Improvements wished for improving the usability:

* Testing framework (19)
  + means to specify inputs and expected outputs / Output model matching (4)
  + code coverage support (3)
  + framework that supports most of the different use cases (e.g., use as a library or application, use of different solvers like clingo or clingo-dl) (2)
  + Automated test case generation from an encoding that is supposed to be correct (regression testing) (2)
  + Unit tests (Tests for small subset of encoding (2)
  + I would love integration with pytest (in Python) and GitHub actions
  + It would be nice if there was an online collection of examples and recipes outside of pdf
  + Make it easy for the user and do not spend too much time configuring
  + Offer some way of defining a test suite combinatorially
  + providing graphs, plots or even parsed information to formats like LaTeX
  + Easy definition of resource limits per run (time and memory) and accumulation of solving statistics (average time, conflicts, constraints, etc.)
  + Command line usage
  + Customizable
  + Possible checks:
    - Check if satisfiable
    - Check if unsatisfiable
    - Check if no other predicates than a specific set is in solution
    - Check if answer contains certain predicates
    - Check for optimality / optimal value
    - Check if problem is solved within 5 seconds
    - Check for warnings
    - Check if all answer sets of 2 encodings coincide
    - Offer some way to check that a test was resolved deterministically (distinguishing "models: 1" from "models: 1+")
  + I don’t intend to use a test-framework specifically for ASP (1)
* Better debugging frameworks (16)
  + Explaining why models are unsatisfiable (6)
    - auto minimality check, given a set of (ground) integrity constraints, what is the minimal set of constraints that need to be excluded to be still satisfiable (5)
  + Having some debugger would be nice. For example, some ASP programs can be evaluated in certain order, just like an imperative program.
  + it was a pain to look into the produced models (“Pretty Print” that does not print the model but the values that makes the predicates true could help)
  + option to show the completion of predicates where possible
  + The second issue is that often particular terms are of interest, where it may be crucial to understand which related ground atoms are derived and how. Here I every now and then observe myself writing quite sophisticated #show statements to specify a small but complete set of output atoms carrying the relevant information.
  + Using clingo from the python API can lead to very unhelpful error messages, making debugging simple issues more difficult than necessary. The error messages could be improved.
* Language Server (8)
  + Automatic warnings about potential typos (variable, predicate, constant) ... Maybe with convention that ending with \_ silences the warning.
  + Inclusion of existing debugging approaches in newer IDEs (VSCode,…)
  + Definitely it would be very helpful to define a recommended development environment, and also debugging tools.
  + Fixers, Linters
  + Automatic satisfiability check
  + Add warnings about predicates being used in optimizations and #show, but nowhere else in any rules
  + automatic support for finding faulty rules
  + automatic detection of inefficient rules
  + Context aware syntax highlighting
  + Some kind of intellisense
* Better benchmark frameworks (4)
  + benchmarking optimization problems is still hard to compare faster/better solution (when a certain solution is found)
* Better Documentation of Clingo Systems (3):
  + the python bindings of clingo are a mess and mostly undocumented
  + the heuristics are difficult to understand from the docs, the "old" versions with --dom-pref still seem to coexist with the new ones…
  + #minimize{ 0@1: #true } trick not documented well
  + Easier examples in the beginning
* Better Evaluation (2)
  + Some standard output syntax for stable models like json or XML in order to do some postprocessing for visualization (e.g. in a timetable or on a grid).
* Better formatting options / Automatic formatting
* Better Optimization
* New Language Features:
  + Named variables (as opposed to ordered variables) for predicates and functions
* output of --text should be readable as input again
* Wrappers for Go
* Cleaner syntax and a high performance REPL
* A cleaner syntax and the inclusion of sorts.
* asprin seems deprecated and not maintained
* time tracking of when a solution is produced (to check in hindsight if the first 10 solutions where produced quickly, or where did it slow down producing solutions)
* Profiling of where the solver spends it’s time
* more expressive error messages
* feature similar to --timeout that stops solving if no progress in quality has been made for XX seconds
* stop search feature that stops if a solution is found in XX% of so far lower bound
* Our problem has a lot of optimization criteria, and it's currently a pain to extract them and show them well on the python side. You have to jump through some hoops to get all your optimization criteria to come out on every solve
* I wish there were better ways to handle arbitrary arity in ASP programs (we run into this a fair bit) but it hasn't been too cumbersome.
* Control and display of random seed in each solving process
* All in Text Mode, usable from Jenkins etc. Build. Maybe optional html Export like with nose2 coverage tools.
* Reimplement and actively maintain a variation of ansunit, preferably without any Python dependencies so that the "clingo" binary is enough to run the test.
* If we get more official uses of asp, we can establish it more in Industry.
* A few things that prolog has should be added to asp (e.g. importing libraries while hiding all but interface atoms).
* Better mac support
* I am not currently updated on the state of RosOClingo, but I really would love this development would continue and grow
* Interactive Clingo:
  + feed clingo with the full file I want to solve
  + clingo give me the first model found but do not stop execution, just wait for my commands
  + my tool read it, make some decision with it and found new facts,
  + I feed clingo with those new facts and ask him to find a new model
* Consider having a discourse forum. It will likely improve community engagement around helping others get started. I think mailing lists have a bit of a barrier to entry
* You Should add an API to wrapp Temporal ASP program in Telingo with Python. Like Clyngor