Technische Universität Braunschweig

ISF

Lab on Model-Driven Software Development

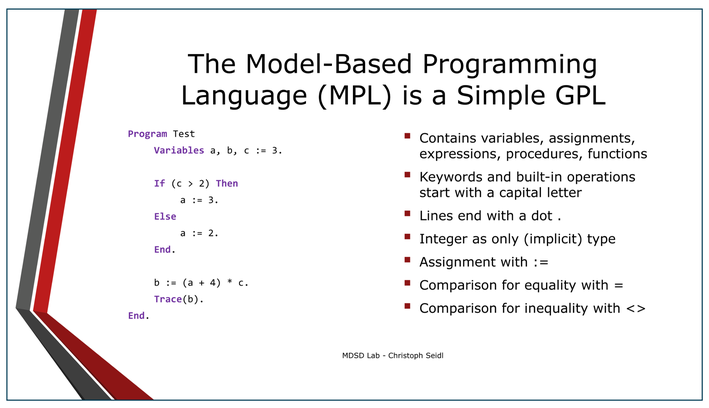
Creating a Model-Based Programming Language

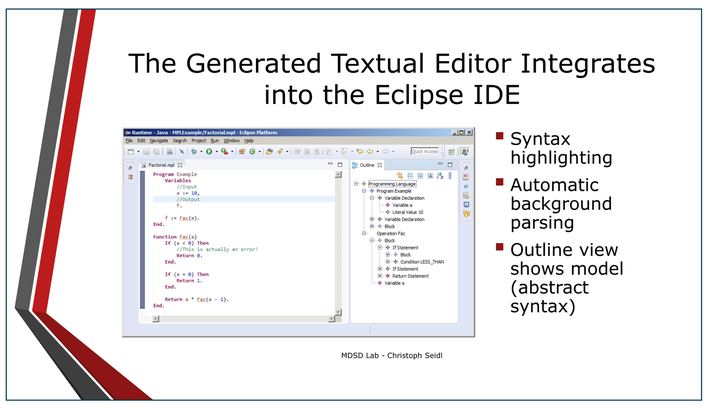
The lab guides students through the process of designing and realizing an own model-based programming language, writing a compiler to an intermediate language and creating an execution environment revolving around the language's interpreter.

Technische Universität Braunschweig | Institute of Software Engineering and Automotive Informatics
Dr.-Ing. Christoph Seidl | c.seidl@tu-braunschweig.de | Phone +49 (0) 531 391-2296

1. Model-Based Programming Language (MPL)

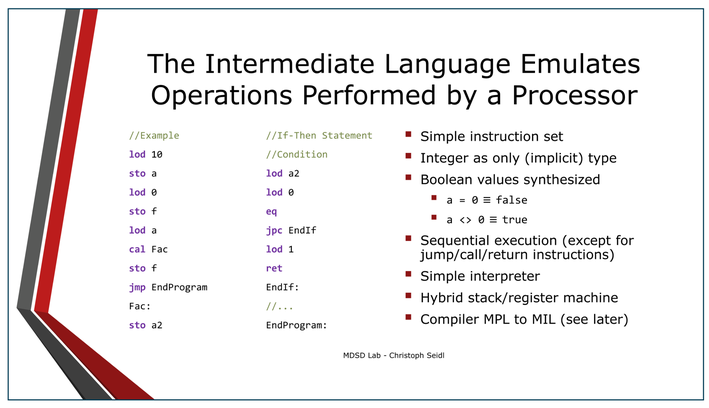
MPL is a Turing complete programming language with variables, calculations, control flow statements and operations. MPL is developed in four revisions with increasing complexity, where the first one is developed in a class-room tutorial and the other three in homework assignments. The resulting metamodel and textual syntax are used to generate an editor.





