Pointing Error Engineering Tool

Architectual Design Document

Document Number:	ASTOS-PEET-ADD-001		Date:
Issue:	1.2		2012-10-15

	Name/Function	Organization	Signature	
Prepared by:	J. Eggert	Astos Solutions	J Eggert	2012-10-15
	S. Weikert	Astos Solutions	[illegible]	2012-10-15
Checked by:	S. Weikert	Astos Solutions	[illegible]	2012-10-15
Product Assurance:				-
Project Management:	S. Weikert	Astos Solutions	[illegible]	2012-10-15

Document Change Record

Issue	Date	Affected Chapter/Section/Page	Reason for Change Brief Description of Change
1.0	2012-07-06	All	Draft issue
1.1	2012-09-07	5.3.1.2	Customer Review
1.2	2012-10-15	All	Software changes

Table of Contents

1 Introduction

This document is the Architectural Design Document (ADD) prepared for the Project performed by Astos Solutions under contract of the European Space Agency.

The purpose of this document is to define the architectural design that meets the requirements stated in the Statement of Work [AD1]. The design is specified by identifying the components, defining the control and data flow between them and stating for each of them the functions to be performed, the data in- and output and the resources utilized.

2 Applicable and Reference Documents

2.1 Applicable Documents

[AD1] Astos Solutions, ASTOS-GEN-RD-002, Java Coding Guidelines, 2010

[AD2] PEE-ECN-SoW-20111014, ESA/ESTEC Work Order "Pointing Error Engineering Tool", Statement of Work, 14 October 2011

[AD3] ESSB-HB-E-003, ESA Pointing Error Engineering Handbook, Issue 1, 19 July 2011

[AD4] ECSS-E-ST-40C; ECSS standard on Space Engineering - Software, March 2009

[AD5] Proposal for ESTEC Frame Contract 19179/05/NL/LvH work order “Pointing Error Engineering Tool Prototyping”, Astos Solutions GmbH,

2.2 Reference Documents

[RD1] ECSS-E-40 Part 1B; Space Engineering, Software, Principles and Requirements

[RD2] Object Management Group, Documents associated with UML Version 2.0, http://www.omg.org/spec/UML/2.0/, Last accessed 07.04.2009

[RD3] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modelling Language, Addison Wesley, 2004

[RD4] ASTOS-PEET-ICD-001_Iss1.1, Pointing Error Engineering Tool Interface Control Document, 2012

3 Terms, Definitions and Abbreviated Terms

3.1 Acronyms

The following abbreviations are used throughout this document.

Acronyms	
ESA	European Space Agency
PES	Pointing error source
PEC	Pointing error contributor
PEET	Pointing Error Engineering Tool
PSD	Power spectral density

3.2 Terminology

The following terminology is used throughout this document.

Container	A special block which can be used to combine several blocks into a logical unit.

4 Software Design Overview

The Pointing Error Engineering Tool (PEET) shall implement the pointing error engineering methodology presented in the ESA PEE Handbook [AD3], with a special focus on pre-phase A (CDF) and Phase A activities. PEET shall be implemented in such a way that it can support the system engineers and control engineers working in CDF and in Phase A studies in performing preliminary pointing error engineering tasks. These tasks include both breakdown activities (i.e. the top-down process of allocating pointing errors to lower level error contributors) and budgeting (i.e. the bottom-up approach of compiling an error budget at high level due to error contributors).

PEET shall support interactive breakdown and budgeting with the possibility for the user to visually examine the response to his actions.

The software consists of two main parts: the Graphical User Interface written in Java and the core written as MATLAB code. The architecture is detailed below.

4.1 Software static architecture

PEET is designed to be based on MATLAB, i.e. it can be used and installed like a MATLAB toolbox and it can be started from the MATLAB command prompt.

All calculations related to pointing error budgeting are realized as MATLAB functions. The graphical user interface of PEET (PEET_GUI) is based on Java technology. Since MATLAB itself is also based on Java this GUI can be run by the built-in MATLAB Java runtime environment (JRE) without starting another instance of a JRE. PEET_GUI and MATLAB can communicate via the JMI, the Java MATLAB interface. Via this interface the Java GUI has access to all MATLAB commands and the MATLAB workspace.

The PEET_GUI comprises several subcomponents:

The system editor (System_Editor) is the central tool to define the pointing system. The user can add PES, summation blocks and subsystems from the block database and can connect them via signals. Those signals represent Pointing Error Contributors. The Tree_View and the Plotting module are used to present the results in terms of pointing error budgets and contributions in terms of time-depending and not time-depending representations, whereas time-series and PSDs can be graphically visualized by means of the Plotting component. The user is able to select for which Pointing Error Index he wants to see the results.

Via the Java_Excel_API an import from Excel and an export to Excel is realized. Each block parameter can be imported from an Excel table, specifying the file path, the sheet, column and row. From the results an Excel table can be exported that contains the results in tabular format.

