Characterizing and Mitigating Self-Admitted Build Debt

I am Tao Xiao, a Master's student, and on behalf of the Software Engineering Team in NAIST, Japan and the University of Waterloo, Canada, we thank you for taking out the time to assist us with your feedback.

We are inviting you to participate because we found that your contributed repositories have added or removed comments that we suspect are related to Self-Admitted Technical Debt (SATD).

Our research aims to analyze build files and their existing SATD, which we refer to as Self-Admitted Build Debt (SABD). More specifically, we set out to characterize SABD, explore its potential for automation, and evaluate SABD mitigation strategies. In this survey, we are soliciting feedback from the developer community on the reasons and purposes for SABD that we identified in our work. Our paper is available at: https://arxiv.org/pdf/2102.09775.pdf

Our survey has a total of ten SABD cases expected to be labeled, each case taking no longer than one minute to complete. Please feel free to skip some cases.

____

Tao Xiao on Behalf of our Team
Nara Institute of Science and Technology, Japan
tao.xiao.ts2@is.naist.jp

* Required

[About Technical Debt] Throughout the software development process, stakeholders strive to build functional, maintainable, and high-quality software. Despite their best efforts, developers inevitably encounter situations where suboptimal solutions, known as technical debt are implemented in a software project [1].

[1] Cunningham, W. (1992). The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger, 4(2), 29-30.

1. You should only complete this survey if you meet all of the conditions below. *

Check all that apply.

☐ I have read the consent form: https://drive.google.com/file/d/1waRTJXUEu_iihNVe8le0_BIczwwLgLCn/view?usp=sharing

☐ I agree to participate.

About You

2. Gender *

Mark only one oval.

- Female
- Male
- Prefer not to say
- Other: ______________________

3. Years of programming experience *

Mark only one oval.

- less than 5 years
- 5-9 years
- 10-14 years
- 15-19 years
- 20-24 years
- 25-29 years
- 30 years and above
- Prefer not to say

4. Years of experience writing/editing Maven specifications *

Mark only one oval.

- less than 3 years
- 3-5 years
- 6-9 years
- 10 years and above
- Prefer not to say

5. Which GitHub projects you participate in? *

Check all that apply.

- ☐ exoplatform/exogtn
- ☐ rhuss/jolokia
- ☐ GoogleCloudPlatform/cloud-bigtable-examples
- ☐ OpenRock/OpenAM
- ☐ sarl/sarl
- ☐ jboss-integration/fuse-bxms-integ
- ☐ RWTH-i5-IDSG/steve
- ☐ svn2github/cytoscape
- ☐ None of the above

6. Participation ID (only for deleting your answers on your request)

____

Case 1

Please look at the following example.

In the following snippet of Maven code, a comment describes a constraint imposed by the version of the maven-war-plugin (working with the 2.0.1 version, but not the 2.0 version).

```
<build>
  <finalName>browser</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <configuration>
        <!-- This is broken in maven-war-plugin 2.0, works in 2.0.1 -->
        <warSourceExcludes>WEB-INF/no-lib/*.jar</warSourceExcludes>
        <archive>
          <manifest>
            <addClasspath>false</addClasspath>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

7. Could you please answer some questions about this case?

Mark only one oval.

- ◯ Answer this case
- ◯ Skip this case *Skip to question 13*

Questions to case 1

Please answer the questions.

In the following snippet of Maven code, a comment describes a constraint imposed by the version of the maven-war-plugin (working with the 2.0.1 version, but not the 2.0 version).

```xml
<build>
  <finalName>browser</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <configuration>
        <!-- This is broken in maven-war-plugin 2.0, works in 2.0.1 -->
        <warSourceExcludes>WEB-INF/no-lib/*.jar</warSourceExcludes>
        <archive>
          <manifest>
            <addClasspath>false</addClasspath>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

8. Is this an example of technical debt in the build system?

   Mark only one oval.

   - Yes
   - No
   - Other: ____________________

9. Which of the following reason do you think this constraint occur?

Mark only one oval.

