Hawk User Guide

Monitoring Tool of MEASURE Platform

Hawk Server Overview

Hawk is a model indexing solution that can take models written with various technologies and turn them into graph-based databases for easier and faster querying. The indexing solution integrates Modelio modeling tool which contains different types of models such as UML, Archimate, BPMN, Analyst, etc.

- Hawk allows to perform fast queries on models (on modelio models in particular) using EOL language. EOL (*Epsilon Object Language*) is a programming language for creating, querying and modifying EMF models (Eclipse Modeling Framework).
- Hawk provide querying engines for executing EOL queries on models such as Modelio models.
- Here is an example of EOL query that return the total number of classes: “return Class.all.size;”. For more information see the following lik: *https://github.com/mondo-project/mondo-hawk/wiki/Example-queries-on-Modelio-based-UML-models*.
- Hawk is integrated in the monitoring and analysis tool “Measure Platform”. It allows developers to collect measures associated to the design phase in the development process.

Running Hawk Server

- Hawk Server is executed in command line interface by executing the script haw-server.sh in the root folder.
- After running the server, you must wait until the indexation of the models of the repositories is completed.
- After the indexation is finished, you will see the following message if the indexation is successful:

```
Updated Hawk instance DataBio (success). 13831 s 341 ms
```

- If the indexation is failed, you will see the following message

```
Updated Hawk instance DataBio (failed). 13831 s 341 ms
```

- The output of the server execution is displayed in console and stored in “hawk.log” file. It can be useful to see the errors that caused the updating failure.
- If you haven’t specified an automatic update in the configuration file, you can run the command ./sync-server.sh to synchronize the indexed model in graph database with the model of the updated local or SVN repository.

Hawk Server Commands

Console Client Commands

You can run commands in the console client or via web services:

- `hawkListBackends`: Lists the available Hawk backends

- `hawkListInstances`: Lists the available Hawk instances
- `hawkStartInstance <name>`: Starts the instance with the provided name
- `hawkStopInstance <name>`: Stops the instance with the provided name
- `hawkSyncInstance <name> [waitForSync:true|false]` : Requests an immediate sync on the instance with the provided name

See the following link for more details: *https://github.com/mondo-project/mondo-hawk/wiki/Console-client*

Web Services

Hawk allows to execute POST http queries with common URL "http://localhost:8080/thrift/hawk/json" and specific body messages to each command.

Here is a list of commands with the specific body message:

- List of instances

  [1,"listInstances",1,1,{}]

- Start an instance

  [1,"startInstance",1,1,{"1":{"str":"<name>"}}]

- Stop an instance

  [1,"stopInstance",1,1,{"1":{"str":"<name>"}}]

- Synchronize an instance and wait for response

  [1,"syncInstance",1,1,{"1":{"str":"<name>"},"2":{"tf":1}}]

- Synchronize an instance and don't wait for response

  [1,"syncInstance",1,1,{"1":{"str":"<name>"},"2":{"tf":0}}]

- Queries
  - Non-temporal query on "Measure" instance

```
[1,"query",1,1,{"1":{"str":"Measure"},"2":{"str":"return
Class.all.size;"},"3":{"str":"org.hawk.epsilon.emc.EOLQueryEngine"},"4":{"r
ec":{}}}]
```

  - Temporal query on "DataBio" instance

```
[1,"query",1,1,{"1":{"str":"DataBio"},"2":{"str":"return
Model.allInstancesNow.size;"},"3":{"str":"org.hawk.timeaware.queries.TimeAw
areEOLQueryEngine"},"4":{"rec":{}}}]
```

For more information about temporal queries see the following link : *https://github.com/mondo-project/mondo-hawk/wiki/Temporal-queries-in-Hawk* .

You can run these commands via a navigator extension like "Postman" or with Linux "curl" command.