The standard MATLAB GUI is not required except to start PEET from the MATLAB command prompt.

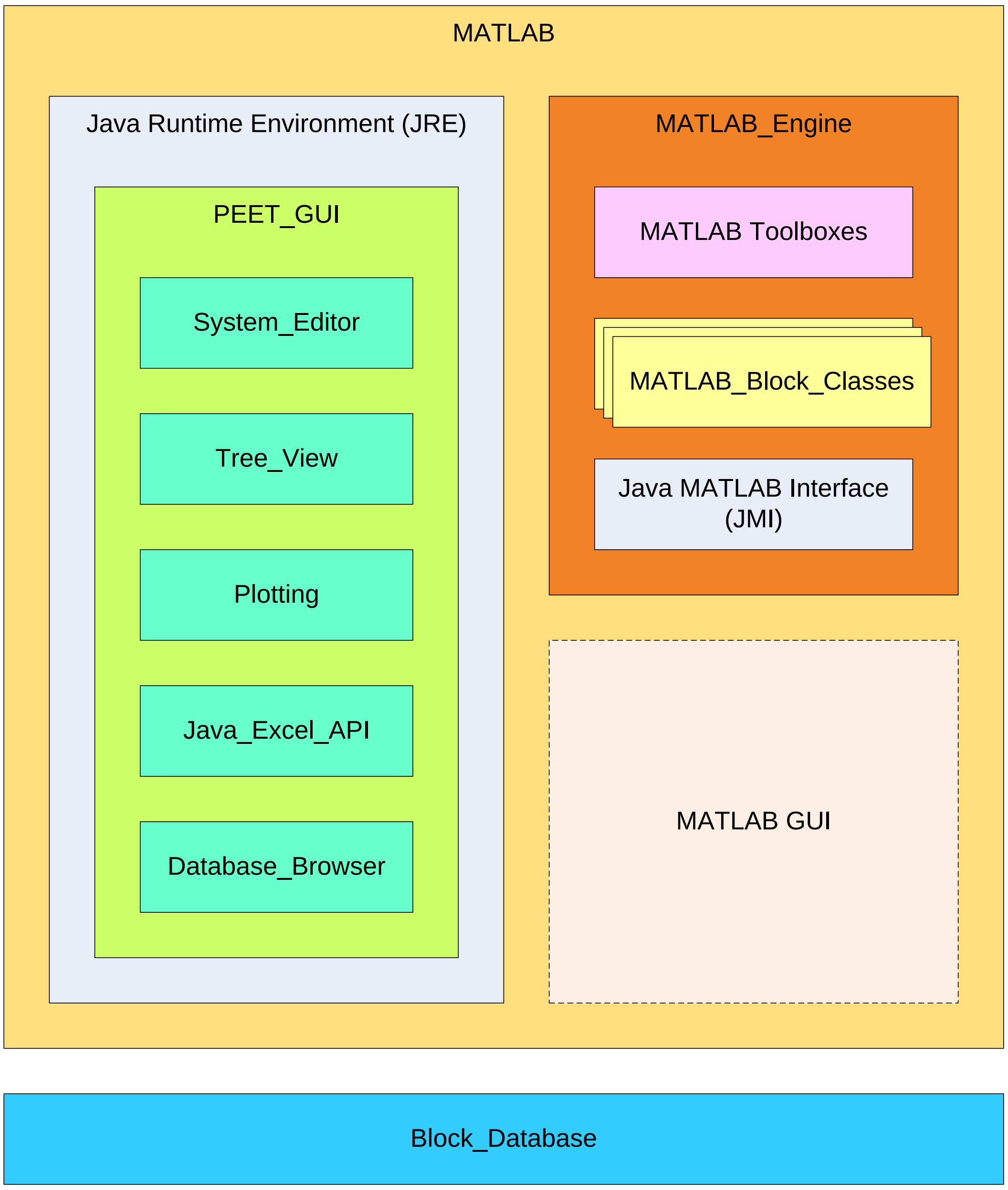

Figure 4-1: PEET static architecture

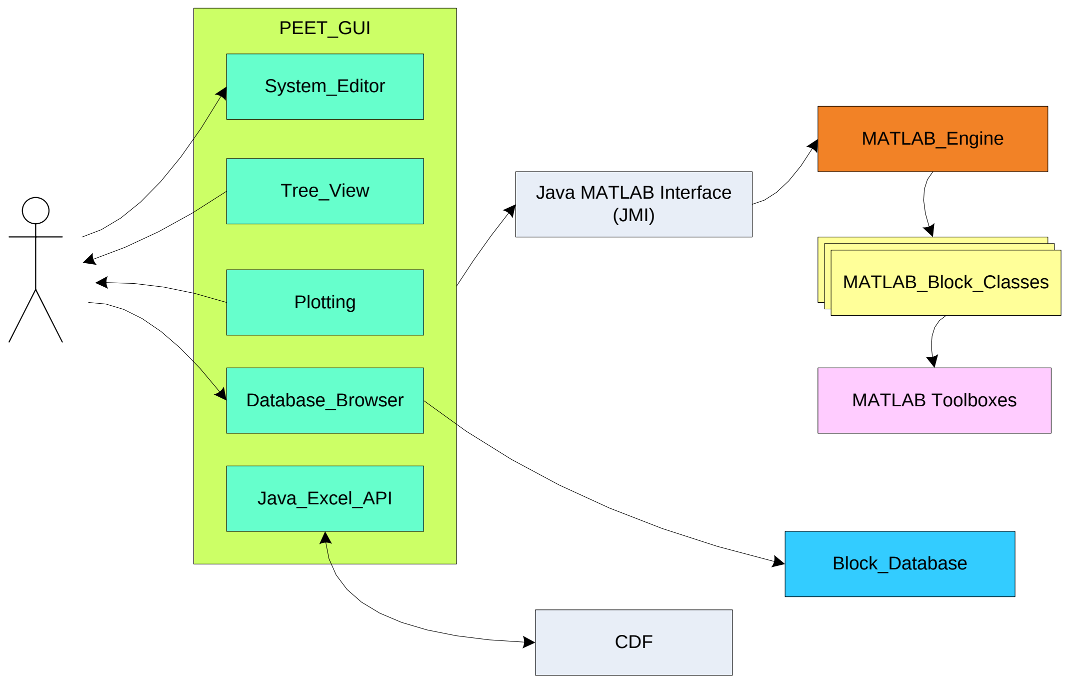

Figure 4-2: PEET component calling sequence and dependencies

4.2 Software dynamic architecture

This section is not applicable to PEET since PEET is not designed to run on real-time systems.

4.3 Software behaviour

The PEET software provides the user a graphical interface to build up pointing error systems and to perform pointing error budgeting and break-down. In a first step the user builds up the pointing system by adding blocks from a database to the pointing system and by connecting the blocks. For each block, the user has to specify the block parameters. Additionally, parameters applicable to the whole system like window time and level of confidence are also defined by the user.

Afterwards, the PEET software can calculate a pointing error. Several pointing error indices can be selected in order to reflect the desired pointing error requirements. The PEET software will visualise the error results.

PEET provides the user the possibility to add new block types by extending the block database or by exporting container blocks to the block database.

4.4 Interfaces context

For external interfaces see the Pointing Error Engineering Tool Interface Control Document [RD4].

4.5 Long lifetime software

This section is not applicable to PEET.

4.6 Memory and CPU budget

The software is intended to be used on a standard PC with standard equipment. There are no special requirements on the PC hardware.

One typical computer that will be used for testing is a standard desktop PC with 2GB RAM, 150GB of free hard disk memory and an AMD Athlon Dual Core 5400B processor.

4.7 Design standards, conventions and procedures

4.7.1 Architectural Design Method

The architecture of a complex and large-scale software can be designed and implemented in different ways. An approach that is able to handle non-functional requirements on the software quality like maintainability, reusability, understandability, structuredness and testability in an easy way is the *object-oriented* approach. This approach is very well suited for implementing physical models and their behaviour. Therefore the extensions and new developments to be performed within this project are done in an object-oriented way, too.

