LibReDE

User Guide

Simon Spinner

Johannes Grohmann

https://bitbucket.org/librede/librede

February 28, 2019

Contents

1 Introduction **2**

2 Installation **4**
2.1 System Requirements 4
2.2 New Installation 4
2.3 Update Existing Installation 5

3 Step by Step Example **7**

4 Building and Running LibReDE from Console **14**

5 Calling LibReDE from Java source code **15**

6 Approaches to Resource Demand Estimation **15**
6.1 Estimation Approaches 15
6.2 State Models 16
6.3 Observation Models 16
6.4 Estimation Algorithms 17

7 Configuration **17**
7.1 Workload Description 17
7.2 Data Sources 18
7.3 Traces 18
7.4 Estimation 19
7.5 Validation 20
7.6 Output 21

8 Extension Points **21**

9 Known Issues **22**

1 Introduction

Librede is a library that provides a set of ready-to-use implementations of approaches to resource demand estimation. In the following, we describe the motivation behind Librede and give an overview on how to use it.

What are resource demands? Our definition is based on classic queueing theory [6]: a resource demand (aka. service demand) is the time a unit of work (e.g., request or transaction) spends obtaining service from a resource (e.g., CPU or hard disk) in a system over all visits. Resource demands are input parameters of widely used stochastic performance formalisms (e.g., Queueing Networks or Queueing Petri Nets) and of architecture-level performance models (e.g., Descartes Modeling Language (DML) [4] or Palladio Component Model (PCM) [1]). Resource demands are random variables and as such follow a certain probability distribution. In the following, when talking of resource demands we implicitly refer to the mean of the random variable if not explicitly noted otherwise.

Why would you use estimation approaches? In order to obtain accurate performance predictions for a system, a performance engineer needs to determine representative values for the resource demands during performance model construction. State-of-the-art monitoring tools can only provide aggregate resource usage statistics on a system- or per-process-level. However, in many applications one process may serve requests of different types with varying resource requirements (due to different computations, caching, etc.). A very fine-grained instrumentation of the system would be necessary to monitor resource demands directly. For many real-world applications this fine-grained monitoring is not feasible as they lack the required instrumentation capabilities or as the instrumentation would incur too high overheads. Instead, we can use statistical estimation techniques to estimate resource demands based on the readily available aggregate monitoring data (e.g., CPU utilization as measured by the operating system, or average response time). Different approaches to resource demand estimation using statistical techniques, e.g., linear regression [10], Kalman filtering [14, 13], or non-linear optimization [9, 7], have been proposed in the literature.

Why should you use Librede? Existing approaches to resource demand estimation differ in their expected input measurements, their resulting outputs and their robustness to data anomalies. As shown in [11], the applicability of certain approaches and the expected accuracy depends heavily on the system under study. For instance, in [11] we show that the relative error of the estimated resource demand can vary between less than 5% and over 100% between different estimation approaches. Therefore, the selection of a suitable estimation approach is very important to obtain reliable performance predictions from a performance model. However, there were no publicly available implementations of the proposed estimation approaches, making comparisons between them difficult as one would have to first implement the different estimation approaches.

Librede provides a library of ready-to-use implementations of different estimation approaches, based on Kalman filtering, linear regression, and non-linear optimization techniques. Performance engineers are relieved from the time-consuming and error-prone implementation of estimation approaches. Thus, Librede simplifies the selection and application of approaches to resource demand estimation for a given system under study. Furthermore, it provides a framework reducing the effort to implement additional, novel estimation approaches through reuse of common functionality.

What are the main features of Librede? Librede supports performance engineers with the following features:

- It contains ready-to-use implementations of 8 estimation approaches (namely response time approximation, service demand law [2], 2 variants of linear regression [10, 5], 2 variants of Kalman filter [13, 14], and 2 variants of nonlinear optimization [9, 8]). Additional estimation approaches can be added through extension points.
- The measurement data which is the input of the estimation is read from standard CSV files. It also offers an extension point to implement additional importers for custom file formats.
- Using cross-validation, the accuracy of the estimated resource demands can be evaluated enabling the comparison of different approaches.
- The resulting resource demands can be exported to CSV files. Additional output formats can be added through an extension point.
- It offers configuration parameters to customize and optimize the estimation (e.g., step size, window size, recursive optimization, approach-specific parameters, etc.).
- An Eclipse-based editor can be used for configuring the estimation (describe workload, configure input data, select estimation approaches, adapt configuration parameters, etc.). The configuration can be saved in a file for later reuse.
- It provides a Java API that allows the integration of resource demand estimation into custom applications. The configuration is provided by an EMF model.

How can you execute Librede? There are different modes in which you can execute Librede:

- The Eclipse editor provides a convenient way to create the estimation configuration and execute the estimation approaches. This mode is targeted at offline analysis, when manually for, e.g., design-time analysis or offline capacity planning purposes. The input measurement data needs to be obtained beforehand by running experiments in a dedicated test environment or using monitoring traces from a production system.
- The standalone console interface enables the execution of Librede outside of Eclipse assuming an existing estimation configuration, created using the Eclipse editor.
- The Java API enables the integration of resource demand estimation in custom programs. In this mode, Librede can be used for online analysis, i.e., new measurements are continuously coming in and the resource demands are updated recursively in a certain interval. For instance, this mode can be used to build autonomic and self-aware systems [3] using performance models to evaluate the impact of changes in the environment on the system performance and automatically adapt their resource allocation.

What are the license terms? All Librede source code files are copyright by the contributors. Librede is distributed as open-source software under the terms of the Eclipse Public License (EPL) 1.0 (see `https://www.eclipse.org/legal/epl-v10.html`).

The library uses open-source third-party libraries. We especially acknowledge the creators of these libraries:

- CERN - European Organization for Nuclear Research, Colt library (`http://acs.lbl.gov/ACSSoftware/colt/`).
- Bayes++ (`http://bayesclasses.sourceforge.net/Bayes++.html`).
- COIN-OR Ipopt [12] (`https://projects.coin-or.org/Ipopt`).
- NNLS (`http://www.netlib.org/lawson-hanson/all`)

2 Installation

2.1 System Requirements

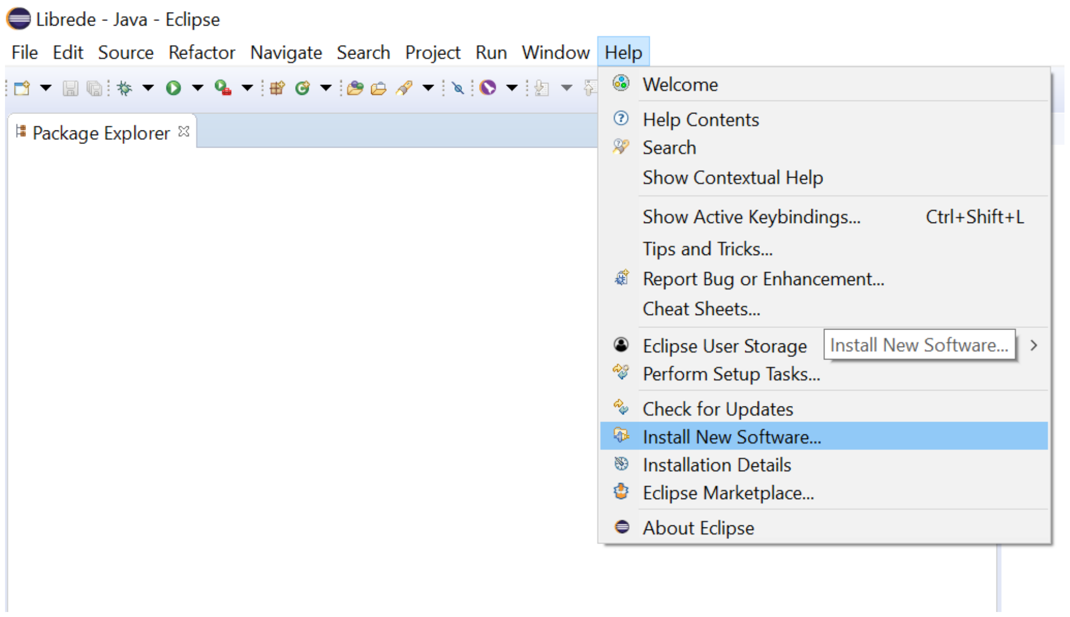
In order to install Librede, your system needs to meet the following prerequisites:

- *Operation System:* Windows 7/8/10 32-bit or 64-bit, Linux 64-bit (MacOS X and Linux 32-bit are currently not supported)
- *Java Runtime Environment:* at least 1.6 (on Linux only 64-bit version supported)
- *Eclipse:* Eclipse Standard 4.4 or higher (download from `http://www.eclipse.org/downloads/`)
- *Other:* On Linux please ensure that the library gfortran is installed. On Windows Librede comes with its own version.

