Pointing Error Engineering Tool

Interface Control Document

esa iFR

Document Number:	ASTOS-PEET-ICD-001		**Date:**
Issue:	1.4		2013-07-26

	Name/Function	**Organization**	**Signature**	
Prepared by:	J. Eggert	Astos Solutions	[illegible]	2013-07-26
	S. Weikert	Astos Solutions	[illegible]	2013-07-26
	H. Su	iFR	[illegible]	2013-07-26
	M.Hirth	iFR	[illegible]	2013-07-26
Checked by:	S. Weikert	Astos Solutions	[illegible]	2013-07-26
Product Assurance:				-
Project Management:	S. Weikert	Astos Solutions	[illegible]	2013-07-26

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
1.3	2012-11-13	5.3.1.1, A.1	Added global parameter
1.4	2013-07-26	5.3.1.2	Updated block list settings

Table of Contents

1 Introduction

This is the Pointing Error Engineering Tool (PEET) Interface Control Document (ICD). This document defines the external interfaces between different software components of PEET in detail, while the Architectural Design Document (ADD) describes the concept and the rationale of these interfaces.

2 Applicable and Reference Documents

2.1 Applicable Documents

[AD1] ASTOS-PEET-SDR-001_Iss1.1, Pointing Error Engineering Tool Software Requirements Document, 2012

[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

None

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 Overview

The software overview is shown in figure 4-1.

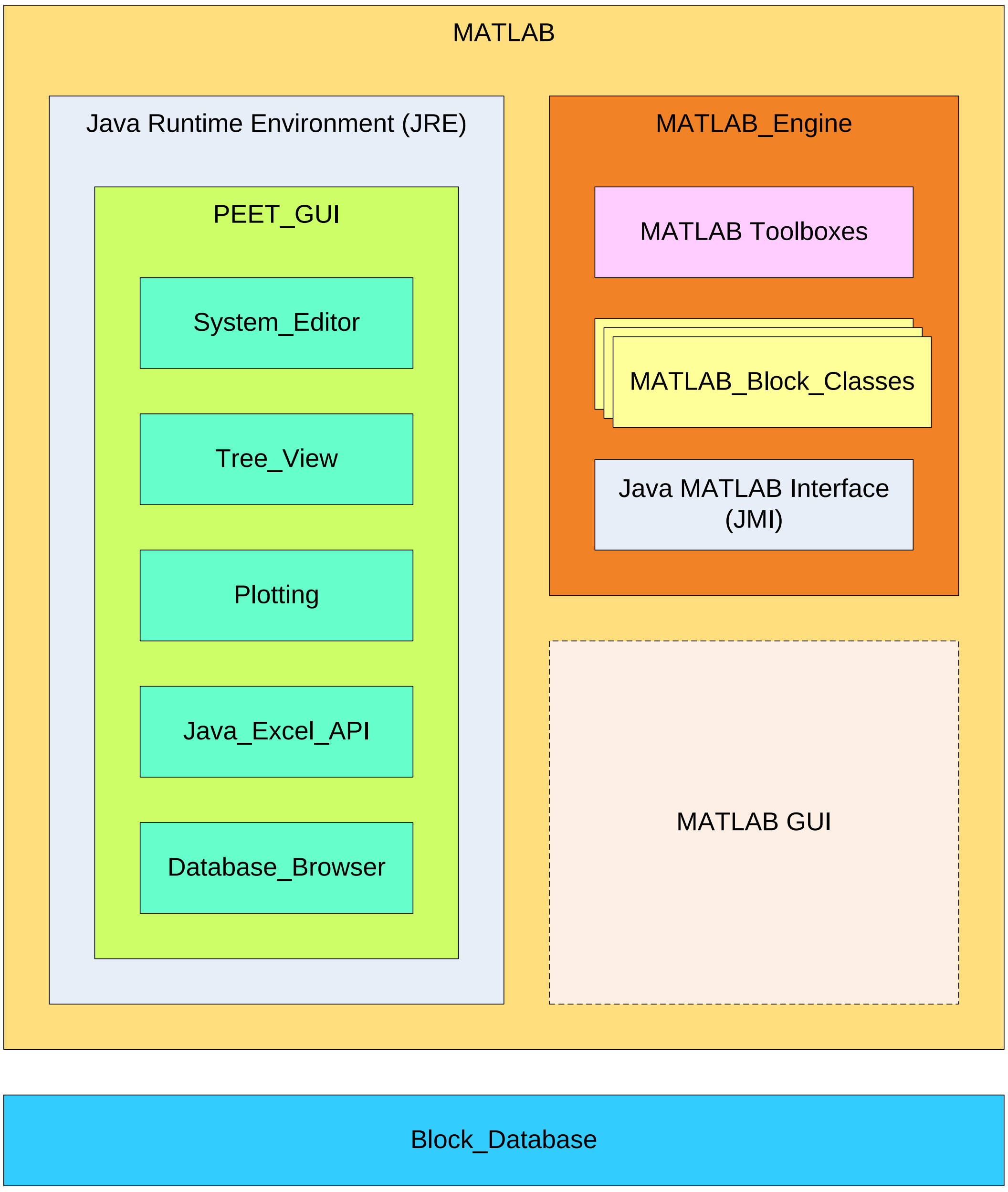

Figure 4-1: PEET static architecture

PEET can be divided into three main components. The PEET_GUI component is providing the graphical user interface and is implemented in Java. The MATLAB_Engine component is responsible for the mathematical computations performed by PEET and is implemented by MATLAB classes. The third component is the Block_Database. The

Block_Database component defines all blocks which can be used to build up a pointing system.

5 Requirements and Design

5.1 General provisions to the requirements in the IRD

All requirements are defined in the Software Requirements Document [AD1].

5.2 Interface requirements

The requirements need not to be further detailed.