The main object-oriented techniques used here are *encapsulation* of data and functional behaviour in classes (so classes are used as first-class entities within the architectural as well as in the implementation model) and *inheritance* and *polymorphism* in order to factor out common behaviour in parent classes.

For documentation and visualization of the detailed design, the Unified Modelling Language (UML) is chosen. UML is a standardized modelling language for the development of object-oriented software systems. Its specification has been published by the Object Management Group (OMG) in version UML 2.0 in 2005 [RD2].

UML offers different diagram types to visualise different views on a software system. For an overview over UML see e.g. [RD3]. Throughout this document the following diagrams are used:

- *component diagrams* for high level overview of the software components and the dependencies between them
- *class diagrams* for the static software structure, i.e. composition of classes and inheritance hierarchy of classes; for each class its attributes and methods are shown as well, so that the interface of the class provided to its potential users is specified as well
- *communication diagrams* for dynamic behaviour of the classes, i.e. flow of control and data between classes

The component diagram gives a high-level view on the software system to be built. It divides the system into components.

A software component is a rather abstract concept. It may be defined in the following way: a software component is an autonomous, encapsulated part of a software system that provides one or more interfaces. It can be used as a black box within a software system and it is (or should be) reusable and with its well-defined boundaries it can be implemented independently or at least in parallel with other software components.

A component in an object-oriented software system can be built up by one class or by a composition of several classes.

The class diagram and the communication diagram offer lower level views on the system and use classes and objects as building blocks. Classes are closer to the implementation level: their translation into code fragments is straight-forward (and may be even done automatically).

When a class is defined with attributes (that are always private members of the class), *getter* (and possibly *setter*) methods will be defined for those attributes if they are needed. Those getter and setter methods are mostly omitted in the diagrams for brevity reasons.