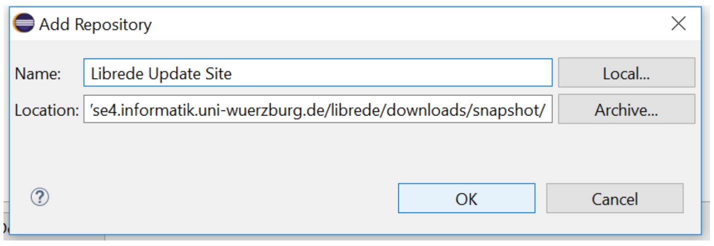
2.2 New Installation

Your can install Librede in as an Eclipse plugin with an update site. Follow these steps:

- In Eclipse go to menu "Help → Install new Software"



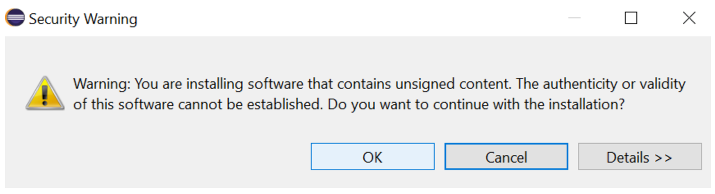
- Add a new repository with `https://se4.informatik.uni-wuerzburg.de/librede/downloads/snapshot/` as location.


Add Repository

Name: Librede Update Site

Location: 'se4.informatik.uni-wuerzburg.de/librede/downloads/snapshot/

- Mark the Librede feature for installation and click next.
- Accept the license agreement.
- Confirm the security warning.


Security Warning

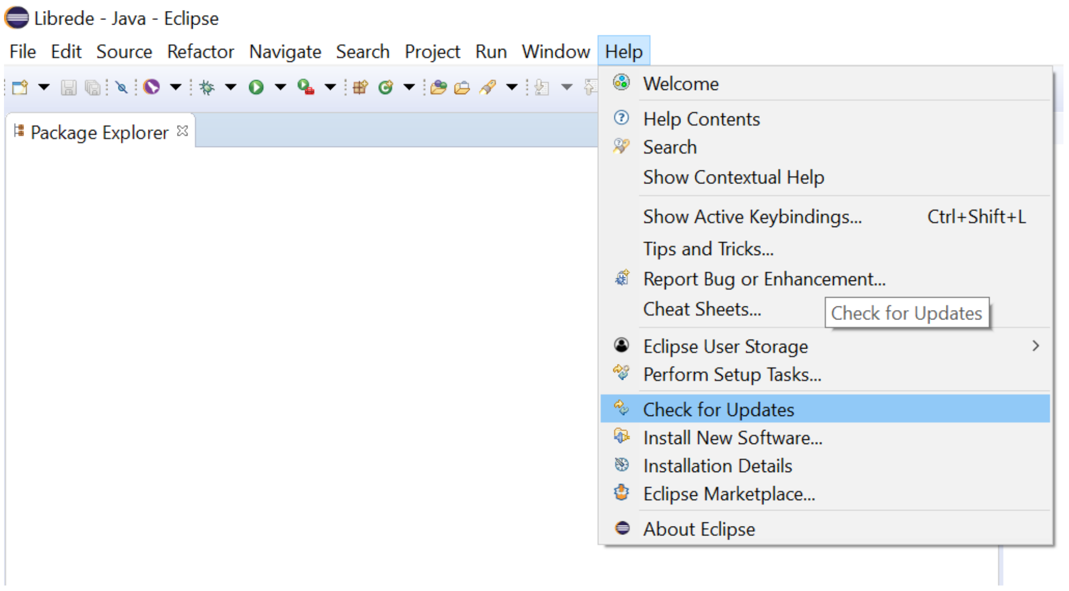
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

- After completion, restart Eclipse.

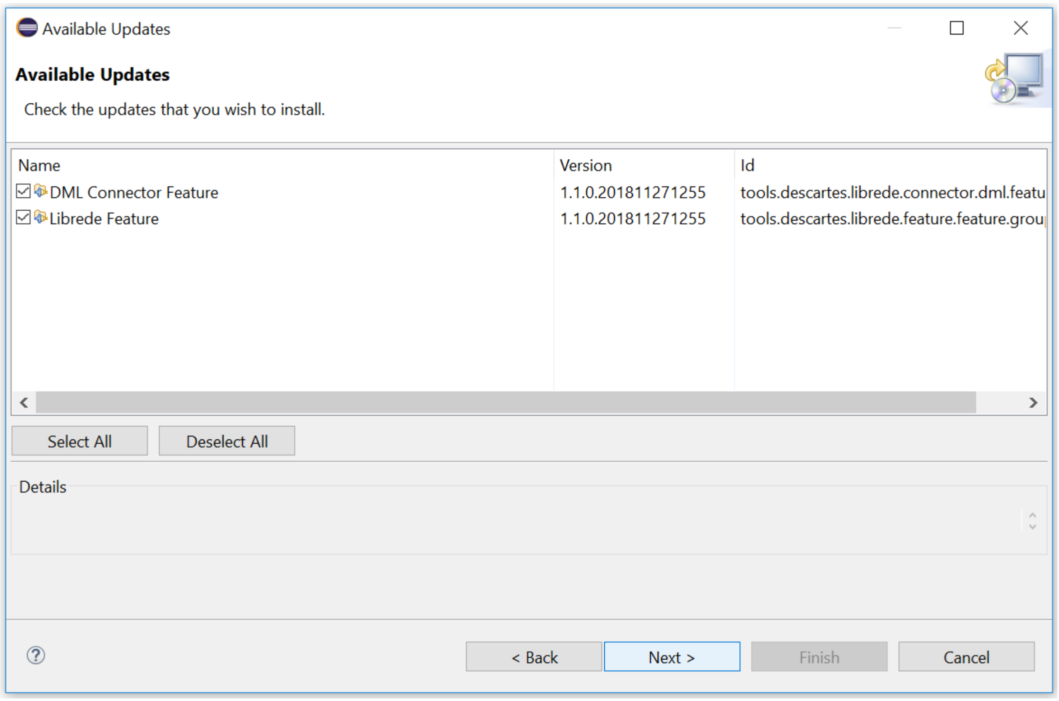
2.3 Update Existing Installation

To update an existing installation of the Librede Eclipse plugin follow these steps:

- In Eclipse go to menu "Help → Check for Updates"

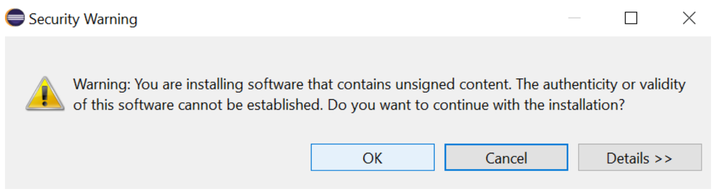


- Wait for the Eclipse operations to complete.
- **IF** a Librede update is available: proceed. **ELSE** No update is available.



Name	Version	Id
☑ DML Connector Feature	1.1.0.201811271255	tools.descartes.librede.connector.dml.featu
☑ Librede Feature	1.1.0.201811271255	tools.descartes.librede.feature.feature.grou

- Accept the license agreement.
- Confirm the security warning.


Security Warning

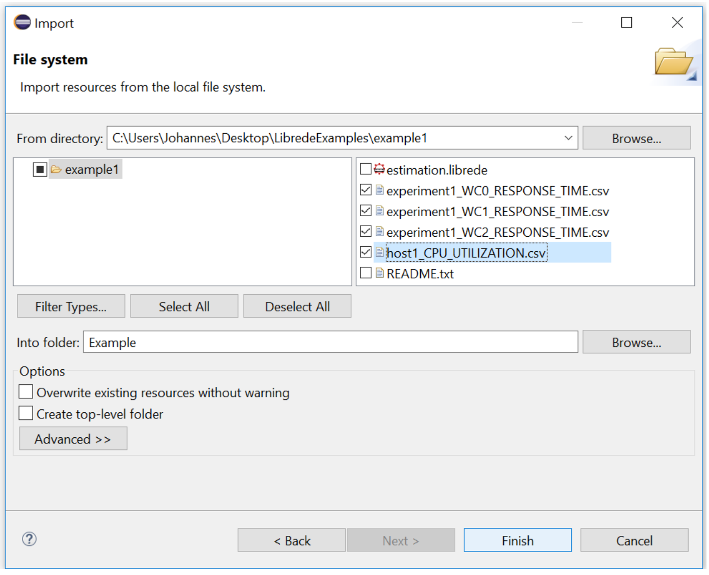
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

- After completion, restart Eclipse.

3 Step by Step Example

This section explains the usage of Librede with a step-by-step example.

- Create a new project in your workspace.
- Download example files from `https://bitbucket.org/librede/librede/downloads/LibredeExamples.zip` and extract the archive to your hard disk.
- Import the measurement traces from the downloaded ZIP archive into your project.


Import

File system

Import resources from the local file system.

