Lab 9: Module Dependency and Cohesion Analysis

Pathan Mohammad Rashid (22110187)

Introduction, Setup, and Tools

In this lab, I analyzed software dependencies and class cohesion using **pydeps** for a Python project and **LCOM** metrics for a Java project. My goals were to generate and interpret module dependency graphs, compute fan-in/fan-out, detect design flaws, and measure class cohesion to suggest refactoring strategies.

I selected the **Requests** library (a simple, yet elegant HTTP library) for Python dependency analysis, and the **Google Guava** library (core Java utilities) for cohesion analysis.

Environment and Tools:

- Operating System: SET-IITGN-VM (in Oracle VM VirtualBox)
- Python Version: 3.9.7
- Java Version: 11.0.16
- **pydeps** v1.3.6 (Python module dependency visualization)
- **LCOM.jar** v1.0 (LCOM1..5 & YALCOM analysis tool)

Setup & Installation:

```
# Update package lists
sudo apt update

# Install Python 3.8+, pip, and Graphviz
sudo apt install -y python3 python3-pip graphviz          # Graphviz
provides `dot` on PATH

# Install Java 11 (or later) and Maven
sudo apt install -y openjdk-11-jdk maven

pip3 install pydeps                              # installs the pydeps CLI
```

Methodology and Execution

1. Python Dependency Analysis with pydeps

Project Setup

I cloned the Requests repository from GitHub:

```
git clone https://github.com/psf/requests.git
cd requests
```

Listing 1: Cloning Requests Repository

Generating Dependency Data

I ran pydeps with the --show-deps option to output JSON for computing fan-in and fan-out:

```
pydeps . --show-deps --deps-output deps.json
```

Listing 2: Running pydeps with JSON output

Dependency Metrics

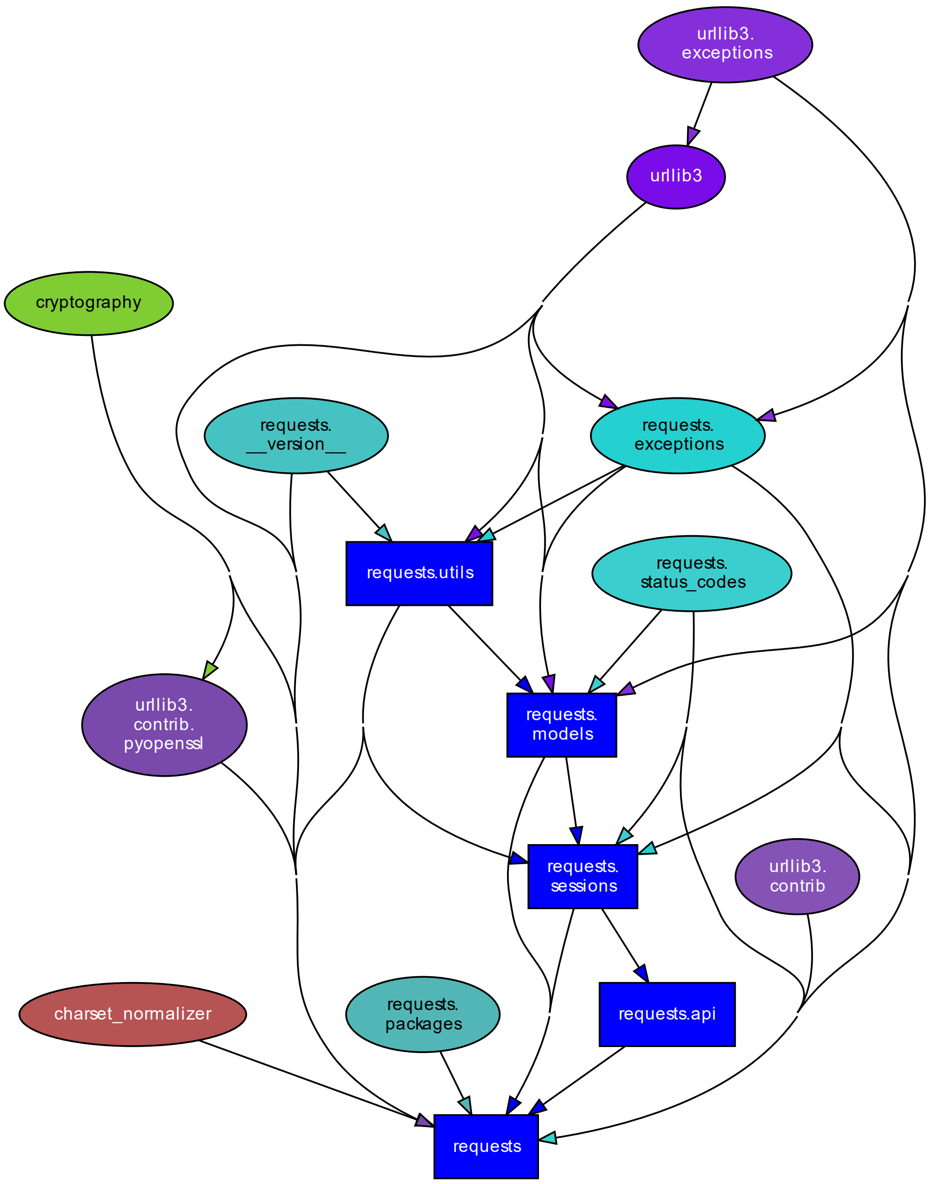

Table 1: Module Fan-In and Fan-Out (pydeps output)