5.3 Interface design

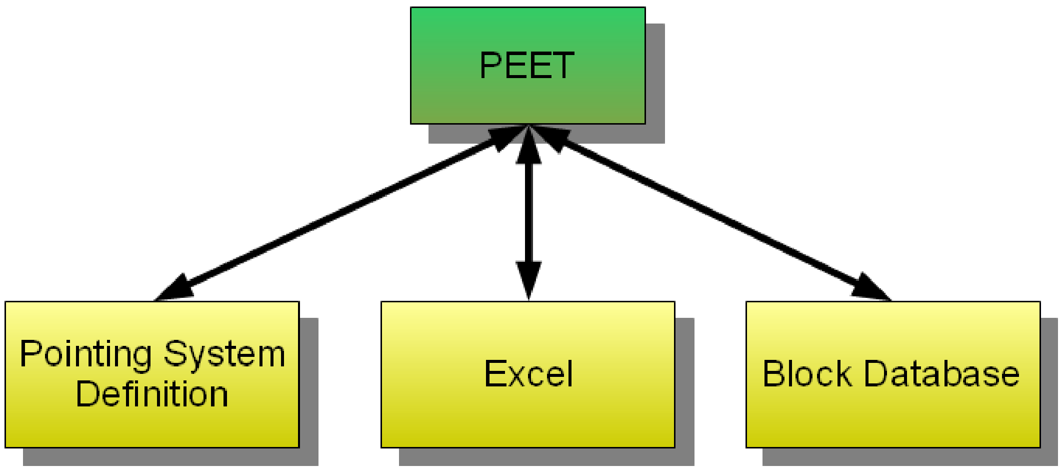

Figure 5-1 External Interfaces Overview

Figure 5-1 gives an overview of the interfaces between PEET and external components. The pointing system definition is realised by a XML file, describing all the data needed to define the pointing system. This data contains all the blocks, subsystems and pointing error sources, which build up the entire pointing system and their user defined parameter settings. It also contains data describing the parameters relevant to the whole pointing system. Also the connection information will be provided by this file. For more details see 5.3.1.

PEET will also provide the user the capability to link parameter values or other user input to data defined in an Excel file or in the MATLAB workspace. In this case the user will specify an excel file, the sheet and the cell which contains the data the user wants to use or he provides the name of a MATLAB variable. For more details about the Excel import capability, see 5.3.2.

PEET will deliver a block database containing a list of basic building blocks. This block database can easily be extended by the user. To extend the block database, the user must provide a Java class which implements the visual representation of the new block type. This new block type must be added to the block database. This will be done by inserting the java class name, the corresponding MATLAB class name and a user defined block category to the block database configuration file. For more details see 5.3.3.

External interfaces between software components of PEET will be described in detail in the chapters 5.3.4 till 5.3.6.

5.3.1 Pointing System definition

The pointing system definition is completely realized as a XML file. This XML file is called PointingSystemDefinition.xml and must be located in the input folder of the problem folder. This file contains all the data needed to define the whole pointing system. For this purpose, the XML file contains the three major sections *<Pointing_System_Settings>, <Block_List>,* and *<Connection_List>.* For a simple example see A.1.

5.3.1.1 Pointing System Settings

The *<Pointing_System_Settings>* section contains all settings which are applied to the whole pointing system. See the following table for all possible entries.

<Pointing_System_Settings>		
<Error_Indices>	List	A list of error indices applicable to the current pointing system
<Correlations>	Record	The definition of the correlations between different PES
<Global_Settings>	Record	The definition of all kind of global parameters.

The `<Error_Indices>` tag contains a list of all available error indices. Each element of the list is encapsulated by an `<Item>` tag. The content of the `<Item>` tag is described in detail in the following table.

Error index definition		
<Type >	String	The type of the error index (see list below)
<ID>	String	A unique identifier for the error index definition
<Statistical_Interpretation>	String	The statistical interpretation. Possible values are `Ensemble`, `Temporal` and `Mixed`
<Confidence_Coefficient>	double	The confidence coefficient used for the final pointing error.

The `<Type>` tag can contain the following values:

- `Absolute Knowledge Error`
- `Absolute Performance Error`
- `Mean Knowledge Error`
- `Mean Performance Error`
- `Relative Knowledge Error`
- `Relative Performance Error`
- `Knowledge Drift Error`
- `Performance Drift Error`
- `Knowledge Reproducibility Error`
- `Performance Reproducibility Error`

In case the `<Type>` tag contains the values `Mean Knowledge Error`, `Mean Performance Error`, `Relative Knowledge Error`, `Relative Performance Error`, `Knowledge Drift Error`, `Performance Drift Error`, `Knowledge Reproducibility Error` or `Performance Reproducibility Error`, the error index definition must also contain the following entries.

Window Time Indices		
<Window_Time>	double	The time interval for the window time

In case the `<Type>` tag contains the values `Knowledge Drift Error`, `Performance Drift Error`, `Knowledge Reproducibility Error` or `Performance Reproducibility Error`, the error index definition must also contain the following entries.

Stability Time Indices		
<Stability_Time>	double	The stability time

The `<Correlations>` tag contains the correlations between different pointing error sources. Each correlation must be encapsulated by an `<Item>` tag. The content of the `<Item>` tag is listed in the following table.

Correlation		
<First_Source>	String	The unique ID of the first PES
<Second_Source>	String	The unique ID of the second PES
<Random_Variable_Correlation>	String	The correlation type for random variables. Possible values are `Uncorrelated` or `Fully correlated`
<Random_Process_Correlation>	String	The correlation type for random processes. Possible values are `Uncorrelated` or `Fully correlated`

The `<Global_Settings>` tag contains all kind of global parameters. These global parameters are listed in the table below.

<Global_Settings>		
<Frequency_Points>	double	The number of frequency points used for the error evaluation.
<Use_User_Frequencies>	boolean	A flag indicating if the evaluation bandwidth should be computed by PEET or if an user defined evaluation bandwidth will be used.
<Line_Of_Sight>	String	The line-of-sight axis. Possible values are `x-Axis`, `y-Axis` or `z-Axis`

In case the flag `<Use_User_Frequencies>` is set to true, two additional parameters are available.

<Global_Settings>		
<Lower_Frequency>	double	The exponent used for the lower frequency.
<Upper_Frequency>	double	The exponent used for the upper frequency.

The lower and upper frequency are always defined as $1x10^{exp}$. For this reason, the two parameters are containing the exponent used for the lower and upper frequency

5.3.1.2 Block List Settings

The `<Block_List>` section defines a list of all blocks added to the pointing system. For each block, this list contains one data block enclosed by an `<Item>` tag, describing the block settings. See the following table for all possible parameters.

General Block Parameters		
<ID>	String	The fully qualified and unique identifier for the block
<Block_Type>	String	The type of the block. Because the user can add new block types to PEET, the values for this tag are not fixed.
<Position>	List	The block position used by the GUI
<Block_Settings>	List	List of special block settings depending on the block type

Every data block owns some general parameters, which are available for every data block, and a `<Block_Settings>` section. The `<Block_Settings>` section contains a list of special block parameter settings which depend on the block type. New block types created by the user and added to PEET should only add their special parameter settings to this section.

