

|| 산업 빅데이터 분석 실제 - 프로젝트 중간 발표 ||

대규모 그리드 컴퓨팅 미들웨어의 코드 품질 예측 모델

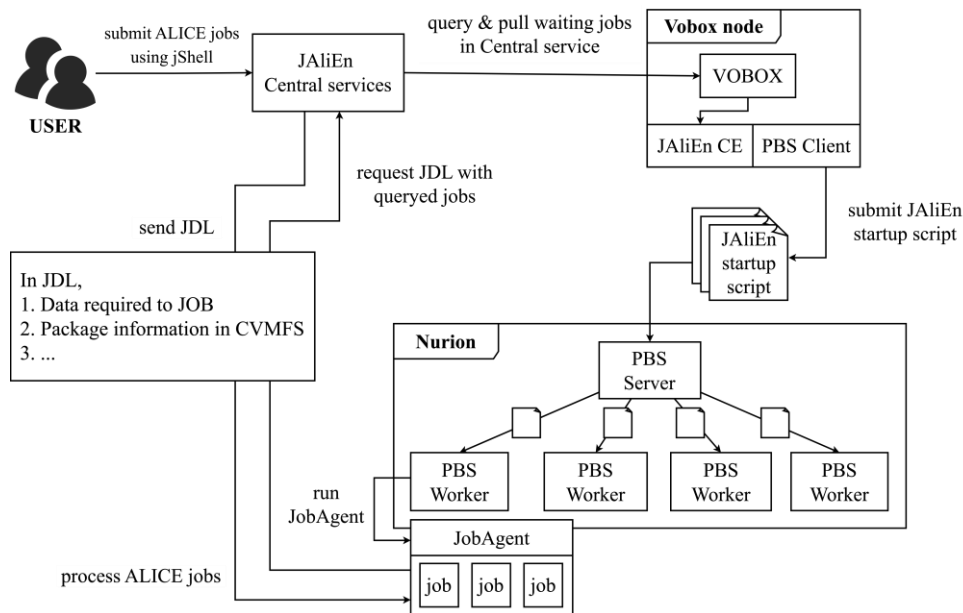
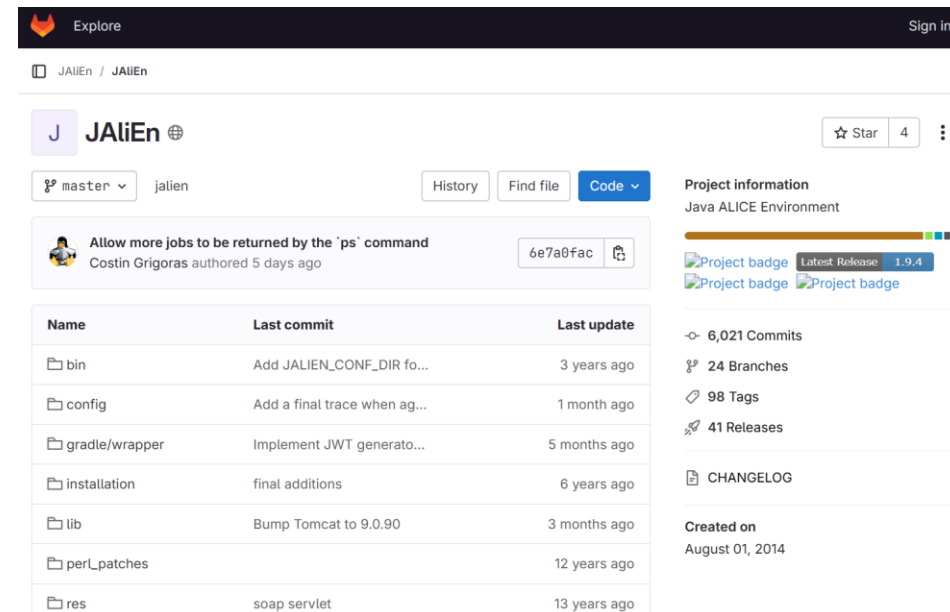
컴퓨터과학과 유현진,
2023299008,
통합과정 4학기

2024.10.28(Mon)