Module	Fan-In	Fan-Out
__main__	0	1
charset_normalizer	1	0
cryptography	3	1
requests	4	15
requests.__version__	2	0
requests.api	1	2
requests.exceptions	4	2
requests.models	2	5
requests.packages	1	0
requests.sessions	2	4
requests.status_codes	3	0
requests.utils	3	4
urllib3	6	2
urllib3.contrib	1	0
urllib3.contrib.pyopenssl	1	2
urllib3.exceptions	4	0

Analysis:

- **Highly Coupled Modules:** requests (Fan-In=4, Fan-Out=15) and urllib3 (Fan-In=6, Fan-Out=2) are central nodes in the dependency graph. requests is the most interconnected, highlighting its role as the core orchestrator.

- **Cyclic Dependencies:** None detected. The graph is acyclic, showing that the system avoids mutual dependencies that can complicate maintenance and testing.

- **Unused / Disconnected Modules:** Modules with Fan-Out = 0 (no dependencies): charset normalizer, requests._version_, requests.packages, requests.status codes, urllib3.contrib, urllib3.exceptions. These may be utility classes or legacy modules.

- **Dependency Depth:** The dependency graph exhibits a maximum depth of 2–3 levels. For example, cryptography and charset normalizer indirectly support modules like requests.utils, which in turn affect requests.models. The relatively shallow hierarchy favors ease of navigation and modular testing.

Impact Assessment

- **Core Module Risk:** The requests module, having the highest Fan-Out (15), acts as the core of the project. Any change in this module may propagate to requests.models, requests.sessions, requests.utils, requests.exceptions, and more, potentially causing widespread breakage or unexpected behavior.

- **Risky Dependencies:** urllib3 (Fan-In=6) is a critical low-level dependency. If modified without backward compatibility, it could break higher-level functionalities in multiple modules relying on its networking stack.

- **Module Isolation:** Modules with low Fan-In and Fan-Out (e.g., requests.status codes, urllib3.contrib) are relatively isolated, posing less risk in change scenarios.

2. Java Class Cohesion Analysis with LCOM

Project Setup

I cloned the Guava repository from GitHub:

```
git clone  https://github.com/google/guava.git
cd guava

git clone https://github.com/tushartushar/LCOM.git
cd LCOM
mvn clean package                              # produces target/LCOM.jar
```

Listing 3: Cloning Guava Repository

Running LCOM Analysis

I executed:

```
java -Xmx2g -jar /home/set-iitgn-vm/Desktop/Lab9/LCOM/target/LCOM.jar -i /home/set-iitgn-
    vm/Desktop/Lab9/LCOM/guava/guava/src -o lcom-output
```

Listing 4: Executing LCOM.jar on Guava

Cohesion Metrics

Java Class	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	YALCOM
`MutableClassToInstanceMap`	214.0	175.0	13.0	13.0	0.86	0.57
`SerializedForm`	0.0	0.0	1.0	1.0	1.0	0.0
`Range`	666.0	429.0	21.0	9.0	0.86	0.21
`RangeLexOrdering`	0.0	0.0	1.0	1.0	0.0	0.0
`AbstractRangeSet`	105.0	0.0	15.0	13.0	0.0	0.87
`FilteredKeySetMultimap`	36.0	0.0	9.0	9.0	0.0	0.67
`EntrySet`	1.0	0.0	2.0	2.0	0.0	1.0
`UnmodifiableListIterator`	3.0	0.0	3.0	3.0	0.0	1.0
`HashMultimapGwtSerializationDependencies`	0.0	0.0	1.0	1.0	0.0	0.0

Table 2: LCOM Metrics for Selected Guava Classes

Analysis: High LCOM values indicate low cohesion—classes like Range (LCOM1=666, YALCOM=0.209) and MutableClassToInstanceMap (LCOM1=214, YALCOM=0.565) are candidates for functional decomposition.

Results and Analysis

- **Dependency Analysis:** requests is highly coupled; no import cycles; several disconnected modules.
- **Cohesion Analysis:** Multiple Guava classes exhibit low cohesion (high LCOM), suggesting refactoring opportunities.

Discussion and Conclusion

This lab taught me the importance of managing coupling and cohesion to improve maintainability. Key challenges included parsing pydeps JSON output and interpreting large LCOM reports. I learned to prioritize low coupling to minimize change impact and to seek high cohesion by grouping related functionality. In future work, I will explore additional metrics such as cyclomatic complexity and automate refactoring suggestions.

References and Resources

- **Requests GitHub:** https://github.com/psf/requests
- **pydeps GitHub:** https://github.com/thebjorn/pydeps
- **Guava GitHub:** https://github.com/google/guava
- **LCOM GitHub:** https://github.com/tushartushar/LCOM.git

GitHub Repository and Drive Link

GitHub Repository: https://github.com/Pathan-Mohammad-Rashid/STT-Lab

Drive Link: 22110187_STT_Labs