**DEMONSTRATION APPLICATION with USER INTERACTION for OREKIT FEATURES**

**Technical Description and Proposal**

 

 ESA SOCIS 2014 – Summer of Code In Space

**Document Status Sheet**

| **Issue** | **Author** | **Date** | **Signature** |
| --- | --- | --- | --- |
| 0.1 | R. Falone | 3/07/2014 |  |
|  |  |  |  |

**Change Record**

| **Issue** | **Page** | **Section** | **Modification** |
| --- | --- | --- | --- |
|  |  |  |  |

#   Table of Content

[Table of Content 3](#_Toc392173446)

[1. Introduction 4](#_Toc392173447)

[1.1 Purpose of the document 4](#_Toc392173448)

[2 Referenced Documents 5](#_Toc392173449)

[2.1 Applicable Documents 5](#_Toc392173450)

[3 DEMO APP with USER INTERACTION for OREKIT FEATURES 6](#_Toc392173451)

[3.1 Introduction to the System 6](#_Toc392173452)

[3.2 Main Project Requirements 6](#_Toc392173453)

[3.3 Adopted Technologies and S/W components 6](#_Toc392173454)

[3.4 Functionalities’ Proposal 7](#_Toc392173455)

[3.4.1 Input Configuration 7](#_Toc392173456)

[3.4.2 Output Configuration 8](#_Toc392173457)

[3.4.3 Functionalities 8](#_Toc392173458)

[3.5 Coverage Matrix: Orekit Packages involved in the current development 8](#_Toc392173459)

[3.6 General Software Architecture 9](#_Toc392173460)

[3.6.1 Overview 9](#_Toc392173461)

[3.6.2 GUI: the User Interface 9](#_Toc392173462)

[3.6.3 Flight Dynamics Core Computation: how Orekit libraries interact in the Demo 11](#_Toc392173463)

[3.6.4 Output Handling: Textual Output, Graphics and Charts 11](#_Toc392173464)

[3.6.5 DB integration: the use of MySQL database for pre-loaded Scenarios 11](#_Toc392173465)

[4 ACTIVITIES’ PLANNING 12](#_Toc392173466)

# Introduction

## Purpose of the document

The purpose of this document is to provide the description and the design requirements of the Demo Application for Orekit features, developed in the context of ESA SOCIS (Summer of Code In Space) 2014 project (in the shifted period 11th June – 11th September). This is a proposal subject to the discussion, modifications and final approval coming from the entire Orekit Developers Community.

# Referenced Documents

## Applicable Documents

The following table lists all documents that are applicable to or referenced by this document.

|  |  |  |
| --- | --- | --- |
| **DOC ID** | **TITLE** | **DESCRIPTION** |
| 1 | Orekit participation to ESA SOCIS 2014 – Projects Proposals | https://www.orekit.org/forge/projects/orekit/wiki/SOCIS |
| 2 | Orekit Documentation | https://www.orekit.org/documentation.html |
| 3 | Orekit API | https://www.orekit.org/site-orekit-6.1/apidocs/index.html |

Table 1 Applicable Documents

# DEMO APP with USER INTERACTION for OREKIT FEATURES

## Introduction to the System

[1] Orekit is a low-level library. As such, it does not provide any visible user interaction and does not display anything fancy. It must be called by a higher-level application that configures it and provides it with the initial data to propagate. It is therefore difficult to explain Orekit features and have new users play with it a little to get a sense of what it can do.

The purpose of this project is to set up a calling application that can be used as a showcase for Orekit features.

All development is done in the Java programming language and uses only free software libraries compatible with the Apache Software License. The developed system will run on any operating system, such as GNU/Linux and Windows.

## Main Project Requirements

In the following, a list of high-level, general requirements for the Demo App is provided.

The Application should be able to select features and run them quickly in front of an audience. The application should have both a set of predefined configurations showing specific features (orbit propagation, events, frames transformations, time scales, attitude ...) and allow users (even new ones) to change the settings and see immediately the effects of the changes. For each feature, links to example code, tutorials and documentation will be available.