4.7.2 Programming Standards

The programming languages used in this project are Java (Java SE 1.6) and MATLAB (m-files).

The Astos Solutions coding guideline for Java [AD1] will be applied to all developed code. These standards cover also requirements on code documentation. Naming conventions defined in [AD1] will be applied also to the MATLAB m-files as far as possible.

4.7.3 Reuse of Components

A set of components will be reused for the realisation of PEET. First of all PEET depends on MATLAB. MATAB serves as the core interpreter for PEET m-files but provides also its Java Runtime Environment for the GUI.

The System Editor of the GUI is realized using the Piccolo2D framework (see http://www.piccolo2d.org). Piccolo2D is licensed under the BSD Public License.

Excel import and Export is based on the Java Excel API (see http://jexcelapi.sourceforge.net). The Java Excel API is licensed under the GNU Library or Lesser General Public License version 2.0 (LGPLv2).

5 Software Design

5.1 General

The Pointing Error Engineering Tool (PEET) is designed as a MATLAB toolbox. The required computations are performed by MATLAB whereas the graphical user interface relies on Java Swing.

5.2 Overall architecture

The Pointing Error Engineering Tool (PEET) consists of the following three main components:

- PEET_GUI
- MATLAB_Engine
- Block_Database

The PEET_GUI component provides the user an interface similar to Simulink. It will manage the whole building process for a pointing system and also the visualisation of the pointing error results. The design goal was to provide users, who are familiar with Simulink, an interface which they can use without further explanations. In order to save the user defined pointing systems, XML based configuration files will be used. These configuration files are only read and written by the PEET_GUI component.
It is the responsibility of the PEET_GUI component to control the calculation of the various pointing error indices. For this reason the PEET_GUI component calls the MATLAB_Engine component in order to exploit the computation power provided by MATLAB.

The MATLAB_Engine component provides the computation power of MATLAB to the PEET_GUI component. This design was chosen in order to reuse already existing mathematical algorithms which are delivered by MATLAB. The MATLAB_Engine component relies on the capabilities provided by the Control System Toolbox, which is an add-on to MATLAB. To implement the various blocks which are defined on the GUI side, MATLAB_Engine uses MATLAB classes.

The Block_Database component defines a database of blocks which can be used inside PEET to build up a pointing system. It is designed as an extendable database file. This database file uses a XML file format to specify all kind of blocks. For each block, it is required to provide information about the java class implementing the user interface for a block type and the MATLAB class, which implements the mathematical behaviour for a block type. The design of the database allows the user to extend the database by simply adding a new section describing the new block type to the database file. It is also possible to export special blocks from the GUI in order to create new block types.

5.3 Software components design - General

Each of the three main components can be further subdivided into smaller subcomponents (see Figure 4-1).

The PEET_GUI component consists of the following subcomponents:

- System_Editor

- Tree_View
- Plotting
- Java_Excel_API
- Database_Browser

The System_Editor subcomponent is responsible for providing the user the ability to build up a pointing system by adding blocks and connecting them. For the graphical representation of the blocks and the connections, the Piccolo2D framework is used. This framework uses some kind of scene graph to manage the graphical elements in order to render them on the screen. Each graphical element must be added as a node to this scene graph. For this reason the base class used for all block types is inherited from the PNode class, which is the base class for all kind of nodes which can be added to the Piccolo2D scene graph. The System_Editor component is also responsible for managing the user input for block parameters. The block parameter dialogs are provided by the java block classes.

The Tree_View subcomponent provides the user the possibility to inspect the pointing error results. It visualises the pointing system in a tree like representation with the pointing error sources to the top and the final pointing error contribution block at the bottom. Some graph layout algorithms are used to build up a graph representation with as less line crossings as possible. Due to the complexity of the line crossing reduction problem, it is not guaranteed that the minimum crossing number can be found.

The Plotting component provides plots for PSDs. It will mainly be used by the Tree_View component to visualise the pointing error results.

The Database_Browser component serves as the user interface to the Block_Database component. It is used to read the database file and to export blocks from the GUI to the database file. The user can choose a block from the Database_Browser component by dragging it form the database browser and dropping it onto the system editor.

The MATLAB_Engine component consists of the following subcomponents:

- MATLAB Toolboxes
- MATLAB_Block_Classes
- Java MATLAB Interface

MATLAB Toolboxes extend the computational functionality of the MATLAB_Engine component. Only the Control System Toolbox is required by PEET. This toolbox provides additional functionality concerning the analysis and design of linear system.

The MATLAB_Block_Classes component names a set of MATLAB classes which implement the mathematical behaviour for the various block types. It is possible that several java block classes map to the same MATLAB block class. This might be the reason if block types share the same behaviour but require different parameter sets.

The Java MATLAB interface (JMI) is provided by MATLAB itself. It serves as an interface between the java classes and the MATLAB_Engine component.

The Block_Database component does not have any subcomponents.

5.4 Software components design - Aspects of each component

5.4.1 General

The various subcomponents are described in detail in the following chapters.