Here is an example of a curl commands

- Synchronize “DataBio” instance

```
curl -s -H 'Content-Type: application/json' -d
'[1,"syncInstance",1,1,{"1":{"str":"DataBio"},"2":{"tf":1}}]'
http://localhost:8080/thrift/hawk/json
```

- List of instances

```
curl -s -H 'Content-Type: application/json' -d '[1,"listInstances",1,1,{}]'
http://localhost:8080/thrift/hawk/json
```

Querying Language

All the queries are written in the *Epsilon Object Language* which is an imperative programming language for creating, querying and modifying EMF models (Eclipse Modeling Framework).

Examples of EOL queries supported by Hawk :

- Returns the number of instances of "Class" in the index:

  ```
  return Class.all.size;
  ```

- Temporal query : Returns the number of instances of "Class" of the last revision:

  ```
  return Class.latest.all.size;
  ```

- Advanced example: loops, variables and custom operations

  ```
  var counts = Sequence {};

  var i = 0;var n = count(0);

  while (n > 0) {  counts.add(Sequence {">" + i, n});  i = i + 1;  n = count(i);}

  return counts;

  operation count(n) {  return Class.all.select(c|c.ownedOperationCount > n).size;}
  ```

Associated Measures

A set of metrics dedicated to the Measure Platform has been developed specifically to make the link between the platform and the Hawk measurement tools.

Hawk allow us to perform queries over all Modelio models including UML, Archimate, Analyst, etc. Some of the measures are basic like the number of UML classes, methods, interfaces, etc. In addition to this, Hawk allows to do complex queries in short time such as “Class Complexity Index”, “Package Dependencies Ratio”, etc.

Measure	Type	Description
HawkMeasure	Generic Measure	Generic Measure which allow to execute a Hawk query provided as parameter of the measure.

NumberOfRequirement	Modelio Requirements	Total number of Requirement defined in the selected scope.
NumberOfTests	Modelio Requirements	Total number of Tests defined in the selected scope.
NumberOfBusinessRule	Modelio Requirements	Total number of Business Rule defined in the selected scope.
NumberOfGoals	Modelio Requirements	Total number of Goals defined in the selected scope.
NumberOfRisks	Modelio Requirements	Total number of Risks defined in the selected scope.
RequirementTracabilityToImplementationIndice	Modelio Requirements	The % of requirement of tracing an implementation model.
RequirementTracabilityToTestIndice	Modelio Requirements	The % of requirement of tracing a test model.
RequirementCoverageIndice	Modelio Requirements	The average number of requirements tracing an architecture model.
RequirementComplexityIndice	Modelio Requirements	The average number of sub requirements defined to refine an existing requirement.
RequirementsSatisfactionQualityIndice	Modelio Requirements	Percentage of requirements that have been satisfied.
SoftwareComponentDecomposition	Modelio UML	The number of software components identified in an application architecture.
ClassDependeciesRatio	Modelio UML	The average number of dependencies from a class.
PackageDependeciesRatio	Modelio UML	The average number of dependencies from a package.
ModelAbstractnessIndex	Modelio UML	The % of abstract classes (and interfaces) divided by the total number of types in a package.
ClassComplexityIndex	Modelio UML	Moy of direct subclasses of a class. A class implementing an interface counts as a direct child of that interface.
InterfaceByComponentRatio	Modelio UML	The average number of dependencies from a class.
NumberOfClasses	Modelio UML	Total number of classes in the selected scope
NumberOfFields	Modelio UML	Total number of fields defined in the selected scope.
NumberOfInterfaces	Modelio UML	Total number of interfaces in the selected scope.
NumberOfMethods	Modelio UML	Total number of methods defined in the selected scope.

NumberOfComponent	Modelio UML	Total number of Components defined in the selected scope.
NumberOfPackage	Modelio UML	Total number of Packages defined in the selected scope.
NumberOfUseCase	Modelio UML	Total number of UseCases defined in the selected scope.
NumberOfActors	Modelio UML	Total number of Actores defined in the selected scope.