The display part will include some simple form-based results (a propagated orbit, an event occurrence report ...), tabular outputs (orbits and attitude ephemerides ...), curves and plots, and animated 2D views (satellite ground tracks, sensors footprints, ground station visibility circles ...). No 3D views are considered here (yet).

The work has been split into the following work packages, each of them will be specifically discussed in following sections [1]:

1. Selection of the features to be demonstrated;
2. For each feature, identification of the most educational use cases to present it;
3. Identification of the configuration parameters and setup of the reference scenarios;
4. Implementation of a top level application with graphical user interface (GUI) to run these use cases with simple textual output;
5. Tabular output for ephemerides;
6. Display of 1D curves with some specific markers for discrete events;
7. Display of 2D ground tracks;
8. Display of sensor footprints and ground station visibility circles;
9. Display of time animation and events triggering feedback.

## Adopted Technologies and S/W components

The system will be based on a Java Platform, with the use of the following libraries:

* **Orekit (v. 6.1)**: central core for Flight Dynamics computation;
* **Apache Commons Math (v. 3.2)**: basic mathematic computation necessary for Orekit at runtime;
* **Swing** and **AWT**: GUI building and user interaction with low-level Orekit core configuration;
* **JFreeChart** (or **charts4j**? To Be Confirmed):  a free chart library for Java that can generate a wide variety of charts. This should be used ad useful tool for 1D curves generation;
* **World Wind** (To Be Confirmed): displaying and interacting with geographic data and representing the wide range of geometric objects needed. 2D limited use for the moment;
* **JDBC** for **MySQL** management (To Be Confirmed): basic and elementary Data Base Architecture to safely load pre-configured scenario’s data, for demonstration purposes.

Each of them will provide a proper interface to interact with the other libraries.

## Functionalities’ Proposal

In the following list, we will provide a summary of the configuration capability of the Tool, available features and possible outputs [2], [3].

### Input Configuration

1. Definition of a Scenario: it should be composed by at least two objects
	1. **Constellation:** it should contain at least one satellite;
	2. **Network:** it should contain at least one Ground Station.
2. Definition of a Satellite object:possibility of configuring different satellite’s characteristics (such as mass, initial state, attitude…);
3. Definition of a Ground Station object: possibility of configuring geodetic coordinates, mask for a ground station’s antenna;
4. Selection and loading of TLE files from a DB or in-local;
5. Selection of forces’ models to be considered in propagation: let the user the possibility to choose what forces consider or not (atmospheric drag, central gravity, third body gravity, solar radiation pressure, forces induced by manoeuvres);
6. Selection among different Time Scales;
7. Selection of propagation window;
8. Selection among different reference frames;
9. Selection among different propagators (Keplerian propagator, Eckstein-Heckler, Numerical propagator);
10. Others to be proposed?

Since the Demo App should be focused more on the input than on the output, a list of further configurable options may be added, even if a more complex and sophisticated output will be neglected for the moment (think about 3D rendering output coming from an attitude manoeuvre). Input configuration will be arranged more complete than output capability, in order to show to a potential audience what Orekit is able to ingest and process, though output capabilities are still not so advanced (**To Be Checked** – Please consider the alternative approach to develop a Demo App which is self-consistent: the only possible inputs are those which can be processed and displayed by the Tool. Which is the best approach, in your opinion?).

### Output Configuration

It should be possible to configure the desirable output. Three types are foreseen at the moment:

* Textual: log-report-like;
* 1D curves (where it is possible);
* 2D displays (where it is possible);

Where it is possible to choose more than one output type, GUI will make available a selection feature to display computation results.

### Functionalities

The Demo App should offer the following actions/features:

* Produce a tabular output of satellite’s ephemerides with configurable time step;
* Display 1D curves for significant user-selected parameters (orbital parameters as a function of time, event triggered transition such as eclipse entering and exiting, umbra entering and exiting, elevation detection, field of view detector, others?);
* Display 2D curves for Satellite’s Ground Track, Sensor Footprint and Ground Station’s Visibility Circles, Time Animation (with time management – start/stop propagation, acceleration, deceleration or normal time evolution with configurable step size) and visual triggering of some discrete events (elevation over a predefined geodetic point on Earth, Perigee, Apogee, Ascending Node, Descending Node, Umbra, Penumbra, Eclipse, alignment with an orbital plane);
* Display of Satellite-Satellite contact;
* Others to be proposed?

