Actor Bug: Communication - Order

This is a classical actor issue where a developer expects messages to arrive in a certain order instead of planning around the non-deterministic nature of actor messaging. Such as message A needs to arrive before message B and if they arrive out of order an error occurs.

It's worth noting here that the majority of the Message Order bugs found for this paper were from Gatling where one one user was reporting on these errors that they found using a tool that injects random delays into the messages of the system. Thus it's unclear if this is unique to Gatling's programmers and this user was able to point them out more thoroughly, or if the other actor developers simply lacked a user doing similar robust debugging message order errors for their own services.

For an example, Gatling uses a DataWriter actor that can become uninitialized after it receives a flush message from a Terminator actor. The Gatling system was built with a classical message order problem where the DataWriter was expected to receive all the messages it needed to before flush arrived from the Terminator actor. However, as messages are non-deterministic these could arrive from these different sources out of the order expected and the DataWriter could receive a scenario message after a flush message which was an unexpected state.

This issue was fixed by adding an extra case for having the programs Terminator end a user based on the scenario message and to output a simple logging message saying the DataWriter received a message after flushing in error.

Link: https://github.com/gatling/gatling/issues/1116

Code:

gatling/core/result/writer/DataWriter.scala

```
     def uninitialized: Receive = {
          case Init(runMessage, scenarios) =>
….

+         case m: DataWriterMessage => logger.error(s"Can't handle $m when in uninitialized state, discarding")
     }

     def initialized: Receive = {
-         case scenarioMessage: ScenarioMessage => onScenarioMessage(scenarioMessage)
+         case scenarioMessage: ScenarioMessage =>
+              onScenarioMessage(scenarioMessage)
+              if (scenarioMessage.event == End) Terminator.endUser
...
```