ICGT23 Artefact Evaluation for

“Advanced Consistency Restoration with Higher-Order Short-Cut Rules” by Lars Fritsche, Jens Kosiol, Adrian Möller and Andy Schürr

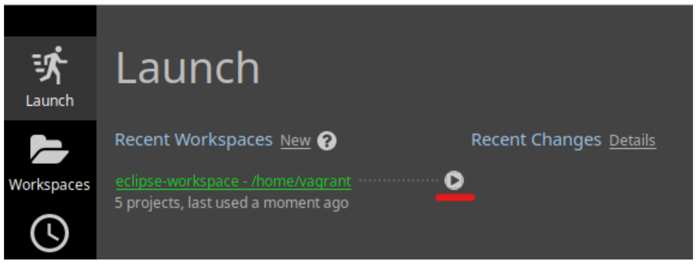
In the following, you will find all the steps inside the virtual maschine to reproduce our results and run the evaluation on your own system.

Eclipse Setup

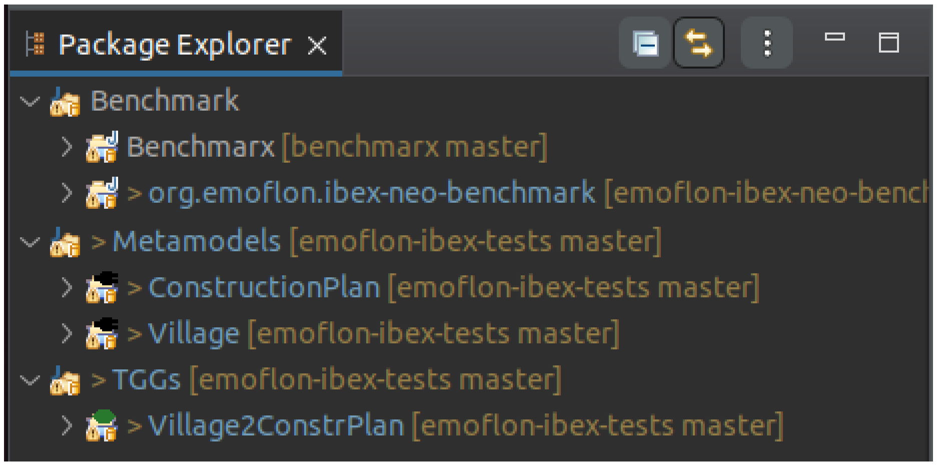
On your desktop you will find a folder, a PDF file, and a link:

- The folder **paper-eval-results** contains the original results used in our paper.
- The **ICGT23-Setup.pdf** file is a copy of this document, but without the first chapter “VM Setup”
- The link **eMoflon::IBeX Eclipse** points to a pre-configured Eclipse version with eMoflon installed.

1. Double-click on the **eMoflon::IBeX Eclipse** link to run eMoflon
2. A quick workspace configuration follows, which you can click through
3. When prompted for the workspace location, make sure that it matches the following one and click the **launch-button**


Launch
Launch
Workspaces
Recent Workspaces New
Recent Changes Details
eclipse-workspace - /home/vagrant
5 projects, last used a moment ago

Now, the Eclipse workspace should be up and running. On the left you should be able to see the three working sets **Benchmark**, **Metamodels,** and **TGGs**.


Package Explorer

```
Benchmark
  Benchmarx [benchmarx master]
  > org.emoflon.ibex-neo-benchmark [emoflon-ibex-neo-bench
> Metamodels [emoflon-ibex-tests master]
  > ConstructionPlan [emoflon-ibex-tests master]
  > Village [emoflon-ibex-tests master]
> TGGs [emoflon-ibex-tests master]
  > Village2ConstrPlan [emoflon-ibex-tests master]
```

