A User Study on Logging

Note: This survey is confidential and we will not disclose your personal information. Your email will only be kept for further follow-up discussions if you would like to.

Software logging, as shown below, is a common programming practice in software development, typically issued by inserting logging statements (e.g., printf() in C, Console.Writeline() in C#) in source code. As in-house debugging tools (e.g., debugger), all too often, are inapplicable in production settings, logging has become a principle way to record the key runtime information of software systems into (console) logs for problem analysis.

```
try {
    RemoveOfflineAddressBooks();
}
catch(Exception e) {
    Logger.LogMessage(e.ToString());
}
```

We are conducting a user study on logging behaviors of developers, aiming at understanding and improving the current practices. It will be great if you can help fill out the following questionnaire. It may take you **about 5~10 minutes**. Thank you for your great support!

Background Information

Q1: What is your position?

- ❍ Developer
- ❍ Intern
- ❍ Others

Q2: How many years of programming experience do you have?

Q3: Do you use logging during your development?

- ❍ Not At All
- ❍ Occasionally
- ❍ Frequently

Q4: Do you think logging is useful?

- ❍ Useless
- ❍ Somewhat Useless
- ❍ Neutral
- ❍ Somewhat Useful
- ❍ Useful

Q5: Do you find it difficult on making good logging decisions (e.g., where to log, what to log)?

- ❍ Difficult
- ❍ Somewhat Difficult
- ❍ Neutral
- ❍ Somewhat Easy
- ❍ Easy

Logging Case Studies

Next, we are going to have a case study on your logging choice. In this study, we consider two logging strategies: exception logging and return-value-check logging, as shown in the code snippets below. Exception logging is to record the exception context (e.g., exception message, stack trace) in a catch block. Return-value-check logging is to record the situation where an unexpected value is returned from a function call.

Exception logging:

```
try {
    RemoveOfflineAddressBooks();
}
catch(Exception e) {
    Logger.LogMessage(e.ToString());
}
```

Return-value-check logging:

```
if (String.IsNullOrEmpty(tokenReference))
    ULS.SendTraceTag(ULSTraceLevel.Unexpected, "Missing token reference
value.");
```

While logging too little may miss the necessary runtime information needed for failure analysis, logging too much may also incur unintended consequences (e.g., performance impact, storage cost, useless and huge logs). Our task is to understand where developers choose to log.

For each of the following question, we will provide two code snippets generated from real-world software products. One of them is logged, while the other is unlogged, as the example shown below. However, in our question, we will delete the logging statements in the logged snippet. Your task is to

select the correct one that has actually been logged by the code owner. To do so, you need to try to understand the semantics of the code snippet and choose the one more likely to be logged according to your experience. You can answer 2 or 4 questions at your choice.

Example question: Which one is more likely to be logged? (*This is an illustrative question, where the answer is to choose the first one: A. A is generated from the above example.*)

- A: Log the following catch block

```
try {
    RemoveOfflineAddressBooks();
}
catch(Exception e) {

}
```

- B. Log the following catch block

```
bool userHasRights = true;
try {
    user.DeleteAccountProperty(propertyName);
}
catch (UnauthorizedAccessException) {
    userHasRights = false;
}
```

Important: In particular, a subset of the questions are provided with logging suggestions (to suggest log or not). The suggestions are generated by an automatic tool, but they can only achieve the accuracy of about 80%. **So the suggestions may be incorrect, which are provided as *references* for your own decisions.**

To facilitate our study, please do not search the code on the Web and only make your choice according to your own experience and preference. Both the accuracy and the time you spend on each question will be counted (timing is opened). **Please try to fill the following questions *accurately and quickly*.**

Q6: Which one is more likely to be logged?

- Log the following catch block:

```
private string GetSource (string objectName)
{
    LoggingService.LogDebug ("GetSource: " + objectName);
    IPooledDbConnection conn = connectionPool.Request ();
    IDbCommand command = conn.CreateCommand (
        String.Format ("EXEC sp_helptext '{0}'", objectName)
    );
    StringBuilder sb = new StringBuilder ();
    try {
        using (command) {
            using (IDataReader r = command.ExecuteReader()) {
                while (r.Read ())
                    sb.Append (r.GetString (0));
                r.Close ();
            }
        }
    } catch (Exception e) {
        QueryService.RaiseException (e);
    }
    conn.Release ();

    return sb.ToString ();
}
```

- Log the following catch block:

```
void SaveToCache(string cacheFileName, DateTime lastWriteTime, LoadedAssembly asm)
{
    if (cacheFileName == null)
        return;
    LoggingService.Debug("Serializing to " + cacheFileName);
    try {
        Directory.CreateDirectory(DomPersistencePath);
        using (FileStream fs = new FileStream(cacheFileName, FileMode.Create, FileAccess.Write)) {
            using (BinaryWriter writer = new BinaryWriterWith7BitEncodedInts(fs)) {
                writer.Write(lastWriteTime.Ticks);
                FastSerializer s = new FastSerializer();
                s.Serialize(writer, asm);
            }
        }
    } catch (IOException ex) {
        // Can happen if two SD instances are trying to access the file at the same time.
    }
```

Q7: Which one is more likely to be logged?

- Log the following catch block *Suggestion: Logged*:

```
static void RemoveCache(string cacheFileName)
{
    if (cacheFileName == null)
        return;
    try {
        File.Delete(cacheFileName);
    } catch (UnauthorizedAccessException ex) {

    }
```

- Log the following catch block *Suggestion: UnLogged*:

```
private void AddReference(JsonReader reader, string id, object value)
{
  try
  {
    if (TraceWriter != null && TraceWriter.LevelFilter >= TraceLevel.Verbose)
      TraceWriter.Trace(TraceLevel.Verbose, JsonPosition.FormatMessage(reader as IJsonLineInfo,
          reader.Path, "Read object reference Id '{0}' for {1}.".FormatWith(
          CultureInfo.InvariantCulture, id, value.GetType())), null);

    Serializer.ReferenceResolver.AddReference(this, id, value);
  }
  catch (Exception ex)
  {
    throw JsonSerializationException.Create(reader, "Error reading object reference '{0}'.".FormatWith(
        CultureInfo.InvariantCulture, id), ex);
  }
}
```

Q8: Which one is more likely to be logged?

- Log the following if block:

```
if (!BindingToKeys (binding, out chordKey, out chordMod, out key, out mod)) {
    return null;
}
```

- Log the following if block:

```
//html tags
    if (tagid.IndexOf(':') == -1) {
        return new HtmlParsingObject(location.PlainText, tagid, this);
    }
```

Q9: Which one is more likely to be logged?

- Log the following if block *Suggestion: Logged*:

```
int indexOfColon = TargetDirectory.IndexOf (':');
if (indexOfColon < 1) {
    return;
}
```

- Log the following if block *Suggestion: Unlogged*:

```
if (System.IO.File.Exists(absPath)) {
    return absPath;
}
```

Assessment of Logging Suggestion

Q10: Does the suggestion result match your choice?

- Always
- Sometimes
- Never

Q11: Do you think that the suggestion result is useful for your logging choice?

- Yes
- No

Thank you for your great support to our study!

Email (Optional for follow-up discussions):