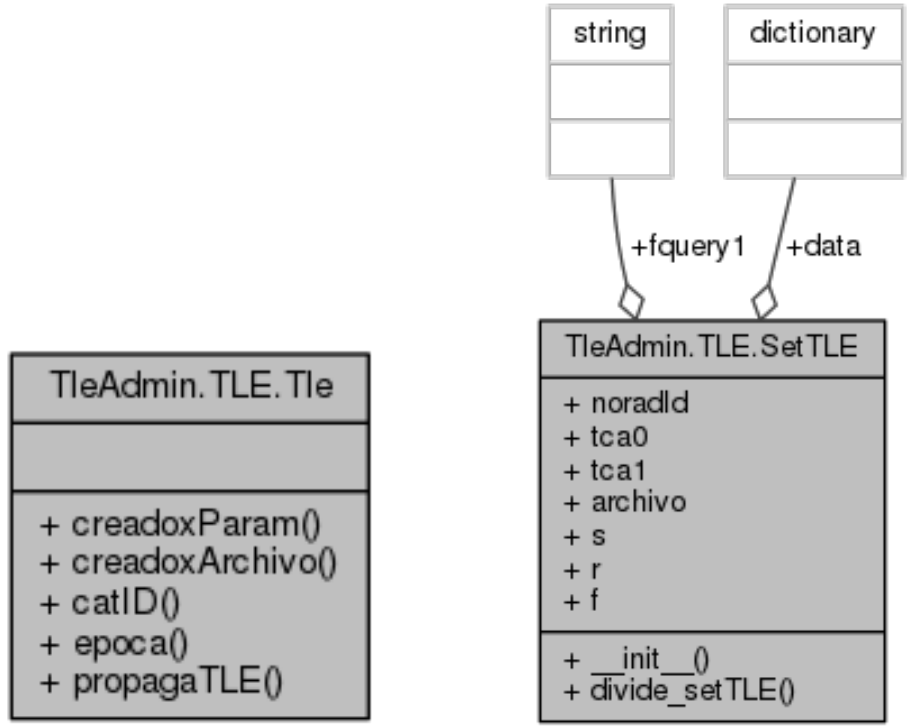
Figura 3.1. Componentes de ARxCODE

ADMINISTRACIÓN DE DATOS: TleAdmin

Este paquete contiene las dos clases que gestionan la descarga de los TLEs de la página Space-Track: **Tle** y **setTLE** (Fig. 3.2). La primera provee un único TLE y todo lo referente a él; se instancia a partir del identificador de NORAD del objeto y una época, o a partir del nombre de un archivo que contiene un único TLE. Entre sus métodos, es fundamental **propagaTLE()**, que propaga la posición del objeto al momento que sea necesario.

La clase **setTLE**, es necesaria particularmente para la implementación del método de Osweiler que requiere un conjunto de TLEs de 15 días para la generación de la matriz de covarianza de la posición del objeto. Esta clase se instancia, indicando el identificador de NORAD del objeto, la fecha de inicio y fin, del set que se requiere. Una vez descargados, genera un único archivo con todos los TLEs del intervalo y los guarda en la carpeta *crudosTLE*.

El método **divide_setTLE()** de la clase **setTLE**, particiona el texto con el conjunto de TLEs y genera un archivo por cada TLE y lo almacena en la carpeta *tle* del paquete. Esta última carpeta, se actualiza con cada corrida del software, de modo que siempre contiene sólo el conjunto de archivos de TLEs que van a ser procesados.



(a) Clase TLE (TleAdmin/TLE/Tle)

(b) Clase setTLE (TleAdmin/TLE/-setTLE)