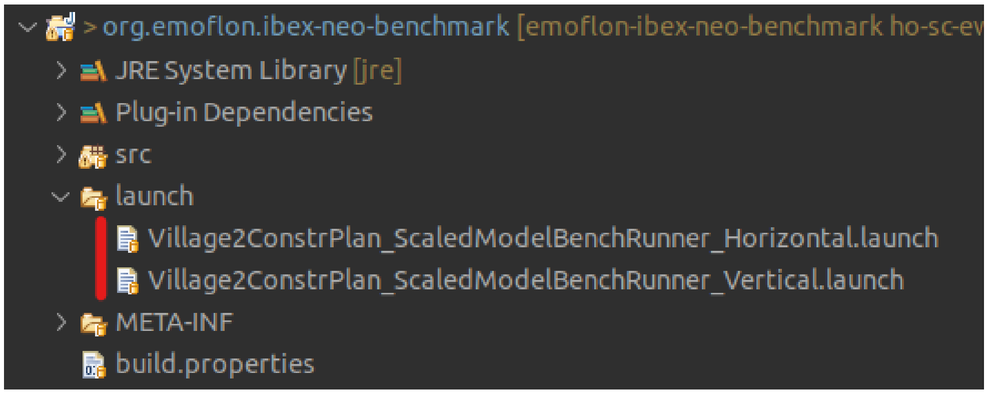
Metamodels contains ecore specifications of the used metamodels from our paper.
TGGs contains our example TGG project.
Benchmark contains the benchmark environment for performance evaluation.

Benchmarking

The benchmark can be run in two different modi:

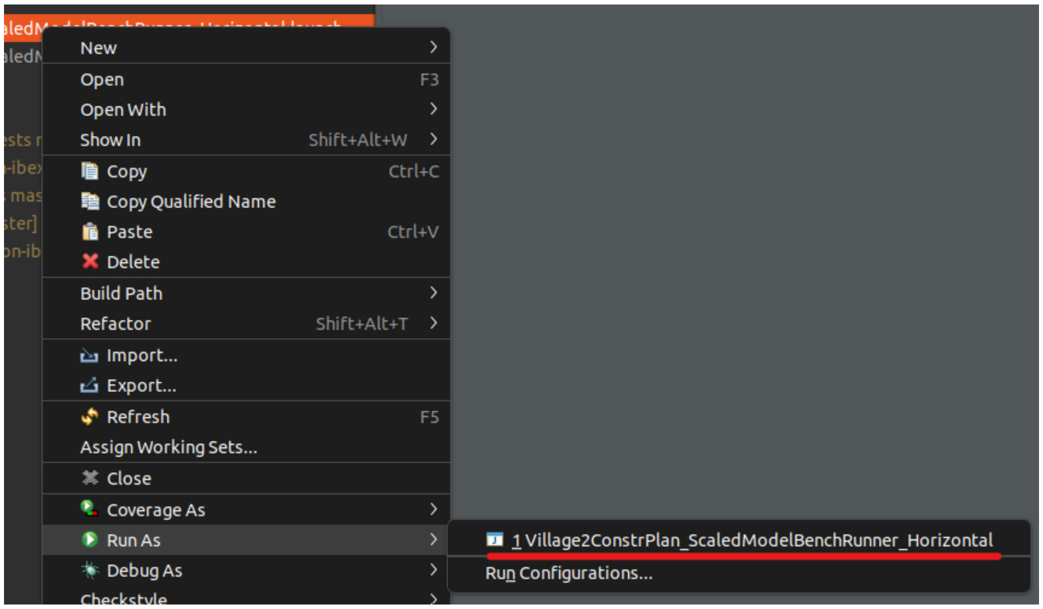
- Horizontal: By increasing the model size, the model grows horizontally
- Vertical: By increasing the model size, the model grows vertically

You can find a **.launch**-file for each modi in the **launch** folder of the **org.emoflon.ibex-neo-benchmark** project, which starts a series of benchmarks in the respective mode.



```
> org.emoflon.ibex-neo-benchmark [emoflon-ibex-neo-benchmark ho-sc-ev
    JRE System Library [jre]
    Plug-in Dependencies
    src
    launch
        Village2ConstrPlan_ScaledModelBenchRunner_Horizontal.launch
        Village2ConstrPlan_ScaledModelBenchRunner_Vertical.launch
    META-INF
    build.properties
```

To run a **.launch**-file, right-click on it and choose **Run As**.