Each block owns a fully qualified and unique identifier. This identifier is build up by the fully qualified identifier of the parent block and the unique block identifier separated by a dot (e.g. parentID.blockID). Block identifiers must only be unique within a container. Because container blocks are also part of other containers or the pointing system itself, the parentID.blockID pattern will create a fully qualified identifier which is unique for all blocks.

The next tables list the possible block settings for every block type provided by PEET.

Container

This block does not provide block settings. The `<Block_Settings>` tag is empty.

Coordinate Transformation

Block Settings		
<Rotation_Sequence>	String	The rotation sequence. Possible values are `1-2-3`, `1-3-2`, `2-1-3`, `2-3-1`, `3-1-2`, `3-2-1`, `1-2-1`, `1-3-1`, `2-1-2`, `2-3-2`, `3-1-3` and `3-2-3`
<Phi>	double	The first rotation angle
<Theta>	double	The second rotation angle
<Psi>	double	The third rotation angle

Dynamic System

Block Settings		
<Representation>	String	The representation type. Possible values are `Transfer function`, `Frequency-Response`, `Zero-Pole-Gain` and `State space`

Depending on the content of the `<Representation>` content, the parameter settings block contains the following data.

Representation: Transfer Function		
<Numerators>	List	A list of 9 coefficients sets for the numerator of the transfer function
<Denominators	List	A list of 9 coefficients sets for the denominator of the transfer function

Each coefficients set must be enclosed by an `<Item>` tag. The content of the `<Item>` tag must be provided in the following format.

```
<Item>
  <Item>0.0</Item>
  ....
</Item>
```

Representation: Frequency-Response		
<Frequency_Response>	List	A list of 9 frequency response matrices

Each frequency-response matrix must be enclosed by an `<Item>` tag. The content of the `<Item>` tag is similar to the format of the correlation matrix, except that the number of columns is fixed to 2 and the number of rows is not fixed.

Representation: Zeros-Poles-Gain		
<Zeros>	List	A list of 9 sets of numerical values
<Poles>	List	A list of 9 sets of numerical values
<Gains>	List	A list of 9 gain values

Each set of numerical values for the `<Zeros>` and `<Poles>` must be enclosed by an `<Item>` tag. The content of the `<Item>` tag is similar to the format of the correlation matrix, except that the number of columns is fixed to 1 and the number of rows is not fixed. Each of the 9 gain values must also be encapsulated by an `<Item>` tag.

Representation: State space		
<State_Variables>	int	Number of state variables

<A_Matrix>	List	The A matrix
<B_Matrix>	List	The B matrix
<C_Matrix>	List	The C matrix
<D_Matrix>	List	The D matrix

Each matrix must stick to the same format as the correlation matrix, except that the number of rows and columns are depending on the number of state variables.

Feedback System

The figure below shows the internal structure of the feedback system block. The nodes labelled A to G are used for the input and output port definition. The blocks labelled 1 to 6 are the internal blocks of the feedback system block.

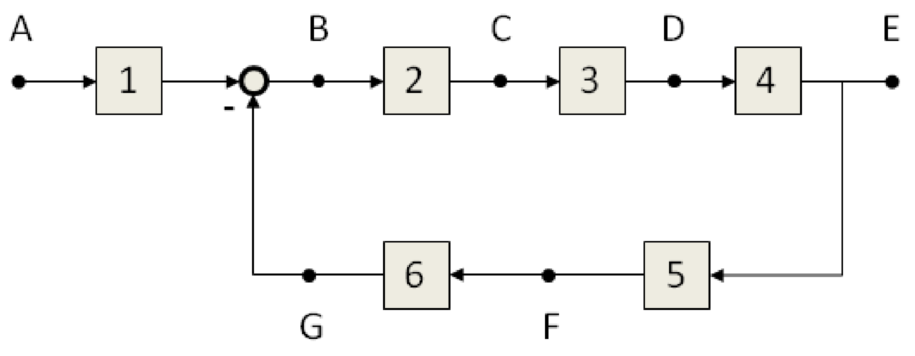

Figure 5-2 Feedback system structure

Block Settings		
<Input_Nodes>	List	A list of strings specifying the input nodes. Possible values are `A, B, C, D, E, F` and `G`
<Output_Node>	String	The ID of the output node. Possible values are `A, B, C, D, E, F` and `G`. It is not allowed to use an output node which is already used in the input nodes list.
<Internal_Block_Settings>	List	List of block settings for the internal blocks

The `<Internal_Block_Settings>` tag contains the block settings for the internal blocks. The internal blocks can use most non-PES block types available in the block database (all which have only input/output). These block types are Coordinate Transformation, Dynamic System, Flexible Plant, PID Controller, Rigid Plant and Static System. Using one of these blocks, controller transfer functions of any kind can be set up inside the feedback system. The parameters required by these block types are listed in the associated sections.

Flexible Plant

Block Settings		
<Modes>	double	The number of modes.
<Inertia>	3x3 matrix	The inertia matrix.
<Coupling_Coefficients>	nx3 matrix	The coupling coefficients for each mode an axis.
<Cantilever_Frequency>	nx1 matrix	The cantilever frequency for each mode.
<Damping_Ratio>	nx1 matrix	The damping ratio for each mode.

Gyro-Stellar Estimator

Block Settings		
<Kalman_Gain_1>	1x3 matrix	A matrix describing the first Kalman gain
<Kalman_Gain_2>	1x3 matrix	A matrix describing the second Kalman gain

Gyro Rate Noise

Block Settings		
< Min_Pole_Order >	double	An optional minimum pole order for rational fit of PSD.
< Max_Pole_Order >	double	An optional maximum pole order for rational fit of PSD.
<Frequency_Point_Number>	double	An optional frequency point number for rational fitting. This tag is only available, if the user has provided data for it.
<Random_Walk_Angle>	double	The magnitude of the angle random walk.
<Random_Walk_Rate>	double	The magnitude of the rate random walk.
<Instability_Bias>	double	The magnitude of the bias instability.
<Quantization_Noise>	double	The magnitude of the quantization noise.
<Time_Window>	double	The time window.

Mapping Block

Block Settings		
<Number_Of_Devices>	int	Number of devices which will be mapped
<Mapping_Matrix>	nx3 Matrix	A matrix describing the mapping for the devices. The number of rows depends on the number of devices

PEC (Pointing)

This block does not have any block settings. The `<Block_Settings>` tag is empty.