Figura 3.2. Clases Tle y setTLE del paquete TleAdmin

ADMINISTRACIÓN DE DATOS: CDM

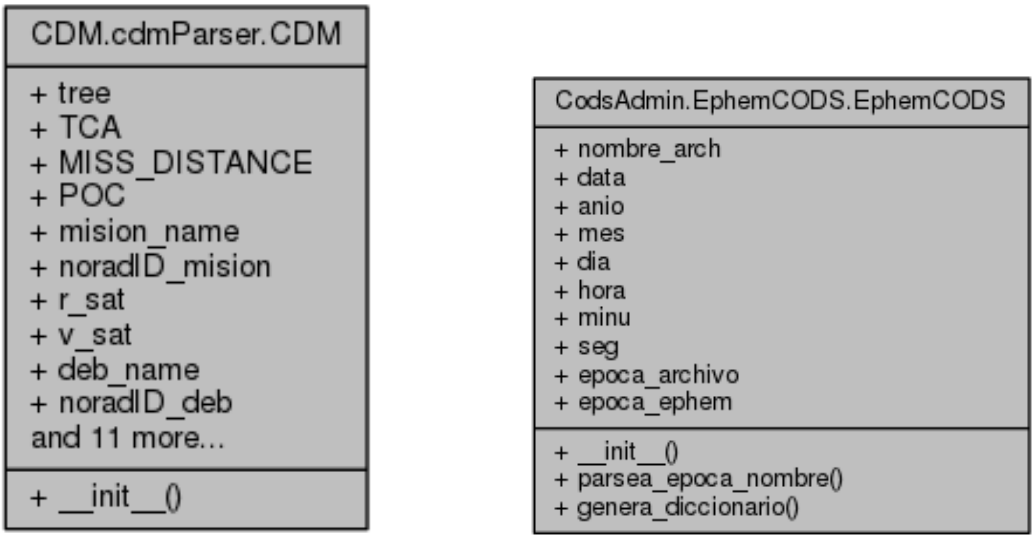
El paquete CDM contiene una carpeta con archivos en formato del CDM, en xml y la clase **CDM** que tiene la capacidad de desglozarlos y extraer los datos del mensajes que serán plasmados en la pantalla de la interfaz gráfica, para la visualización del operador.

ADMINISTRACIÓN DE DATOS: CodsAdmin

Este paquete administra los productos orbitales generados por el departamento de Dinámica Orbital CONAE Orbyt Dynamics Services (CODS). Cuenta con varias carpetas que almacenan los productos previamente descargados y contiene a la clase **EphemCODS**, que tiene la capacidad de extraer las efemérides de los archivos y de identificar el periodo de datos que abarca el archivo a partir del nombre.

PROCESAMIENTO: AjustarTle

Este paquete incluye el módulo con el mismo nombre *AjustarTle*, que cuenta con varias funciones que reagrupan la información del conjunto de TLEs a fin de poder hacer ordenamientos y compara-



(a) Clase CDM (CDM/CDM).

(b) Clase EphemCODS (CodsAdmin/EphemCODS)

Figura 3.3. Clases CDM y EphemCODS para el pareseo de los mensajes de alerta CDM y los productos orbitales de CODS

ciones, entre los TLE de distintas fechas que son propagados a una misma época (práctica necesaria para la implementación del método de Osweiler para la construcción de las matrices de covarianza).

PROCESAMIENTO: Comparar

En el paquete *Comparar* se nuclean todos los módulos y funciones que permiten la selección y la extracción de los datos de los productos de Dinámica Orbital (CODS), y sus respectivas comparaciones con los resultados que provienen de la propagación de los TLEs. Este tipo de comparaciones son necesarias en la estimación de errores que se comenten al utilizar propagaciones de los TLE con SGP4.

PROCESAMIENTO: Estadística

Dentro de este paquete se encuentran los desarrollos referidos a los cálculos estadísticos, más precisamente los módulos que calculan las matrices de covarianza. En este paquete se encuentra el módulo que implementa el método de Osweiler, (Osweiler, 2006) **matrizOsweiler()**.

PROCESAMIENTO: Encuentro

Este es el paquete núcleo del software. En él se encuentra la clase **Encuentro** (Fig. 3.4), que instancia la generación del encuentro, incorporando ambos objetos mediante sus identificadores de NORAD y el TCA, generando las propagaciones necesarias, y las matrices de error, para calcular la mínima distancia y la probabilidad de colisión. A tal fin, cuenta con los métodos (Fig. 3.4): **calculaMacom-**

binada(), **proyecta_alplano_encuentro()**, **calculaPoC_circ()** y **calculaPoC_gral()**