A benchmark series comprises three different scenarios (**ADD_ROOT**, **REMOVE_ROOT**, **MOVE_ROW**). Each one is run with **no repair** framework, with the **legacy repair** framework, and with the **higher-order short-cut repair** framework.

For each combination, the **model size** is varied.
In horizontal mode, this is combined with a variation of the number of **model changes**.

Executing the benchmarks as described above, after some time, you should see outputs like this:

```
scaledModel;250;50;HORIZONTAL;ADD_ROOT;NONE;3252;1.784;2.193;118;1.0;1050;900;52607;0;300;350;true
scaledModel;250;50;HORIZONTAL;ADD_ROOT;NONE;3252;1.371;1.511;118;1.0;1050;900;52607;0;300;350;true
scaledModel;250;50;HORIZONTAL;ADD_ROOT;NONE;3252;1.438;2.351;118;1.0;1050;900;52607;0;300;350;true
scaledModel;250;50;HORIZONTAL;ADD_ROOT;NONE;3252;1.448;2.066;119;1.0;1050;900;52607;0;300;350;true
scaledModel;250;50;HORIZONTAL;ADD_ROOT;NONE;3252;1.679;1.767;118;1.0;1050;900;52607;0;300;350;true
scaledModel;500;50;HORIZONTAL;ADD_ROOT;NONE;6502;2.372;2.252;197;1.0;1050;900;91107;0;300;350;true
```

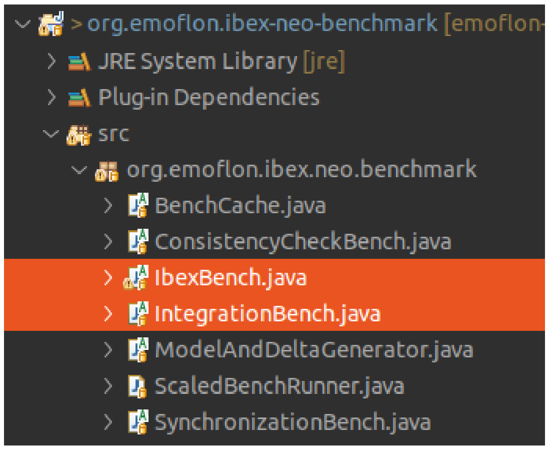
When the measurements are finished, the runner calculates average and median values for each run based on this output and appends them to the console.

model_scale: linear scaling factor for the generated model
num_of_changes: number of generated changes
elts: number of generated elements on source and target side
avg_init: average initialization time including initial pattern matching of generated model
median_init: median initialization time including initial pattern matching of generated model
avg_resolve: average time to propagate changes including repair operations
median_resolve: median time to propagate changes including repair operations
avg_ram: average memory used by the JVM
median_ram: median memory used by the JVM

In addition, we collected the number of created elements, deleted elements, found matches, repaired matches, revoked matches, and applied matches.

Implementation

The main benchmark implementation can be found in **IbexBench.java** and **IntegrationBench.java**.



```
> org.emoflon.ibex-neo-benchmark [emoflon-
  JRE System Library [jre]
  Plug-in Dependencies
  src
    org.emoflon.ibex.neo.benchmark
      BenchCache.java
      ConsistencyCheckBench.java
      IbexBench.java
      IntegrationBench.java
      ModelAndDeltaGenerator.java
      ScaledBenchRunner.java
      SynchronizationBench.java
```

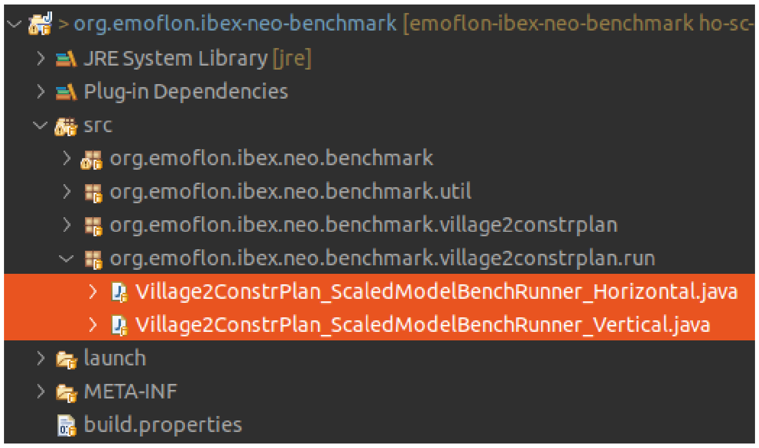
IbexBench::genAndBench is called, which first initializes an eMoflon stub and a model and delta generator, which generates the model and all deltas.