## Coverage Matrix: Orekit Packages involved in the current development

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Packages/Features | S/C state config | GSconfig | Propagationconfig | Frame selection/conversion | Time and PropagationWindow | 1D curvesdisplay | 2DGraphic display |
| org.orekit.attitudes | ✓ |  |  |  |  | ✓ |  |
| org.orekit.bodies |  |  |  | ✓ |  |  | ✓ |
| org.orekit.forces |  |  | ✓ |  |  |  |  |
| org.orekit.frames |  |  |  | ✓ |  |  | ✓ |
| org.orekit.orbits | ✓ |  |  |  |  | ✓ | ✓ |
| org.orekit.propagation.analytical |  |  | ✓ |  |  | ✓ | ✓ |
| org.orekit.propagation.analytical.tle | ✓ |  | ✓ |  |  |  | ✓ |
| org.orekit.propagation.conversion |  |  | ✓ |  |  |  | ✓ |
| org.orekit.propagation.events |  |  | ✓ |  |  | ✓ | ✓ |
| org.orekit.propagation.integration |  |  | ✓ |  |  | ✓ | ✓ |
| org.orekit.propagation.numerical |  |  | ✓ |  |  | ✓ | ✓ |
| org.orekit.propagation.sampling |  |  | ✓ |  |  | ✓ | ✓ |
| org.orekit.time |  |  |  |  | ✓ | ✓ | ✓ |
| org.orekit.utils |  | ✓ |  |  |  |  | ✓ |

Table : coverage and use of Orekit's packages

In this section we want to give a precise mapping of the use of Orekit packages. The above matrix (Table 1) helps to give an idea of how many (and which) Orekit capabilities will be covered (and used) by the Tool.

Not all Orekit’s packages are mentioned, it means that their usage is not estimated at the present time, though the document and the planned work is subject to mentors’ and Orekit Developers’ Community’s review. (If you think other aspects and relevant Orekit’s functionalities must be added and developed, please point out the missing aspects).

## General Software Architecture

### Overview

In this section we just want to provide an overall quick view of the Tool Architecture. Main parts will be displayed in the UML diagram with following and successive more detailed explanation.



At the moment, we can imagine that Architecture of specific above modules can vary with a significant rapidity. That’s why, in this project phase I will not provide details about single modules, letting the final draft of the architectural document provide these aspects.

### GUI: the User Interface

In this section, the only one populated at the moment, we want to provide a general overall of the GUI Demo App. This is in order to inspect the endorsement of the Orekit Community about how much “appealing” the GUI is or should be further improved. Your suggestions are welcome.

In the following, you can find some screenshots showing the GUI proposal.



Figure : Preliminar GUI - Textual Output



Figure : Preliminar GUI - Charting Feature



Figure : Preliminar GUI - 2D Display

### Flight Dynamics Core Computation: how Orekit libraries interact in the Demo

TBD

### Output Handling: Textual Output, Graphics and Charts

TBD

### DB integration: the use of MySQL database for pre-loaded Scenarios

TBD

# ACTIVITIES’ PLANNING

The project will be roughly developed as described in the following main phases:

* **11th June – 3rd July**: feasibility study. Analysis and study of Orekit’s capabilities, selection of the features to be demonstrated, in-depth research of the best Java Open Source Libraries useful to the development of the Demo App, simple demos and partial coding to test their real match to Orekit Demo Project, drafting of the present document in order to discuss the architectural guidelines of the project with mentors and Orekit Developers’ Community;
* **4th July – 3rd August:** start of the proper coding phase. At the end of this period, most of the Orekit’s selected features should have been implemented, with the possibility of displaying results in textual form; writing and completion of the present document, updating existing parts and adding architectural missing sections relevant to Orekit packages;
* **4th August – 11th September**: starting displaying coding phase. Processing data for Displaying Libraries (such as World Wind or similar) may request a significant effort. So, writing code for smart interfaces and making different libraries to correctly communicate one with each other, is estimated to be time expensive and critical for the success and fulfilment of the Project. Some temporal windows are foreseen, too, in order to provide complete and final documentation.