From directory: C:\Users\Johannes\Desktop\LibredeExamples\example1

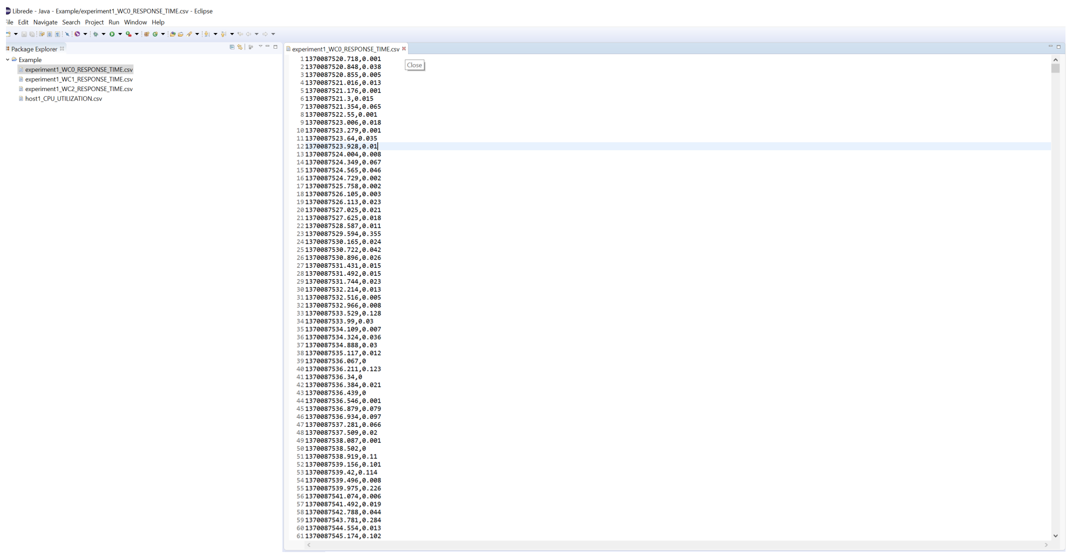
example1

☐ estimation.librede
☑ experiment1_WC0_RESPONSE_TIME.csv
☑ experiment1_WC1_RESPONSE_TIME.csv
☑ experiment1_WC2_RESPONSE_TIME.csv
☑ host1_CPU_UTILIZATION.csv
☐ README.txt

Into folder: Example

Options
☐ Overwrite existing resources without warning
☐ Create top-level folder

- Check the structure of the imported measurement traces. The traces are standard CSV files with two or more columns. The first column contains a unix timestamp (counting seconds from 1.1.1970), the second and any other columns contain measurements from a system.

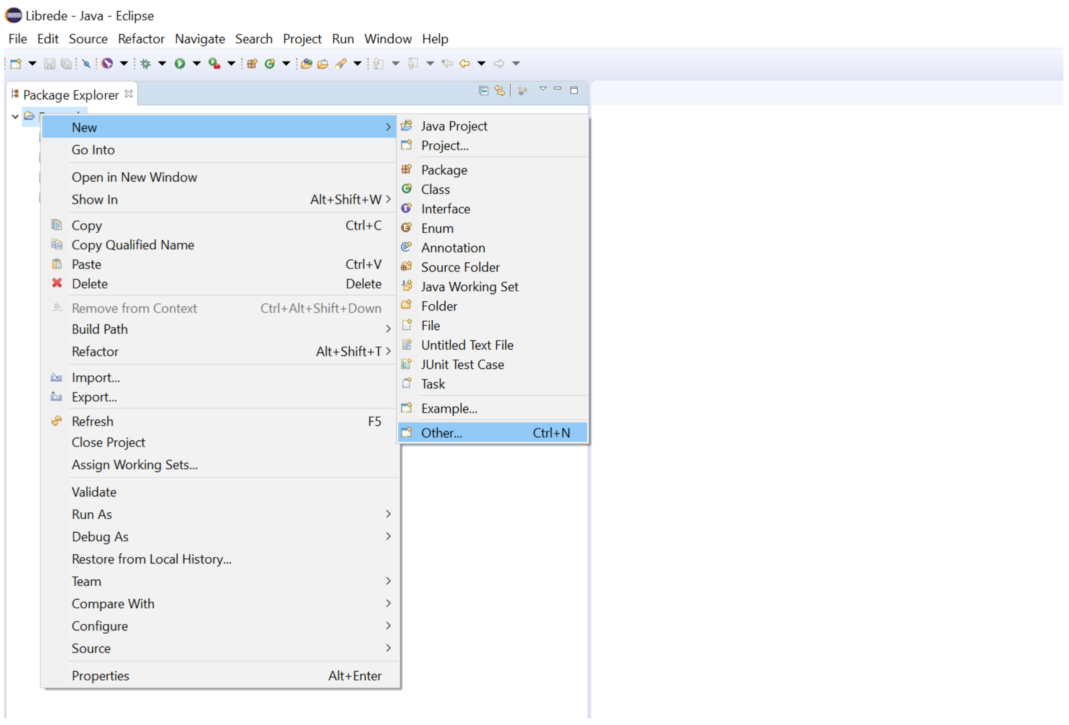


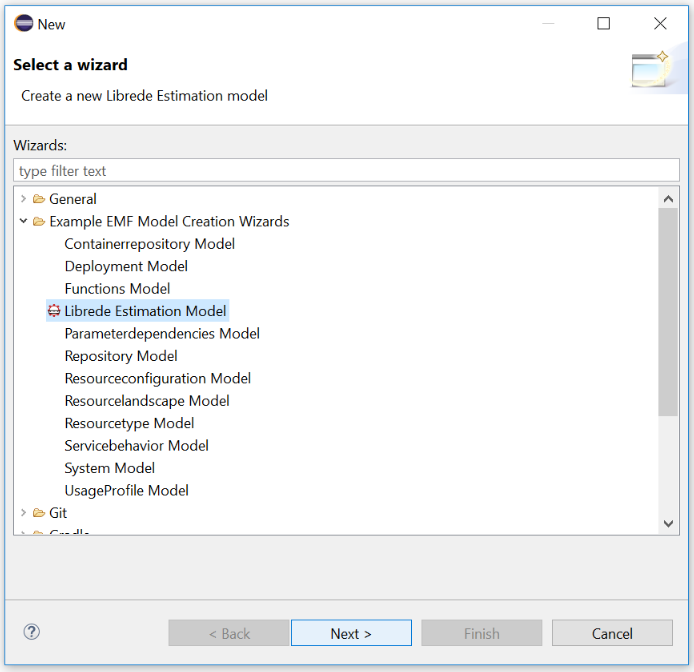
Librede - Java - Example/experiment1_WC0_RESPONSE_TIME.csv - Eclipse
File Edit Navigate Search Project Run Window Help
Package Explorer
Example
experiment1_WC0_RESPONSE_TIME.csv
experiment1_WC1_RESPONSE_TIME.csv
experiment1_WC2_RESPONSE_TIME.csv
host1_CPU_UTILIZATION.csv
experiment1_WC0_RESPONSE_TIME.csv
Close

```
1370087520.718,0.001
1370087520.848,0.038
1370087520.855,0.005
1370087521.016,0.013
1370087521.176,0.001
1370087521.3,0.015
1370087521.354,0.065
1370087522.55,0.001
1370087523.006,0.018
1370087523.279,0.001
1370087523.64,0.035
1370087523.928,0.01
1370087524.004,0.008
1370087524.349,0.067
1370087524.565,0.046
1370087524.729,0.002
1370087525.758,0.002
1370087526.105,0.003
1370087526.113,0.023
1370087527.025,0.021
1370087527.625,0.018
1370087528.587,0.011
1370087529.594,0.355
1370087530.165,0.024
1370087530.722,0.042
1370087530.896,0.026
1370087531.431,0.015
1370087531.492,0.015
1370087531.744,0.023
1370087532.214,0.013
1370087532.516,0.005
1370087532.966,0.008
1370087533.529,0.128
1370087533.99,0.03
1370087534.109,0.007
1370087534.324,0.036
1370087534.888,0.03
1370087535.117,0.012
1370087536.067,0
1370087536.211,0.123
1370087536.34,0
1370087536.384,0.021
1370087536.439,0
1370087536.546,0.001
1370087536.879,0.079
1370087536.934,0.097
1370087537.281,0.066
1370087537.509,0.02
1370087538.087,0.001
1370087538.502,0
1370087538.919,0.11
1370087539.156,0.101
1370087539.42,0.114
1370087539.496,0.008
1370087539.975,0.226
1370087541.074,0.006
1370087541.492,0.019
1370087542.788,0.044
1370087543.781,0.284
1370087544.554,0.013
1370087545.174,0.102
```