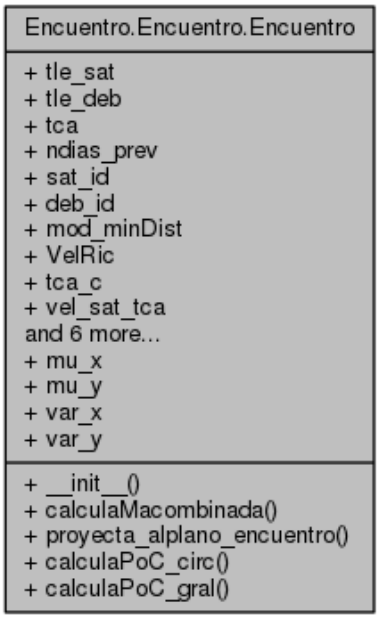


Figura 3.4. Clase Encuentro para el cálculo de los parámetros del acercamiento

INTERFAZ GRÁFICA Y VISUALIZACIÓN: Aplicación

Este paquete contiene, principalmente el módulo con el formulario de interfaz de ARxCODE **frm_main**. Dentro del mismo se ubican las clases que heredan de la estructura de QT, para el desarrollo de la aplicación, y son estas clases las que invocan los distintos módulos del resto del código para el desarrollo de los procesos.

INTERFAZ GRÁFICA Y VISUALIZACIÓN: visual

Visual es el paquete que agrupa todos los módulos de generación de gráficos y ploteos. Los gráficos se utilizaron principalmente para el estudio de la tendencia de errores de los TLE y otros análisis y validaciones. Dentro de este paquete se encuentra el módulo que genera las trayectorias de las órbitas, proyectadas sobre el mapa en el momento del encuentro.

Sistemas de Referencia

Este paquete contiene módulos para realizar transformaciones entre los distintos sistemas de referencia. Entre ellos, los dos más utilizados son: *teme2tod* y *ricSis*, ya descriptos en la Sec. **??**.

Validaciones

Reúne todos los módulos que se desarrollaron para la verificación y validación de los resultados que se describen a continuación (Sec. **??**), y cuyos valores se analizan y publican en la sección de Resultados y Validaciones (Sec. **??**).

3.2. Interfaces

Por cuestiones de tiempo y de accesibilidad, para el desarrollo de este trabajo, los datos que provee el departamento de Dinámica Orbital fueron descargados y se extraen de un directorio, al igual que los mensajes de alerta CDM, que fueron descargados de páginas de internet , ya que no nos han facilitado ninguno vinculado a la misión operativa con la que trabajamos, por motivos de confidencialidad.

Se realizó la automatización de la descarga de TLEs de la página Space-Track y se habilitó en la interfaz la pantalla que permite la carga manual de los datos del encuentro. Como resultado, las interfaces implementadas, son (Fig. 3.5):

- Conexión a Space-Track para la solicitud de TLEs.
- Administración de las efemérides orbitales de los directorios de CodsAdmin.
- Administración de los CDM del directorio CDM, a través de la intervención del operador.
- Carga Manual de datos de un encuentro realizada por el operador.

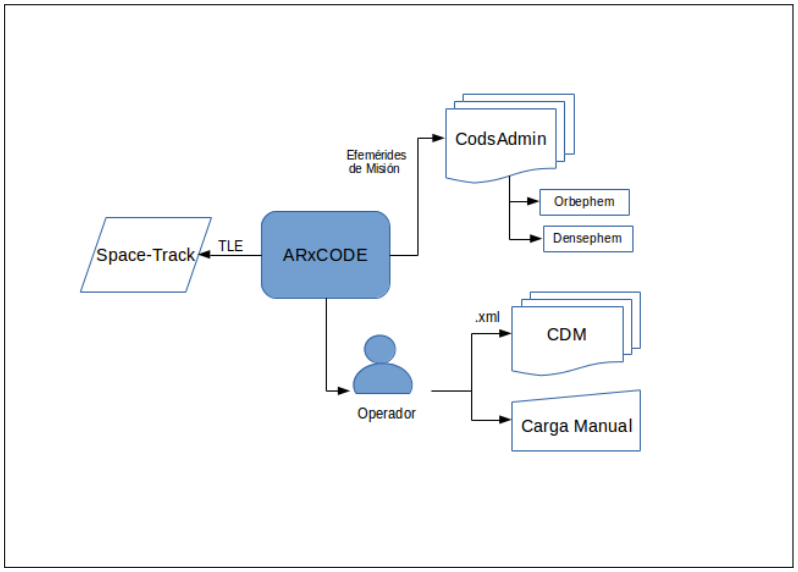


Figura 3.5. Diagrama de Interfaces Implementadas en ARxCODE

3.3. Seguimiento y Control

3.3.1. Verificación y Validación

Verificación

Se realizaron distintas pruebas de unidad y de integración. Los distintos casos de prueba se diseñaban a partir de datos de prueba de la bibliografía recabada para cada una de las metodologías o sobre situaciones reales. Los resultados de la ejecución de las pruebas se comparaban luego con los valores de los casos publicados o la situaciones reales. Un breve esquema de este procedimiento puede verse en la Figura 3.6.

