ns2measure Reference Manual

Generated by Doxygen 1.3.7

Tue May 30 08:09:56 2006

Contents

Chapter 1

ns2measure Hierarchical Index

1.1 ns2measure Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

ns2measure Class Index

2.1 ns2measure Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

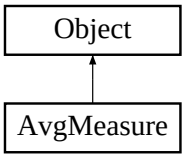
ns2measure Class Documentation

3.1 AvgMeasure Class Reference

An AvgMeasure is a set of populations for averaged metrics.

`#include <measure.h>`

Inheritance diagram for AvgMeasure::



Public Member Functions

- **AvgMeasure** ()

  Create an emptry AvgMeasure.

- ~**AvgMeasure** ()

  Do nothing.

- void **addSample** (sample_t x, unsigned int **id**)

  Add a sample to a population.

- **Population** & **getPopulation** (unsigned int **id**)

  Return the population of a given index.

- bool **getValid** (unsigned int **id**)

  Return true if the population with a given index exists.

- unsigned int **getSize** () const

  Return the number of populations in this measure.

Private Attributes

- std::vector< **Population** > **populations**

  Array of populations.

- std::vector< bool > **valid**

  Bit array to check if the i-th entry of populations is valid.

3.1.1 Detailed Description

An AvgMeasure is a set of populations for averaged metrics.

The documentation for this class was generated from the following files:

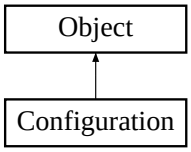
- measure.h
- measure.cc

3.2 Configuration Class Reference

One instance of this class stores information from the configuration file.

`#include <configuration.h>`

Inheritance diagram for Configuration::



Public Member Functions

- **Configuration** ()

  Create an empty Configuration object.

- ~**Configuration** ()

  Do nothing.

- void **parse** (std::string inputFileName)

  Parse the configuration file, and close it immediately after.

- std::string **getOutputFileName** () const

  Get the output file name.

- unsigned int **getMinReplics** () const

  Get the minimum number of replics.

- unsigned int **getMaxReplics** () const

  Get the maximum number of replics.

- std::string **getHeaderName** () const

  Get the header file name information.

- std::string **getTrailerName** () const

  Get the trailer file name information.

- void **getDescAvg** (bool &valid, **MetricDescAvg** &dsc, std::string s, unsigned int **id**)

  Get the descriptor of an averaged metric.

- void **getDescDst** (bool &valid, **MetricDescDst** &dsc, std::string s, unsigned int **id**)

  Get the descriptor of a distribution metric.

- void **dump** (std::ostream &os)

  Debug function to dump to an ostream the database content.

Private Member Functions

- void **insert** (std::string s, unsigned int **id**, const **MetricDescAvg** &dsc)

  Insert an averaged metric descriptor.

- void **insert** (std::string s, unsigned int **id**, std::string what, const **MetricDescAvg** &dsc)

  Insert a distribution metric descriptor.

- std::string **getNextWord** (std::istream &is, bool required=false)

  Get the next word from configuration file.

Private Attributes

- std::string **outputFileName**

  Output(p. 20) *file name.*

- unsigned int **minReplics**

  Minimum number of replics. No minimum => 0.

- unsigned int **maxReplics**

  Maximum number of replics. No maximum => 0.

- std::string **headerName**

  Header file name.

- std::string **trailerName**

  Trailer file name.

- std::map< std::string, std::vector< **MetricDescAvg** > > **avg**

  Descriptors for averaged metrics.

- std::map< std::string, std::vector< **MetricDescDst** > > **dst**

  Descriptors for distribution metrics.

3.2.1 Detailed Description

One instance of this class stores information from the configuration file.

3.2.2 Member Function Documentation

3.2.2.1 std::string Configuration::getNextWord (std::istream & *is*, bool *required* = false) [private]

Get the next word from configuration file.

If a comment character '#' is found, the entire line is skipped. If the end of file is reached, std::string::npos is returned. If the next word is required, the an exception is thown if the end of file is reached.