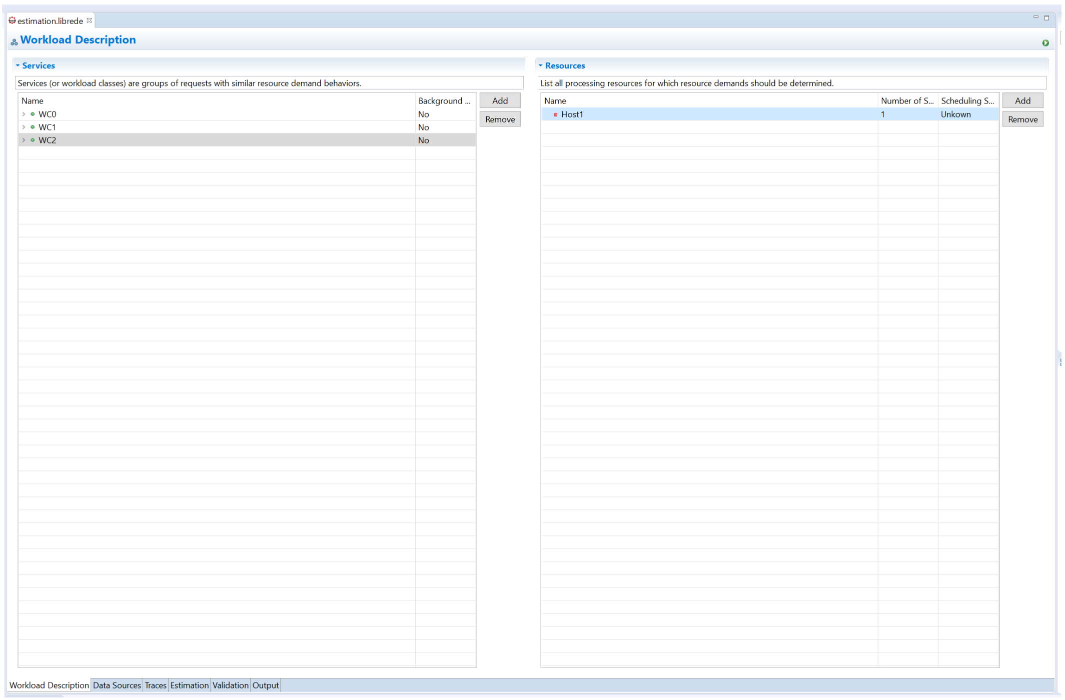
The monitoring data must be available as time series data with associated timestamps for each sample. The library can work on time series with individual events (e.g., arrival times and response times of individual requests) or on fixed sampled time-aggregated data (e.g., average response times or average throughput). If the input data consists of time series with individual events, the library automatically computes the required time-aggregated data.

- Create a new Librede estimation model in this project using the wizard ”Librede Estimation Model” (see screenshots below). A complete estimation model can be found in the examples ZIP archive. However, note that the configuration just serves as an example and does not work right away (as the file-paths have to be adapted). Therefore, we will create a new estimation model in the next steps.





- The estimation approaches require a description of the workload of the system under study. The workload description consists of the services (also known as workload classes), which are distinguished during estimation, and the resources for which resource demands will be estimated. Add services WC0, WC1, WC2 and resource Host1 in the workload description. Set number of servers to 1 and the scheduling strategy to ”Unkown”, as we do not want to make any assumptions on the scheduling strategy.



Services

Services (or workload classes) are groups of requests with similar resource demand behaviors.

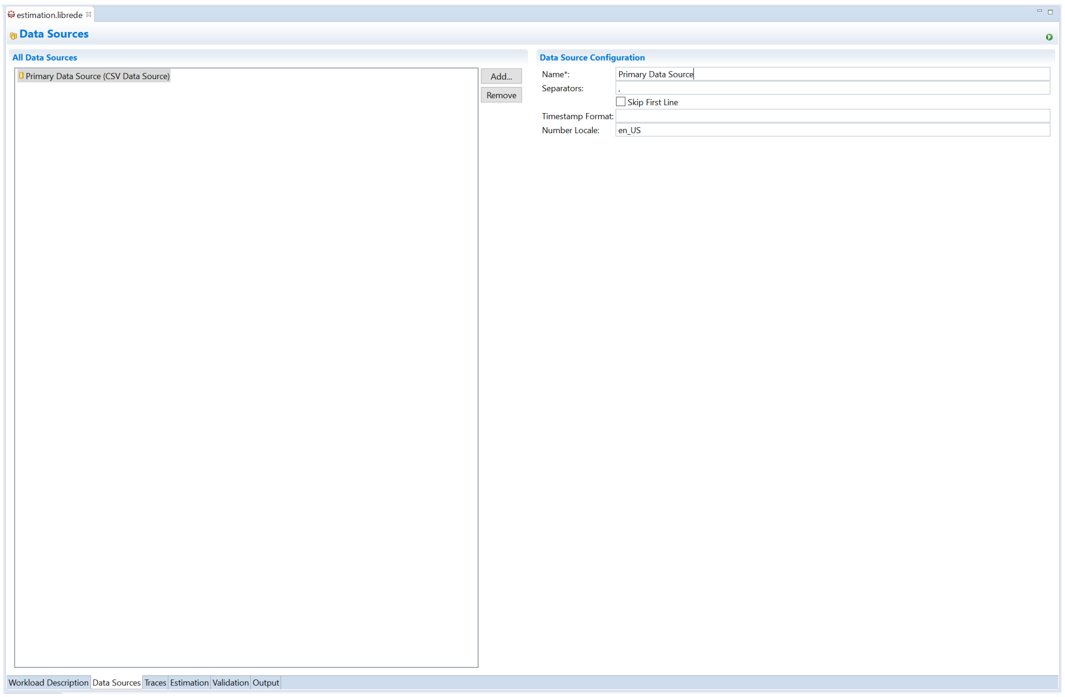
Name	Background ...
WC0	No
WC1	No
WC2	No

Resources

List all processing resources for which resource demands should be determined.

Name	Number of S...	Scheduling S...
Host1	1	Unkown

- Go to page *Data Sources* and add a new datasource with default values. The name property is just for easier identification. The separator char may be changed depending on the format of the CSV files used.



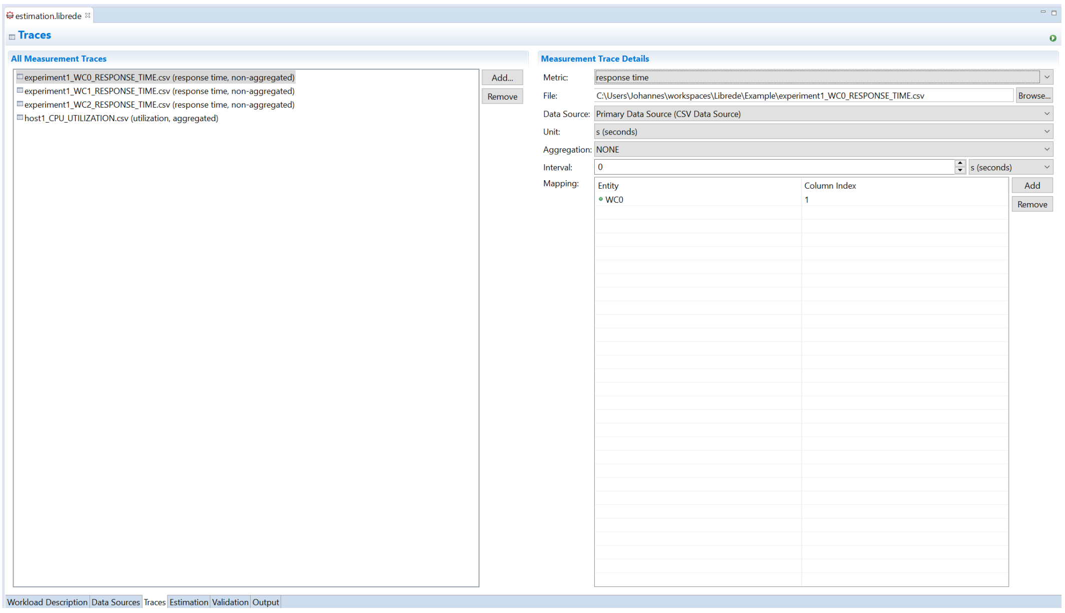
All Data Sources

Primary Data Source (CSV Data Source)

Data Source Configuration

Name*:	Primary Data Source
Separators:	,
	☐ Skip First Line
Timestamp Format:	
Number Locale:	en_US

- In the next step you define the input measurement traces. Go to page *Traces* in the editor and add a new measurement trace.


estimation.librede

Traces

All Measurement Traces

- experiment1_WC0_RESPONSE_TIME.csv (response time, non-aggregated)
- experiment1_WC1_RESPONSE_TIME.csv (response time, non-aggregated)
- experiment1_WC2_RESPONSE_TIME.csv (response time, non-aggregated)
- host1_CPU_UTILIZATION.csv (utilization, aggregated)

