To the reviewers,

Thank you for your constructive suggestions. Below we have indicated (in red) how we have addressed your concerns in the final version of the paper.

===========================================================================

PLATEAU 2016 Review #1A

---------------------------------------------------------------------------

Paper #1: Discount Method for Programming Language Evaluation

---------------------------------------------------------------------------

Overall merit: 3. Weak accept

Reviewer expertise: 4. Expert

===== Paper summary =====

Describes an adaptation of discount usability testing to programming language evaluation and then applies it to Quorum, demonstrating that it can reveal usability problems of varying severity.

===== Comments for author =====

This paper addresses a gap in programming language design methods: the need for rapid ways of getting early feedback about syntax and semantics before building anything. Adapting the discount usability methods from HCI is a sensible idea. The authors chose to adapt think aloud usability testing, similar to those used by Pane et al. (2002) to develop the HANDS language. The paper provides some useful guidance on how to perform an evaluation, including several good PL-specific ideas about how to simulate IDEs and the specific steps to prepare for an evaluation.

As a research paper, the results are a work in process. The paper does not provide strong evidence that the method is "discount" or effective at detecting problems. Some of the best work on this was done by Nielsen et al. (1990) in their evaluation of the Heuristic Evaluation technique, in which they measured the tradeoff between the number of evaluators and the number of issues discovered. A similar evaluation in this work would have revealed the tradeoffs in the number of issues found and the number of participants included. The authors actually have the data to show this: how many unique issues did each participant reveal? That would have revealed how much return on investment each participant offers to a language designer.

A short comment about this has been added to the paper

The work also has a weak motivation. User interfaces are designed in the tens of thousands a year, and on very tight schedules, and so the need for discount evaluation methods is great. Programming languages are designed slowly and much less frequently. Is there really a need to evaluate them quickly? One argument against this is JavaScript: it was developed quickly and many of its issues could have been found with discount methods. A study that identified its longstanding issues in just a few hours would have been more persuasive.

A study of JavaScript would indeed be interesting and is now on our list for future work.

Although only a few programming languages make it into mainstream, hundreds of domain specific languages are designed every year. As a teacher of language design, I see at least 10 new languages every year and have thus felt the need for a language design evaluation method that can be applied in the early phases of a language design.

===========================================================================

PLATEAU 2016 Review #1B

---------------------------------------------------------------------------

Paper #1: Discount Method for Programming Language Evaluation

---------------------------------------------------------------------------

Overall merit: 5. Strong accept

Reviewer expertise: 3. Knowledgeable

===== Paper summary =====

The paper proposes a lightweight evaluation technique for programming languages that can be applied early in the design process, requiring only five or six test subjects. In previous work they applied the technique to C# and F# in the context of an IDE and implementation; here they apply it to Quorum with no IDE or implementation. Their results are roughly comparable to previous studies on Quorum that used more standard techniques.

===== Comments for author =====

This paper is to be commended on two points. First, with the increasing importance of domain-specific languages there are bound to be an increasing number of programming language designs, so a lightweight evaluation technique will be of more and more value. Second, the authors chose to evaluate by comparing with work other than their own, C# and F# in one instance and Quorum in another.

My main recommendation for revision is that in Section 4.1 the authors should be careful to make clear where they are describing results from the previous study of Quorum and where they are describing their own results. As far as I can tell, the bulleted list of page 5 is summarising the previous results, whereas the material on page 6 summarises the new results. It would be better if each bullet point was clearly divided into two segments, the first listing the previous results from Quorum (as now) and the second giving the comparable results from the study (including, if there are no comparable results from the current study, just noting that).

Done

The real question is whether this technique can help designers uncover conceptual problems in a language design. The two studies reported to date are equivocal on this key point. With Quorum, the problems identified appeared to have more to do with mismatches between the experienced programmers they tested upon and the novice audience that Quorum is designed for. The planned studies with students designing programming languages for their course at Aalborg seem perfect for gathering evidence as to whether the technique can be effective.

