Concern-Oriented Reuse Project

Part 1

Group 20:
Ilan Mamontov
Pier-Luc Picard

Concern and Features

Monitoring is an important concern for software development. This concern tracks various metrics to help the developer find problems and bottlenecks, allowing him to quickly fix and optimize his code. To do so, the concern has to present the relevant information clearly by generating reports with provided logs and alerting when something goes wrong. The following features are part of our monitoring concern.

Monitor is the main controller. The monitor is essentially keeping the data (info frames) between our features synced up. As soon as a new frame is created containing logs, the monitor keeps track of it. The report or alert feature then takes use of these frames to do their function.

Resource monitoring is our first mandatory feature. It has its own separate timer that invokes the collection of all information. Data from memory, networks, CPU, and disk are all being collected within info frames. This data can then be used to identify memory leaks, compile statistics and monitor the overall performance of the application.

Report is our second mandatory feature. Report has a simple, but important functionality. It's main role is to gather all data stored within different info frames and generate a report that displays the data in a convenient way.

Function Call is our first optional feature. Monitoring Function Call enables us to track a few important metrics regarding function calls. This feature tracks the input, output and execution time among other things. This allows us to find bottlenecks, and bugs much easier. Data is stored within info frames which monitor then keep track of them.

Alert is our second optional feature. Alert uses a set of defined rules to evaluate the frames provided by both resource and function call features. If something unusual is detected, alert sends a notification containing the frame that triggered it.

Engine is the third optional feature, while being part of the mandatory Resource feature. This feature main responsibility is to log all relevant information regarding the engine. Data like stack and heap are being stored.

Finally, Garbage Collection is our final optional feature. Its role extends from that of the engine. This feature tracks garbage collection data done by the engine.

Impacts and Justification

The optional features have three very important impacts being: increasing complexity, increasing cost and gaining additional information.

As we add more features to our system, we are increasing the complexity of the application. For example, adding an Alert system increases the complexity by adding a new layer to maintain.

Adding Alert will also increase the cost of the application, either by requiring more manpower, or by increasing the resource consumption (CPU, memory etc.). At the same time, Alert will provide additional information to the developers. Having an Alert feature in place will allow a quick reaction to an issue.

Complexity

1. We gave Function Call the largest impact on the complexity as it requires a complex setup to monitor every single call made in the system.
2. Engine is second as it require a single complex setup to the language engine.
3. Alert is third, requiring less setup than Engine, but still adding a new layer of monitoring to the application.
4. Finally, Garbage Collection is just an extension to the already setup Engine.

Cost

1. Function Call has the largest cost due to requiring a lot of resources when monitoring every single call. Either by requiring a lot of log storage, or by requiring a lot of manpower to setup and maintain this feature.
2. Engine also has a relatively large cost associated due to monitoring a complex system, but requires less resources than monitoring every single function call
3. Both Alert and Garbage Collection have a small cost due to not requiring a lot of resources or manpower for maintainability.

Information

1. Engine is the feature that provides the most additional information. It provides us with information about the Heap, Thread, Stack, and more. These are very important when diagnosing an issue.
2. Function Call provides almost as much as Engine. This feature provides us with more high-level information about our application. Such information allows us to optimize certain parts of the code.
3. Alert does not provide with more statistics about our application, but provides an important notification when an issue is encountered.
4. Finally, Garbage Collection provides us with very specific details about the engine. These details can be very important to some, but is often there just for very specific issues (e.g. memory leaks)

To conclude, these features provide information that are critical when trying to address performance related issues and ensure that problems are solved quickly. Without monitoring, developers are debugging blindly when trying to optimise a software application. Furthermore, developers can release a product quickly and optimise only the relevant and critical parts of the system according to the provided monitoring data.