

Assessment rubric for Systems Simulation projects (authors):

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