- ◯ Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- ◯ Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- ◯ Recursive call - Coherence issues, recursive calls to invoke another build file.
- ◯ Document - Inadequate project description issues, such as licensing and metadata specification.
- ◯ Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- ◯ Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- ◯ Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- ◯ Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- ◯ None of the above
- ◯ Other: ______________________

10. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- ◯ Document for later fix - Document an issue that should be revisited in the future.
- ◯ Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- ◯ Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- ◯ Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- ◯ Placeholder for later extension - Document an extension point for later enhancement(s).
- ◯ Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- ◯ None of the above
- ◯ Other: ____________________

11. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: ____________________

12. Any comments (optional)

____

____

____

____

____

Case 2

Please look at the following example.

In the following snippet of Maven code, a comment describes a constraint imposed to resolve dependency issue that commons-logging:commons-logging dependency is required by org.apache.httpcomponents:httpclient-osgi dependency.

```
<!-- Dependency required by httpclient-osgi via transitive dependencies which need to include since
     httpclient-osgi is a fat-jar. In fact this is a workaround for httpclient-osgi's borked pom.xml -->
<dependency>
  <groupId>commons-logging</groupId>
  <artifactId>commons-logging</artifactId>
  <version>1.1.1</version>
  <scope>test</scope>
</dependency>
```

13. Could you please answer some questions about this case?

Mark only one oval.

◯ Answer this case

◯ Skip this case *Skip to question 19*

Questions to case 2

Please answer the questions.

In the following snippet of Maven code, a comment describes a constraint imposed to resolve dependency issue that commons-logging:commons-logging dependency is required by org.apache.httpcomponents:httpclient-osgi dependency.

```
<!-- Dependency required by httpclient-osgi via transitive dependencies which need to include since
     httpclient-osgi is a fat-jar. In fact this is a workaround for httpclient-osgi's borked pom.xml -->
<dependency>
  <groupId>commons-logging</groupId>
  <artifactId>commons-logging</artifactId>
  <version>1.1.1</version>
  <scope>test</scope>
</dependency>
```

14. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

15. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ____________________

16. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- ◯ Document for later fix - Document an issue that should be revisited in the future.
- ◯ Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- ◯ Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- ◯ Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- ◯ Placeholder for later extension - Document an extension point for later enhancement(s).
- ◯ Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- ◯ None of the above
- ◯ Other: ______________________

17. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: ______________________

18. Any comments (optional)

____

____

____

____

____

Case 3

Please look at the following example.

In the following snippet of Maven code, a comment describes an option that uncomments parent tag to call the common plugins and properties in com.google.cloud:bigtable-samples Parent POM when it passes check styles.

```
<!-- Parent POM defines common plugins and properties.
     TODO: use the parent when this sample passes checkstyles.
     See: https://github.com/GoogleCloudPlatform/cloud-bigtable-examples/issues/59
<parent>
  <groupId>com.google.cloud</groupId>
  <artifactId>bigtable-samples</artifactId>
  <version>1.0.0</version>
  <relativePath>..</relativePath>
</parent> -->
```

19. Could you please answer some questions about this case?

Mark only one oval.

◯ Answer this case

◯ Skip this case *Skip to question 25*

Questions to case 3

Please answer the questions.

In the following snippet of Maven code, a comment describes an option that uncomments parent tag to call the common plugins and properties in com.google.cloud:bigtable-samples Parent POM when it passes check styles.

```
<!-- Parent POM defines common plugins and properties.
     TODO: use the parent when this sample passes checkstyles.
     See: https://github.com/GoogleCloudPlatform/cloud-bigtable-examples/issues/59
<parent>
  <groupId>com.google.cloud</groupId>
  <artifactId>bigtable-samples</artifactId>
  <version>1.0.0</version>
  <relativePath>..</relativePath>
</parent> -->
```

20. Is this an example of technical debt in the build system?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: ____________________

21. Which of the following reason do you think this constraint occur?

Mark only one oval.

