# Assessment rubric for Systems Simulation projects (authors):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Criterion** | | **Expert (1)** | | | | **Gifted (2)** | **Competent (3)** | **Learner (4): Project should …** | **(5)** |
| **Argument: Presentation** | | Uses ***concise, precise and accessible text and graphics*** elements to present research and null hypotheses, and project description. | | | | Presents project description and hypotheses using ***concise and precise text and graphics*** elements. | Presents project description and hypotheses using ***concise and precise text***. | … present ***project description, research and null hypotheses in easily accessible language***. |  |
| **Argument: Operational** | | States the ***research and null hypothesis*** in a form that ***is refutable***. | | | | ***Research and null hypotheses are mostly refutable***. | ***Research and null hypotheses are only partly refutable***. | … ***operationalise*** research and null hypotheses. |  |
| **Argument: Logic** | | The ***project tests*** consistently and measurably the correctness of ***the hypotheses***. | | | | ***Project mostly tests*** the correctness of ***the hypotheses***. | ***Project tests only partly*** the correctness of ***the hypotheses***. | … clarify ***sufficient conditions*** for the hypotheses. |  |
| **Argument: Methods** | | ***Experimental design fulfils all requirements of the project description to test hypotheses***. | | | | ***Experiment fulfils most project requirements*** to test hypotheses. | ***Experiment partly fulfils the project requirements*** to test hypotheses. | … ensure that ***experiment robustly tests the hypotheses***. |  |
| **Argument: Analysis** | | Presents experimental results in such a way that ***readers immediately understand how they (dis-)confirm the research hypothesis***. | | | | Presents results in such a way that ***readers can follow how they (dis-) confirm the research hypothesis***. | Presents results in such a way that ***they plausibly (dis-)confirm the research hypothesis***. | … ***derive*** (dis-)confirmation of ***hypotheses*** convincingly ***from experimental results***. |  |
| **Argument: Discussion** | | The wider ***meaning and relevance*** of the conclusions is ***immediately apparent***. | | | | Wider ***meaning and relevance*** of the conclusions is ***apparent***. | Wider ***relevance*** of the conclusions is only ***recognisable*** with effort. | … ***present wider relevance*** of the work for other domains. |  |
| **Argument: Software-Apparatus** | | ***Software-code fulfils all requirements*** of the project without logical gaps, checks for user errors and displays appropriate outputs. | | | | ***Code fulfils all requirements*** without errors, with some inappropriate output. Checks for some user errors. | ***Code delivers correct results***, but displays them incorrectly. Checks for some user and range errors. | … ***fulfil all experimental requirements*** correctly and check user input. |  |
| **Software: Presentation** | | ***The code is clearly structured and formatted.*** Clear code-blocks, methods, indentation and line-breaks facilitate easy understanding. | | | | ***Code is easy to follow*** despite minor formatting, indentation or bracketing errors. | ***Code is mostly easy to follow***, but formatting increases the difficulty of this task. | … use clear ***formatting*** to ***express the execution flow*** clearly for lay readers. |  |
| **Software: Coherence** | | ***Code-structure, naming and comments emphasise*** clearly ***the unifying intention behind*** allmodules and ***code-components***. | | | | ***Comments express the intention of the components***,but ***naming is*** sometimes ***confusing***. | ***Comments express intentions***,but ***naming does not***. | … use ***code-structure and naming*** to ***communicate*** clearly its unifying ***intention***. |  |
| **Software: Typing** | | ***Uses*** primitive and custom ***types efficiently and correctly*** to structure code cleanly and conceptually. | | | | ***Uses types appropriately*** to structure code efficiently and transparently. | ***Uses types appropriately***, but based on inappropriate or unclear conceptual structures. | … use ***typing*** as a tool for ***designing software around conceptual structures***. |  |
| **Software: Control** | | ***Control structures (recursion, convolution, iteration) support effective algorithm design***. | | | | ***Employs control structures mostly appropriately*** to algorithmic intent. | ***Uses control structures appropriately,*** but in inappropriate algorithms. | … use ***control structures*** as a tool for ***algorithmic design***. |  |
| **Software: Modularity** | | ***Modules***, methods and interfaces ***possess*** and encapsulate a ***transparent*** ***Intention and responsibility*** to minimise redundancy and error propagation (rippling). | | | | ***Modularity is transparent***, but leaky partitioning of responsibilities permits redundancy and error propagation. | ***Modularity is understandable***, but the code employs global access to data, so allowing the dangers of redundancy and error propagation. | … clearly ***partition modules***, methods and interfaces ***based on*** their ***responsibility and intention***. |  |
| **Software: Efficiency** | | ***Code is very efficient***, minimises operation complexity and stores multiply accessed data, ***without sacrificing readability***. | | | | ***Code is efficient without sacrificing readability*** and comprehensibility. | ***Code is efficient with little loss of readability*** or comprehensibility. | … organise ***communication*** between code-components ***efficiently and readably***. |  |
| **Total =** |  | | **/ 13 =** |  |  |  |  |  |  |