PEC (Position)

Block Settings

<Summation_Rule>	string	The summation rule for the total error. Possible values are `Exact` and `Worst case`
<Attitude_Contributor_Count>	int	The number of attitude N of attitude contributors (creates N additional inputs)
<Attitude_Coupling_Vector >	Nx3 Matrix	Matrix containing the elements of the N coupling vectors corresponding to the attitude contributor inputs

PES

Block Settings		
<Signal_Dimension>	String	The dimension of the signal. Possible values are `1D` or `3D`.
<Use_Constant_Part>	boolean	A flag indicating if the pointing error source contains a time constant part.
<Use_Random_Part>	boolean	A flag indicating if the pointing error source contains a time random part.
<Time_Constant>	-	The time constant parameters of the PES
<Time_Random>	-	The time random parameters of the PES

The `<Time_Constant>` tag contains all data relevant to the time constant part of the pointing error source definition. The next table will show the content for this tag.

Time Constant		
<Distribution_Type>	String	The distribution type of the time constant part. Possible values are `Discrete, Uniform, Bimodal, Gaussian` and `Rayleigh`

Depending on the value of the distribution type, the following parameters must be provided.

Distribution: Discrete		
<Mean_Value>	double / 1x3 matrix	The mean value of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector

Distribution: Uniform		
<Minimum>	double / 1x3 matrix	The minimum value of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector
<Maximum>	double / 1x3 matrix	The maximum value of the distribution. If the PES is one dimensional, this is a scalar value,

		otherwise it is a three dimensional vector
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

Distribution: Bimodal		
<Amplitude>	double / 1x3 matrix	The amplitude of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

Distribution: Gaussian		
<Mean_Value>	double / 1x3 matrix	The mean value of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector
<Standard_Deviation>	double / 1x3 matrix	The standard deviation of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector.
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

Distribution: Rayleigh		
<Rayleigh_Parameter>	double / 1x3 matrix	The parameter for a Rayleigh distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector.
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

The `<Time_Random>` tag contains all parameters relevant to the time random part of the pointing error source.

Time Random		
<Representation>	String	The representation of the time random part. Possible values are `Random variable` and `Random process`

Depending on the representation type, different parameters are available inside the `<Time_Random>` tag.

The parameters for the random variable representation are listed below.

Random Variable		
<Distribution_Type>	String	The distribution type used for the random variable. Possible values are `Uniform`, `Gaussian` and `Drift`

The parameters required by the *Uniform* are equal to the parameters for the time constant part.

The list below lists all available parameters for the Gaussian distribution.

Gaussian		
<Mean_Value>	double / 1x3 matrix	The mean value of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector
<Standard_Deviation_Distribution>	String	The distribution type of the standard deviation. Possible values are `Discrete` and `Uniform`
<Standard_Deviation >	double / 1x3 matrix	The standard deviation of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector. This tag is only available, if variance distribution is set to `Discrete`.
<Minimum>	double / 3x3 matrix	The variance of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional matrix. This tag is only available, if variance distribution is set to `Uniform`.
<Maximum>	double / 3x3 matrix	The variance of the distribution. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional matrix. This tag is only available, if variance distribution is set to `Uniform`.

The parameters needed by the `Drift` distribution are listed in the next table.

Drift		
<Reset_Time>	Double / 1x3 matrix	The reset time for the drift. In case of a three dimensional error source, a reset time for each axis is stored.
<Drift_Distribution>	String	The distribution type of the drift. Possible values are *Discrete*, *Uniform*, *Gaussian* and *Bimodal*

Depending on the value for <Drift_Distribution>, there are additional parameters.

Drift_Distribution: Discrete		
<Drift_Rate>	double /1x3 matrix	The drift rate. In case of a three dimensional error source, a drift rate for each axis is defined.

Drift_Distribution: Uniform		
<Minimum_Rate>	double / 1x3 matrix	The minimum drift rate. In case of a three dimensional error source, a minimum drift rate for each axis is defined.
<Maximum_Rate>	double / 1x3 matrix	The maximum drift rate. In case of a three dimensional error source, a maximum drift rate for each axis is defined.

Drift_Distribution: Gaussian		
<Mean_Rate>	double / 1x3 matrix	The mean drift rate. In case of a three dimensional error source, a mean drift rate for each axis is defined.
<Variance>	double / 3x3 matrix	The variance of the drift rate. In case of a three dimensional error source, a three dimensional variance matrix is defined.

Drift_Distribution: Bimodal		
<Amplitude>	double /1x3 matrix	The amplitude. In case of a three dimensional error source, an amplitude for each axis is defined.

For the random process definition, the following parameters are defined:

Random Process		
<Type>	String	The type of the random process. Possible values are `Time series, PSD, Covariance` and `Periodic`

Depending on the random process type, the `<Time_Random>` tag contains the following data.

Random Process Type: Time Series		
<Min_Pole_Order>	double	The minimum pole order used for the rational fit of the time series data.
<Max_Pole_Order>	double	The maximum pole order used for the rational fit of the time series data.
<Time_Series>	nx2 matrix /	The time series data. In case of a three

	nx4 matrix	dimensional data, time series data for each axis is defined.

Random Process Type: PSD		
<PSD_Representation>	String	The representation of the PSD. Possible values are `Transfer Function, Frequency-Response, Spectrum magnitude, Zeros-Poles-Gain` and `State space`

Depending on the PSD representation, the `<Time_Random>` part contains additional data. This additional data is similar to the data described for the Dynamic System. In case of the PSD representation type `Spectrum magnitude`, the following parameters are available.

Spectrum magnitude		
<Frequency_Magnitude>	nx2 matrix / nx4 matrix	The frequency magnitude data, In case of a three dimensional error source, a magnitude for each axis is defined.
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

Random Process Type: Covariance		
<Sampling_Time>	double	The sampling time
<Variance>	double / 1x3 matrix	The variance. If the PES is one dimensional, this is a scalar value, otherwise it is a three dimensional vector.
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`

Random Process Type: Periodic		
<Amplitude_Distribution>	String	The distribution used for the amplitude. Possible values are `Discrete` or `Uniform`.
<Correlation>	String	Only available in case of a three dimensional error source. The correlation can either be `Uncorrelated` or `Fully correlated`
<Frequency>	Matrix	A matrix containing frequency-amplitude data. Depending on the amplitude distribution and the signal dimension, the matrix dimensions are nx2 (1D, discrete), nx3 (1D, uniform), nx4 (3D, discrete) or nx7 (3D, uniform).

