Modeling DevOps

Metamodel

The **Pipeline** is the initial class of our metamodel, containing an attribute titled “Name.” In EMF, we use the “Name0” due to the problems in generating the IDE when more than one class has the same attribute. Objects of the class pipeline can contain several When and Job objects. The pipeline can have references to Tools_frameworks.

Objects of the class **When** have three attributes “Name,” Timer,” and “Trigger.” The attributes indicate the time or action necessary to start the Pipeline in a DevOps process.

Objects of the class **Job** have only one attribute, its “Name.” An object of the class Job can contain several objects of the class Step. Objects of this class can refer to objects of the same class, indicating that the referred step is necessary. Objects of this class can also have several references to Tool_framework objects to indicate the tools necessary for that step. Jobs can be asynchronous if defined with the supertype Parallel_job.

Objects of the class **Step** represent several commands used for a more specific goal than the Job class. This class has one attribute, “Name.” It contains several Command objects. It can reference objects of its class to indicate dependencies. It can also reference artifacts. Dependencies of one step transfer to the following Steps if the jobs in which they are contained are not of the supertype Parallel_job.

The class **Artifact** represents software artifacts such as code from a repository. It has one attribute, “Name.”

The class **Tool_framework** represents the tools and frameworks, such as command line tools and operative systems.

The class **Command** represents commands and their arguments. It possesses the attributes “Description,” containing an optional command description, and “Parameter” for the command parameters. It references one “Tool_framework,” which is the tool used in the command. It also references “Parameters” that function and environment variables.

The **Parameter** class represents environment variables.

Example

We used a GitHub actions pipeline written in yaml to test our metamodel. The yaml file specifies that it requires ubuntu's latest version and Node.js version [10. x, 8. x]. It then specifies several jobs involving the checkout, installation, testing, and building of the source code. It then specifies the steps necessary for deploying the code to Heroku.

In this model, we created a Pipeline object referencing two Tool_frameworks representing the Node.js environment and the ubuntu operative system. We created three jobs. The jobs are, Install, Test, Build, and Deploy.

We modeled the source code as an Artifact referenced in the Install Job's first Step. This Step contains a command that uses the Yarn Tool_framework.

In the Job Test, we created one Step and one Command that uses the Yarn Tool_framework to do the testing.

The Job Build is similar to the Job test, but the Command used the Yarn Tool_framework with different arguments for building the code.

The job Deploy contains one Step, and this Step contains two Jobs. One of those Jobs uses the GIt Tool_framework to deploy to add the remote Heroku repository. This Command takes two Parameters representing the Heroku key and app name. Finally, we create a command using the Git Tool_framework to deploy the application to the remote Heroku repository.