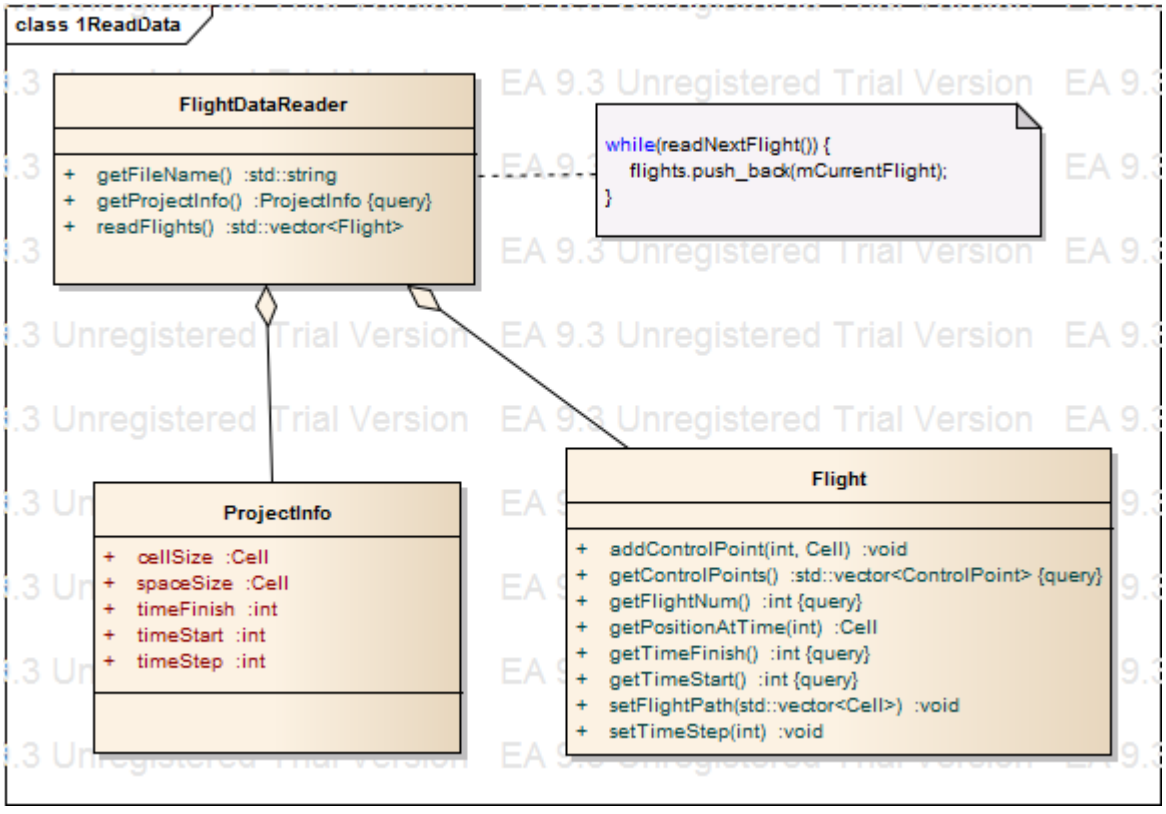
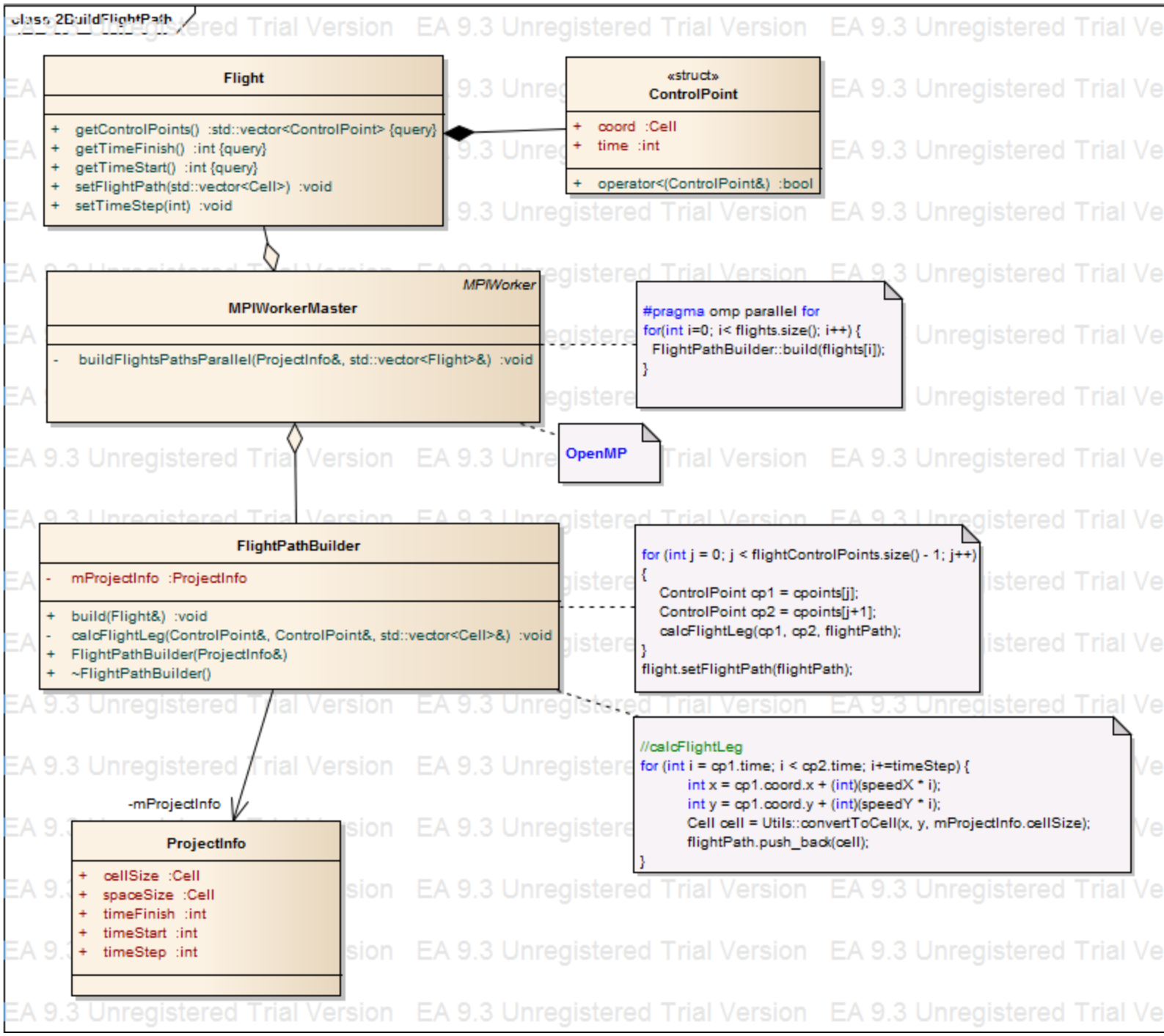
Critical Level Calculation Documentation

[Critical Level Calculation Documentation](#)
- [1. Read flights data](#)
- [2. Build flights path](#)
- [3. Critical Degree Calculation](#)
- [4. MPI](#)

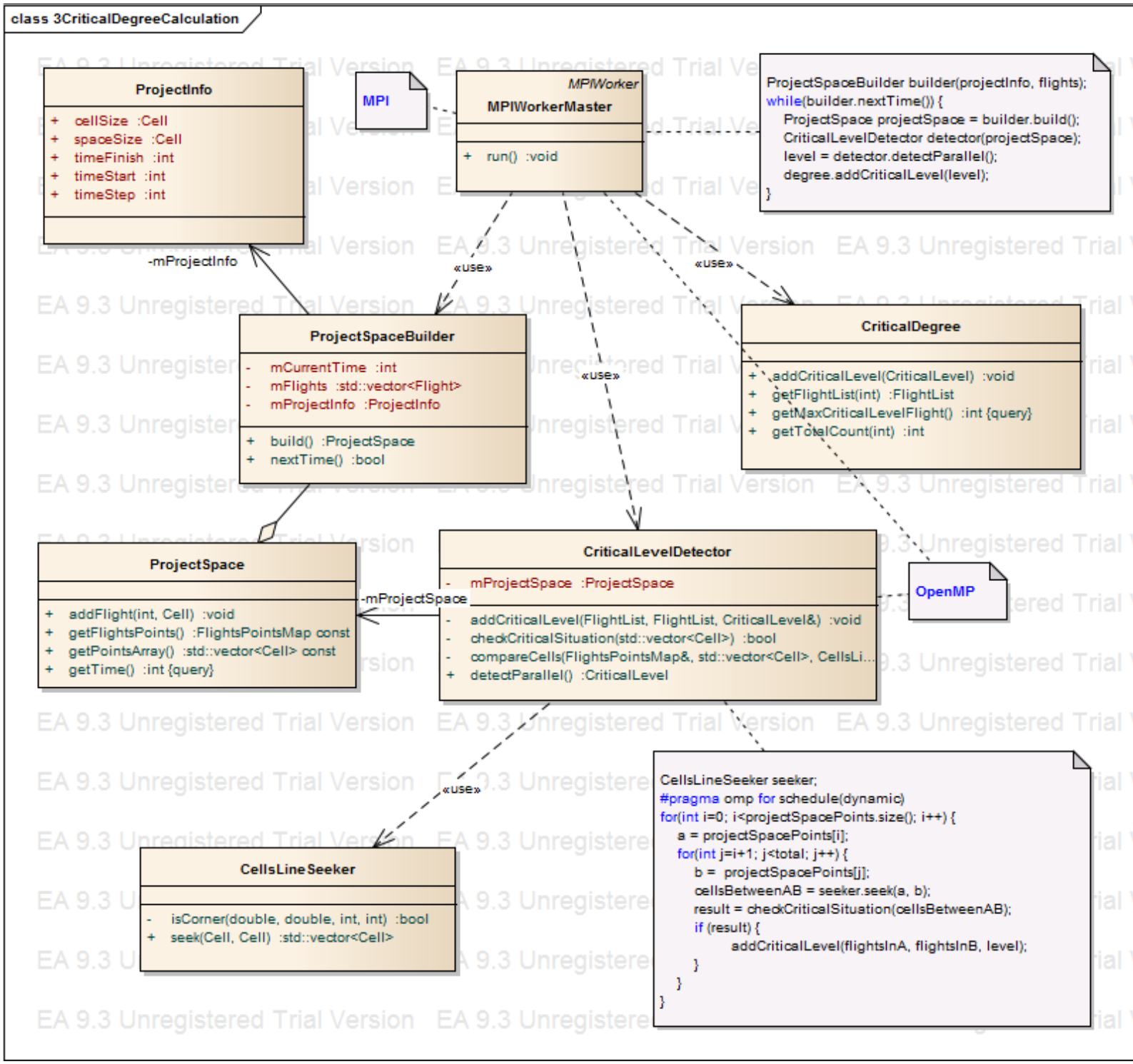
1. Read flights data



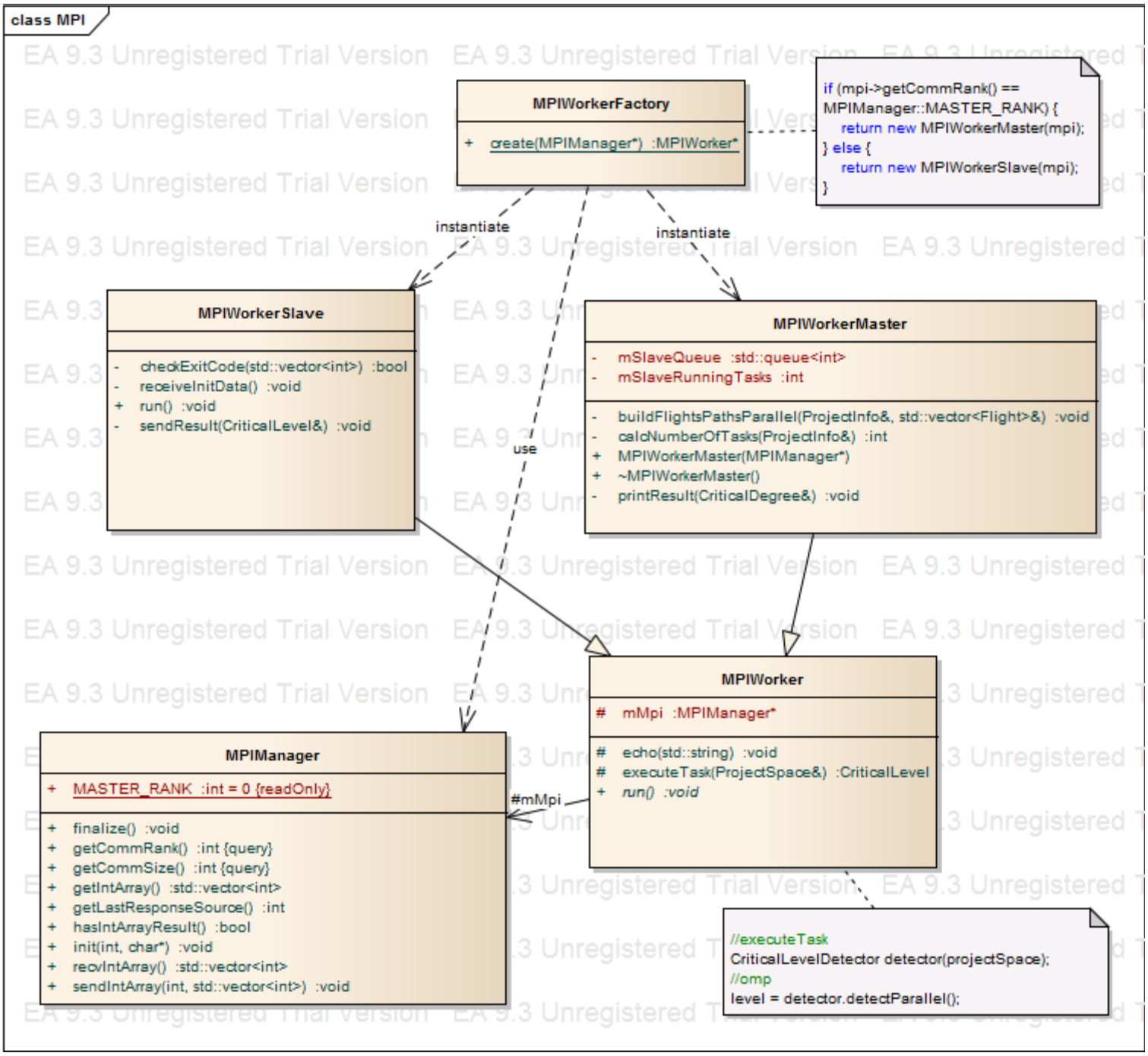
2. Build flights path