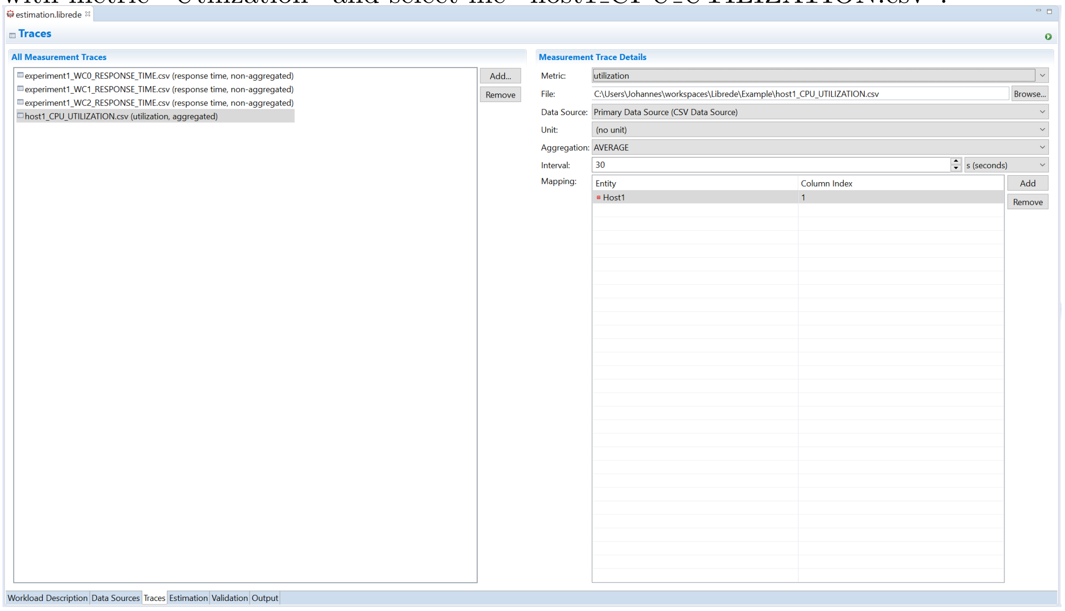
Measurement Trace Details

Metric: response time
File: C:\Users\Johannes\workspaces\Librede\Example\experiment1_WC0_RESPONSE_TIME.csv
Data Source: Primary Data Source (CSV Data Source)
Unit: s (seconds)
Aggregation: NONE
Interval: 0 s (seconds)
Mapping:

Entity	Column Index
WC0	1

Now select the newly created trace and edit the details. Set the *Metric* to "response time". For *File*, select the path to "experiment1_WC0_RESPONSE_TIME.csv" in the current project. Select the previously created data source. The measurements are given in seconds, so for *Unit*, we select "s (seconds)". Our traces are not aggregated, therefore we set *Aggregation* to "NONE" and the interval to 0 seconds. This means that the observations are not aggregated over a time-interval (i.e., in our case we have the response times of individual requests). Then, add a *mapping* to specify the association to service WC0. Column index 1 refers to the first non-timestamp column. Repeat these steps for WC1 and WC2. Make sure, that the other traces map to their corresponding entities (WC1 and WC2, respectively).

- Next, add a fourth trace describing the CPU Utilization of Host1. Create a new trace with metric "Utilization" and select file "host1_CPU_UTILIZATION.csv".


estimation.librede

Traces

All Measurement Traces

- experiment1_WC0_RESPONSE_TIME.csv (response time, non-aggregated)
- experiment1_WC1_RESPONSE_TIME.csv (response time, non-aggregated)
- experiment1_WC2_RESPONSE_TIME.csv (response time, non-aggregated)
- host1_CPU_UTILIZATION.csv (utilization, aggregated)

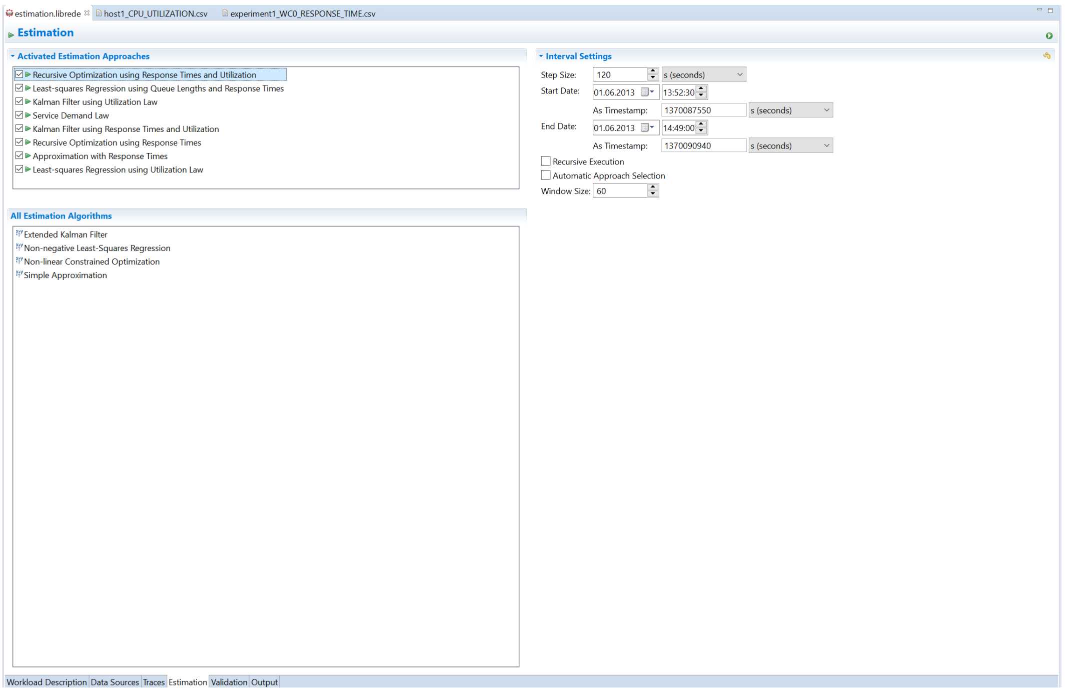
Measurement Trace Details

Metric: utilization
File: C:\Users\Johannes\workspaces\Librede\Example\host1_CPU_UTILIZATION.csv
Data Source: Primary Data Source (CSV Data Source)
Unit: (no unit)
Aggregation: AVERAGE
Interval: 30 s (seconds)
Mapping:

Entity	Column Index
Host1	1

We use the same data source as for all other traces. Leave *Unit*' to "(no unit)" as the utilization is given as a relative share. Our measurement traces describe the average utilization over the last 30 seconds. Therefore, set the aggregation to "AVERAGE" and the interval to 30 seconds. Create a mapping between resource "host1" and column 1.

- Next, switch to page *Estimation* to configure the estimation approach(es) and set the estimation interval.



estimation.librede | host1_CPU_UTILIZATION.csv | experiment1_WC0_RESPONSE_TIME.csv

Estimation

Activated Estimation Approaches

- ☑ Recursive Optimization using Response Times and Utilization
- ☑ Least-squares Regression using Queue Lengths and Response Times
- ☑ Kalman Filter using Utilization Law
- ☑ Service Demand Law
- ☑ Kalman Filter using Response Times and Utilization
- ☑ Recursive Optimization using Response Times
- ☑ Approximation with Response Times
- ☑ Least-squares Regression using Utilization Law

All Estimation Algorithms

- Extended Kalman Filter
- Non-negative Least-Squares Regression
- Non-linear Constrained Optimization
- Simple Approximation

Interval Settings

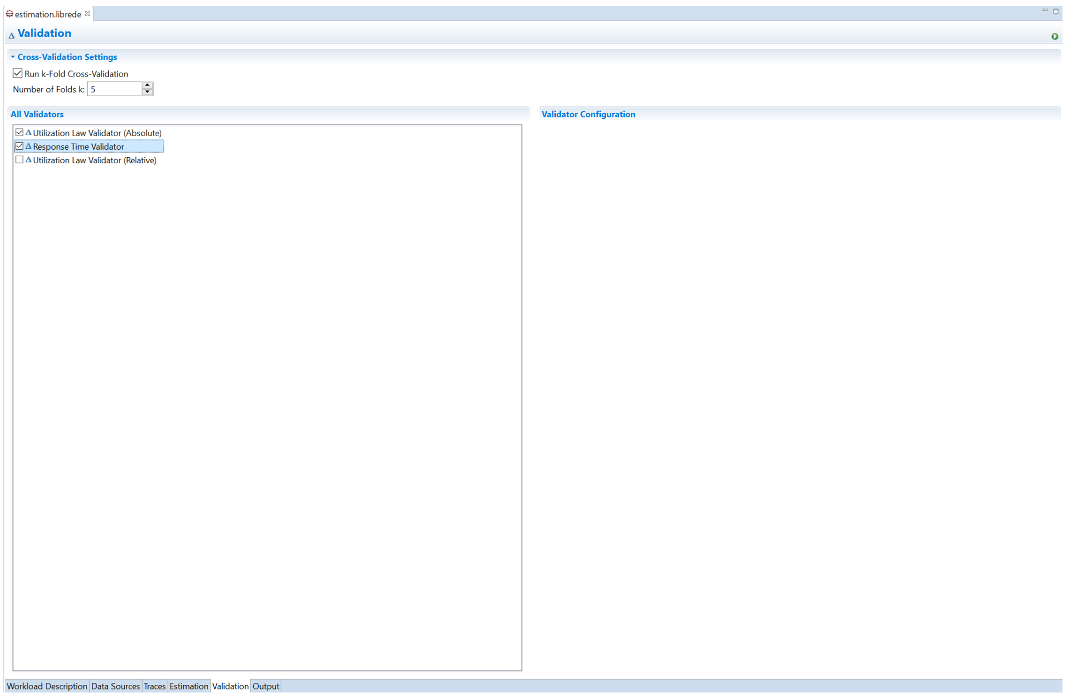
Step Size: 120 s (seconds)
Start Date: 01.06.2013 13:52:30
As Timestamp: 1370087550 s (seconds)
End Date: 01.06.2013 14:49:00
As Timestamp: 1370090940 s (seconds)
☐ Recursive Execution
☐ Automatic Approach Selection
Window Size: 60

The *step* size defines in which interval the estimates are updated. If there are traces with aggregated measurements, the step size must be equals or larger than their aggregation interval. However, the step size should not be too large as this may be result in too few samples for estimation. Use 120 seconds for this example.