- ⬭ Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- ⬭ Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- ⬭ Recursive call - Coherence issues, recursive calls to invoke another build file.
- ⬭ Document - Inadequate project description issues, such as licensing and metadata specification.
- ⬭ Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- ⬭ Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- ⬭ Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- ⬭ Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- ⬭ None of the above
- ⬭ Other: ____________________

22. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- ◯ Document for later fix - Document an issue that should be revisited in the future.
- ◯ Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- ◯ Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- ◯ Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- ◯ Placeholder for later extension - Document an extension point for later enhancement(s).
- ◯ Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- ◯ None of the above
- ◯ Other: ____________________

23. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: ____________________

24. Any comments (optional)

____

____

____

____

____

Case 4

Please look at the following example.

In the following snippet of Maven code, a comment describes that the license information needs to be added to Manifest.

```
<manifestEntries>
    <Specification-Title>OpenAM Auth Scripted Device Print</Specification-Title>
    <Specification-Version>${project.version} - ${maven.build.timestamp}</Specification-Version>
    <Specification-Vendor>ForgeRock</Specification-Vendor>
    <Implementation-Title>OpenAM Auth Scripted</Implementation-Title>
    <Implementation-Version>${project.version} - ${maven.build.timestamp}
    </Implementation-Version>
    <Implementation-Vendor>ForgeRock</Implementation-Vendor>
    <Built-By>${user.name}</Built-By>
    <Build-Jdk>${java.version}</Build-Jdk>
    <Build-Time>${maven.build.timestamp}</Build-Time>
    <Version>${project.version}</Version>
    <Create-On>${maven.build.timestamp}</Create-On>
    <Revision>${git.short.sha1}</Revision>
    <!-- TODO Add License to Manifest -->
    <License />
</manifestEntries>
```

25. Could you please answer some questions about this case?

Mark only one oval.

- ◯ Answer this case
- ◯ Skip this case *Skip to question 31*

Questions to case 4

Please answer the questions.

In the following snippet of Maven code, a comment describes that the license information needs to be added to Manifest.

```xml
<manifestEntries>
    <Specification-Title>OpenAM Auth Scripted Device Print</Specification-Title>
    <Specification-Version>${project.version} - ${maven.build.timestamp}</Specification-Version>
    <Specification-Vendor>ForgeRock</Specification-Vendor>
    <Implementation-Title>OpenAM Auth Scripted</Implementation-Title>
    <Implementation-Version>${project.version} - ${maven.build.timestamp}
    </Implementation-Version>
    <Implementation-Vendor>ForgeRock</Implementation-Vendor>
    <Built-By>${user.name}</Built-By>
    <Build-Jdk>${java.version}</Build-Jdk>
    <Build-Time>${maven.build.timestamp}</Build-Time>
    <Version>${project.version}</Version>
    <Create-On>${maven.build.timestamp}</Create-On>
    <Revision>${git.short.sha1}</Revision>
    <!-- TODO Add License to Manifest -->
    <License />
</manifestEntries>
```

26. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ______________________

27. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ______________________

28. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- Document for later fix - Document an issue that should be revisited in the future.
- Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- Placeholder for later extension - Document an extension point for later enhancement(s).
- Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- None of the above
- Other: ____________________

29. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- Yes
- No
- Other: ____________________

30. Any comments (optional)

Case 5

Please look at the following example.

In the following snippet of Maven code, a comment describes that the reason for the extra classpath element.

```
<!-- FIXME: This extra classpath element is defined for fixing a bug into the Maven's Tycho compiler
         that avoids to load the SWT bundle fragment into the classpath. It causes a class not found
         exception for a SWT widget. -->
<plugin>
        <groupId>org.eclipse.tycho</groupId>
        <artifactId>tycho-compiler-plugin</artifactId>
        <version>${tycho.version}</version>
        <configuration>
                <extraClasspathElements>
                        <extraClasspathElement>
                                <groupId>org.eclipse.platform</groupId>
                                <artifactId>org.eclipse.swt.gtk.linux.x86_64</artifactId>
                                <version>${swt.version}</version>
                        </extraClasspathElement>
                </extraClasspathElements>
        </configuration>
</plugin>
```

31. Could you please answer some questions about this case?

Mark only one oval.

