Claims of the Artifact

We believe that this artifact should be awarded both (a) Open Research Object (ORO) and (b) Research Object Reviewed (ROR) -- Reusable badges. To qualify for either badge, artifacts must be functional, and so we first make a case that our artifact is functional before describing how it satisfies the criteria of the ORO and ROR badges.

Functional

The ICSA artifact evaluation track website deems an artifact to be Functional if it satisfies the criteria below:

```
In order to obtain any badge every artifact will be required to be Functional, that is, it m
consistent, complete, exercisable, and include appropriate evidence of verification and vali
```

Where the definitions of documented, consistent, complete, and exercisable come from the **Artifacts Evaluated -- Functional v1.1** criteria laid out in the ACM policy on Artifact Review and Badging Version 1.1:

```
* Documented: At minimum, an inventory of artifacts is included, and sufficient description
  artifacts to be exercised.

* Consistent: The artifacts are relevant to the associated paper, and contribute in some inh
  generation of its main results.

* Complete: To the extent possible, all components relevant to the paper in question are inc
  (Proprietary artifacts need not be included. If they are required to exercise the package
  documented, along with instructions on how to obtain them. Proxies for proprietary data sh
  demonstrate the analysis.)

* Exercisable: Included scripts and/or software used to generate the results in the associat
  successfully executed, and included data can be accessed and appropriately manipulated.
```

Below, we argue how this artifact satifies each of the above criteria:

- **Documented:** An outline of the replication package and a detailed description of its contents are provided in README.rst. Installation instructions for the artifact are given in INSTALL.rst, and steps for reproducing the results of the paper are given in README.rst.

- **Consistent:** The artifacts provided in this replication package include the source code and executables for ROSDiscover toolchain (i.e., an implementation of the technique presented in the paper), prebuilt images for the evaluation dataset (as well as their sources), the infrastructure used to run all of the experiments in the paper, and the raw and processed results themselves.
- **Complete:** See above. All of the necessary software, data, and instructions for reproducing the results of the paper are included in this replication package.
- **Exercisable:** We provide a Docker-based setup for reliably reproducing the results of the experiments reported in the paper, complete with instructions for installing and running that setup.

Open Research Object (ORO)

The NISO Recommended Practice on Reproducibility Badging and Definitions give the following description of the ORO badge:

```
This badge signals that author-created digital objects used in the research (including data
are permanently archived in a public repository that assigns a global identifier and guarant
persistence, and are made available via standard open licenses that maximize artifact availa
```

Similarly, the ICSA artifact evaluation track website describes artifacts with the ORO badge as being:

```
Functional + placed on a publicly accessible archival repository.
A DOI or link to this repository along with a unique identifier for the object is provided (
```

This replication package is hosted on Zenodo, a long-term archival service that will exist as long as its host, the CERN laboratory, which has an experiment programme defined for at least the next 20 years. The replication package can be accessed via the following DOI: https://doi.org/10.5281/zenodo.5834633.

Research Object Reviewed (ROR) -- Reusable

The ICSA artifact evaluation track gives the following criteria for this badge:

```
Functional + very carefully documented and well-structured to the extent that reuse and repu
In particular, norms and standards of the research community for artifacts of this type are
```

This replication package contains components that are designed for reuse and repurposing:

- **ROSDiscover (https://github.com/rosqual/rosdiscover):** provides an implementation of the technique described in the paper. It is designed to, among other purposes, recover run-time architectures from ROS applications, provided in the form of a Docker image and an accompanying configuration file. Further instructions on the general use of ROSDiscover can be found in its README file, available either in its archival form in the `deps/rosdiscover` directory of this artifact, or, preferably, in its up-to-date form on GitHub at:https://github.com/rosqual/rosdiscover.

- **ROSWire (https://github.com/rosqual/roswire):** is a standalone Python library, used as part of the ROSDiscover toolchain, that provides extensive functionality for building static and dynamic tools for ROS that accept Docker images as their input (rather than assuming that those tools are located on the same machine as the subject of the analysis). ROSWire has been used for a variety of purposes including, but not limited to, the following:
  - Afzoon Afzal. 2021. Automated Testing of Robotic and Cyberphysical Systems. PhD Thesis. Carnegie Mellon University, USA.
  - Deborah S. Katz. 2020. Identification of Software Failures in Complex Systems Using Low-Level Execution Data. PhD Thesis. Carnegie Mellon University, USA.
  - A. Afzal, C. Le Goues and C. S. Timperley, ”Mithra: Anomaly Detection as an Oracle for Cyberphysical Systems,” in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2021.3120680.
- **Our evaluation dataset:** provides executable, historically accurate Docker images for several popular open-source ROS systems and architectural misconfigurations in those systems. Producing these images is non-trivial and requires many careful steps to end up with Docker images that are exercisable, complete, and historically accurate. This dataset and its associated images may be used by other researchers who wish to look into misconfigurations in ROS, or static analysis of ROS systems more generally.
- **Our experiment infrastructure:** can be both (a) reused in other studies of ROS architectural misconfiguration bugs, and (b) repurposed for the purpose of evaluating other static and dynamic analysis techniques for ROS systems (e.g., program repair, testing, fuzzing, static analysis). In particular, we provide infrastructure for building historically accurate Docker images of ROS systems that can be reused to build versions of other ROS systems.

Throughout all of these artifacts, we follow software engineering best practices and the norms and practices of relevant ecosystems (e.g., Docker, Python).