5.4.2 Component "System_Editor"

Identifier
PEET_GUISystem_Editor

Type
Java class

Purpose
The software requirements touched by this component are:

Req.#	Description
REQ-PEET-FUNC-0020	PEET shall provide a graphical interface that is similar to MATLAB/Simulink and provides the ability to specify PESs, systems and that allows routing of error sources between systems and PESs
REQ-PEET-FUNC-0060	PEET shall allow the use of different error source representations (times series, variances, covariance matrices, mean values, PSDs)
REQ-PEET-FUNC-0070	PEET shall be able to account for correlation between signals (at least full or no correlation)
REQ-PEET-FUNC-0100	PEET shall allow the inclusion of (fixed-structure) feedback systems
REQ-PEET-FUNC-0110	PEET shall be able to store and load systems
REQ-PEET-FUNC-0120	Each system shall be customisable by means of a fixed set of parameters
REQ-PEET-FUNC-0140	The statistical distribution types used to specify error sources shall include at least normal (Gaussian), uniform, and Rayleigh distributions
REQ-PEET-DESI-0020	THE GUI of PEET shall be implemented in Java
REQ-PEET-DESI-0040	The "Piccolo2D" framework shall be used for the implementation of the system editor
REQ-PEET-DESI-0100	A double-click on a library block (except the "Container" block) shall open a dialog where the user can specify block settings/parameters
REQ-PEET-DESI-0110	A "Container" block similar to the MATLAB/Simulink "System" block shall serve as a container for subsystems. Double-clicking the "Container" block shall open the subsystem in a separate window
REQ-PEET-MAIN-0010	The software code shall be well documented such that it can be

	easily analysed
REQ-PEET-MAIN-0020	The software code shall be produced according to a clear coding guideline that contains - naming conventions for classes, methods and variables - formatting rules - implementation advices

Function

The System_Editor component enables the user to create a pointing system by adding or removing blocks from the pointing system and to connect them. The user interface required for this task is provided by this component. It provides also the user the possibility to configure the block parameters.

Subordinates

The System_Editor owns the following subordinates:

- MATLAB_Engine.Java MATLAB interface

Dependencies

There are no dependencies for the System_Editor component.

Interfaces

The System_Editor component communicates with the MATLAB_Engine component via the Java MATLAB Interface. It will provide each MATLAB block class the user defined block parameters and the signal data for the input signals. It will receive the output signal for each block from the MATLAB block classes.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

The internal data of this component consists of a PeetCanvas object which is a subclass of PCanvas. The PCanvas class is provided by the Piccolo2D framework and manages the scene graph used by the Piccolo2D framework. Additionally, each block contains the user defined parameter data.

5.4.3 Component "Tree_View"

Identifier

PEET_GUI.Tree_View

Type

Java class

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-FUNC-0010	PEET shall allow the specification of a pointing error breakdown or budget visualized as a tree.
REQ-PEET-FUNC-0030	For each error index, an error tree shall be implementable.
REQ-PEET-FUNC-0040	PEET shall allow the visualization of breakdown and budgeting values simultaneously.
REQ-PEET-FUNC-0080	PEET shall allow different visualization types in the error tree (variances or PSD)
REQ-PEET-DESI-0020	THE GUI of PEET shall be implemented in Java
REQ-PEET-MAIN-0010	The software code shall be well documented such that it can be easily analysed
REQ-PEET-MAIN-0020	The software code shall be produced according to a clear coding guideline that contains - naming conventions for classes, methods and variables - formatting rules - implementation advices

Function

The Tree_View component is responsible for visualising the pointing error calculation results. It provides a tree like representation of the pointing system. By selecting one node inside the tree, the relevant data for the selected block will be shown to the user. The Tree_View component can visualise PSDs and time series data.

Subordinates

The System_Editor owns the following subordinates:

- MATLAB_Engine.Java MATLAB interface

Dependencies

There are no dependencies for the Tree_View component.

Interfaces

The Tree_View component communicates with the MATLAB_Engine component via the Java MATLAB Interface. It is responsible for providing the required error index to the MATLAB_Engine. It will receive all the data relevant for visualising the pointing error results from the MATLAB block classes.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

The internal data of this component consists of a PCanvas object This PCanvas class is provided by the Piccolo2D framework and manages the scene graph used by the Piccolo2D framework. Additionally, it owns some plot objects responsible for plotting time series or PSD data.

5.4.4 Component "Plotting"

Identifier

PEET_GUI.Plotting

Type

A set of java classes.

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-FUNC-0080	PEET shall allow different visualization types in the error tree (variances or PSD)
REQ-PEET-DESI-0020	THE GUI of PEET shall be implemented in Java
REQ-PEET-MAIN-0010	The software code shall be well documented such that it can be easily analysed
REQ-PEET-MAIN-0020	The software code shall be produced according to a clear coding guideline that contains - naming conventions for classes, methods and variables - formatting rules - implementation advices

Function

The Plotting component provides some plotting capabilities for power spectrum distribution (PSD) data.

