Declarative Dynamic Object Reclassification (Artifact)

Riccardo Sieve
University of Oslo, Norway

Eduard Kamburjan
IT University of Copenhagen, Denmark

Ferruccio Damiani
University of Turin, Italy

Einar Broch Johnsen
University of Oslo, Norway

Abstract

This artifact contains the GreenhouseDT API, a backend API for the GreenhouseDT system, which implements the declarative dynamic object reclassification in the SMOL language, as specified and described in the paper *"Declarative Dynamic Object Reclassification"*. It also contains a Python to reproduce the results of the paper. This artifact description describes how the artifact is structured, as well as how to build and run it.

2012 ACM Subject Classification Software and its engineering → Abstraction, modeling and modularity; Software and its engineering → Object oriented languages

Keywords and phrases Dynamic Object Reclassification, Dynamic Software Updates, Featherweight Java, Knowledge Bases, Semantic Reflection, Type Soundness

Digital Object Identifier 10.4230/DARTS.11.2.6

Related Article Riccardo Sieve, Eduard Kamburjan, Ferruccio Damiani, and Einar Broch Johnsen, "Declarative Dynamic Object Reclassification", in 39th European Conference on Object-Oriented Programming (ECOOP 2025), LIPIcs, Vol. 333, pp. 29:1–29:31, 2025.
`https://doi.org/10.4230/LIPIcs.ECOOP.2025.29`

Related Conference 39th European Conference on Object-Oriented Programming (ECOOP 2025), June 30–July 2, 2025, Bergen, Norway

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2025 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

This artifact contains (1) the implementation of the declarative dynamic object reclassification in the SMOL language, and (2) scripts in order to reproduce the experiments described in the paper. Furthermore, it describes how to use the system as a backend API for the GreenhouseDT system.

Note that the reproducing all the set of experiments described in the paper will be time-consuming as the system is provided with a basic triple store without strong reasoning capabilities.

2 Content

This artifact contains three main components: (1) The GreenhouseDT API itself, (2) a `setup.py` Python script to install the artifact, and (3) the `ecoop-artifact-test.py` Python script to run the tests described in the Results section of the paper.

GreenhouseDT API

The API is located in `/GreenHouseAPI`. The API is built using `Kotlin` and `Spring Boot`, and is designed to be run in a Docker container. The API exposes a set of endpoints that can be used to interact with the system and the different components that are to be used in the adaptation process. The API is structured as follows:

- `src/main/kotlin/org/smolang/greenhouse/api` contains the main implementation of the API.
- `src/main/resources` contains the configuration files for the API, the ontology, and the SMOL code.
- `Dockerfile` is used to build the Docker image for the API.
- `README.md` contains the documentation for the API.

Python scripts

The Python scripts are located in the main folder and contain the experiments described in the Results section of the paper.

The setup script configures the artifact and checks that all processes are running. Note that the `requests` and `influxdb-client` packages have to be installed for the test to be run. It is posisble to install them from the `requirements.txt` file using pip. The setup script can install them by adding the `-pip` argument.

The script then setup the architecture using `docker compose`, using by default the prebuilt `x86` images. It is possible to use the `arm` version of the images by adding the `-arm` argument to the script, or build the images instead of using the prebuilt one with the `-build` argument.

The test script should be run in a python environment, but is by default run using the python version installed directly, provided the installation of the `requests` and `influxdb-client` libraries. The script updates the API with the different values and checks the results of the different tests. The results prompted during the execution will match the results from the experiments section for the pumps, plus the tests on adaptation of the plants.

3 Getting the artifact

The artifact containing the code and the docker compose file to run the system is available at `https://doi.org/10.5281/zenodo.15361832`. The newest version of the code for the Backend API can be found at `https://github.com/sievericcardo/greenouse-dt-api`. The code can also be manually compiled using the gradle wrapper in the API folder with `./gradlew build`.

4 Tested platforms

We tested the implementation on (1) a Macbook Pro 14 with M2 Pro chip, running MacOS Sequoia 15.3, (2) a Windows machine with a Intel i7-3930k CPU and 16GB RAM, and (3) a Linux machine, running Ubuntu 22.04 with a Intel i7-1355U CPU and 16GB RAM. The Docker container is not dependent on a specific architecture.

5 License

The artifact is available under license Creative Commons Attribution 4.0 International.

Table 1 Plant states based on NDVI values. We show the results for 4 plants.

Plant ID	NDVI	Expected State	Actual State
After Instantiation			
Plant 1	0.0	Unhealthy	Unhealthy
Plant 2	0.0	Unhealthy	Unhealthy
Plant 3	0.0	Unhealthy	Unhealthy
Plant 4	0.0	Unhealthy	Unhealthy
After First Reclassification			
Plant 1	-1.0	Dead	Dead
Plant 2	0.1	Unhealthy	Unhealthy
Plant 3	0.95	Healthy	Healthy
Plant 4	0.75	Healthy	Healthy
After Second Reclassification			
Plant 1	1.0	Healthy	Healthy
Plant 2	-0.84	Dead	Dead
Plant 3	-0.75	Dead	Dead
Plant 4	0.1	Unhealthy	Unhealthy

6 MD5 sum of the artifact

332aaeb0e86d326ca3b1d9dea2d79df0

7 Size of the artifact

2.6 GB

A Running the tests and comparing the results

To check the reproducibility of experiments, and to ensure that the artifact is functional and can be reusable, the `ecoop-artifact-test.py` contains the tests on both plants and pumps; the first set of tests checks the adaptation process against plants as shown in Table 1.

The second set of tests checks the adaptation process against pumps, as described in Table 1 in Section 5 of the paper.