The main statement of the reviewer 1 is:

"Until the confusion about the purpose of the OCL assertions is cleared up, I cannot accept this paper. "

Regarding this purpose, in the before the last paragraph of Introduction, I stated:

"From the first version of the standard, the level of exigencies required for OCL specifications has grown in a natural manner. At the beginning, the understandability and the correctness of specifications were among the most cited topics in the literature. The use of the OCL was focused on specifying the model in a complete, easy understandable and unequivocal manner. Now, these do not suffice. In design by contract, when assertion evaluation fails, the users have to fix the problem. This supposes to be informed about the reasons of failures. Fortunately, using OCL this information can be obtained in a natural manner.

Regarding the applications (purposes of assertions), in design by contract, [Meyer97] pp. 389 states:

"There are four main applications:

- Help in writing correct software.
- Documentation aid.
- Support for testing, debugging and quality assurance.
- Support for software fault tolerance.

Only the last two assume the ability to monitor assertions at run time."

Therefore, I would like to ask the reviewer in the most polite manner to explain me which is "the confusion made in the paper about the purpose of the OCL assertions". Unfortunately, in the review I didn't find this information.

The most detailed statement of the reviewer 1 is:

"To take an example from the paper, **an invariant** stating that all books have unique titles could appear in a high-level requirements specification for a publishing house. A straightforward assertion might contain something like "b1.title = b2.title implies b1 = b2".

At runtime, in every implementation that satisfies the spec, there will never be two books with the same title. Therefore, there will never be the need to identify a book that breaks the rule."

I am sorry to say that the above mentioned example, presented in pp 3 of my proposal, was totally tacked out from the context! In my proposal, the example was included only to highlight that many OCL specifications published in the literature are incorrect, so these need to be fixed.

"All these three specifications are incorrect and inappropriate. If among the Book instances there are at least two with the same title, then the value returned after evaluating any of the specifications above will be `false` for any instance of the `Book class` (see Figure 2.1). For the objects having a unique title, this evaluation result is wrong. This proves that the above specifications are incorrect with respect to the semantic of invariants; only in case of instances violating the constraint, the evaluation result must be false. In our concrete example, those are the instances whose title appears at least twice."

As regarding the runtime evaluation, in my proposal (pp 2), I stated: "Evaluating invariants is useful in case of existent models (static evaluation) and in case of persistent objects loaded from databases or from different files. The evaluation of pre and postconditions is especially useful at runtime enabling to update the value of the mentioned attribute in a consistent manner with the semantics of the application." So, I cannot understand the statement of the reviewer 1: "**At runtime**, in every implementation that satisfies the spec, there will never be two books with the same title. Therefore, there will never be the need to identify a book that breaks the rule.".

I consider that the following quote from [Meyer97] pp 394 clarify the reviewer question about: "How much assertion monitoring? ":

"What level of assertion tracing should you enable? The answer is a tradeoff between the following considerations: how much you trust the correctness of your software; how crucial it is to get the utmost efficiency; how serious the consequences of an undetected run-time error can be."

…

"When you are debugging a system, or more generally testing it prior to release, you should enable assertion monitoring at the highest level for the classes of the system (although not necessarily for the libraries that it uses, as explained next). This ability is one of the principal contributions to software development of the method presented in this book. Until they have actually had the experience of testing a large, assertion-loaded system using the assertion monitoring mechanisms described in this section, few people realize the power of these ideas and how profoundly they affect the practice of software development."

Finally, the last important notice of reviewer 1 is:

"The paper's unsupported claim that "this is a reasonable price for having a correct and useful invariant" can be questioned: does "useful" only mean "supports debugging of faulty code purporting to satisfy a spec"?"

Let's suppose an assertion A. In my opinion, if an assertion specification, AS1, could offer more useful information than another specification of the same assertion AS2, then AS1 is more useful compared with AS2. This is just because in case of using AS1 instead of AS2 we can fix the bug easier. In the paper, both AS1 and AS2 were mentioned explicitly for all the examples. Therefore I stated that even in cases when AS1 is a little bit more complex than AS2, this is a reasonable price to be paid. I didn't said that ""useful" only mean "supports debugging of faulty code purporting to satisfy a spec ". I just stated that using a little bit more complex specifications is a reasonable price for obtaining more information.

Just a simple example from the paper:

Suppose that AS2 is the pattern of specifying a collection whose elements satisfy a property:

```
context Book::setTitleE(t: String)
pre uniqueTitleE1:
Book.allInstances->forAll(b | b.title <> t)
```

Then, the following AS1 specification:

```
context Book::setTitle(t: String)
pre uniqueTitleP:
Book.allInstances->reject(b| b.title <> t)->isEmpty
```

is more useful that AS2 because we can compute the set of books that does not comply with the condition required for the title.

The key statement of the reviewer 2 seems to be:

"In order to be acceptable I suggest that the paper is reorganized to remove the subjective nature of the claims and to increase the objectivity of the content."

Unfortunately I didn't understand which of my statements are subjective and why. Therefore I would like to ask the reviewer to give me at least two examples of subjective claims.

Another statement of the reviewer 2 that I didn't succeed to fully understand is the following:

"It certainly lacks technical analysis (such as semantics or the definition of new language features to overcome the problems)."

In my paper, I didn't suggested that my proposal requires new language features. Contrary, the solution I proposed uses only the language constructs existent in the standard. I wouldn't say that a solution could not consider the definition of new language feature. However, I stated that, (even in this last case), a new modality of conceiving assertions is needed. In the last section, conclusion, I said:

"This **requires a different modality of conceiving OCL assertions. The design of assertions is centered around the information we need to obtain in case of failure. In case of the specification patterns that we have proposed, the correspondent existing specifications can be automatically transformed in the new proposed form.**"

For that reason, I would like to ask the reviewer to explain me his statement more clearly, if possible.

[Meyer97] - Bertrand Meyer - Object-Oriented Software Construction (2nd Edition) - Prentice Hall PTR, 1997, ISBN 0136291554