3.2.2.2 void Configuration::insert (std::string *s*, unsigned int *id*, std::string *what*, const MetricDescAvg & *dsc*) [private]

Insert a distribution metric descriptor.

The function allows a metric descriptor to be overriden. The what argument specifies what submetric is to updated.

3.2.2.3 void Configuration::insert (std::string *s*, unsigned int *id*, const MetricDescAvg & *dsc*) [private]

Insert an averaged metric descriptor.

The function allows a metric descriptor to be overriden.

The documentation for this class was generated from the following files:

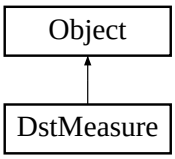
- configuration.h
- configuration.cc

3.3 DstMeasure Class Reference

A DstMeasure is a set of populations for distribution metrics.

`#include <measure.h>`

Inheritance diagram for DstMeasure::



Public Member Functions

- **DstMeasure** ()

  Create an emptry DstMeasure.

- ~**DstMeasure** ()

  Do nothing.

- void **addSample** (sample_t x, unsigned int **id**, unsigned int bin)

  Add a sample to a population bin.

- **Population** & **getPopulation** (unsigned int **id**, unsigned int bin)

  Return the population of a given index/bin.

- **Population** & **getPopulationCDF** (unsigned int **id**, unsigned int bin)

  Return the CDF population of a given index/bin.

- **Population** & **getMeanPopulation** (unsigned int **id**)

  Return the mean population.

- **Population** & **getMedianPopulation** (unsigned int **id**)

  Return the mediam population.

- **Population** & **getPercentile95Population** (unsigned int **id**)

  Return the 95th percentile population.

- **Population** & **getPercentile99Population** (unsigned int **id**)

  Return the 99th percentile population.

- void **computeDerivedStatistics** (unsigned int **id**)

  Compute the derived statistics (mean, quantiles) if not already done.

- bool **getValid** (unsigned int **id**, unsigned int bin)

  Return true if the population with a given index exists.

- unsigned int **getSize** () const

Return the number of populations in this measure.

- unsigned int **getSize** (unsigned int **id**)
  Return the number of bins in the given index.
- void **setBinSize** (sample_t s)
  Set the bin size.
- void **setDistLower** (sample_t s)
  Set the distribution lower bound.
- sample_t **getBinSize** () const
  Get the bin size.
- sample_t **getDistLower** () const
  Get the distribution lower bound.

Private Attributes

- std::vector< std::vector< **Population** > > **populations**
  Array of array of populations.
- std::vector< std::vector< **Population** > > **populationsCDF**
  Array of array of populations. Cumulative wrt the previous bins.
- std::vector< std::vector< bool > > **valid**
  Bit array to check if the i-th entry of populations is valid.
- sample_t **binSize**
  Bin size.
- sample_t **distLower**
  Distribution lower bound.
- bool **binSizeSet**
  True if the bin size has been set.
- bool **distLowerSet**
  True if the distribution lower bound has been set.
- std::vector< **Population** > **meanPopulations**
  Populations of average values.
- std::vector< **Population** > **medianPopulations**
  Populations of median values.
- std::vector< **Population** > **percentile95Populations**
  Populations of 95th percentile values.

- std::vector< **Population** > **percentile99Populations**

  Populations of 99th percentile values.

- std::vector< unsigned int > **derivedLast**

  Record the last size of the bins populations.

3.3.1 Detailed Description

A DstMeasure is a set of populations for distribution metrics.

Each element in a DstMeasure is the probability that a given sample fits in a given bin. The size of the bin and the minimum value of the distribution are part of the DstMeasure data structure. In addition to the cumulative distribution function and the probability mass function, the mean value and quantiles are derived from the DstMeasure.

The bin size and the mininmum data structure must be set before computing the quantile values. Also, samples must be added in bin order. If they are not, then the cumulative values will not be meaningful.