The authors suggest that a sample sheet be used as the sole means to convey the intended design of the programming language; the authors might say more about alternatives, including grammars and formal methods. It would have been helpful to include a sample sheet (or fragment thereof) for Quorum in the paper. Most importantly, all of the experimental materials and results should be made available online and the URL given in the paper.

Due to lack of space, the sample sheet and task sheet are not included. A URL has been added.

The paper refers to a previous study on C# and F#, but that study does not appear to have been published and I could not find it online. Therefore, it would be appropriate to also include more detail of the previous study when revising this paper to prepare the final version.

Lack of space prevents us from elaborating on this study. A URL has been added.

[Added after reading Referee A's review: Applying your technique to evaluate JavaScript is an excellent idea.]

===========================================================================

PLATEAU 2016 Review #1C

---------------------------------------------------------------------------

Paper #1: Discount Method for Programming Language Evaluation

---------------------------------------------------------------------------

Overall merit: 1. Reject

Reviewer expertise: 3. Knowledgeable

===== Paper summary =====

The paper describes a method for evaluating programming languages without resource-intensive user studies. The method involves giving a small group of subjects tasks and interviewing them about their experiences. The method is designed to be used while a language is in the design phase, notably before the language has been implemented.

===== Comments for author =====

Lightweight methods for evaluating programming languages prior to implementation are an important research topic. I like the problem that this paper attempts to address.

Unfortunately, the paper is significantly lacking in detail. I would expect a paper on this topic to have enough detail that someone else could try the proposed evaluation approach on their own, simply based on the content of the paper. This paper fails to meet this standard.

For example, most of the Method section (section 2) contains what appears to be boilerplate for conducting any study with users (such as "prepare setup", "conduct a pilot test", and "start experiment"). What here is the proposed method, versus what is just good practice for user studies? The discussion suggests that the sample sheet is key (which I would believe). But then the paper needs to include an example of both tasks and the sample sheet, so that the reader can understand the proposed method.

Sample tasks and the sample sheet are also important for gauging the validity of the experiment. If the sample sheet contains code fragments that are analogous to the tasks given during the experiment, then the method might just be testing the copy/paste ability of the participants. Details about the materials used is essential to assess whether the experiment is valid.

Due to lack of space, the sample sheet and task sheet are not included. A URL has been added.

The "Analyse Data" section needs to be more specific about the criteria checked. What counts as "a problem encountered during the test"? Something the participant initially got wrong? Took too long to solve? Did less efficiently than usual? These metrics are critical and need to be part of the paper.

I sensed from page 6 that the criteria were along the lines of "what did the programmer like or expect", but that's an odd metric. It would suggest that the new language is designed to do just what a programmer's current languages do (in which case there would be little point to switching languages, except for syntax). Is this analysis checking for syntax design, or construct design?

Admittedly, these criteria are rather informal and could in the future be refined by more formal evaluation criteria based on e.g. the cognitive dimensions framework.

This is now mentioned in the paper.

Similarly, the paper should include the interview questions.

Lack of space prevents us from including the interview questions. A URL has been added.

In addition, the results seem misaligned with the goals of the paper. The paper claims to be about the evaluation method, but the results section is all about issues discovered with Quorum. The authors should adjust the writing to decide so that the introduction and the results are focused on similar goals.

Quorum is used as a test case for the method. As such we mainly confirm issues already known about Quorum, which to us indicate that the method is able to help language designers find such issues quickly and with less resources than by other methods.

The work needs to be related to Green and Blackwell's "Cognitive Dimensions" framework, which also provides a way to do lightweight HCI-based analysis. While that work has been applied more to evaluating interfaces than languages, the dimensions are concrete, raising specific ways in which a system (here, the language) might not be as usable as the designers intended. The Cognitive Dimensions papers might also give the authors an idea of how to present a system such as theirs, so that it has the appropriate level of detail to be useful to readers.

We have to admit that we had not heard of the Cognitive Dimensions" framework before. So thank you for pointed out this work to us.