❖ 현재 그리드 미들웨어(CERN JAliEn) 현황

- 코드 규모가 크고, 여러 개발자가 참여하여 개발 및 유지보수 중 (58438 lines, 654 classes)
- 코드 규모와 오픈소스라는 특징에 반해 소프트웨어 안정성이 검증되지 않음
- 현재는 프로덕션 환경에서 구동해보며 정상적으로 동작되는지만 확인

⇒ **미들웨어의 품질 예측을 통해, 안정성과 유지보수 효율성을 증대시키는 것이 목표**

The screenshot shows the GitHub repository for JAliEn. The repository is named **JAliEn** and is described as **Java ALICE Environment**. It has 4 stars and 4 forks. The latest release is **1.9.4**. The repository contains 6,021 commits, 24 branches, 98 tags, and 41 releases. The **CHANGELOG** is visible. The repository was created on August 01, 2014.

Name	Last commit	Last update
bin	Add JALIEN_CONF_DIR fo...	3 years ago
config	Add a final trace when ag...	1 month ago
gradle/wrapper	Implement JWT generato...	5 months ago
installation	final additions	6 years ago
lib	Bump Tomcat to 9.0.90	3 months ago
perl_patches		12 years ago
res	soap servlet	13 years ago

❖ "JAliEn의 SpotBugs 분석 결과 데이터"

[데이터 출처]

- SpotBugs라는 정적 코드 분석 도구를 사용해 생성된 XML 형식의 파일
 - SpotBugs : Java 프로그램의 잠재적인 버그와 코드 문제를 자동으로 검출하는 도구로, 소프트웨어 품질 향상에 사용



[데이터 내용]

- Java 코드 파일의 메타데이터
- 발견된 다양한 유형의 버그와 그 위치에 대한 정보

Bad practice Warnings

CodeWarning

CN alien.io.protocols.Xrd3cpGW.clone() does not call super.clone()

[Bug type CN_IDIOM_NO_SUPER_CALL \(click for details\)](#)

In class alien.io.protocols.Xrd3cpGW

In method alien.io.protocols.Xrd3cpGW.clone()

At Xrd3cpGW.java:[lines 261-265]



```
<BugInstance type="DCN_NULL_POINTER_EXCEPTION" priority="2" rank="17" abbrev="DCN" category="STYLE"
instanceHash="2614e468d8b29b92e785bd6b819669a9" instanceOccurrenceNum="0" instanceOccurrenceMax="0">
  <ShortMessage>NullPointerException caught</ShortMessage>
  <LongMessage>Do not catch NullPointerException like in alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)
</LongMessage>
  <Class classname="alien.ArchiveMemberDelete" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="42" end="692" sourcefile="ArchiveMemberDelete.java"
sourcepath="alien/ArchiveMemberDelete.java" relSourcepath="java/alien/ArchiveMemberDelete.java">
      <Message>At ArchiveMemberDelete.java:[lines 42-692]</Message>
    </SourceLine>
  </Class>
  <Message>In class alien.ArchiveMemberDelete</Message>
  <Method classname="alien.ArchiveMemberDelete" name="deleteArchiveMember" signature="(Ljava/lang/String;Z)V"
isStatic="true" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="104" end="343" startBytecode="0" endBytecode="3828"
sourcefile="ArchiveMemberDelete.java" sourcepath="alien/ArchiveMemberDelete.java"
relSourcepath="java/alien/ArchiveMemberDelete.java">
      <Message>In method alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)</Message>
    </SourceLine>
  </Method>
  <SourceLine classname="alien.ArchiveMemberDelete" primary="true" start=
endBytecode="97" sourcefile="ArchiveMemberDelete.java" sourcepath="ali
relSourcepath="java/alien/ArchiveMemberDelete.java">
    <Message>At ArchiveMemberDelete.java:[line 122]</Message>
  </SourceLine>
</BugInstance>
```

XML 파일 ⇒ CSV 파일
(변환 예정)

❖ XML 형식이므로, 태그별로 정리

1. 최상위 태그 - **BugCollection**

- **BugCollection** 은 파일의 최상위 태그로, 전체 분석의 메타데이터를 포함
 - Ex. SpotBugs의 버전, 타임스탬프, 분석 대상 프로젝트 이름

```
<BugCollection version="4.8.6" sequence="0" timestamp="1729580045580"
analysisTimestamp="1729580045580" release="">
```

