Software Language Engineering

Language Characteristics
(“Steckbrief”) for
SimLang (SimulationLanguage)

Deniz Schmidt
Software Engineering
RWTH Aachen University

http://www.se-rwth.de/

Language/Tool at a Glance

Name:	SimLang (+ Weather)
Developed by:	Schmidt, Deniz
Based on:	-

- Purpose of the language / tool:
  - language itself describes (vehicle) simulation scenarios consisting of different settings and entities
  - the project's pipeline takes a model in form of a .sim file and finally provides a container which holds all the model's data in java
  - MontiSim uses model-data acquired through this project

Technical Briefing

- Can be found in: github:
  - https://github.com/MontiSim/SimulationLanguage

- Open accessible: Yes
- MC version: 4.5.3 (on April, 10th, 18)
- Uses: NumberUnit fork, Weather sub-language, MontiSim's simulation-environment for adapter

- Current state:
  - SimLang + Weather grammar, CoCos
  - SimLang symbol table creation
  - SimLangContainer
  - SimLang -> MontiSim adapter

Language Details (1: Syntax)

- Grammars: SimLang.mc4, Weather.mc4

- #of Nonterminals: 34 (SimLang), 16 (Weather)
- state of grammar: stable

- Most interesting nonterminals: AlternativeInput, Simulation, Weather, Channel

- Comments:
  - Disregarding artifacts, the max. current scope depth is 2
  - Attributes with unique input-formats require unique modeling of alternative inputs

Language Details (2: CoCo's)

- #of coco's: 18 (SimLang), 8 (Weather)
- state of coco: stable
- Cocos explained here: Model-driven development of configurable vehicle simulations (Bachelorthesis)
- Cocos implemented here:
  de.monticore.lang.montisim.simlang.cocos (package)
  de.monticore.lang.montisim.weather.cocos (package)
- Comments:
  - cocos still need to get reviewed regarding completeness and correctness

Language Details (3: Symbols)

- #of Symbol-creating nonterminals: 25 (SimLang), 12 (Weather)
- #of Symbol-Kinds: 25+12
- List of nonterminals creating symbols: / Symbol-Kind:

List of nonterminals creating symbols:	Symbol-Kind:
SimulationDuration	SimulationDurationKind
Weather	WeatherKind
ExplicitVehicle	ExplicitVehicleKind

- Comments:
  - Every simulation attribute (settings + entities) receives it's own symbol, kind and resolving filter.

Language Details (4: Scopes)

- #of Scope-creating nonterminals: 3 (SimLang), 1 (Weather)
- #of Scopes: 4
- List of nonterminals with scopes:

List of nonterminals with scopes:	Their function:
SimLangCompilationUnit	artifact scope
Simulation	holds simulation attributes
Channel	holds channel settings
WeatherScope	holds weather attributes

- Comments:
  - Simulation, Channel and WeatherScope are basically Kleene-Closures of interfaced rules.

```
-> symbol scope Simulation = "sim" Name "{"
   ((SimulationSetting | SimulationEntity) ";")*
   "}";
```

Backend: Functionality

- What functions are offered?
  - SimLangTool:
    - SimLangLang SIMLANG_LANGUAGE = new SimLangLang();

    Static reference to a language instance.
    - void main(String[] args)

    Expects only a filepath to a .sim model. Creates a ST and performs CoCo checks.
    - ASTSimLangCompilationUnit parse(String model) –

    Attempts to parse a given file
    - void checkDefaultCoCos(ASTSimLangCompilationUnit ast)

    Performs CoCo-checks on an AST
    - Scope createSymbolTable(SimLangLang lang, ASTSimLangCompilationUnit ast)

    Creates a ST from a given AST
    - SimLangContainer parseIntoContainer(String model) –

    Runs a given model through the project pipeline

Backend: Functionality

- What functions are offered?
  - SimLangContainer
    - Recommended instantiation using the SimLangTool function.
    - Resolves all attributes from the ST and places them inside private Optionals
    - If attribute was defined in the model -> Optional is filled
    - Access Optionals via getters
      - Eg.: Optional<String> getMapName()

    - Recommended usage:
      - Make an adapter-class inherit or use this class
      - Access (+transform?) the model attributes in SimLang data formats
      - hand them to the next processor.

Plans for Tools and Language

- Future of the language/tool?
  - Extend attributes if needed
  - Clean-up ST infrastructure, possibly generate kinds and resolving filters
  - Create GUI for modeling
  - Aggregate with CarLang