Subordinates

- There are no subordinates for this component.

Dependencies

There are no dependencies for the Plotting component.

Interfaces

The plotting component receives the plot data from the Tree_View component.

Resources

No special resources are required by this component.

References

No references exist for this component

Data

The internal data of this component consists of two dimensional data sets which represent the plot data.

5.4.5 Component "Block_Database"

Identifier

Block_Database

Type

XML text file

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-DESI-0020	THE GUI of PEET shall be implemented in Java
REQ-PEET-DESI-0070	The GUI shall read the list of available blocks from a file that contains at least the name, a description and the number of inputs for each available block
REQ-PEET-DESI-0080	It shall be possible to extend PEET by further blocks
REQ-PEET-DESI-0090	Adding further blocks shall require not more than adding them to the block list, providing a JAR file with the block classes, adding the JAR file to the class path and adding the corresponding MATLAB classes (as m-file to the foreseen folder)
REQ-PEET-DATA-0010	The database shall comprise a star tracker
REQ-PEET-DATA-0012	The database shall comprise a gyroscope
REQ-PEET-DATA-0015	The database shall comprise a reaction wheel
REQ-PEET-DATA-0020	The database shall comprise a generic transfer function
REQ-PEET-DATA-0030	The database shall comprise a rigid and a flexible plant model
REQ-PEET-DATA-0040	The database shall comprise a PID controller
REQ-PEET-DATA-0050	The database shall comprise a feedback loop with a fixed structure
REQ-PEET-DATA-0060	The database shall comprise a fixed gain observer

Function

The Block_Database component defines all block types which can be used for building up a pointing system. The database can be extended by the user by adding new sections to the database file or by exporting blocks from the GUI.

Subordinates

There are no subordinates for this component.

Dependencies

This component does not have any dependencies.

Interfaces

There are no internal interfaces for this component.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

This component does not have internal data.

5.4.6 Component "Database_Browser"

Identifier

PEET_GUI.Database_Browser

Type

Java class

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-DESI-0020	The GUI of PEET shall be implemented in Java
REQ-PEET-DESI-0050	On Java side each block shall be represented by one Java class
REQ-PEET-MAIN-0010	The software code shall be well documented such that it can be easily analysed
REQ-PEET-MAIN-0020	The software code shall be produced according to a clear coding guideline that contains - naming conventions for classes, methods and variables - formatting rules - implementation advices

Function

The Database_Browser is responsible for providing a graphical interface to the block database. The user can add blocks to the current pointing system by dragging it from the database browser window onto the system editor window.

Subordinates

The System_Editor owns the following subordinates:

- Block_Database

Dependencies

The Database_Browser depends on the Block_Database component.

Interfaces

The Database_Browser component reads the available block types from the XML database file.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

The internal data of this component consists of a set of PCanvas objects. Each PCanvas object is responsible for managing the scene graph for a specific block category.

5.4.7 Component "Java_Excel_API"

Identifier

PEET_GUI.Java_Excel_API

Type

Java jar file

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-INTF-0010	PEET shall provide an interface to MS Excel that allows the import of system (block) parameters
REQ-PEET-INTF-0015	PEET shall provide an interface to Excel that allows exporting the results computed by PEET in tabular format. These exports shall cover at least all sources and individual contributors as well as the final budget values for each index.

Function

The Java_Excel_API is responsible for importing data from and exporting data to an Excel file.

Subordinates

There exist no subordinates for this component.

Dependencies

There are no dependencies for this object.

Interfaces

There are no internal interfaces for this component.

Resources

No special resources are required by this component.

References

There exist no references for this component.

Data

This component does not contain internal data.

5.4.8 Component "MATLAB Toolboxes"

Identifier

MATLAB_Engine.MATLAB Toolboxes

Type

A set of MATLAB toolboxes

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-FUNC-0050	PEET shall allow the calculation of a pointing error budget according to the computation rules defined in [AD2].
REQ-PEET-FUNC-0130	PEET shall provide blocks that are equivalent and parameter compatible with the MATLAB functions tf, ss, frd, and zpk. No data conversion between the parameters required for the MATLAB functions and their representations in PEET shall be necessary.
REQ-PEET-DESI-0010	The core of PEET shall be implemented in MATLAB

Function

The MATLAB toolboxes extend the basic functionality of the MATLAB_Engine component. The MATLAB_Engine component relies on the Control System Toolbox.

Subordinates

There exist no subordinates for this component.

Dependencies

There are no dependencies for this object.

Interfaces

There are no internal interfaces for this component.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

This component does not contain internal data.

5.4.9 Component "MATLAB_Block_Classes"

Identifier

MATLAB_Engine.MATLAB_Block_Classes

Type

A set of MATLAB classes.