2. 프로젝트 정보 - **Project**

- 분석 대상 프로젝트의 이름과 함께, 분석에 포함된 개별 클래스 파일 나열
- **Jar** 태그는 분석에 포함된 각 파일의 경로 및 분석한 클래스 나열

3. 버그 인스턴스 - **BugInstance**

- 코드에서 발견된 각 버그에 대한 세부 정보
 - Ex. **type** (버그 유형), **priority** (우선순위), **rank** (심각도 수준)
- 버그의 구체적 위치는 **SourceLine** 태그에서 확인 가능

4. 버그 패턴 - **BugPattern**

- SpotBugs에서 정의한 특정 버그 유형의 설명 및 고유 식별자 보유

5. 클래스 및 메소드 정보 - **Class**, **Method**

- **Class** 태그: 버그가 발견된 클래스에 대한 정보, 해당 클래스 내 **Method** 태그는 구체적으로 버그가 발생한 메소드 표시
- **Method** 태그: 메소드 이름, 접근 제한자(예: public, private), 리턴 타입 등 포함

6. 소스 코드 라인 정보 - **SourceLine**

- 버그가 발견된 구체적인 코드 라인을 표시 및 파일의 위치, 라인 번호 등의 정보 제공

❖ XML 형식이므로, 태그별로 정리

1. 최상위 태그 - **BugCollection**

- **BugCollection** 은 파일의 최상위 태그로, 전체 분석의 메타데이터를 포함
 - Ex. SpotBugs의 버전, 타임스탬프, 분석 대상 프로젝트 이름

2. 프로젝트 정보 - **Project**

- 분석 대상 프로젝트의 이름과 함께, 분석에 포함된 개별 클래스 파일 나열
- **Jar** 태그는 분석에 포함된 각 파일의 경로 및 분석한 클래스 나열

3. 버그 인스턴스 - **BugInstance**

- 코드에서 발견된 각 버그에 대한 세부 정보
 - Ex. **type** (버그 유형), **priority** (우선순위), **rank** (심각도 수준)
- 버그의 구체적 위치는 **SourceLine** 태그에서 확인 가능

4. 버그 패턴 - **BugPattern**

- SpotBugs에서 정의한 특정 버그 유형의 설명 및 고유 식별자 보유

5. 클래스 및 메소드 정보 - **Class**, **Method**

- **Class** 태그: 버그가 발견된 클래스에 대한 정보, 해당 클래스 내 **Method** 태그는 구체적으로 버그가 발생한 메소드 표시
- **Method** 태그: 메소드 이름, 접근 제한자(예: public, private), 리턴 타입 등 포함

6. 소스 코드 라인 정보 - **SourceLine**

- 버그가 발견된 구체적인 코드 라인을 표시 및 파일의 위치, 라인 번호 등의 정보 제공

```
<BugInstance type="DCN_NULLPOINTER_EXCEPTION" priority="2" rank="17" abbrev="DCN" category="STYLE"
instanceHash="2614e468d8b29b92e785bd6b819669a9" instanceOccurrenceNum="0" instanceOccurrenceMax="0">
  <ShortMessage>NullPointerException caught</ShortMessage>
  <LongMessage>Do not catch NullPointerException like in
  alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)</LongMessage>
  <Class classname="alien.ArchiveMemberDelete" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="42" end="692"
    sourcefile="ArchiveMemberDelete.java" sourcepath="alien/ArchiveMemberDelete.java"
    relSourcepath="java/alien/ArchiveMemberDelete.java">
      <Message>At ArchiveMemberDelete.java:[lines 42-692]</Message>
    </SourceLine>
    <Message>In class alien.ArchiveMemberDelete</Message>
  </Class>
  <Method classname="alien.ArchiveMemberDelete" name="deleteArchiveMember" signature="
  (Ljava/lang/String;Z)V" isStatic="true" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="104" end="343" startBytecode="0"
    endBytecode="3828" sourcefile="ArchiveMemberDelete.java"
    sourcepath="alien/ArchiveMemberDelete.java"
    relSourcepath="java/alien/ArchiveMemberDelete.java"/>
    <Message>In method alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)</Message>
  </Method>
  <SourceLine classname="alien.ArchiveMemberDelete" primary="true" start="122" end="122"
  startBytecode="97" endBytecode="97" sourcefile="ArchiveMemberDelete.java"
  sourcepath="alien/ArchiveMemberDelete.java" relSourcepath="java/alien/ArchiveMemberDelete.java">
    <Message>At ArchiveMemberDelete.java:[line 122]</Message>
  </SourceLine>
</BugInstance>
```