◯ Answer this case

◯ Skip this case *Skip to question 37*

Questions to case 5

Please answer the questions.

In the following snippet of Maven code, a comment describes that the reason for the extra classpath element.

```
<!-- FIXME: This extra classpath element is defined for fixing a bug into the Maven's Tycho compiler
         that avoids to load the SWT bundle fragment into the classpath. It causes a class not found
         exception for a SWT widget. -->
<plugin>
        <groupId>org.eclipse.tycho</groupId>
        <artifactId>tycho-compiler-plugin</artifactId>
        <version>${tycho.version}</version>
        <configuration>
                <extraClasspathElements>
                        <extraClasspathElement>
                                <groupId>org.eclipse.platform</groupId>
                                <artifactId>org.eclipse.swt.gtk.linux.x86_64</artifactId>
                                <version>${swt.version}</version>
                        </extraClasspathElement>
                </extraClasspathElements>
        </configuration>
</plugin>
```

32. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

33. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ______________________________

34. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- Document for later fix - Document an issue that should be revisited in the future.
- Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- Placeholder for later extension - Document an extension point for later enhancement(s).
- Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- None of the above
- Other: ____________________

35. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- Yes
- No
- Other: ____________________

36. Any comments (optional)

Case 6

Please look at the following example.

In the following snippet of Maven code, a comment describes that there is a workaround for kie-ci-osgi.

```
<!-- Temporary workaround. This should be removed once the Karaf is properly configured with
    the remote repos, so that it can download the kie-ci-osgi itself (instead of relying on the outer Maven
    test build). -->
<dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-ci-osgi</artifactId>
    <scope>test</scope>
</dependency>
```

37. Could you please answer some questions about this case?

Mark only one oval.

- ◯ Answer this case
- ◯ Skip this case *Skip to question 43*

Questions to case 6

Please answer the questions.

In the following snippet of Maven code, a comment describes that there is a workaround for kie-ci-osgi.

```
<!-- Temporary workaround. This should be removed once the Karaf is properly configured with
    the remote repos, so that it can download the kie-ci-osgi itself (instead of relying on the outer Maven
    test build). -->
<dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-ci-osgi</artifactId>
    <scope>test</scope>
</dependency>
```

38. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

39. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ____________________

40. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- Document for later fix - Document an issue that should be revisited in the future.
- Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- Placeholder for later extension - Document an extension point for later enhancement(s).
- Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- None of the above
- Other: ______________________

41. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- Yes
- No
- Other: ______________________

42. Any comments (optional)

Case 7

Please look at the following example.

In the following snippet of Maven code, a comment describes a future activity.

```
<!-- FIXME: Remove asap if the project compile without -->
<dependency>
        <groupId>org.eclipse.platform</groupId>
        <artifactId>org.eclipse.core.runtime</artifactId>
        <version>3.12.0</version>
</dependency>
```

43. Could you please answer some questions about this case?

Mark only one oval.

- ◯ Answer this case
- ◯ Skip this case *Skip to question 49*

Questions to case 7

Please answer the questions.

In the following snippet of Maven code, a comment describes a future activity.

```
<!-- FIXME: Remove asap if the project compile without -->
<dependency>
        <groupId>org.eclipse.platform</groupId>
        <artifactId>org.eclipse.core.runtime</artifactId>
        <version>3.12.0</version>
</dependency>
```

44. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

45. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ____________________

46. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- Document for later fix - Document an issue that should be revisited in the future.
- Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- Placeholder for later extension - Document an extension point for later enhancement(s).
- Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- None of the above
- Other: ______________________

47. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- Yes
- No
- Other: ______________________

48. Any comments (optional)

Case 8

Please look at the following example.