PID Controller

Block Settings		
<Proportional_Gains>	1x3 matrix	A vector of the proportional controller gains.
<Integral_Gains>	1x3 matrix	A vector of the integral controller gains.
<Differential_Gains>	1x3 matrix	A vector of the differential controller gains.

Reaction Wheel Force

Block Settings		
<Wheel_Mass>	double	The mass of the reaction wheel.
<Speed>	double	The rotational speed of the reaction wheel
<Broadband_Force_Noise>	String	The representation type of the broadband noise. Possible values are `Standard deviation`, `Band-limited` and `PSD`.
<Radial_Mode>	-	The radial mode parameters of the block.
<Axial_Mode>	-	The axial mode parameters of the block. The presence of this tag is optional.

The `<Radial_Mode>` tag contains all data relevant to the radial mode of the reaction wheel force model. The next table shows the content for this tag.

Radial Mode		
<Translation_Mode_Frequency>	double	The mode frequency of the axial translation mode.
<Translation_Mode_Damping>	double	The damping of the axial translation mode.
<PSD>	-	Parameters for the radial broadband noise in PSD representation. Only available if broadband noise is set to `PSD`.Parameter data corresponds to content of table 'Random Process Type: PSD'
<Noise_Standard_Deviation>	double	Standard deviation of the radial broadband noise. Only available if broadband noise is set to `Standard deviation` or `Band-limited`.
<Noise_Bandwidth>	double	The bandwidth of the radial broadband noise. Only available if broadband noise is set to `Band-limited`.
<Tonal_Disturbance>	String	The representation type of the tonal disturbance. Possible values are `by imbalance` and `by harmonics`.
<Static_Imbalance_Coefficient>	double	The static imbalance coefficient. Only required in case of tonal disturbance `By imbalance`.

<Harmonics_Count>	int	Number N of harmonics to realized. Only required in case of tonal disturbance `By harmonics`.
<Amplitude_Coefficients>	1xN matrix	Vector of N amplitude coefficients. Only required in case of tonal disturbance `By harmonics`.
<Harmonic_Numbers>	1xN matrix	Vector of N harmonic numbers corresponding to the N amplitude coefficients. Only required in case of tonal disturbance `By harmonics`.

The `<Axial_Mode>` tag contains all data relevant to the axial mode of the reaction wheel force model. As the parameters are fully equivalent to those of the radial mode, they will not be repeated again.

Reaction Wheel Torque

Block Settings		
<Inertia_About_Spin_Axis>	double	The inertia Izz about the wheel spin axis
<Inertia_Perpendicular_Spin_Axis>	double	The inertia Irr perpendicular to the wheel spin axis.
<Speed>	double	The rotational speed of the reaction wheel.
<Rock_Mode_Frequency>	double	The mode frequency of the rocking mode.
<Rock_Mode_Damping>	double	The damping of the rocking mode.
<Broadband_Torque_Noise>	String	The representation type of the broadband noise. Possible values are `Standard deviation, Band-limited` and `PSD`.
<PSD>	-	Parameters for the broadband noise in PSD representation. Only available if broadband noise is set to `PSD`. Parameter data corresponds to content of table 'Random Process Type: PSD'
<Noise_Standard_Deviation>	double	Standard deviation of the broadband noise. Only available if broadband noise is set to `Standard deviation` or `Band-limited`.
<Noise_Bandwidth>	double	The bandwidth of the broadband noise. Only available if broadband noise is set to `Band-limited`.
<Tonal_Disturbance>	String	The representation type of the tonal disturbance. Possible values are `by imbalance` and `by harmonics`.

<Dynamicc_Imbalance_Coefficient>	double	The dynamic imbalance coefficient. Only required in case of tonal disturbance `By imbalance`.
<Harmonics_Count>	int	Number N of harmonics to realized. Only required in case of tonal disturbance `By harmonics`.
<Amplitude_Coefficients>	1xN matrix	Vector of N amplitude coefficients. Only required in case of tonal disturbance `By harmonics`.
<Harmonic_Numbers>	1xN matrix	Vector of N harmonic numbers corresponding to the N amplitude coefficients. Only required in case of tonal disturbance `By harmonics`.

Rigid Plant

The parameters required by the Plant block are listed below.

Plant Block Settings		
<Inertia>	3x3 matrix	A 3x3 matrix containing the inertia

Static System

Block Settings		
<System_Matrix>	3x3 matrix	The system matrix

Star Tracker Noise

Block Settings		
<Detector_Pixel>	double	Number of detector pixels
<Field_Of_View>	double	Field of view of the sensor camera head
<FOV_Noise_Level>	double	The low frequency level of the field of view noise for all axes
<Avg_Number_Stars>	double	Average number of stars tracked by the sensor
<Angular_Velocity>	double	Average spacecraft angular velocity
<Alpha_Angle>	double	The angle between the star image direction of motion on the detector matrix and the reference axis
<Beta_Angle>	double	The angle between the sensor boresight and the spacecraft rotation axis
<Boresight_Noise_Level>	double	The low frequency noise level for the sensor boresight axis
<Cross_Axes_Noise_Level>	double	The low frequency noise level for the cross boresight axes

<Filter_Dumping_Coef>	double	The damping coefficient used for the pixel noise transfer function
<Centroiding_Window_Size>	double	The size of the centroiding window in pixels

SUM

The `<Block_Settings>` contains only one entry listed below

Block Settings		
<Number_Of_Inputs>	int	The number of input ports of the summation block

5.3.1.3 Connection Settings

The third main section `<Connection_List>` defines the interconnection of blocks. This section is a list containing all connections in the pointing system. Each connection is defined by a data block enclosed by the `<Item>` tag. See the following table for all possible parameters.

Connection Parameters		
<From>	List	The definition of the source block
<To>	List	The definition of the sink block
<Vertex_List>	List	A list of vertices describing the path of the connection used inside the GUI

The `<From>` tag and the `<To>` tag are containing the unique ID of the block and the port number to which the connection is attached. The format of those 2 tags is shown below.

```
<From>
  <Block_ID>ID of the block</Block_ID>
  <Output_Port_Index>0</Output_Port_Index>
</From>

<To>
  <Block_ID>ID of the block</Block_ID>
  <Input_Port_Index>4</Input_Port_Index>
</To>
```

The `<Vertex_List>` tag contains a list of points used for the graphical representation of the connection. The format of the positional data is shown below.

```
<Item>
  <X>0.0</X>
  <Y>0.0</Y>
</Item>
```

5.3.2 Excel

For the interface to Excel, PEET uses the Java Excel API.