- The *start date* and *end date* needs to specified based on the timestamps in the input measurement traces. Choose a start date of 01.06.2013 13:52:30 and an end date of 01.06.2013 14:49:00 for our example. Note that these timestamps get converted to unix timestamps (as shown below the dates) according to your local timezone. However, the given timestamps have to match to the timestamps in the measurement traces. The example traces were measured in CEST (UTC+2). Therefore, make sure to adapt the given timestamps to your timezone, or (preferably) just enter the unix timestamps 1370087550 as start and 1370090940 as end date. The date will the adapt accordingly.

- The *recursive execution* flag determines whether the resource demands at the end of each estimation interval will be written to the output or only the final result. The parameter *window size* can be used to control the maximum number of samples considered for estimation. Set the window size to 60 in the example. The flag *Automatic Approach Selection* automatically analyses the results and recommends the favorable approach. Disable both flags for this example.

- Check that all estimation approaches are activated. Some of the estimation approaches are configurable to optimize approach-specific parameters. Use the default values here.



☑ Run k-Fold Cross-Validation

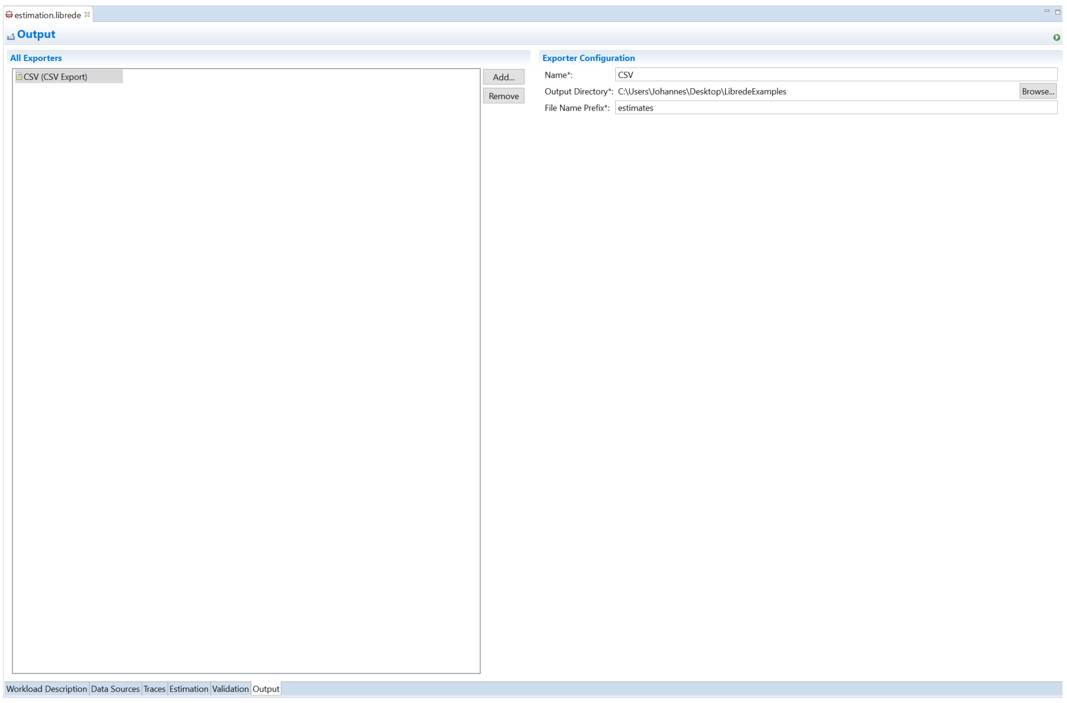
Number of Folds k: 5

☑ Utilization Law Validator (Absolute)

☑ Response Time Validator

☐ Utilization Law Validator (Relative)

- Switch to page *Validation*. Ensure that the k-fold Cross-Validation is enabled and set the number of folds to 5. Also, ensure that Response Time Validator and Utilization Law Validator (absolute) are enabled.



Name*: CSV

Output Directory*: C:\Users\Johannes\Desktop\LibredeExamples

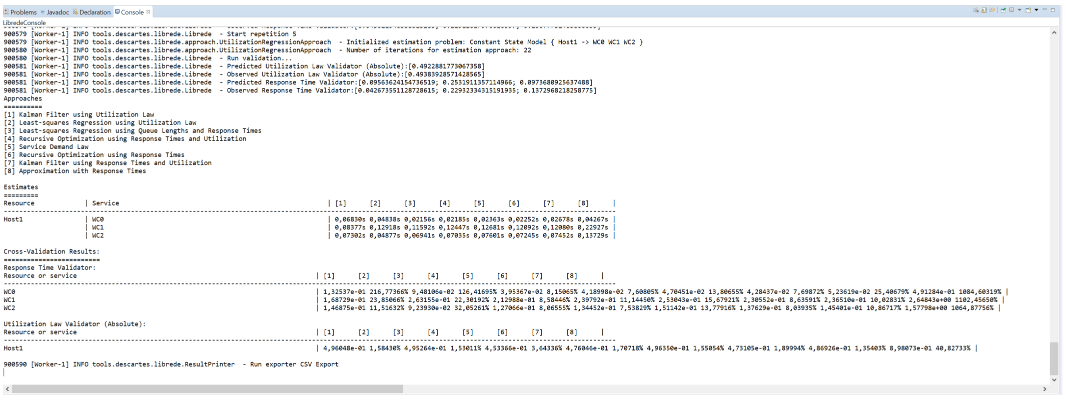
File Name Prefix*: estimates

- Go to page "Output" to configure the persistency of the results. Add a new CSV export. The *name* property is just for easier identification. Set the *output directory* to a directory of your choice. The value of property *file name prefix* will be prepended to each created output file. Set it to "estimates".

  Note: It is not necessary to configure a file output. If you are fine with console output, you can skip this step.

- Start the estimation by clicking the green arrow in the upper right corner of the editor.
- The results of the estimation and the cross-validation are printed to the console and the output files.

