Minutes meeting 2016-04-19

TU coach:

- Fill in deadlines for the coming one/two weeks.
- Weekly meetings on monday 11.00.
- List of **all** deadlines in project plan.
- Project plan hand-in tomorrow morning.
- As detailed research questions for research report as possible; hand-in is also tomorrow morning.
- Send minutes of every meeting (client and/or TU coach) to Hendrik.
- For research report, compare notebook/playground with REPLs.
  - Maple had this as one of the first, but it was confusing (changes were not propagated to already evaluated expressions)
  - There is an online python notebook somewhere.

Client:

- We get our own repo in the MetaBorg GitHub organization.
- We work embedded at the TU Lab, monday to friday starting between 9:00 and 9.30.
- Sign document for the apache license, will go through Hendrik.
- Discussed features:
  - Commandline interface (shell) for any language defined in spoofax: it should work with expressions in that language. Perhaps limit this to only a subset of the expressions in the language.
  - Same features that editors have, e.g. syntax checked expressions, syntax highlighting, indications of errors, et cetera.
  - Integration with Eclipse (IntelliJ too?)
  - Features should be as generic as possible: as much should be in spoofax-core instead of in editor plugins.
  - Basically a mini (line-)editor for a single expression. Questions:
  - Support multiline, or hit return and evaluate?
  - A natural thing to do is to call the interpreter and print the value, but perhaps we should do something else, such as arbitrary number of commands to e.g. ask for the type of a variable (enable GDB style interaction/debugging).
  - Define command language, evaluator for that language.
  - Decide whether single expressions should be evaluated.
  - Evaluate expression in context of module: namebinding et cetera (in the editor); link to modules (interplay with interpreter).
  - The shell has a state that it should be able to save and restore (R-REPL has this).
  - Maybe: hover over variables to see their values, types, et cetera.
  - Commands should work between repl and other editor features (see autocad).
  - notebook/playground thing.
- Discussed Non-features:

    - None discussed so far.
- Discuss architecture with gabriel.
- Draw inspiration for features from existing REPLs and Spoofax's featureset, e.g. do all REPLs work in context of a project or otherwise? REPLs to study: Lisp REPL, Scheme REPL, Dr Racket REPL, Python REPL, Haskell REPL, R REPL, Autocad REPL.
- From the previous bullet point, the research report and the discussed features we will make a document outlining all features we think would deliver a viable product to the client. The client will then give their feedback on this list of features.