In the following snippet of Maven code, a comment warns the reason for this constraint.

```
<plugin>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-maven-plugin</artifactId>
    <version>5.2.4</version>

    <!-- Must be in the same phase as Jooq -->
    <executions>
        <execution>
            <phase>generate-sources</phase>
            <goals>
                <goal>migrate</goal>
            </goals>
        </execution>
    </executions>

    <configuration>
        <!-- we need this because of extensions. they have a higher version number
             namespace reserved (e.g. 8.x.x). new migrations in core version should
             be executed in the extended version as well -->
        <outOfOrder>true</outOfOrder>

        <!-- Because maven produces this warning after upgrading from 4.2.0 to 5.1.0:
             [WARNING] Could not find schema history table `stevedb`.`flyway_schema_history`, but found
             `stevedb`.`schema_version` instead. You are seeing this message because Flyway changed its
             default for flyway.table in version 5.0.0 to flyway_schema_history and you are still relying
             on the old default (schema_version). Set flyway.table=schema_version in your configuration to
             fix this. This fallback mechanism will be removed in Flyway 6.0.0. -->
        <table>schema_version</table>

        <cleanDisabled>true</cleanDisabled>
        <driver>com.mysql.cj.jdbc.Driver</driver>
        <url>${jdbcUrl}</url>
        <user>${db.user}</user>
        <password>${db.password}</password>
        <schemas>
            <schema>${db.schema}</schema>
        </schemas>
        <locations>
            <location>filesystem:src/main/resources/db/migration</location>
        </locations>
    </configuration>
</plugin>
```

49. Could you please answer some questions about this case?

Mark only one oval.

- Answer this case
- Skip this case *Skip to question 55*

Questions to case 8

Please answer the questions.

In the following snippet of Maven code, a comment warns the reason for this constraint.

```xml
<plugin>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-maven-plugin</artifactId>
    <version>5.2.4</version>

    <!-- Must be in the same phase as Jooq -->
    <executions>
        <execution>
            <phase>generate-sources</phase>
            <goals>
                <goal>migrate</goal>
            </goals>
        </execution>
    </executions>

    <configuration>
        <!-- we need this because of extensions. they have a higher version number
             namespace reserved (e.g. 8.x.x). new migrations in core version should
             be executed in the extended version as well -->
        <outOfOrder>true</outOfOrder>

        <!-- Because maven produces this warning after upgrading from 4.2.0 to 5.1.0:
             [WARNING] Could not find schema history table `stevedb`.`flyway_schema_history`, but found
             `stevedb`.`schema_version` instead. You are seeing this message because Flyway changed its
             default for flyway.table in version 5.0.0 to flyway_schema_history and you are still relying
             on the old default (schema_version). Set flyway.table=schema_version in your configuration to
             fix this. This fallback mechanism will be removed in Flyway 6.0.0. -->
        <table>schema_version</table>

        <cleanDisabled>true</cleanDisabled>
        <driver>com.mysql.cj.jdbc.Driver</driver>
        <url>${jdbcUrl}</url>
        <user>${db.user}</user>
        <password>${db.password}</password>
        <schemas>
            <schema>${db.schema}</schema>
        </schemas>
        <locations>
            <location>filesystem:src/main/resources/db/migration</location>
        </locations>
    </configuration>
</plugin>
```

50. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

51. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ____________________

52. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- ⬭ Document for later fix - Document an issue that should be revisited in the future.
- ⬭ Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- ⬭ Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- ⬭ Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- ⬭ Placeholder for later extension - Document an extension point for later enhancement(s).
- ⬭ Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- ⬭ None of the above
- ⬭ Other: ______________________

53. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- ⬭ Yes
- ⬭ No
- ⬭ Other: ______________________

54. Any comments (optional)

____
____
____
____
____

Case 9

Please look at the following example.

In the following snippet of Maven code, a comment describes the reason for uncommenting dependency tag.

```
<!-- 3rd Party Libraries -->
<!-- Latest version of iText has broken OSGi metadata and we have to wait until they fix it... -->
<!-- <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>itextpdf</artifactId>
        <version>5.1.1</version>
</dependency>
```

55. Could you please answer some questions about this case?

Mark only one oval.

◯ Answer this case

◯ Skip this case *Skip to question 61*

Questions to case 9

Please answer the questions.