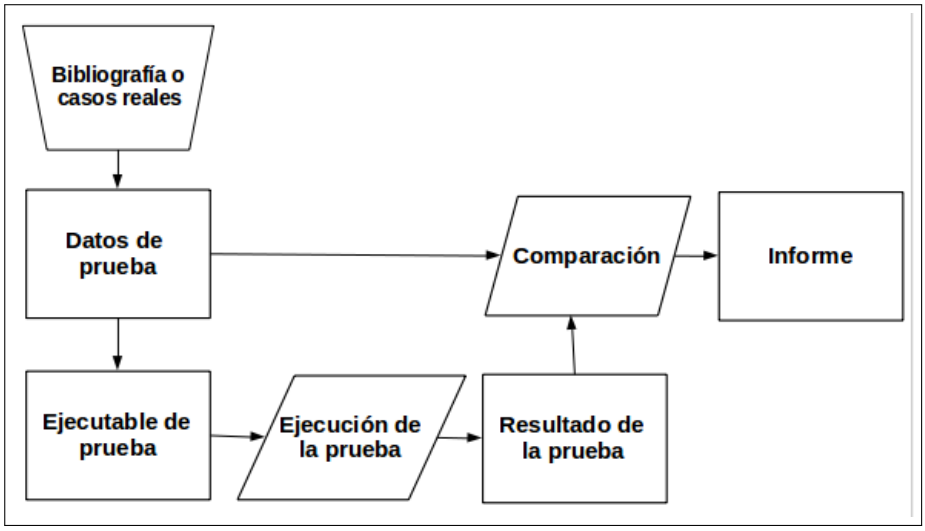


Figura 3.6. Esquema de la metodología para realizar las pruebas de unidad sobre los módulos.

A medida que se desarrollaba un nuevo módulo, se realizaban los casos de prueba para ese módulo y una prueba de integración, que incorporaba todos los módulos hasta el momento verificados.

TleAdmin y CDM Dado que tanto los TLE como los CDM son los elementos fundamentales que dan inicio al análisis de la probabilidad de colisión, es fundamental tener un control sobre la correcta adquisición de estos inputs. Sobre ellos se diseñaron pruebas funcionales. Se prueba siempre que los TLE de ambos objetos hayan sido adquiridos; si esto no fuera así ya sea por complicaciones en la conexión a internet, o porque los TLE para las fechas solicitadas no están disponibles, el software interrumpe el proceso e indica a cuál de las situaciones se debe el problema. Mientras que para el caso de los CDM, el programa se interrumpe si el archivo ingresado no respeta el formato .xml capaz de procesar y anuncia un mensaje de alerta cuando el CDM a sido procesado pero no todos los campos de interés se encuentran en él.

Comparar y SistReferencia Las pruebas realizadas para garantizar los correctos procedimientos respecto de la propagación y las transformaciones de los sistemas de referencia fueron fundamentales, pero sólo se realizaron en las primeras etapas del desarrollo, ya que no dependen de cada ejecución en particular, sino de la correcta implementación de los algoritmos.

Estadística Para verificar la correcta implementación del método de Osweiler para la generación de las matrices de covarianza, se generó un caso de prueba que permite comparar los resultados con los que se publican en el trabajo del autor. Dada la dependencia de este módulo con el módulo de administración de los TLE, es importante que esta prueba es naturalmente de integración y es muy importante que se realice frecuentemente.

Comparar y CodsAdmin El procedimiento que se realiza para la generación de la matriz de propagación de errores involucra la utilización de los datos de misión (o Datos de CODS). Esta tarea fue realmente compleja y se realizaron varias ejecuciones de la prueba sobre la manipulación de estos datos, ya que los mismos muchas veces no respetaban un formato estandarizado por ejemplo sobre las fechas o contenían intervalos sin datos significativos, etc. Estas verificaciones se realizaron una sola vez hasta lograr que los datos resultaran ordenados dentro del período de análisis que se utilizó para generar la matriz de propagación de errores. Por cuestiones de tiempo no se implementó una verificación automatizada sobre estos inputs, lo que debe ser tenido en cuenta si se desean generar matrices de propagación de errores actualizadas o en forma dinámica.