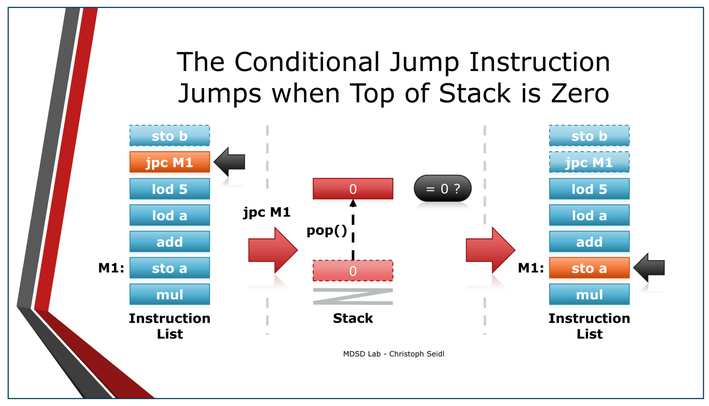
2. Model-Based Intermediate Language (MIL)

MIL features an instruction set similar to that of processors with a total of 19 instructions, which are easy to execute. MIL is developed in four revisions that align with those of MPL. In class-room tutorials and homework assignments, students design a metamodel and textual syntax for MIL while learning the semantics of each instruction.

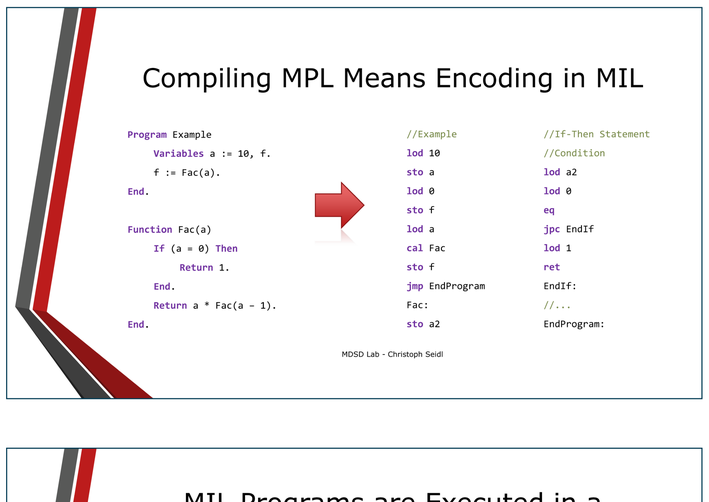


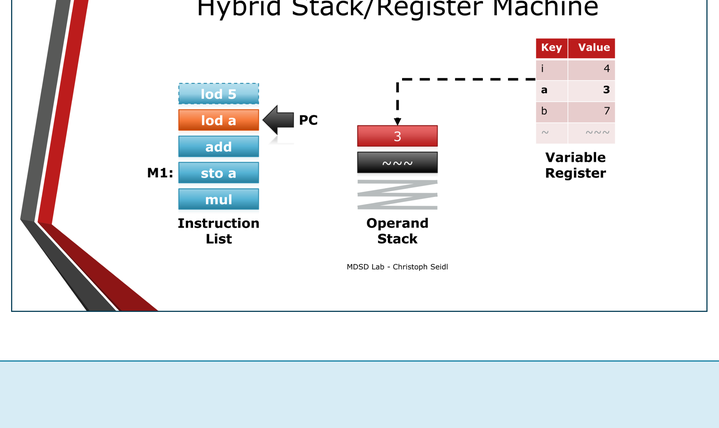
3. Execution Environment

Instead of executing MPL directly, it is compiled to MIL by model-to-model transformation. Students devise and implement appropriate translations from MPL constructs to MIL instructions. The execution environment for MIL is a hybrid stack/register machine students implement in Java. The implementation is relatively straight forward, which makes it portable so that MIL instructions and, thus, MPL code can be executed on different platforms when necessary.



Metamodeling
Constraints
Textual Syntaxes
Model Transformation
Model Interpreter
IDE Integration





4. Extensions to Language Infrastructure

Students follow up on the guided tasks in free exercises where they can choose from a number of potential topics to extend and refine the language infrastructure, e.g., introducing a type system to MPL, creating a graphical programming language that compiles to MIL, writing a debugger for MIL or compiling MIL to a binary format with an interpreter for embedded devices. To complete the lab, students have to realize at least one free exercise topic.