**OEG20 Open Math Exercise Format - Collaborative Design Document**

The goal of this document is to collaboratively design a specification for interactive math exercises, so that interactive math exercises can become more interoperable, can easily be maintained/improved and be better preserved to remain usable for future generations.

In order to achieve this, we will collaboratively try to:

1. create an overview of past and present initiatives that have tried to create an open format or specification for interactive math exercises
2. identify what we might learn from those initiatives
3. formulate design criteria that a good and usable specification or format should adhere to as well as what we should avoid
4. design collaboratively and constructively critique possible formats and specifications

**Scope & Assumptions**

- We assume a (non-mathematical) **open format for online exercises can be used as a basis**. For example QTI.
- The **scope and focus** of this action lab are the **mathematical specific aspects** of the open format. These are at least the following:
    - *generation of dynamic parameters including symbolic expressions* to support parameterised exercises. For example:
        - let a be a number in the range 1 to 5
        - let f be the formula f(x)=x^a
    - *definition of correctness of a mathematical answer*
        - the answer of the student should describe the same function as f (e.g. if a is 2, x*x should be seen as correct)

# 1. Literature review: past and present initiatives

*Please add to the list of past and present initiatives.*

**Summary**

Based on our first literature review it seems there are three Open Math Exercise Format initiatives: MyOpenMath, MathDox, and R/Exams. Note that the R/Exams does not natively support symbolic computations.
In addition there are several Mathematical Document Languages, such as MathML and OpenMath.
Finally, there are signs (link required) that QTI 3.0 will also (partly) support numerical/mathematical exercises.

**Open Math Exercise Format Initiatives**

- MyOpenMath
- MathDox
- R/Exams
- LibreText

**General Open Exercise Format Initiatives**

- QTI (3.0)

Mathematical Document Languages

- MathML
- OpenMath
- OMDoc

# 2. Lessons learned

*Please contribute pros and cons of the listed initiatives and what we might learn from them.*

*If you think there are general lessons to be learned, not tied to a specific initiative, please list those at the bottom of this section.*

**Open Math Exercise Format Initiatives**

- MyOpenMath
    - Pros
        - Clear documentation and tutorials (link, link,
        - Allows for parameterisation
        - Uses open-source software (IMathAS) as basis
    - Cons
        - parameters defined by code with a wide variety of very specific functions (example, link)
        - nested functions in parameter definition making it harder to display in an easy to use interface (e.g. “$answer = normalcdf(round($z,2)”)
        - text in parameters, making translations to multiple languages more cumbersome
- MathDox
    - Pros:
        - Definition of parameters and comparison defined in document instead of using CAS functions directly
        - Documentation available (link)
    - Cons:
        - Parts of documentation (temporarily?) offline (link)
        - relation between defined parameters and display indirect (based on order?)
- RExams
    - Pros
        - clear documentation (link)
        - allows parameter definition for randomization
        - uses open-source software (R) as basis
        - Can be used in many  and electronic learning system (link). ated by adher
    - Cons

        - only supports numerical questions natively (“Exercise types include multiple-choice or single-choice questions, numeric or text answers, or combinations of these.”)
        - focused on generation not on interactivity and automatic grading of mathematical answers
        - parameters defined by R code (see example)

**Mathematical Document Languages**
These languages could be used to define mathematical expressions inside the open format. Please add any other languages which would be suitable to explore.

- MathML
    - Pros
        - actively developed/refreshed (link)
        - structured approach and supported as mathematical language by multiple CAS systems
    - Cons
        - not natively supported by modern browsers natively as was intended (link)
- OpenMath
    - Pros
        - Focusses on the semantic meaning of the mathematical expressions instead of presentation: “There is a strong relationship to the MathML recommendation from the Worldwide Web Consortium, and a large overlap between the two developer communities. MathML deals principally with the presentation of mathematical objects, while OpenMath is solely concerned with their semantic meaning or content.” (link)
    - Cons
        -
- OMDoc
    - Pros
        - extension on OpenMath to be used for defining mathematical expressions in a more broader sense
    - Cons
        - no longer maintained? (latest news in 2016, latest github commit on nov 2019)

**General Insights and Lessons Learned**

-

# 3. Design Criteria and What to Avoid

*The goals of this section:*

*a)* formulate design criteria that a good and usable specification or format should adhere *b)* list what we should avoid

*How you can contribute:*
*Please add to the design criteria and explain why that would be an important and useful criterion to judge possible designs on.*
*Also please add to the list of things to avoid. How could this work turn out to be completely useless or unfeasible?*

Design Criteria:

- -
- Human readable:
    - Why: the best way to ensure people will always be able to open and inspect the exercise, is if they can easily read the exercise without an interface.
    - Suggested by: Pim Bellinga
- Easy to understand:
    - Why: if the logic behind an exercise is easy to understand it is more likely that other people know what it tests, which makes it more likely that the exercise is adopted in other curricula.
    - Suggested by: Eric Bouwers
- Allow for an intuitive, visual editor-interface based on the format:
    - Why: not all content creators know how to code in a text-based format. To increase the community of content creators a visual editor (think scratch) is a plus. Note: this can also make it easier for non-programmers to tailor an exercise to their particular context.
    - Suggested by: Eric Bouwers
-
- Parameters should be be specified based on what you want, not on how it should be calculated
    - Why: this set-up makes it easier to understand the goal of the exercise. In addition, this makes the language less dependent on a specific implementation which improves the interoperability of the format.
      Suggested by: Thijs Gillebaart

What to Avoid:

- One OEG jury member: “Interoperability is also an issue per se... It reminds me of metadata... for years we have had discussions on the right format for metadata, and the result is not convincing. Why wouldn't it be the same problem here? Why not consider using parsers to be able to interoperate?”

# 4. Possible Designs and Critiques

Design Suggestion 1: specification that many formats can adhere to instead of one format

- If we look back at what to avoid, the ‘one format to rule them all’ seems to be the unachievable holy grail.

- The disadvantage of having to create ‘translators’ for each design however is that it can be brittle (what if the format changes), might require one to parse the full code, and requires many custom-built translators.
- A middle way could be a specification, that different formats can adhere to. As long as they are ‘specification compliant’ then one can be assured that one format can be translated into another format without complications.