5.3.2.1 Import

To import data from an Excel file, the user must provide a string of the form

```
[FileName.xls]SheetName!Cell
```

For Cell, the user can either specify a single cell or a range of cells.

5.3.2.2 Export

Data export to Excel files is supported by writing the data to a new Excel file. If a file already exists, this file will be overwritten with the exported data.

5.3.3 Block Database

PEET provides a huge block database to the user. This block database contains a lot of basic blocks from which a pointing system can be build up. The block database is defined by a XML file named block_database.xml which is located in the bin folder of the PEET installation. This XML file defines all block types available to PEET. For a small example see A.2.

The block database is extendable by simply creating a java class, a MATLAB class and by adding a new entry to the block_database.xml file. Additionally, the path to the java class must be added to the static java class path of MATLAB. For the java class, PEET provides an abstract base class named AbstractBlock, which defines the java interface common to all blocks. The user is free to choose a category name. There is no restriction to this. The structure of the XML database file will be described in the next chapter.

5.3.3.1 Database XML structure

The root tag of the database file is `<Block_Database>`. This tag contains a list of block definitions. Each block definition must be surrounded by the `<Block>` tag. The general block data contained by the `<Block>` tag is summarized in the following table.

General block definition fields		
<Record_Type>	String	The type of the block definition. Possible values are `Block` and `User Defined Container`
<Block_Type>	String	A string identifying the name of the block type
<Block_Category>	String	A string identifying the name of the block category

Depending on the record type, the `<Block>` tag contains additional data. The record type `Block` defines a single block. The additional tags are listed in the next table.

Fields for record type Block		
<Java_Class>	String	The name of the java class implementing the graphical user interface for the block type
<Matlab_Class>	String	The name of the MATLAB class implementing the mathematical behaviour of the block type

The record type `User Defined Container` represents a special kind of block definition. This definition is used by block types which are created by the container block export function of the GUI. The additional tags for this kind of block definition are listed in the next table.

Fields for record type User Defined Container		
<Internal_Blocks>	List	A list of blocks contained by the surrounding container block
<Internal_Connections>	List	A list of connections inside the container block

The internal block list contains the definition of all blocks internal to the container block, which was exported as block type. Each internal block must be surrounded by an `<Item>` tag. The content of the `<Item>` tag depends on the block type but is similar to the formats described in chapter 5.3.1.2.

The `<Internal_Connections>` tag contains a list of all connections inside the container block type. Each connection must be surrounded by an `<Item>` tag. The content of the `<Item>` tag is similar to the format described in chapter 5.3.1.3.

5.3.4 MATLAB class interface

This section describes the external interface of the MATLAB block classes. Each MATLAB class must provide the following three functions:

- Class constructor

The constructor function has the same name as the MATLAB class and does not require any parameters.

- `[returnCode, returnMessage] initializeParams(obj, parameterStruct)`

The `initializeParams` function takes a reference to the MATLAB block class instance and a reference to the MATLAB parameter structure class instance containing the block settings. It returns a return code and a message. The return code must be an integer value. Possible values are 0 (success), 1 (warning) and 2 (error). In case of return code 1 and 2, the `returnMessage` contains the information about what has happened. In case of a successful function call (return code 0), `returnMessage` can be any kind of string.

- `[outputSignal, returnCode, returnMessage] computeOutput(obj, inputSignals)`

The `computeOutput` function takes a reference to the MATLAB block class instance and a list of input signals and returns the output signal, a return code and a message. The structure for the input signals and the output signal is described in chapter 5.3.5. The return code must be an integer value. Possible values are 0 (success), 1 (warning) and 2 (error). In case of return code 1 and 2, the `returnMessage` contains the information about what has happened. In case of a successful function call (return code 0), `returnMessage` can be any kind of string.

5.3.5 Signal data structure definition

This chapter describes the structure of the signal data used in the interface of the MATLAB classes. The table below shows all fields provided by the signal class.

Signal Structure			
Field alias	Subfields alias	Data type	Description
randVariable	-	-	Time-constant random variable
	meanVal	Structure	Mean value on x, y, z axis, for each statistical interpretation.
	covMat	Structure	Variance of x, y, z axis on diagonal, and covariance between axes for remaining entries, for each statistical interpretation.
constRandVariable	-	-	Time-random random variable
	meanVal	Structure	Mean value on x, y, z axis, for each statistical interpretation.
	covMat	Structure	Variance of x, y, z axis on diagonal, and covariance between axes for remaining entries, for each statistical interpretation.
randProcess	-	-	Random process
	type	String	Type of the random process
	data	Structure	A structure containing the random process signal data depending on the type.
periodic	-	-	Periodic process
	frequency	MATLAB numerical	Frequency of periodic signal
	minCovarianceMat	3×3 matrix	Minimum variance of x, y, z axis on diagonal, and covariance between axes for the rest entries
	maxCovarianceMat	3×3 matrix	Maximum variance of x, y, z axis on diagonal, and covariance between axes for the rest entries
drift	-	-	Drift error
	meanVal	Structure	Mean value on x, y, z axis, for each statistical interpretation.
	covMat	Structure	Variance of x, y, z axis on diagonal, and covariance between axes for remaining entries, for each statistical interpretation.
	resetTime	Structure	Reset time of x, y, z for each statistical interpretation.
history		Signal history class	Record signal history of system transfer

isConstRandVariable	-	MATLAB numerical	Flag indicating if the signal owns a constant random variable part
isRandVariable	-	MATLAB numerical	Flag indicating if the signal owns a random variable part
isDrift	-	MATLAB numerical	Flag indicating if the signal owns a drift part
isRandProcess	-	MATLAB numerical	Flag indicating if the signal owns a random process part
isPeriodic	-	MATLAB numerical	Flag indicating if the signal owns a periodic part
dim	-	MATLAB numerical	Signal dimension

5.3.6 Metric filter interface

The `MetricFilter` class is responsible for calculating the various error indices. The interface to this MATLAB class is described in this chapter. The `MetricFilter` class must provide the following three functions:

- `MetricFilter()`

The constructor function has the same name as the `MetricFilter` class and does not require any parameters.

- `[returnCode, returnMessage] initializeParams(obj, paramStruct)`

The `initializeParams` function takes a reference to the MATLAB metric filter class instance and a reference to the MATLAB parameter structure class instance containing the metric filter settings. It returns a return code and a message. The return code must be an integer value. Possible values are 0 (success), 1 (warning) and 2 (error). In case of return code 1 and 2, the `returnMessage` contains the information about what has happened. In case of a successful function call (return code 0), `returnMessage` can be any kindo of string.