Encuentro Las pruebas realizada en el módulo para la estimación de las probabilidades de colisión tienen también una fuerte dependencia del módulo que gestiona los TLE, mientras que pueden configurarse o no para que la matriz de covarianza que utilizan sea generada por el módulo de Osweiler o no. Salvo para el caso del método de Lei-Chen, ya que en su trabajo existe un ejemplo que indica todos los parámetros que se utilizan en la ecuación final, y entonces se puede hacer una prueba sólo para esa unidad.

Aplicación En lo que respecta a la aplicación que nuclea y despliega la interfaz gráfica, se realizaron inspecciones para depurar la correcta inicialización de variables y el flujo de las distintas vistas.

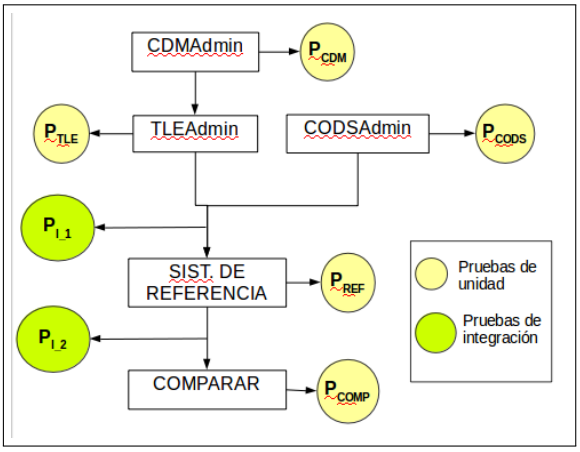


Figura 3.7. Casos de prueba para los módulos que interactúan con los datos de Misión.

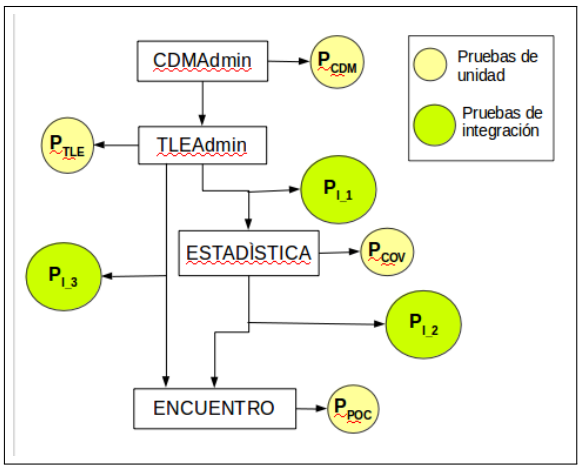


Figura 3.8. Casos de prueba del bloque completo, desde los inputs hasta el cálculo de la PoC.

Validación

Al tratarse de un sistema que implementa distintas metodologías para el cálculo de parámetros, el control más exhaustivo se basó en analizar que los resultados de los algoritmos implementados fueran coherentes y coincidieran con los que existían en publicaciones bibliográficas o situaciones reales que se podían reproducir.

En conclusión se han realizado distintas pruebas e inspecciones, pero no pruebas exhaustivas. Un desarrollo detallado de los resultados de las pruebas y validaciones se describe en la Sec. **??**

3.3.2. Gestión de la Configuración

Al ser ARxCODE un prototipo sencillo desarrollado por una sola persona, no fue necesario implementar una compleja gestión de configuración. Se utilizó el repositorio Git (Ver Sec. 2.2) para el control de versiones, al que día a día se incorporaba el proyecto. Se pautó realizar un release una vez por mes, ya que era la frecuencia con la que se planificó que existieran nuevos módulos y pruebas de unidad y/o de integración.

3.3.3. Aseguramiento de la Calidad

Se han definido tres instancias fundamentales para la evaluación y el control de la calidad del software durante el desarrollo.