  Note that your results are probably not identical to the ones shown here. Execution result can differ slightly for every execution. This is due to random selections of the k validation folds and random states of some individual estimators.



```
900579 [Worker-1] INFO tools.descartes.librede.Librede  - Start repetition 5
900579 [Worker-1] INFO tools.descartes.librede.approach.UtilizationRegressionApproach  - Initialized estimation problem: Constant State Model { Host1 -> WC0 WC1 WC2 }
900580 [Worker-1] INFO tools.descartes.librede.approach.UtilizationRegressionApproach  - Number of iterations for estimation approach: 22
900580 [Worker-1] INFO tools.descartes.librede.Librede  - Run validation...
900581 [Worker-1] INFO tools.descartes.librede.Librede  - Predicted Utilization Law Validator (Absolute):[0.4922881773067358]
900581 [Worker-1] INFO tools.descartes.librede.Librede  - Observed Utilization Law Validator (Absolute):[0.49383928571428565]
900581 [Worker-1] INFO tools.descartes.librede.Librede  - Predicted Response Time Validator:[0.09563624154736519; 0.2531911357114966; 0.0973680925637488]
900581 [Worker-1] INFO tools.descartes.librede.Librede  - Observed Response Time Validator:[0.04267355112872861 5; 0.22932334315191935; 0.1372968218258775]
Approaches
==========
[1] Kalman Filter using Utilization Law
[2] Least-squares Regression using Utilization Law
[3] Least-squares Regression using Queue Lengths and Response Times
[4] Recursive Optimization using Response Times and Utilization
[5] Service Demand Law
[6] Recursive Optimization using Response Times
[7] Kalman Filter using Response Times and Utilization
[8] Approximation with Response Times

Estimates
=========
Resource             | Service                                                    | [1]      [2]      [3]      [4]      [5]      [6]      [7]      [8]      |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Host1                | WC0                                                        | 0,06830s 0,04838s 0,02156s 0,02185s 0,02363s 0,02252s 0,02678s 0,04267s |
                     | WC1                                                        | 0,08377s 0,12918s 0,11592s 0,12447s 0,12681s 0,12092s 0,12080s 0,22927s |
                     | WC2                                                        | 0,07302s 0,04877s 0,06941s 0,07035s 0,07601s 0,07245s 0,07452s 0,13729s |

Cross-Validation Results:
========================
Response Time Validator:
Resource or service                                                      | [1]      [2]      [3]      [4]      [5]      [6]      [7]      [8]      |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
WC0                                                                      | 1,32537e-01 216,77366% 9,48106e-02 126,41695% 3,95367e-02 8,15065% 4,18998e-02 7,60805% 4,70451e-02 13,80655% 4,28437e-02 7,69872% 5,23619e-02 25,40679% 4,91284e-01 1084,60319% |
WC1                                                                      | 1,68729e-01 23,85066% 2,63155e-01 22,30192% 2,12988e-01 8,58446% 2,39792e-01 11,14450% 2,53043e-01 15,67921% 2,30552e-01 8,63591% 2,36510e-01 10,02831% 2,64843e+00 1102,45650% |
WC2                                                                      | 1,46875e-01 11,51632% 9,23930e-02 32,05261% 1,27066e-01 8,06555% 1,34452e-01 7,53829% 1,51142e-01 13,77916% 1,37629e-01 8,03935% 1,45401e-01 10,86717% 1,57798e+00 1064,87756% |

Utilization Law Validator (Absolute):
Resource or service                                                      | [1]      [2]      [3]      [4]      [5]      [6]      [7]      [8]      |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Host1                                                                    | 4,96048e-01 1,58430% 4,95264e-01 1,53011% 4,53366e-01 3,64336% 4,76046e-01 1,70718% 4,96350e-01 1,55054% 4,73105e-01 1,89994% 4,86926e-01 1,35403% 8,98073e-01 40,82733% |

900590 [Worker-1] INFO tools.descartes.librede.ResultPrinter  - Run exporter CSV Export
```

  And the generated output-files should be in the folder that you configured previously.

4 Building and Running LibReDE from Console

It is also possible to run LibReDE from console. For this, you have to first download and compile the source files it using Maven. This can be done by the following steps:

1. Clone the git repository from Bitbucket: "git clone https://bitbucket.org/librede/librede.git"
2. *(Optional)* Checkout the development branch: "git checkout develop"
3. Switch to the "tools.descartes.librede.releng" directory inside the downloaded "librede" directory.
4. Execute the maven packaging: "mvn clean install -DskipTests".
5. Switch to the "tools.descartes.librede.releng.standalone/target/standalone/console" directory and execute "librede.bat" or "librede.sh". You need to provide a valid librede configuration file to be executed.

5 Calling LibReDE from Java source code

The main functionality is encapsulated inside the `Librede` class in the `tools.descartes.librede` package. Here, you have different methods for executing LibReDE. The easiest one would be the static method `execute(LibredeConfiguration conf)`. It accepts a `LibredeConfiguration` as created by the GUI editor. However, this is just an XML-file that has to match the defined EMF-structure and can therefore also manipulated with any standard text editor. After termination, `execute(LibredeConfiguration conf)` will return an instance of `LibredeResults`, containing the estimates along with the error values in the different validation folds.

Any appropriate file can be loaded using the static method `loadConfiguration(Path path)`. This method tries to load the file on the given path and returns a `LibredeConfiguration`, if valid. The configuration implements a normal object structure. Therefore, it is also possible to modify or create `LibredeConfiguration` objects using normal Java commands, since EMF (Eclipse Modeling Framework) was used here. Its meta-model definition can be found in `tools.descartes.librede.model`

6 Approaches to Resource Demand Estimation

An *estimation approach* in Librede consists of the following three separate building blocks: *state model*, *observation model*, and *estimation alogrithm*. In the following we describe these components in more detail and document the currently available implementations.

6.1 Estimation Approaches

Table 6.1 shows for each estimation approach currently implemented by Librede which implementations of the state model, observation model and estimation algorithm are used. Details on the implementations can be found in the following sections.

Table 1: The table shows which implementations of state model, observation model and estimation algorithm are used by an estimation approach.

Estimation Approach	**State Model**	**Observation Model**	**Estimation Algorithm**
Service Demand Law	Constant	Service Demand Law	Simple approximation
Approximation with Response Times	Constant	Response time approximation	Simple approximation
Least-squares Regression using Utilization Law	Constant	Utilization Law	Nonneg. least squares regression
Least-squares Regression using Queue Lengths and Response Times	Constant	Response time equation	Nonneg. least squares regression
Kalman Filter using Utilization law	Constant	Utilization Law	Kalman Filter
Kalman Filter using Response times and Utilization	Constant	Response time equation, Utilization Law	Kalman Filter
Recursive Optimization using Response Times	Constant	Response time equation	Interior-point, non-linear, constrained optimization
Recursive Optimization using Response Times and Utilization	Constant	Response time equation, Utilization Law	Interior-point, non-linear, constrained optimization

6.2 State Models

The state model contains information about the hidden state vector $\mathbf{x}$ that should be estimated. In our case the hidden state are the resource demands. The state model contains the following information:

- The size of the vector of resource demands to be estimated. The size is usually set to $M \cdot N$ where M denotes the number of resources and N the number of services.
- A set of constraints encoding additional knowledge of the value of certain resource demands. For example, it is possible to specify certain upper and lower bounds on the resource demands.
- Initial values for the resource demand estimates. This is used by Kalman filter techniques and nonlinear optimization technique as starting point.
- A function $\mathbf{x}_{n+1} = f(\mathbf{x}_n)$ that returns the expected next state vector $\mathbf{x}_{n+1}$ based on the current estimated state vector $\mathbf{x}_n$. This can be used to encode additional knowledge about changes in the resource demands. Currently, only the Kalman filter techniques can exploit such knowledge.

Librede currently provides one implementation of a *Constant State Model*, where $x_{n+1} = x_n$ is assumed.

6.3 Observation Models

The observation model describes the function $\mathbf{y}_n = h(\mathbf{x}_n)$ between the current state vector $\mathbf{x}_n$ and the current observations vector $\mathbf{y}_n$. The observations vector $\mathbf{y}_n$ consists of values observed at the system of interest (e.g., average response time or CPU utilization). The function h is is the combination of different output functions. The following output equations are currently provided by Librede:

- *Response time approximation:* approximates the resource demand $D_{i,c}$ of workload class c at resource i with the observed response time $R_{i,c}$ of workload class c at resource i. This is only valid if queueing delays are insignificant.
- *Service Demand Law:* the resource demand $D_{i,c}$ of workload class c at resource i) is $D_{i,c} = \frac{U_{i,c}}{X_{i,c}}$ ($U_{i,c}$: utilization due to workload class i, and $X_{i,c}$ is the throughput of workload class c at resource i). The utilization $U_{i,c}$ is derived from the aggregate utilization U_i using the apportionment scheme used in [2].
- *Utilization Law:* the relationship between the average utilization U_i of resource i and the resource demands is $U_i = \sum_{c=1}^{C} X_{i,c} \cdot D_{i,c}$ (C: is the number of workload classes, $X_{i,c}$ is the throughput of workload class c at resource i, and $D_{i,c}$: resource demand of workload class c at resource i).
- *Response time equation:* the relationship between the end-to-end response time R_c of workload class c and the resource demands is $R_c = \sum_{j=1}^{I} \frac{D_{j,c}}{1-\sum_{k=1}^{C} \lambda_{i,k} \cdot D_{i,k}}$ (I: number of resources, C: number of workload classes, $\lambda i, c$: arrival rate of workload class c at resource i, and $D_{i,c}$: resource demand of workload class c at resource i).

6.4 Estimation Algorithms

The estimation algorithm takes the state model and the observation models and then estimates the resource demands using statistical techniques. Librede currently provides the following estimation algorithms:

- *Simple approximation:* approximates the resource demand with the average, maximum, minimum or sum of the output of the observation model.
- *Nonnegative least squares regression:* only applicable to observation models with a single output.
- *Kalman filter:* applicable to all non-linear observation models with no state constraints. Uses the Bayes++ library.
- *Interior-point, non-linear, constrained optimization:* applicable to all kinds of state and observation models. Uses the Ipopt library.

7 Configuration

This section describes all configuration parameters that can be changed with the editor. The editor (and the corresponding configuration file) consists of multiple pages. In total there are six configuration pages:

1. Workload Description
2. Data Sources
3. Traces
4. Estimation
5. Validation
6. Output

In the following, we will elaborate on each of those.

7.1 Workload Description

The workload description contains information about the workload that is estimated here. This includes information about which services exist, how these service are connected, what resources exist (type and number of servers), and which services are executed on which resources. Parameters are listed in the following table.