- `[outputSignal, returnCode, returnMessage]`
  `computeOutput(obj, signal)`

The `computeOutput` function takes a reference to the MATLAB metric filter class instance and a reference to the signal class instance which represents the final error contributor. It returns the data for the specified error index, a return code and a message. The `outputSignal` contains the signal data after the metric filter was applied. The return code must be an integer value. Possible values are 0 (success), 1 (warning) and 2 (error). In case of return code 1 and 2, the `returnMessage` contains the information about what has happened. In case of a successful function call (return code 0), `returnMessage` can be any kind of string.

6 Validation Requirements

6.1 REQ-PEET-INTF-0010

PEET shall provide an interface to MS Excel that allows the import of system (block) parameters.

This requirement will be validated by testing the import of Excel data. In order to test the import from an Excel file, the desired data must be specified as defined in chapter 5.3.2.1. Afterwards the returned data must be manually compared to the data contained in the Excel file.

6.2 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.

This requirement will be tested by exporting the results to an Excel file. Afterwards the content of the Excel file must be manually inspected.

6.3 REQ-PEET-INTF-0020

GUI and core of PEET shall communicate via the JMI interface of MATLAB.

This requirement will be tested by running several PEET test cases. If the test cases are all running and are not producing any kind of error, this requirement is successfully validated.

6.4 REQ-PEET-INTF-0030

PEET shall be compatible with MATLAB 2011b.

This requirement will be tested by running PEET on a MATLAB 2011b installation. If the test cases are all running and are not producing any kind of error, this requirement is successfully validated.

6.5 Validation Matrix

	Manual testing	**Test cases**
REQ-PEET-INTF-0010	**X**	
REQ-PEET-INTF-0015	**X**	
REQ-PEET-INTF-0020		**X**
REQ-PEET-INTF-0030		**X**

Appendix A XML structure examples

A.1 Pointing System definition

This section shows the PointingSystemDefinition.xml file for a simple pointing system containing 2 PES, a summation block and the final PEC (Pointing) block. The structure of the pointing system is shown in figure A.1-1.