3.3.2 Member Data Documentation

3.3.2.1 std::vector< std::vector<Population> > DstMeasure::populationsCDF [private]

Array of array of populations. Cumulative wrt the previous bins.

Note that the populations and populationsCDF always have the same size and structure. Thus, the valid structure is meaningful for both these data structures.

The documentation for this class was generated from the following files:

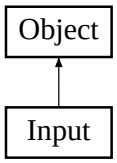
- measure.h
- measure.cc

3.4 Input Class Reference

Class for reading the input file according to the configuration.

`#include <input.h>`

Inheritance diagram for Input::



Public Member Functions

- void **readSingleRun** (std::istream &fileIn, std::ostream *fileOut=0, bool recover=false)
  Read a single run from an input file.
- **Input** (**Configuration** &c, **Metrics** &m)
  Create an empty Input object.
- ~**Input** ()
  Do nothing.
- void **loadData** (std::string fileIn, std::string fileOut)
  Reads data from a client.
- bool **recoverData** (std::string saveFile)
  Recover a (possibly damaged) save data file.
- bool **checkConfidence** ()
  Check whether the confidence level is reached. If so, return true.
- bool **check** ()
  Check if no more simulations are needed. If so, return true.
- const std::set< unsigned int > & **getRunIdentifiers** () const
  Get the set of run identifiers.

Private Attributes

- **Configuration** & **configuration**
  Configuration(p. 7) *object used to parse input data.*
- **Metrics** & **metrics**
  Metrics(p. 17) *database.*
- std::set< unsigned int > **runIdentifiers**
  Set of run identifiers.

3.4.1 Detailed Description

Class for reading the input file according to the configuration.

project: measure filename: **input.h**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: definition of input classes and functions

3.4.2 Member Function Documentation

3.4.2.1 void Input::loadData (std::string *fileIn*, std::string *fileOut*)

Reads data from a client.

fileIn and fileOut are the unix descriptors of the input and ouput files, respectively, which must have been already opened for reading and writing, respectively.

This function returns when one of the following conditions becomes true:

- the maximum number of replics has been reached
- all the relevant metrics with check == true have a confidence interval below the threshold

Provided that:

- the minimum number of replics has been reached

This function also appends data read from fileIn to the outputfile specified in the configuration file.

3.4.2.2 void Input::readSingleRun (std::istream & *fileIn*, std::ostream * *fileOut* = 0, bool *recover* = false)

Read a single run from an input file.

If fileOut != 0, the input is copied to fileOut. The recover flag is set to true if you want to gather all metrics for debugging or recovering purposes.

The documentation for this class was generated from the following files:

- input.h
- input.cc

3.5 MetricDescAvg Struct Reference

Descriptor for averaged metrics.

`#include <configuration.h>`

Public Member Functions

- **MetricDescAvg** ()

  Create by default a non-relevant metric descriptor.

- bool **isRelevant** () const

  Return true if this metric is relevant.

Public Attributes

- bool **relevant**

  False if this metric should be ignored.

- bool **output**

  True if this metric generates an output sample.

- bool **check**

  True if the confidence interval is checked for simulation termination.

- double **outCL**

  Output(p. 20) *confidence level. Only meaningful if output == true.*

- double **CL**

  Confidence level. Only meaningful if check == true.

- double **threshold**

  Confidence threshold. Only meaningful if check == true.

3.5.1 Detailed Description

Descriptor for averaged metrics.

project: measure filename: **configuration.h**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: definition of **Configuration**(p. 7) class

The documentation for this struct was generated from the following file:

- configuration.h

3.6 MetricDescDst Struct Reference

Descriptor for distribution metrics.

`#include <configuration.h>`

Public Member Functions

- **MetricDescDst** ()

  Create by default a non-relevent metric descriptor.

- bool **isRelevant** () const

  Return true if this metric is relevant.

Public Attributes

- **MetricDescAvg pmf**

  Descriptor for the Probability Mass Function.