Sin duda el primer punto tiene que ver con el correcto acceso a los TLE que son los inputs básicos para iniciar el procesamiento. Del mismo modo, la ingesta de un CDM defectuoso puede producir distintos fallos ya que al menos los mínimos datos de iniciación deben ser ingresados correctamente. Finalmente, todo el ciclo de los distintos procesos, a saber: solicitud de los TLE correspondientes, generación de matrices, cómputo de la mínima distancia y de la PoC, así como todos los procesos de transformaciones de coordenadas deben realizarse en forma fluida.

A fin de garantizar un control sobre la calidad mínima en los puntos mencionados, una prueba general que contempla las tres pruebas integrales sobre el circuito completo se realizó antes de cada release mensual. Con la misma frecuencia, a continuación se realizaron, pruebas sin acceso a internet y por ende sin acceso a la página proveedora de los TLE y sobre CDM defectuosos para verificar que el software interrumpía su proceder e informaba con un mensaje específico a qué se debía la interrupción.

No obstante, a medida que se realizaban más y más pruebas, sumando nuevos datos de validación, otros puntos importantes fueron descubriéndose. De modo que se incorporaron pruebas que permiten medir la diferencia de épocas entre el TLE y el tiempo de mayor acercamiento, a fin de garantizar que se informe al operador si las propagaciones orbitales superen los 7 días para el caso de las mínimas distancias, o los 15 días para la generación de matrices.

En la misma dirección fue importante incorporar un mensaje de alerta, cuando las matrices de covarianza generadas superan ciertos valores establecidos, ya que el proceso resulta finalizado exito-

samente pero los datos no son confiables (Ver Tabla 3.1).

Tabla 3.1

Pruebas de Aseguramiento de la Calidad que se realizaron antes de cada release, aproximadamente una vez al mes.

Prueba	Resultado esperado
Prueba general	MSJ: *El proceso ha finalizado correctamente*
Prueba sin internet o sin devolución de TLE del sitio web	INTERRUPIÓN: *El TLE no se ha generado correctamente*
Prueba con CDM defectuoso	INTERRUPIÓN: *El CDM ingresado es incorrecto o está incompleto*
Prueba con TLE muy antiguos	MSJ de Alerta: *El TLE utilizado en el problema no es de una época confiable*
Prueba con matrices de covarianza con errores groseros	MSJ de Alerta: *El TLE utilizado en el problema no es de una época confiable*

Bibliografía

Akella, M. R. and Alfriend, K. T. (2000). Probability of Collision Between Space Objects. *Journal of Guidance Control Dynamics*, 23:769–772.

Alfano, S. (2007). Review of conjunction probability methods for short-term encounters (aas 07-148). *Advances in the Astronautical Sciences*, 127(1):719.

Alfano, S. (2008). Method for determining maximum conjunction probability of rectangular-shaped objects. US Patent 7,383,153.

Analytical Graphics, Inc. (2016). System tool kit - stk 11x64.

Arsenault, J., Chaffee, L., and Kuhlman, J. (1964). General ephemeris routine formulation document. Technical report, Report ESDTDR-64-522, Aeronutronic Publ. U-2731.

Babiker, F., Doyon, M., and Abbasi, V. (2012). The canadian space agency (csa) collision risk assessment and mitigation system (crams): Sharing the development and the operational challenges. In *SpaceOps Conference*.

Bowman, B. (1971). A first order semi-analytic perturbation theory for highly eccentric 12 hour resonating satellite orbits. *NORAD document, Nov*.

Brandon Rhodes (2015). sgp4-1.4.

Brouwer, D. (1959). Solution of the problem of artificial satellite theory without drag. *The Astronomical Journal*, 64:378.

Brouwer, D. and Hori, G.-i. (1961). Theoretical evaluation of atmospheric drag effects in the motion of an artificial satellite. *The Astronomical Journal*, 66:193.

CCSDS (2013). *Conjunction Data Message - CCSDS 508.0-B-1 - Recommended Standard*. CCSDS Secretariat.

Chan, K. (2003). Improved analytical expressions for computing spacecraft collision probabilities. *Advances in the Astronautical Sciences*, 114:1197–1216.

Cranford, K. (1969). An improved analytical drag theory for the artificial satellite problem. In *Astrodynamics Conference*, page 925.

ESA (2013). Space Debris. http://www.esa.int/Our_Activities/Operations/Space_Debris.