Parameter	**Description**
Services	
Name	A human-readable identifier for the service.
Background Service	Whether or not this service runs in the background.
Resources	

Parameter	Description
Name	A human-readable identifier for the resource.
Number of Servers	The number of servers processing requests in parallel (e.g., number of cores of a CPU).
Scheduling Strategy	A scheduling strategy that best models the processing at the resource. This information may be exploited by some of the estimation approaches. If set to *unkown*, the estimation approaches do not make any assumptions on the scheduling.

7.2 Data Sources

The data sources describe the type of data source that will be used for this estimation. Detailed configuration parameters for each data source are listed in the following table.

Parameter	Description
CSV Data Source	
Name	A human-readable identifier for the data source. Only for the display on the user interface.
Separators	A Java regular expression for matching the separator chars in the CSV file.
Skip First Line	Determines whether the first non-comment line in the CSV file is interpreted as data values. Comments are expected to start with "#".
Timestamp Format	The format of the timestamp of all traces of this data source. The timestamp format can be either a pattern as expected by java.util.SimpleDataFormat, or if it is a numerical timestamp, a specifier of the form [xx] where xx specifies the time unit, e.g., [ms].
Number Locale	The locale of the number as to be interpreted by Java. Default value is "en_US".

Other (custom) data source types might support more and/or different parameter sets.

7.3 Traces

This page describes single data source instances (e.g., files), what data types and aggregations each trace contains as well as which service or resource (as specified in the workload description) it is referring to. For each trace, you have to specify different parameters as defined in the following table.

Parameter	Description
Measurement Trace	
Metric	Specifies the metric of the measurements in the trace.
File	A path to a file containing measurement data. Only for the display on the user interface.

Parameter	Description
Data Source	Reference to a data source that is used to load the contents of the file.
Unit	The unit of the data of the trace. Available options are usually dependent on the chosen metric.
Aggregation	The measurement data is often provided as aggregated values over a fixed sampling interval. The interval specifies the time duration over which the data in the file is aggregated. If set to *NONE*, the data is assumed to be non-aggregated, i.e., the values represent measurements of individual requests (e.g., in the case of response time).
Interval	The time interval of one aggregated and reported metric (amount and unit). This is ignored, if Aggregation is set to *NONE*.
Mapping	The mapping specifies the associated entity of the workload description (service or resource) for each column in the trace. A mapping indicates that the observations in the measurement trace correspond to the specified resource or service.

7.4 Estimation

This page is for specifying the actual estimation. Here you can specify several parameters concerning the estimation approaches, next to selecting which approaches to execute. Some settings apply for all algorithms, while some parameters are specific to certain estimation algorithm types. The description of all available parameters can be found in the following table.

Parameter	Description
Interval Settings	
Step Size	The step size specifies the time interval in which the estimates are updated. If an estimation approach is based on time-aggregated measurements, this parameter also determines the aggregation interval of its input data. This aggregation interval can be changed independently from the aggregation interval of the input measurement traces. If both intervals are different, the input data will be converted accordingly. However, the step size must be greater or equal to the aggregation interval of the input measurements and should ideally be a multiple of it in order to avoid inaccuracies.
Start Date	Any measurements before that date will not be included in the estimation. The time here refers to the timestamps in the input time series. There exists a button to automatically detect the start time from the input files.

Parameter	Description
End Time	Any measurements after that date will not be included in the estimation. The time here refers to the timestamps in the input time series. There exists a button to automatically detect the start time from the input files. You can also configure the end time to be in the future. This is especially important, when online and continuous execution is targeted.
Recursive Estimation	If this flag is set, all transient resource demands estimates will be output. Otherwise only the end result will be given.
Approach Selection	Enables automatic approach selection. If set to true, all estimation approaches will be executed and the best one chosen based on cross-validation.
Window Size	Defines the size of the sliding window on the input measurements for recursive estimation. The size is specified in number of step (i.e., the product of window size and step size gives the actual sliding time window). Smaller values will improve the adaptivity of the estimator if the resource demands change over time. However, it may result in lower accuracy, if the window is too small.
Kalman Filter	
State Noise Covariance	A constant value used to fill the state noise covariance vector q_k. This is an internal parameter of Bayes++ for the prediction of the next state $x_{k\|k-1} = x_{k-1\|k-2} + G_k w_k$. The vector q_k is the covariance of w_k.
State Noise Coupling	A constant value used to fill the state noise coupling matrix G_k. This is an internal parameter of Bayes++ for the prediction of the next state $x_{k\|k-1} = x_{k-1\|k-2} + G_k w_k$.
Observe Noise Covariance	A constant value used to fill the diagonal of the observe noise covariance matrix. This is an internal parameter of Bayes++.
Recursive Optimization	
Solution Tolerance	Desired convergence tolerance of the solution.
Upper Bounds Infinity Value	Values greater than this value are considered as infinity.
Lower Bounds Infinity Value	Values smaller than this value are considered as negative infinity.
Log verbosity	Sets the default verbosity level for console output. The larger this value the more detailed is the output. The valid range for this integer option is $0 \leq$ `print_level` ≤ 12 and its default value is 5.

7.5 Validation

On the validation page, different options for validating the acquired estimates are configurable. Next to selecting the different validators to apply, the following options are possible.

Parameter	Description
Cross-Validation Settings	
k-fold Cross-Validation	If enabled, a k-fold cross validation is executed after estimation.
Number of Folds	Specifies k, the number of folds for the cross-validation. The cross-validation splits the input time series into k randomly chosen, equally sized folds and uses each fold once for validation.

Additional (custom) validators might offer further parameter settings.

7.6 Output

In the last section, you can specify type and location of the output (apart from logging to console).

Parameter	Description
CSV Export	
Name	A human-readable identifier for the exporter. This is only relevant for the user interface.
Output Directory	The directory where the output files should be stored.
File Name Prefix	A prefix that is prepended to the name of all generated files.

Other (custom) exporters might offer further parameter settings.

8 Extension Points

Librede provides a set of interfaces that can be implemented to extend its functionality. The following extensions are currently supported:

- `tools.descartes.librede.repository.IMetric`: Provide additional metrics that are supported as input data.
- `tools.descartes.librede.datasource.IDataSource`: Support additional file formats for the input measurement traces.
- `tools.descartes.librede.approaches.IEstimationApproach`: Create new estimation approaches.
- `tools.descartes.librede.validation.IValidator`: Create new validators to evaluate the accuracy of the resource demand estimates.
- `tools.descartes.librede.export.IExporter`: Support additional file formats for the output.

The class `tools.descartes.librede.registry.Registry` can be used to register new extensions. For more details see source code and javadocs.

9 Known Issues

This section contains a list of known bugs and issues that might be addressed in further releases. If you encounter any problems or bugs, not yet listed here, please contact the maintainers of LibReDE (see `http://descartes.tools/librede`).

- Approaches based on "Recursive Optimization" can not be executed in parallel.

References

[1] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82:3–22, Jan 2009.

[2] F. Brosig, N. Huber, and S. Kounev. Automated extraction of architecture-level performance models of distributed component-based systems. In *26th IEEE/ACM Intl. Conf. On Automated Software Engineering*, Nov 2011.

[3] Samuel Kounev, Fabian Brosig, Nikolaus Huber, and Ralf Reussner. Towards self-aware performance and resource management in modern service-oriented systems. In *Proceedings of the 7th IEEE International Conference on Services Computing (SCC 2010), July 5-10, Miami, Florida, USA*. IEEE Computer Society, July 2010.

[4] Samuel Kounev, Brosig Fabian, and Huber Nikolaus. Descartes meta-model (dmm).

[5] Stephan Kraft, Sergio Pacheco-Sanchez, Giuliano Casale, and Stephen Dawson. Estimating service resource consumption from response time measurements. In *VALUETOOLS '09*, pages 1–10, 2009.

[6] Edward D Lazowska, John Zahorjan, G Scott Graham, and Kenneth C Sevcik. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[7] Z. Liu, L. Wynter, C. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Perf. Eval.*, 63(1):36–60, 2006.

[8] Zhen Liu, Laura Wynter, Cathy H. Xia, and Fan Zhang. Parameter inference of queueing models for IT systems using end-to-end measurements. *Perform. Evaluation*, 63(1):36–60, 2006.

[9] D. Menascé. Computing missing service demand parameters for performance models. In *Proc. of the 2008 Computer Measurement Group (CMG) Conference*, pages 241–248, 2008.

[10] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. In *Proc. of the 1995 conf. of the Centre for Advanced Studies on Collaborative research*, page 54. IBM Press, 1995.

[11] Simon Spinner. Evaluating approaches to resource demand estimation. Master's thesis, Karlsruhe Institute of Technology (KIT), Am Fasanengarten 5, 76131 Karlsruhe, Germany, 7 2011.

[12] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[13] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong. Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications. In *2012 IEEE 5th Intl. Conf. on Cloud Computing*, pages 439–446, Jun 2012.

[14] T. Zheng, M. Woodside, and M. Litoiu. Performance model estimation and tracking using optimal filters. *IEEE Trans. on Soft. Eng.*, 34(3):391–406, 2008.