```
prepare input artefacts()
                                    finished ≔ false
                                    readOnlyBlackboardView := Read-Only Blackboard View .construct (blackboard)
                                    measurementControllerBlackboardView := Measurement Controller Blackboard View .construct (blackboard)
                                    analyserBlackboardView ≔ Analyser Blackboard View .construct (blackboard)
                                    final Judge Blackboard View := Final Judge Blackboard View .construct (blackboard)
                                    while (¬finished) do
                                       Mifasurement Controller .can measure (readOnlyBlackboardView) then
                                      Measurement Controller .measure (measurementControllerBlackboardView)
                                        else
                                            analysersIteratAmatysers.iterator()
                                            while ¬analysersIterator.current (Contribute (readOnlyBlackboardView) do
                                                 analysersIterator.next()
                                            od
                                            if analysers Iterator. curroamt () on tribute (read Only Blackboard View) \\
                                                analysersIterator.com/teibt(te (analyserBlackboardView)
                                                fimiah&ddge.judge(finalJudgeBlackboardView)
                                        fi
                                    od
                                    write results()
                                                                                             Beagle Controller
                     + main( PCM repository files: File [1..*], IDs of stuff to measure: String [1..*], execution files: File [1..
                                                        1...
                                                                                                                                                           - Analysers {unique}
                                                                - Measurement Controller
                                                                                                                                                «Interface»
                                        Measurement Controller
                                                                                                                                             Result Analyser
                    + can measure( blackboard: Read-Only Blackboard View [1]
                     + measure( blackboard: Measurement Controller Blackboard View [1]
                                                                                                                + can Contribute( blackboard: Read-Only Blackboard View [1]): Boolean [
                                                                                                                + contribute( blackboard: Analyser Blackboard View [1]
                                                                1
return | blackboard. get RDIAs to be measured ()
                                                                                                         - Final Judge
    U blackboard. get SEFF Branches to be measured ()
                                                                                                            «Interface»
    U blackboard. get SEFF Loops to be measured ()| > 0
                                                                                                           Final Judge
                                                                             + judge( blackboard: Final Judge Blackboard View [1]): Boolean [
                              - Measurement Tools {unique,
                                     «Interface»
                               Measurement Tool
                       + measure( null: Measurement Order [1]
```