- **MetricDescAvg cdf**

  Descriptor for the Cumulative Distribution Function.

- **MetricDescAvg mean**

  Descriptor for the mean value.

- **MetricDescAvg median**

  Descriptor for he median value.

- **MetricDescAvg percentile95**

  Descriptor for the 95th percentile value.

- **MetricDescAvg percentile99**

  Descriptor for the 99th percentile value.

3.6.1 Detailed Description

Descriptor for distribution metrics.

The documentation for this struct was generated from the following file:

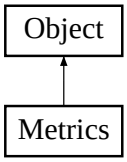
- configuration.h

3.7 Metrics Class Reference

A Metrics object contain all the **AvgMeasure**(p. 5) and **DstMeasure**(p. 10) objects.

`#include <measure.h>`

Inheritance diagram for Metrics::



Public Member Functions

- **Metrics** ()

  Create an empty Metrics object.

- ~**Metrics** ()

  Do nothing.

- void **addSample** (std::string m, sample_t x, unsigned int **id**)

  Add a sample to an averaged measure.

- void **addSample** (std::string m, sample_t x, unsigned int **id**, unsigned int bin)

  Add a sample to a distribution measure.

- void **setBinSize** (std::string m, sample_t binSize)

  Set the bin size of a distribution measure.

- void **setDistLower** (std::string m, sample_t distLower)

  Set the lower bound of a distribution measure.

- std::map< std::string, **AvgMeasure** > & **getAvgMeasures** ()

  Return the set of average measures.

- std::map< std::string, **DstMeasure** > & **getDstMeasures** ()

  Return the set of distribution measures.

- void **dump** (std::ostream &os)

  Debug function to dump the content of the Metrics object into a stream.

Private Attributes

- std::map< std::string, **AvgMeasure** > **avgMeasures**
- std::map< std::string, **DstMeasure** > **dstMeasures**

3.7.1 Detailed Description

A Metrics object contain all the **AvgMeasure**(p. 5) and **DstMeasure**(p. 10) objects.

The documentation for this class was generated from the following files:

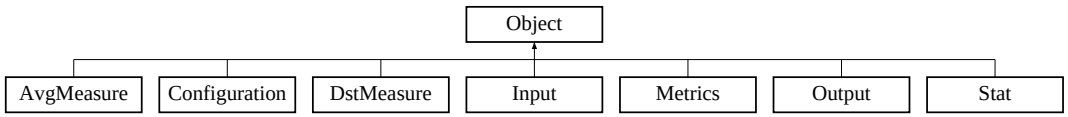
- measure.h
- measure.cc

3.8 Object Class Reference

Object superclass. All other classes should inherit from this class.

`#include <object.h>`

Inheritance diagram for Object::



Public Member Functions

- **Object** (std::string name)

  Default construction only allowed.

- unsigned int **getId** () const

  Access function that returns the object unique identifier.

- std::string **getName** () const

  Access function that returns the class name of this object.

- virtual ~**Object** ()

  Virtual destructor, which makes this a pure virtual class.

Private Attributes

- std::string **className**

  Name of the derived class.

- unsigned int **id**

  Unique numerical sequential identifier.

3.8.1 Detailed Description

Object superclass. All other classes should inherit from this class.

project: measure filename: **object.h**(p. ??) author: C. Cicconetti <`c.cicconetti@iet.unipi.it`> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: definition of the object superclass

The documentation for this class was generated from the following file:

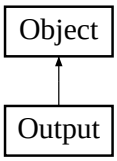
- object.h

3.9 Output Class Reference

Class for writing output files.

`#include <output.h>`

Inheritance diagram for Output::



Public Member Functions

- **Output** (**Configuration** &c, **Metrics** &m)

  Create an empty Output object.

- ~**Output** ()

  Do nothing.

- void **generate** ()

  Generate output graphs.

