B. *Beliefs about the Nature of the Scientific Challenges*

1) Verification and Validation are Difficult

Belief: According to Carver et al. [12], and others, verification and validation in software development for CSc is difficult and strictly scientific [13, 15, 20, 21]. That is, CSc developers spend more time debugging their theories of physical phenomena than debugging SE related issues (system testing, continuous integration, etc). If this belief were true, then much of the standard SE testing infrastructure would need extensions before it can be applied to CSc. For example, while unit tests and system tests are certainly useful, CSc projects would need a separate level of tests for domain-specific testing.

Notes: According to Carver et al. [12], verification and validation of scientific software can be difficult for several reasons:

- Lack of suitable test oracles [21]. Complex distributed hardware environments with no comparable software [13],
- Scientists' suspicion that the problems of the software are the results of their scientific theory [18],
- Lack of physical experimentation and experimental validation is impractical [12].

Modeling Assumptions: As stated above in §II-A, in order to bridge between the terminology of the belief and the Github data, we assume that (1) V&V is associated with testing; (2) the amount of testing is an indicator for V&V efforts; and (3) the amount of failed tests indicating how rigorous the software is being validated and verified. This second assumption is shared by Vasilescu et al. [34, 35], where the number and proportion of commits are treated as indicators for developing efforts of the repositories. For the third assumption, we can say the amount of failed tests (FT) and failed builds (FB) in Travis CI are indicators for measuring the V&V rigor results with respect to the efforts. We define four attributes, where higher values indicate more rigorous testing within the projects:

- *FB_Ratio* = sum(FB) / all_builds : how often do builds fail?
- *Average_FT_per_FB* = sum(FT) / FB : for each failed build, what is the number of expected failed tests?
- *FT_Ratio_across_FB* = sum(FT) / sum(Tests_across_FB) : what is the proportion of failed tests across all failed builds?
- *FT_Ratio_per_FB* = median(FT_per_FB / Tests_per_FB) : for each failed build, what is the proportion of failed tests?

Prediction: Verification and validation in CSc is more difficult than in SE if the observed CSc efforts (2) with respect to CSc rigor (3) in this area is larger than in SE. To justify strictly scientific development, we should expect far more scientific testing than SE testing.

Observed: We used a simple approach to show that CSc software verification and validation are heavily focused on scientific issues. Among the commits that were labeled testing, we classified them based on science, SE, or other testing. The labeling guidelines are similar to the ones used for the whole development cycle mentioned in §II-C. We achieved a "substantial agreement" with the IRR between the two reviewers with Cohen's $\kappa = 0.78$ [48]. Table V shows that scientific testing is the largest type of commit in our labeled Testing commits sample (47%). Far less effort is spent on SE testing (only 24%).

TABLE V
DISTRIBUTION OF TESTING COMMITS.

	Absolute	%
Science	289	47%
SE	146	24%
Other	173	29%

To show that the CSc V&V is "as much or more difficult" than in SE, recall that Figure 2 showed that 15% of the commits were associated with CSc testing, whereas 6% were associated with SE testing. This SE data comes from a recent study [47] which randomly sampled 20 highly starred projects from Github that satisfy our sanity checks of Table II. Note that 15% is 2.5 times larger than 6%. Table VI summarizes the median and interquartile range of testing rigor for both CSc and SE projects. Across all four metrics, CSc projects are reported to test more rigorously than SE projects. However, by the statistical Scott-Knott test[2], the difference in testing rigor is not distinguishable between CSc and SE. This is interesting as we are aware that scientific developers have to put 250% more effort into testing than SE developers, yet CSc's testing is as rigorous as SE's testing. That is to say, the nature of verification and validation is more or at much as difficult for scientific developers as it is for traditional software engineers.

TABLE VI
MEDIAN AND INTERQUARTILE RANGE (IQR) SUMMARY FOR THREE ATTRIBUTES PORTRAYING THE TESTING RIGOR OF CSC AND SE PROJECTS. IQR IS THE DELTA BETWEEN THE 75TH AND 25TH PERCENTILE.

Metric	Project	Median	IQR
FB Ratio	SE	38%	33%
	CSc	25%	20%
FT per FB	CSc	20	27
	SE	11	19
FT Ratio across FB	CSc	3%	2%
	SE	1%	7%
FT Ratio per FB	CSc	26%	19%
	SE	20%	27%

Conclusion: We **endorse** the belief that within CSc, software development's verification and validation are difficult and are mostly concerned with scientific issues.

Discussion: This result is somewhat unexpected since it runs counter to standard beliefs in the SE literature. For example, Brookes argues that unit tests and systems tests will consume half the time of any project [53]. One of our conjectures is that the larger V&V effort in CSc is due to the nature of CSc problems. CSc software is written to solve endless problems in nature (most of which are beyond human's understanding), with requirements that are not known upfront and the software's state is incrementally improved. CSc V&V has to cover both scientific and engineering concerns while SE V&V at some points would mature to only focusing on verification, especially because SE software is largely focused on products.

More intuitively, by looking at the *Testing* and *Bug-fixing* attributes from Figure 2, the bug-fixing activities from SE software development are almost three times more than CSc, which is the direct result from testing 2.5 times less than CSc. Essentially, the *less* developers test, the *more* bugs developers

[2]Scott-Knott recursively divides treatments, stopping if a significance test or an effect size test reports that sub-divisions are not statistically different [51, 52]. We use a bootstrap test to check for significance differences (at the 95% confidence level) and the *A12* test to check for small effects ($A12 \geq 0.6$). This procedure has been endorsed in the recent SE literature [46, 51].

have to potentially fix. After shipping software products, SE developers are more reluctant to test the software while for CSc developers, scientific software research and development might be a continuous process as knowledge grows.

Moreover, a conflating factor that might make us doubt this observation is if CSc code bases were always much smaller than the SE code bases. If that were true, then even if tasks had a larger percentage effort (e.g., Table V's "scientific testing") then "relatively more" might actually mean "less" (in absolute terms). However, as discussed in §II-B, our data does not show that SE projects are larger & more active than CSc projects.