# Project summary

| | |
|---|---|
| Project name | Base Languages for MPS |
| Acronym | BLangs |
| Supervisor | Pavel Parízek (parizek@d3s.mff.cuni.cz) |
| Consultants | Václav Pech, JetBrains (pech@d3s.mff.cuni.cz) |
| Abstract | The goal of this project is to add support for two general-purpose programming languages, JavaScript and C#, into MPS – an open-source language workbench. For each language, participants have to implement its abstract syntax, editor, and text generator. As a result, the project should deliver implementation of the languages packaged as MPS plugins that will be distributed freely under an open-source license. |

## Motivation

JetBrains MPS (http://www.jetbrains.com/mps) is an open-source language workbench focusing on Domain-specific Languages (DSL). Code written in DSLs needs to be automatically transformed into one or more General-purpose languages (GPL), such as Java, JavaScript, C# or XML, before triggering the compiler of the target platform and generating binary code that is actually executed.

At the moment MPS supports only Java and XML as the target GPLs. Both languages have been implemented in MPS. Industry users have been rising interest in other GPLs being supported – JavaScript, Python and C# are among the most demanded. A few open-source projects have already been started (e.g., for JavaScript - https://github.com/mar9000/ecmascript4mps).

With the limited resources that JetBrains can put into open-source activities we would welcome efforts that would lead to generally available functional implementations of these GPLs.

# Project description

Unlike many competing language workbenches and in contrast to the industry prevalent approach, MPS uses a projectional (structured) editor for editing code. The developer directly manipulates the program in its tree (AST) form. The code is always represented as AST, including the persistence format, which avoids the need for parsing text.

This approach was chosen in order to give language authors:

- greater flexibility of the language syntaxes – tabular, graphical, textual or form-like notations are all possible
- ability to switch between notations on-the-fly and thus view the same code in different ways depending on the task at hands
- modularity of languages, which enables languages to be easily combined – extended, embeded, reused or referrenced from one another

Languages in MPS are defined using principles of object-oriented programming, instead of grammars.

Since MPS does not rely on parsing, but instead uses a projectional (structured) editor and layered code generation, existing GPLs that should be made editable and generatable in MPS need to be re-implemented using the MPS language definition facilities. Only then can these GPLs be used as targets for code generation in MPS.

The main goal of this project is to implement support for two additional GPLs, namely JavaScript and C#, into the MPS workbench, possibly enhancing existing prototypes or contributing to existing open-source projects. Participants should implement full support for one of these languages, and basic support for the other. The decision whether to focus primarily on JavaScript or C# will be made at the start of the project based on student preferences.

Basic support for a GPL requires at least definition of its structure (abstract syntax), simple editor (concrete syntax), and a module for conversion into text. More advanced features, such as smooth editing, type rules and data-flow, make usage of the GPL much easier. An implementation of each GPL will be a separate sub-project and will be packaged separately. The outputs of the project will be distributed as MPS language plugins under the open-source Apache 2 license and available for free download. All outputs including documentation will be in English.

# Expected effort

Number of participants: 5

Demands on participants:

- A good command of Java is preferred, since Java is used as the base for all language definition facilities in MPS.

Completion date: 9 months since the start

Main project tasks and overall schedule:

- Participants will have to learn how to use MPS and how to define languages in it. [1-2 months]
- Participants will have to familiarize with the respective target GPLs' syntaxes and study the prototypes, where they exist. [1 month]
- Set up the development infrastructure (VCS, CI), create automated build scripts and plan their efforts. [0.5 month]
- Add support for the target languages into MPS following their official grammar definitions. As indicated above, this will include implementation of the abstract syntax, editor, text generation module, and some of the advanced features such as type rules. [5 months]
- Package the languages as plugins for MPS, document the projects and enable future evolution of these projects. [1 month]

MPS is a stable tool evolved and supported by JetBrains that imposes no extra risks on the project. A consultant from JetBrains will be available to help the team members familiarize with MPS at the beginning of the project and get over difficulties in using the tool. In addition, regular meetings with

the consultant are planned for the whole duration of the project.

The existing prototype implementations of some GPLs may or may not provide good starting points for the project, and so will require careful investigation before making a decision whether to build upon them.

In general, the project is feasible for a 5-member team and a 9 month realization time span.

# Project characterization

*The project targets the following areas (mark suitable areas):*

| | |
|---|---|
| Discrete models and algorithms | |
| | discrete mathematics and algorithms |
| | geometry and mathematical structures in informatics |
| | optimization |
| Theoretical informatics | |
| | theoretical informatics |
| Software and data engineering | |
| x | software engineering |
| x | software development |
| x | web engineering |
| | databases |
| | big data analysis and processing |
| Software systems | |
| | system programming |
| | dependable systems |
| | high-performance systems |
| Mathematical linguistics | |
| | computer and formal linguistics |
| | statistical methods and machine learning in computer linguistics |
| Artificial intelligence | |
| | smart agents |
| | machine learning |
| | robotics |
| Computer graphics and game development | |
| | computer graphics |
| | game development |