❖ XML 형식이므로, 태그별로 정리

1. 최상위 태그 - **BugCollection**

- **BugCollection** 은 파일의 최상위 태그로, 전체 분석의 메타데이터를 포함
 - Ex. SpotBugs의 버전, 타임스탬프, 분석 대상 프로젝트 이름

2. 프로젝트 정보 - **Project**

- 분석 대상 프로젝트의 이름과 함께, 분석에 포함된 개별 클래스 파일 나열
- **Jar** 태그는 분석에 포함된 각 파일의 경로 및 분석한 클래스 나열

3. 버그 인스턴스 - **BugInstance**

- 코드에서 발견된 각 버그에 대한 세부 정보
 - Ex. **type** (버그 유형), **priority** (우선순위), **rank** (심각도 수준)
- 버그의 구체적 위치는 **SourceLine** 태그에서 확인 가능

4. 버그 패턴 - **BugPattern**

- SpotBugs에서 정의한 특정 버그 유형의 설명 및 고유 식별자 보유

5. 클래스 및 메소드 정보 - **Class**, **Method**

- **Class** 태그: 버그가 발견된 클래스에 대한 정보, 해당 클래스 내 **Method** 태그는 체적으로 버그가 발생한 메소드 표시
- **Method** 태그: 메소드 이름, 접근 제한자(예: public, private), 리턴 타입 등 포함

6. 소스 코드 라인 정보 - **SourceLine**

- 버그가 발견된 구체적인 코드 라인을 표시 및 파일의 위치, 라인 번호 등의 정보 제공

```
<BugPattern type="CN_IDIOM_NO_SUPER_CALL" abbrev="CN" category="BAD_PRACTICE">
  <ShortDescription>clone method does not call super.clone()</ShortDescription>
  <Details>
    <![CDATA[ <p> This non-final class defines a clone() method that does not call super.clone(). If
    this class ("<i>A</i>") is extended by a subclass ("<i>B</i>"), and the subclass <i>B</i> calls
    super.clone(), then it is likely that <i>B</i>'s clone() method will return an object of type
    <i>A</i>, which violates the standard contract for clone().</p> <p> If all clone() methods call
    super.clone(), then they are guaranteed to use Object.clone(), which always returns an object of
    the correct type.</p> ]]>
  </Details>
</BugPattern>
```

❖ XML 형식이므로, 태그별로 정리

1. 최상위 태그 - **BugCollection**

- **BugCollection** 은 파일의 최상위 태그로, 전체 분석의 메타데이터를 포함
 - Ex. SpotBugs의 버전, 타임스탬프, 분석 대상 프로젝트 이름

2. 프로젝트 정보 - **Project**

- 분석 대상 프로젝트의 이름과 함께, 분석에 포함된 개별 클래스 파일 나열
- **Jar** 태그는 분석에 포함된 각 파일의 경로 및 분석한 클래스 나열

3. 버그 인스턴스 - **BugInstance**

- 코드에서 발견된 버그에 대한 세부 정보
 - Ex. **type** (버그 유형), **priority** (우선순위), **rank** (심각도 수준)
- 버그의 구체적 위치는 **SourceLine** 태그에서 확인 가능

4. 버그 패턴 - **BugPattern**

- SpotBugs에서 정의한 특정 버그 유형의 설명 및 고유 식별자 보유

5. 클래스 및 메소드 정보 - **Class**, **Method**

- **Class** 태그: 버그가 발견된 클래스에 대한 정보, 해당 클래스 내 **Method** 태그는 구체적으로 버그가 발생한 메소드 표시
- **Method** 태그: 메소드 이름, 접근 제한자(예: public, private), 리턴 타입 등 포함