Private Attributes

- **Configuration** & **configuration**

  Configuration(p. 7) *object used to parse input data.*

- **Metrics** & **metrics**

  Metrics(p. 17) *database.*

3.9.1 Detailed Description

Class for writing output files.

project: measure filename: **output.h**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: definition of the Output class

3.9.2 Member Function Documentation

3.9.2.1 void Output::generate ()

Generate output graphs.

project: measure filename: **output.cc**(p. **??**) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: body of the Output class

The documentation for this class was generated from the following files:

- output.h
- output.cc

3.10 Population Class Reference

A Population is a set of samples collected from the same sensor.

`#include <measure.h>`

Public Member Functions

- **Population** ()

  Create an empty population.

- ~**Population** ()

  The destructor does nothing.

- unsigned int **getSize** () const

  Return the number of elements.

- void **addSample** (sample_t x)

  Add a sample to the population.

- sample_t **getSample** (bool &valid, unsigned int i)

  Return the i-th sample.

- double **mean** (bool &valid)

  Return the mean of the population.

- double **confInterval** (bool &valid, double cl)

  Return the confidence interval of the population.

- void **dump** (std::ostream &os)

  Debug function to print the values to an output stream.

Private Attributes

- std::vector< sample_t > **population**

  Vector of samples. It is the population itself.

3.10.1 Detailed Description

A Population is a set of samples collected from the same sensor.

project: measure filename: **measure.h**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: definition of the main project classes

3.10.2 Member Function Documentation

3.10.2.1 void Population::addSample (sample_t *x*)

Add a sample to the population.

project: measure filename: **measure.cc**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: body of the main project classes

The documentation for this class was generated from the following files:

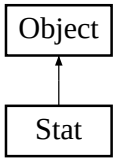
- measure.h
- measure.cc

3.11 Stat Class Reference

Utility static class containing statistical functions.

`#include <stat.h>`

Inheritance diagram for Stat::



Public Member Functions

- **Stat** ()
  Default constructor. Invoked once. Does nothing.
- ~**Stat** ()
  Distructor. Does nothing.

Static Public Member Functions

- double **confInterval** (bool &valid, const std::vector< sample_t > &samples, double cl)
  Return the confidence interval of a set of samples.
- double **mean** (bool &valid, const std::vector< sample_t > &samples)
  Return the mean of a set of samples.

Static Private Member Functions

- double **t_student** (double cl, int df)
  Function to access the t-student table.

Static Private Attributes

- double **t_table** [30][4]
  Static table containing the t-student values.

3.11.1 Detailed Description

Utility static class containing statistical functions.

project: measure filename: **stat.h**(p. ??) author: C. Cicconetti <`c.cicconetti@iet.unipi.it`> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: statistical classes and functions

3.11.2 Member Function Documentation

3.11.2.1 double Stat::confInterval (bool & *valid*, const std::vector< sample_t > & *samples*, double *cl*) [static]

Return the confidence interval of a set of samples.

The validity bit is false if the number of samples is smaller than or equal to 1, or if the confidence level is outside [0, 1].

3.11.2.2 double Stat::mean (bool & *valid*, const std::vector< sample_t > & *samples*) [static]

Return the mean of a set of samples.

The validity bit is false if the number of samples is zero.

3.11.2.3 double Stat::t_student (double *cl*, int *df*) [static, private]

Function to access the t-student table.

Access to the t-student table through the number of degrees of freedom and the confidence level. Return -1.0 if the the number of the degrees of freedom is smaller than or equal to 1, or if the confidence level is outside [0, 1].

3.11.3 Member Data Documentation

3.11.3.1 double Stat::t_table [static, private]

Static table containing the t-student values.

project: measure filename: **stat.cc**(p. ??) author: C. Cicconetti <c.cicconetti@iet.unipi.it> year: 2006 affiliation: Dipartimento di Ingegneria dell'Informazione University of Pisa, Italy description: body of statistical functions and classes

The documentation for this class was generated from the following files:

- stat.h
- stat.cc

Index