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-FUNC-0050	PEET shall allow the calculation of a pointing error budget according to the computation rules defined in [AD2].
REQ-PEET-FUNC-0070	PEET shall be able to account for correlation between signals (at least full or no correlation)
REQ-PEET-FUNC-0090	PEET shall be able to compute a 3-axis error budget
REQ-PEET-FUNC-0095	PEET shall be able to compute an error budget in terms of LOS (line-of sight) and ALOS (around line-of sight) errors
REQ-PEET-FUNC-0100	PEET shall allow the inclusion of (fixed-structure) feedback systems
REQ-PEET-FUNC-0120	Each system shall be customisable by means of a fixed set of parameters
REQ-PEET-FUNC-0130	PEET shall provide blocks that are equivalent and parameter compatible with the Matlab functions tf, ss, frd, and zpk. No data conversion between the parameters required for the MATLAB functions and their representations in PEET shall be necessary.
REQ-PEET-FUNC-0140	The statistical distribution types used to specify error sources shall include at least normal (Gaussian), uniform, and Rayleigh distributions
REQ-PEET-INTF-0020	GUI and core of PEET shall communicate via the JMI interface of MATLAB
REQ-PEET-INTF-0030	PEET shall be compatible with MATLAB 2011b

REQ-PEET-DESI-0010	The core of PEET shall be implemented in MATLAB
REQ-PEET-DESI-0030	Each block shall be represented by one MATLAB class
REQ-PEET-DESI-0080	It shall be possible to extend PEET by further blocks

Function

The MATLAB_Block_Classes component provides the MATLAB classes which implement the mathematical behaviour of the various block types. The Java block type classes and the MATLAB block type classes communicate via the Java MATLAB interface. For extensibility, all MATLAB block classes belong to the same class hierarchy. This class hierarchy can be extended by the user in order to add additional block types.

Subordinates

The MATLAB_Block_Classes component owns the following subordinates:

- MATLAB_Engine.MATLAB Toolboxes

Dependencies

The MATLAB_Block_Classes component depends on the Control System Toolbox.

Interfaces

The MATLAB_Block_Classes component communicates with the System_Editor and the Tree_View component via the Java MATLAB Interface. Each MATLAB block class will receive the user defined block parameters and the input signal data from the System_Editor. The MATLAB block classes will provide the signal data of the block output port to the System_Editor and Tree_View component.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

This component handles data required for the output signal calculation. Each MATLAB block class owns its own internal block parameters. These parameters will be initialized and modified by the corresponding Java block class. The input signal data required for the output signal calculation will also be provided by the corresponding java block class.

5.4.10 Component "Java MATLAB Interface"

Identifier

MATLAB_Engine.Java MATLAB Interface

Type

Java jar file

Purpose

The software requirements touched by this component are:

Req.#	Description
REQ-PEET-INTF-0020	GUI and core of PEET shall communicate via the JMI interface of MATLAB
REQ-PEET-INTF-0030	PEET shall be compatible with MATLAB 2011b

Function

The Java MATLAB Interface is directly provided by MATLAB. It enables java classes to communicate with MATLAB.

Subordinates

There are no subordinates for this component.

Dependencies

There are no dependencies for this component.

Interfaces

No internal interfaces exist for this component.

Resources

No special resources are required by this component.

References

No references exist for this component.

Data

There exists no internal data for this component.

5.5 Internal interface design

The Interface Control Document [RD4] describes some interfaces which are also used internally between software components. The signal data structure is used as internal interface between the MATLAB_Block_Classes, the System_Editor and the Tree_View and is described in the ICD [RD4] in chapter 5.3.5.

Also the metric filter interface is used as an internal interface between the MATLAB_Engine component and the Tree_View component. This interface is described in the ICD [RD4] in chapter 5.3.6

6 Requirements to design components traceability

6.1 Traceability matrix software requirements → software components

Req.#	Software components
REQ-PEET-FUNC-0010	PEET_GUI.Tree_View
REQ-PEET-FUNC-0020	PEET_GUI.System_Editor
REQ-PEET-FUNC-0030	PEET_GUI.Tree_View
REQ-PEET-FUNC-0040	PEET_GUI.Tree_View
REQ-PEET-FUNC-0050	MATLAB_Engine.MATLAB Toolboxes, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0060	PEET_GUI.System_Editor
REQ-PEET-FUNC-0070	PEET_GUI.System_Editor, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0080	PEET_GUI.Tree_View, PEET_GUI.Plotting
REQ-PEET-FUNC-0090	MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0095	MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0100	PEET_GUI.System_Editor, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0110	PEET_GUI
REQ-PEET-FUNC-0120	PEET_GUI.System_Editor, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0130	MATLAB_Engine.MATLAB Toolboxes, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-FUNC-0140	PEET_GUI.System_Editor, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-INTF-0010	PEET_GUI.Java_Excel_API
REQ-PEET-INTF-0015	PEET_GUI.Java_Excel_API
REQ-PEET-INTF-0020	MATLAB_Engine.MATLAB_Block_Classes, MATLAB_Engine.Java MATLAB Interface
REQ-PEET-INTF-0030	MATLAB_Engine.Java MATLAB Interface
REQ-PEET-RESC-0010	-
REQ-PEET-DESI-0010	MATLAB_Engine.MATLAB Toolboxes, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-DESI-0020	PEET_GUI.System_Editor, PEET_GUI.Tree_View, PEET_GUI.Plotting, Block_Database, PEET_GUI.Database_Browser
REQ-PEET-DESI-0030	MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-DESI-0040	PEET_GUI.System_Editor
REQ-PEET-DESI-0050	PEET_GUI.Database_Browser
REQ-PEET-DESI-0060	-
REQ-PEET-DESI-0070	Block_Database