3. Critical Degree Calculation



4. MPI


class MPI
EA 9.3 Unregistered Trial Version
MPIWorkerFactory
+ create(MPIManager*) :MPIWorker*
if (mpi->getCommRank() == MPIManager::MASTER_RANK) {
return new MPIWorkerMaster(mpi);
} else {
return new MPIWorkerSlave(mpi);
}
instantiate
instantiate
use
MPIWorkerSlave
- checkExitCode(std::vector<int>) :bool
- receiveInitData() :void
+ run() :void
- sendResult(CriticalLevel&) :void
MPIWorkerMaster
- mSlaveQueue :std::queue<int>
- mSlaveRunningTasks :int
- buildFlightsPathsParallel(ProjectInfo&, std::vector<Flight>&) :void
- calcNumberOfTasks(ProjectInfo&) :int
+ MPIWorkerMaster(MPIManager*)
+ ~MPIWorkerMaster()
- printResult(CriticalDegree&) :void
MPIWorker
mMpi :MPIManager*
echo(std::string) :void
executeTask(ProjectSpace&) :CriticalLevel
+ run() :void
#mMpi
MPIManager
+ MASTER_RANK :int = 0 {readOnly}
+ finalize() :void
+ getCommRank() :int {query}
+ getCommSize() :int {query}
+ getIntArray() :std::vector<int>
+ getLastResponseSource() :int
+ hasIntArrayResult() :bool
+ init(int, char*) :void
+ recvIntArray() :std::vector<int>
+ sendIntArray(int, std::vector<int>) :void
//executeTask
CriticalLevelDetector detector(projectSpace);
//omp
level = detector.detectParallel();