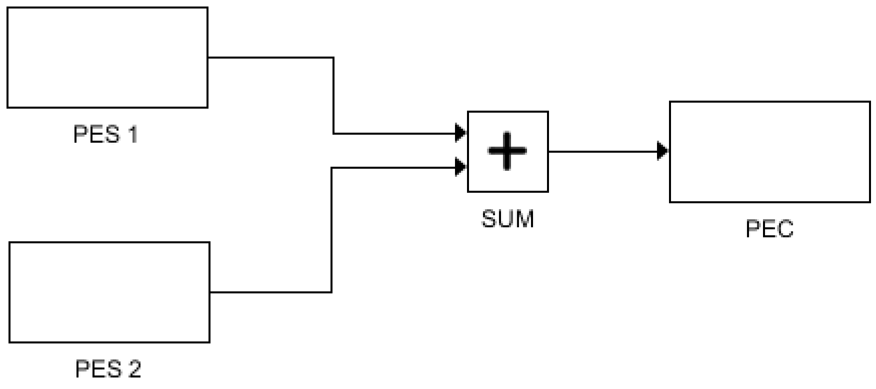

Figure A.1-1 Structure of the pointing system

For this pointing system, the pointing system definition file would look like the XML structure below.

```
<Pointing_System> <!-- Root node of the pointing system definition -->
  <Pointing_System_Settings> <!-- Pointing system settings (see 5.3.1.1) -->
    <Error_Indices> <!-- Error index definitions -->
      <Item>
        <Type>Absolute Knowledge Error</Type>
        <ID>Absolute Knowledge Error</ID>
        <Statistical_Interpretation>Temporal</Statistical_Interpretation>
        <Confidence_Coefficient>3.0</Coefficient>
      </Item>
      <Item>
        <Type>Absolute Performance Error</Type>
        <ID>Absolute Performance Error</ID>
        <Statistical_Interpretation>Mixed</Statistical_Interpretation>
        <Confidence_Coefficient>3.0</Confidence_Coefficient>
      </Item>
    </Error_Indices>
    <Correlations><!-- correlations between the error sources -->
      <Item>
        <First_Source>PES 1</First_Source>
        <Second_Source>PES 2</Second_Source>
        <Type>Uncorrelated</Type>
      </Item>
    </Correlations>
    <Global_Settings>
      <Frequency_Resolution>100000.0</Frequency_Resolution>
      <Use_User_Frequencies>False</Use_User_Frequencies>
      <Line_Of_Sight>x-Axis</Line_Of_Sight>
    </Global_Settings>
  </Pointing_System_Settings>
  <Block_List> <!-- Block list (see 5.3.1.2) -->
```

```
    <Item>
      <ID>SUM</ID>
      <Block_Type>SUM</Block_Type>
      <Position>
        <X>352.0</X>
        <Y>192.0</Y>
      </Position>
      <Block_Settings>
        <Number_Of_Inputs>2</Number_Of_Inputs>
      </Block_Settings>
    </Item>
    <Item>
      <ID>PEC</ID>
      <Block_Type>PEC Pointing</Block_Type>
      <Position>
        <X>453.0</X>
        <Y>187.0</Y>
      </Position>
      <Block_Settings></Block_Settings>
    </Item>
    <Item>
      <ID>PES 1</ID>
      <Block_Type>PES</Block_Type>
      <Position>
        <X>122.0</X>
        <Y>140.0</Y>
      </Position>
      <Block_Settings>
        <Signal_Dimension>3D</Signal_Dimension>
        <Time_Constant>
          <Distribution_Type>Discrete</Distribution_Type>
          <Mean_Value>
            <Item>
              <Item>1.0</Item>
              <Item>3.0</Item>
              <Item>2.0</Item>
            </Item>
          </Mean_Value>
        </Time_Constant>
        <Time_Random>
          <Representation>Random variable</Representation>
          <Distribution_Type>Uniform</Distribution_Type>
          <Minimum>[C:\data.xls]Sheet1!A3:C3</Minimum> <!-- Excel import -->
          <Maximum>
            <Item>
              <Item>2.0</Item>
              <Item>2.5</Item>
              <Item>0.0</Item>
            </Item>
          </Maximum>
          <Correlation>Fully_Correlated</Correlation>
        </Time_Random>
      </Block_Settings>
    </Item>
    <Item>
      <ID>PES 2</ID>
      <Block_Type>PES</Block_Type>
      <Position>
        <X>123.0</X>
        <Y>257.0</Y>
      </Position>
      <Block_Settings>
        <Signal_Dimension>3D</Signal_Dimension>
        <Time_Constant>
          <Distribution_Type>Rayleigh</Distribution_Type>
          <Rayleigh_Parameter>
            <Item>
              <Item>1.75</Item>
              <Item>1.3</Item>
              <Item>2.6</Item>
            </Item>
          </Rayleigh_Parameter>
          <Correlation>Uncorrelated</Correlation>
        </Time_Constant>
        <Time_Random>
          <Representation>Random process</Representation>
          <Type>Covariance</Type>
          <Sampling_Time>127.0</Sampling_Time>
          <Correlation>Uncorrelated</Correlation>
          <Variance>
            <Item>
              <Item>3.2</Item>
              <Item>3.34</Item>
              <Item>3.0</Item>
            </Item>
          </Variance>
        </Time_Random>
      </Block_Settings>
    </Item>
  </Block_List>
  <Connection_List> <!-- Connection list (see 5.3.1.3) -->
    <Item>
      <From>
        <Block_ID>SUM</Block_ID>
        <Output_Port_Index>0</Output_Port_Index>
      </From>
      <To>
        <Block_ID>PEC</Block_ID>
        <Input_Port_Index>0</Input_Port_Index>
      </To>
      <Vertex_List>
        <Item>
          <X>392.0</X>
          <Y>212.0</Y>
        </Item>
        <Item>
          <X>447.0</X>
          <Y>212.0</Y>
        </Item>
      </Vertex_List>
    </Item>
    <Item>
      <From>
        <Block_ID>PES 2</Block_ID>
        <Output_Port_Index>0</Output_Port_Index>
      </From>
      <To>
        <Block_ID>SUM</Block_ID>
        <Input_Port_Index>1</Input_Port_Index>
      </To>
      <Vertex_List>
        <Item>
          <X>223.0</X>
          <Y>282.0</Y>
        </Item>
        <Item>
          <X>284.5</X>
          <Y>282.0</Y>
        </Item>
        <Item>
```

```
          <X>284.5</X>
          <Y>220.33333333333334</Y>
        </Item>
        <Item>
          <X>346.0</X>
          <Y>220.33333333333334</Y>
        </Item>
      </Vertex_List>
    </Item>
    <Item>
      <From>
        <Block_ID>PES 1</Block_ID>
        <Output_Port_Index>0</Output_Port_Index>
      </From>
      <To>
        <Block_ID>SUM</Block_ID>
        <Input_Port_Index>0</Input_Port_Index>
      </To>
      <Vertex_List>
        <Item>
          <X>222.0</X>
          <Y>165.0</Y>
        </Item>
        <Item>
          <X>285.0</X>
          <Y>165.0</Y>
        </Item>
        <Item>
          <X>285.0</X>
          <Y>203.66666666666666</Y>
        </Item>
        <Item>
          <X>346.0</X>
          <Y>203.66666666666666</Y>
        </Item>
      </Vertex_List>
    </Item>
  </Connection_List>
</Pointing_System>
```

A.2 Block Database

The XML file example below shows a small block database containing a summation block and an exported block type. This example shows the structure of both possible block type definitions (see 5.3.3 for details).

```
<Block_Database>
  <Block> <!-- Simple block type definition -->
    <Record_Type>Block</Record_Type>
    <Block_Type>SUM</Block_Type> <!-- Name of the block type -->
    <Block_Category>Basic Blocks</Block_Category>
    <Java_Class>int_.esa.peet.blocks.SummationBlock</Java_Class>
    <Matlab_Class>Summation</Matlab_Class>
  </Block>
  <Block> <!-- User defined block type exported by the GUI -->
    <Record_Type>User Defined Container</Record_Type>
    <Block_Type>User defined type</Block_Type> <!-- Name of the block type -->
    <Block_Category>Static System Blocks</Block_Category>
    <Internal_Blocks> <!-- Internal blocks which are defining the behaviour -->
      <Item>
        <ID>Input</ID>
        <Block_Type>Input Port</Block_Type>
        <Position>
          <X>104.0</X>
          <Y>290.0</Y>
        </Position>
        <Block_Settings></Block_Settings>
      </Item>
      <Item>
        <ID>Output</ID>
        <Block_Type>Output Port</Block_Type>
        <Position>
          <X>472.0</X>
          <Y>290.0</Y>
        </Position>
        <Block_Settings></Block_Settings>
      </Item>
      <Item>
        <ID>Static System</ID>
        <Block_Type>Static System</Block_Type>
        <Position>
          <X>295.0</X>
          <Y>290.0</Y>
        </Position>
        <Block_Settings>
          <System_Matrix>
            <Item>
              <Item>2.0</Item>
              <Item>0.0</Item>
              <Item>0.0</Item>
            </Item>
            <Item>
              <Item>0.0</Item>
              <Item>3.0</Item>
              <Item>0.0</Item>
            </Item>
            <Item>
              <Item>0.0</Item>
              <Item>0.0</Item>
              <Item>2.7</Item>
            </Item>
          </System_Matrix>
        </Block_Settings>
      </Item>
    </Internal_Blocks>
    <Internal_Connections> <!-- Connections between the internal blocks -->
      <Item>
        <From>
          <Block_ID>Input</Block_ID>
          <Output_Port_Index>0</Output_Port_Index>
        </From>
        <To>
          <Block_ID>Static System</Block_ID>
          <Input_Port_Index>0</Input_Port_Index>
        </To>
        <Vertex_List>
          <Item>
            <X>204.0</X>
            <Y>315.0</Y>
          </Item>
          <Item>
            <X>289.0</X>
            <Y>315.0</Y>
          </Item>
        </Vertex_List>
      </Item>
      <Item>
        <From>
          <Block_ID>Static System</Block_ID>
          <Output_Port_Index>0</Output_Port_Index>
        </From>
        <To>
          <Block_ID>Output</Block_ID>
          <Input_Port_Index>0</Input_Port_Index>
        </To>
        <Vertex_List>
          <Item>
            <X>395.0</X>
            <Y>315.0</Y>
          </Item>
          <Item>
            <X>466.0</X>
            <Y>315.0</Y>
          </Item>
        </Vertex_List>
      </Item>
    </Internal_Connections>
  </Block>
</Block_Database>
```