In the following snippet of Maven code, a comment describes the reason for uncommenting dependency tag.

```
<!-- 3rd Party Libraries -->
<!-- Latest version of iText has broken OSGi metadata and we have to wait until they fix it... -->
<!-- <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>itextpdf</artifactId>
        <version>5.1.1</version>
</dependency>
```

56. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

57. Which of the following reason do you think this constraint occur?

Mark only one oval.

- Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- Recursive call - Coherence issues, recursive calls to invoke another build file.
- Document - Inadequate project description issues, such as licensing and metadata specification.
- Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- None of the above
- Other: ____________________

58. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- ◯ Document for later fix - Document an issue that should be revisited in the future.
- ◯ Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- ◯ Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- ◯ Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- ◯ Placeholder for later extension - Document an extension point for later enhancement(s).
- ◯ Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- ◯ None of the above
- ◯ Other: ______________________

59. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: ______________________

60. Any comments (optional)

Case 10

Please look at the following example.

In the following snippet of Maven code, a comment describes that dependency should be avatica, but not avatica-core.

```
<dependencies>
    <dependency>
        <groupId>org.apache.calcite</groupId>
        <artifactId>calcite-core</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.apache.calcite.avatica</groupId>
                <artifactId>avatica-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <!-- It should be avatica(the shaded one), not avatica-core, since the inconsistency protobuf dependency with Hadoop -->
    <dependency>
        <groupId>org.apache.calcite.avatica</groupId>
        <artifactId>avatica</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.apache.calcite.avatica</groupId>
                <artifactId>avatica-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
```

61. Could you please answer some questions about this case?

Mark only one oval.

- ◯ Answer this case
- ◯ Skip this case and Submit

Questions to case 10

Please answer the questions.

In the following snippet of Maven code, a comment describes that dependency should be avatica, but not avatica-core.

```
<dependencies>
    <dependency>
        <groupId>org.apache.calcite</groupId>
        <artifactId>calcite-core</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.apache.calcite.avatica</groupId>
                <artifactId>avatica-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <!-- It should be avatica(the shaded one), not avatica-core, since the inconsistency protobuf dependency with Hadoop -->
    <dependency>
        <groupId>org.apache.calcite.avatica</groupId>
        <artifactId>avatica</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.apache.calcite.avatica</groupId>
                <artifactId>avatica-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
```

62. Is this an example of technical debt in the build system?

Mark only one oval.

- Yes
- No
- Other: ____________________

63. Which of the following reason do you think this constraint occur?

Mark only one oval.

- ◯ Limitation - Constraints imposed by the design or implementation of third-party libraries or development tools.
- ◯ Dependency - Dependency issues due to unavailable artifacts or assets, such as missing or stale dependencies, dependency conflicts, or dependency resolution in post-install phase.
- ◯ Recursive call - Coherence issues, recursive calls to invoke another build file.
- ◯ Document - Inadequate project description issues, such as licensing and metadata specification.
- ◯ Build break - Broken builds (i.e., failures that occur during the build process) in build files.
- ◯ Compiler setting - Configuration issues during the compilation process, such as compiler configuration and symbol visibility.
- ◯ Code smell - Violations of fundamentals of design principles, i.e., instances of poor coding practice in build files.
- ◯ Change propagation - Changes that need to be propagated to keep software artifacts in sync during updates.
- ◯ None of the above
- ◯ Other: ______________________

64. Which of the following purpose do you think developers leave this comment?

Mark only one oval.

- Document for later fix - Document an issue that should be revisited in the future.
- Document workaround - Explicitly document constraints imposed by design or implementation choices. The comment contains workaround-related keywords, such as "workaround" and "temporary".
- Warning for future developers - Warn other developers to pay attention to an aspect of the solution that may not be clear from its structure or content.
- Document suboptimal implementation choice - Explain why a problematic solution has been adopted.
- Placeholder for later extension - Document an extension point for later enhancement(s).
- Silence build warnings - Defer or ignore warnings emitted by underlying tools.
- None of the above
- Other: ____________________

65. Have you personally experienced a similar kind of technical debt?

Mark only one oval.

- Yes
- No
- Other: ____________________

66. Any comments (optional)