REQ-PEET-DESI-0080	Block_Database, MATLAB_Engine.MATLAB_Block_Classes
REQ-PEET-DESI-0090	Block_Database
REQ-PEET-DESI-0100	PEET_GUI.System_Editor
REQ-PEET-DESI-0110	PEET_GUI.System_Editor
REQ-PEET-DESI-0120	-
REQ-PEET-DESI-0130	-
REQ-PEET-DESI-0140	-
REQ-PEET-PORT-0010	-
REQ-PEET-PORT-0020	-
REQ-PEET-PORT-0030	-
REQ-PEET-PORT-0040	-
REQ-PEET-MAIN-0010	PEET_GUI.System_Editor, PEET_GUI.Tree_View, PEET_GUI.Plotting, PEET_GUI.Database_Browser, PEET_GUI
REQ-PEET-MAIN-0020	PEET_GUI.System_Editor, PEET_GUI.Tree_View, PEET_GUI.Plotting, PEET_GUI.Database_Browser, PEET_GUI
REQ-PEET-CONF-0010	-
REQ-PEET-DATA-0010	Block_Database
REQ-PEET-DATA-0012	Block_Database
REQ-PEET-DATA-0015	Block_Database
REQ-PEET-DATA-0020	Block_Database
REQ-PEET-DATA-0030	Block_Database
REQ-PEET-DATA-0040	Block_Database
REQ-PEET-DATA-0050	Block_Database
REQ-PEET-HUMF-0010	-
REQ-PEET-INST-0010	-

6.2 Traceability matrix software components → software requirements

Software components	Req.#
PEET_GUI.Tree_View	REQ-PEET-FUNC-0010, REQ-PEET-FUNC-0030, REQ-PEET-FUNC-0040, REQ-PEET-FUNC-0080, REQ-PEET-DESI-0020, REQ-PEET-MAIN-0010, REQ-PEET-MAIN-0020
PEET_GUI.System_Editor	REQ-PEET-FUNC-0020, REQ-PEET-FUNC-0060, REQ-PEET-FUNC-0070, REQ-PEET-FUNC-0100, REQ-PEET-FUNC-0120, REQ-PEET-FUNC-0140, REQ-PEET-DESI-0020, REQ-PEET-DESI-0040, REQ-PEET-DESI-0100, REQ-PEET-DESI-0110, REQ-PEET-MAIN-0010, REQ-PEET-MAIN-0020

PEET_GUI.Plotting	REQ-PEET-FUNC-0080, REQ-PEET-DESI-0020, REQ-PEET-MAIN-0010, REQ-PEET-MAIN-0020
Block_Database	REQ-PEET-DESI-0020, REQ-PEET-DESI-0070, REQ-PEET-DESI-0080, REQ-PEET-DESI-0090, REQ-PEET-DATA-0010, REQ-PEET-DATA-0012, REQ-PEET-DATA-0015, REQ-PEET-DATA-0020, REQ-PEET-DATA-0030, REQ-PEET-DATA-0040, REQ-PEET-DATA-0050
PEET_GUI.Database_Browser	REQ-PEET-DESI-0020, REQ-PEET-DESI-0050, REQ-PEET-MAIN-0010, REQ-PEET-MAIN-0020
PEET_GUI.Java_Excel_API	REQ-PEET-INTF-0010, REQ-PEET-INTF-0015
MATLAB_Engine.MATLAB Toolboxes	REQ-PEET-FUNC-0050, REQ-PEET-FUNC-0130, REQ-PEET-DESI-0010
MATLAB_Engine.MATLAB_Block_Classes	REQ-PEET-FUNC-0050, REQ-PEET-FUNC-0070, REQ-PEET-FUNC-0090, REQ-PEET-FUNC-0095, REQ-PEET-FUNC-0100, REQ-PEET-FUNC-0120, REQ-PEET-FUNC-0130, REQ-PEET-FUNC-0140, REQ-PEET-INTF-0020, REQ-PEET-DESI-0010, REQ-PEET-DESI-0030, REQ-PEET-DESI-0080
MATLAB_Engine.Java MATLAB Interface	REQ-PEET-INTF-0020, REQ-PEET-INTF-0030
PEET_GUI	REQ-PEET-FUNC-0110, REQ-PEET-MAIN-0010, REQ-PEET-MAIN-0020