Flohrer, T., Krag, H., and Klinkrad, H. (2008). Assessment and categorization of tle orbit errors for the us ssn catalogue. *risk*, 8(9):10–11.

Hoots, F. R. and Roehrich, R. L. (1980a). Models for propagation of norad element sets. Technical report, AEROSPACE DEFENSE COMMAND PETERSON AFB CO OFFICE OF ASTRODYNAMICS.

Hoots, F. R. and Roehrich, R. L. (1980b). Spacetrack report no. 3 models for propagation of norad element sets. Technical report, Project Spacetrack Reports, Office of Astrodynamics, Aerospace Defense Center.

Hoots, F. R. and Roehrich, R. L. (1998). A History of Analytical Orbit Modeling in the United States Space Surveillance System. Third US-Russian Space Surveillance Workshop. Washington, D.C.

Hoots, F. R., Schumacher Jr, P. W., and Glover, R. A. (2004). History of analytical orbit modeling in the us space surveillance system. *Journal of Guidance, Control, and Dynamics*, 27(2):174–185.

Hujsak, R. S. (1979). A restricted four body solution for resonating satellites without drag. Technical report, AEROSPACE DEFENSE COMMAND PETERSON AFB CO OFFICE OF ASTRODYNAMICS.

IADC (2007). *Space Debris Mitigation Guidelines*. IADC.

J.R. Alarcón Rodríguez, F.M. Martínez-Fadrique, H. K. (2004). Development of a collision risk assessment tool. *Advances in Space Research*, 34:1120–1124.

Karacalioglu, A. G. and Stupl, J. (2016). The impact of new trends in satellite launches on the orbital debris environment.

Kelso, T. et al. (2007). Validation of sgp4 and is-gps-200d against gps precision ephemerides.

Kessler, D. and Cour-Palais, B. (1978). Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, 83:2637–2646.

Klinkrad, H. (2006a). *Space Debris. Models and Risk Analysis*. Springer.

Klinkrad, H. (2006b). *Space Debris. Models and Risk Analysis*, chapter 8, pages 215–241. Springer.

Kozai, Y. (1962). Second-order solution of artificial satellite theory without air drag. *The Astronomical Journal*, 67:446.

Krag, H., Klinkrad, H., and Alarcon-Rodriguez, J. (2007). Assessment of orbit uncertainties for collision risk predictions at esa. In *Second IAASS conference "Space safety in a global world*, pages 14–16.

Lane, M. (1965). The development of an artificial satellite theory using a power-law atmospheric density representation. In *2nd Aerospace Sciences Meeting*, page 35.

Lane, M. H. and Hoots, F. R. (1979). General perturbations theories derived from the 1965 lane drag theory. Technical report, AEROSPACE DEFENSE COMMAND PETERSON AFB CO OFFICE OF ASTRODYNAMICS.

Laporte, F. (2014). Jac software, dedicated to the analysis of conjunction messages. *AIAA*, 1774.

LEI, B. X. C. (2009). A rapid algorithm of space debris collision probability based on space compression and infinite series [j]. *Acta Mathematicae Applicatae Sinica*, 2:015.

Lei, C., Xian-Zong, B., Yan-Gang, L., and Ke-Bo, L. (2017). *Orbital Data Applications for Space Objects*. Springer.

Lyddane, R. (1963). Small eccentricities or inclinations in the brouwer theory of the artificial satellite. *The Astronomical Journal*, 68:555.

Montenbruck, O. and Gill, E. (2012). *Satellite orbits: models, methods and applications*. Springer Science & Business Media.

NASA (2014). Space missions and satellite box score. *Orbital Debris Quarterly News*, 18.

NASA (2017a). The od environment in numbers. *Orbital Debris Quarterly News*, 21.

NASA (2017b). Orbital Debris Program Office. http://orbitaldebris.jsc.nasa.gov/.

NASA (2017c). Space missions and satellite box score. *Orbital Debris Quarterly News*, 21.

Osweiler, V. P. (2006). *Covariance estimation and autocorrelation of NORAD Two-line Element Sets*.

Wang, R., Liu, J., and Zhang, Q. (2009). Propagation errors analysis of tle data. *Advances in Space Research*, 43(7):1065–1069.

Xu, X.-L. and Xiong, Y.-Q. (2014). A method for calculating probability of collision between space objects. *Research in Astronomy and Astrophysics*, 14(5):601.