In **IntegrationBench::applyDeltaAndRun**, after initiating a dry run of eMoflon, the deltas are applied and eMoflon is run twice to propagate the changes. Finally, the benchmark results are collected and printed.

Troubleshooting

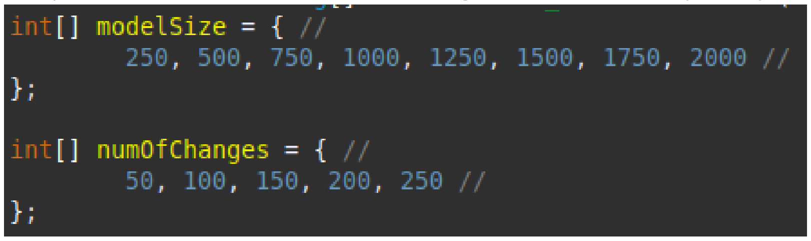
Note that our evaluation was performed on a workstation with 64GB RAM and an AMD Ryzen 9 3900x. eMoflon relies on incremental pattern matching, which at this moment still consumes a lot of memory. However, the currently employed technology highly benefits from multi core setups. However, this means that if your system has lower specs than this, you should alter the configurations of the benchmark.

For this, open a runner class:



```
> org.emoflon.ibex-neo-benchmark [emoflon-ibex-neo-benchmark ho-sc-
  JRE System Library [jre]
  Plug-in Dependencies
  src
    org.emoflon.ibex.neo.benchmark
    org.emoflon.ibex.neo.benchmark.util
    org.emoflon.ibex.neo.benchmark.village2constrplan
    org.emoflon.ibex.neo.benchmark.village2constrplan.run
      Village2ConstrPlan_ScaledModelBenchRunner_Horizontal.java
      Village2ConstrPlan_ScaledModelBenchRunner_Vertical.java
  launch
  META-INF
  build.properties
```

Adapt the **modelSize** and **numOfChanges** variables to fit your system:

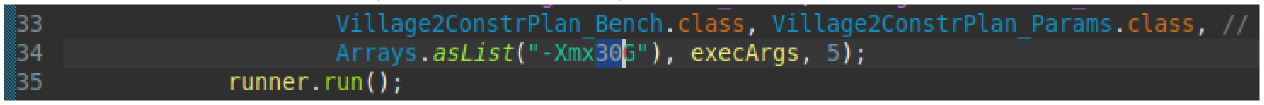


```java
int[] modelSize = { //
        250, 500, 750, 1000, 1250, 1500, 1750, 2000 //
};

int[] numOfChanges = { //
        50, 100, 150, 200, 250 //
};
```

If your system has less memory and also less cores, consider choosing lower values for **modelSize**. Note that due to the implementation the values for **numOfChanges** should be equal or less than those of **modelSize**.

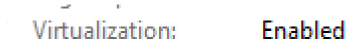
Problems with JVM memory:
If you run into errors while executing the benchmarks, try to increase the JVM memory limit. It can be found in the runners at line 34 (current value 30 GB):



```java
33                        Village2ConstrPlan_Bench.class, Village2ConstrPlan_Params.class, //
34                        Arrays.asList("-Xmx30G"), execArgs, 5);
35              runner.run();
```

Problems with the VM:
If there are problems when starting the VM, please check if there are any warnings from VirtualBox.
We recommend reading them thoroughly and solving them as recommended by the tool.

If you get an error stating that virtualization has been deactivated for your system, you may have to activate this feature in your BIOS.
If you are using Windows, you can find out if virtualization is enabled via accessing the **Task Manager -> Performance**.



```
Virtualization:        Enabled
```