6. 소스 코드 라인 정보 - **SourceLine**

- 버그가 발견된 구체적인 코드 라인을 표시 및 파일의 위치, 라인 번호 등의 정보 제공

```
<BugInstance type="DCN_NULLPOINTER_EXCEPTION" priority="2" rank="17" abbrev="DCN" category="STYLE"
instanceHash="2614e468d8b29b92e785bd6b819669a9" instanceOccurrenceNum="0" instanceOccurrenceMax="0">
  <ShortMessage>NullPointerException caught</ShortMessage>
  <LongMessage>Do not catch NullPointerException like in
  alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)</LongMessage>
  <Class classname="alien.ArchiveMemberDelete" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="42" end="692"
    sourcefile="ArchiveMemberDelete.java" sourcepath="alien/ArchiveMemberDelete.java"
    relSourcepath="java/alien/ArchiveMemberDelete.java">
      <Message>At ArchiveMemberDelete.java:[lines 42-692]</Message>
    </SourceLine>
    <Message>In class alien.ArchiveMemberDelete</Message>
  </Class>
  <Method classname="alien.ArchiveMemberDelete" name="deleteArchiveMember" signature="
  (Ljava/lang/String;Z)V" isStatic="true" primary="true">
    <SourceLine classname="alien.ArchiveMemberDelete" start="104" end="343" startBytecode="0"
    endBytecode="3828" sourcefile="ArchiveMemberDelete.java"
    sourcepath="alien/ArchiveMemberDelete.java"
    relSourcepath="java/alien/ArchiveMemberDelete.java"/>
    <Message>In method alien.ArchiveMemberDelete.deleteArchiveMember(String, boolean)</Message>
  </Method>
  <SourceLine classname="alien.ArchiveMemberDelete" primary="true" start="122" end="122"
  startBytecode="97" endBytecode="97" sourcefile="ArchiveMemberDelete.java"
  sourcepath="alien/ArchiveMemberDelete.java" relSourcepath="java/alien/ArchiveMemberDelete.java">
    <Message>At ArchiveMemberDelete.java:[line 122]</Message>
  </SourceLine>
</BugInstance>
```


❖ “버그 패턴 유형 분석 + 코드 품질 예측”을 위해 SpotBugs 데이터로부터 분석해야 할 주요 항목

- **버그 패턴 유형 분석**

- 버그 유형 (BugInstance 태그의 type)
- 버그 심각도 (priority, rank)
- 버그 발생 위치 (SourceLine 태그)
- 버그 패턴 설명 (BugPattern 태그)

- **코드 품질 예측**

- 클래스 수 및 메소드 수 (Class, Method 태그)
- 메소드 길이
- 코드 복잡도 측정



" 버그 발생 빈도 & 코드 메타데이터의 상관관계 분석 "

버그가 자주 발생하는 코드의 특징을 파악하여,
복잡도나 코드 길이 등이 버그 발생에 미치는 영향 분석

❖ 크게 3가지 효과를 기대할 수 있음

[공통]

"유지보수 효율성 증대"

대규모 코드베이스와 복잡한 아키텍처로 이루어져 있음
품질 저하 영역을 조기 발견 시 유지보수 비용을 크게 감축 가능

⇒ 이는 그리드 컴퓨팅 인프라 운영 비용 절감 및 효율성을 높이는 데 기여

[유사 버그 패턴 분석]

"버그 예방 및 소프트웨어 안정성 향상"

유사한 패턴의 버그가 반복적으로 발생 시,
특정 코딩 스타일이나 구조적 문제가 있을 가능성이 높음

⇒ 이러한 버그 패턴 파악은 비슷한 유형의 오류를 방지

[코드 품질 예측 모델 개발]

"안정성 향상"

그리드 컴퓨팅은 다수의 클러스터에서 분산 작업을 수행
일부 오류가 전체 그리드 환경의 성능과 안정성에 큰 영향

⇒ 취약한 코드 영역을 사전에 식별하고 개선함으로써,
전반적인 시스템 안정성 강화 기대