ACTUS-MP

Meta-Programming Framework

Mark Greenslade June ‘23

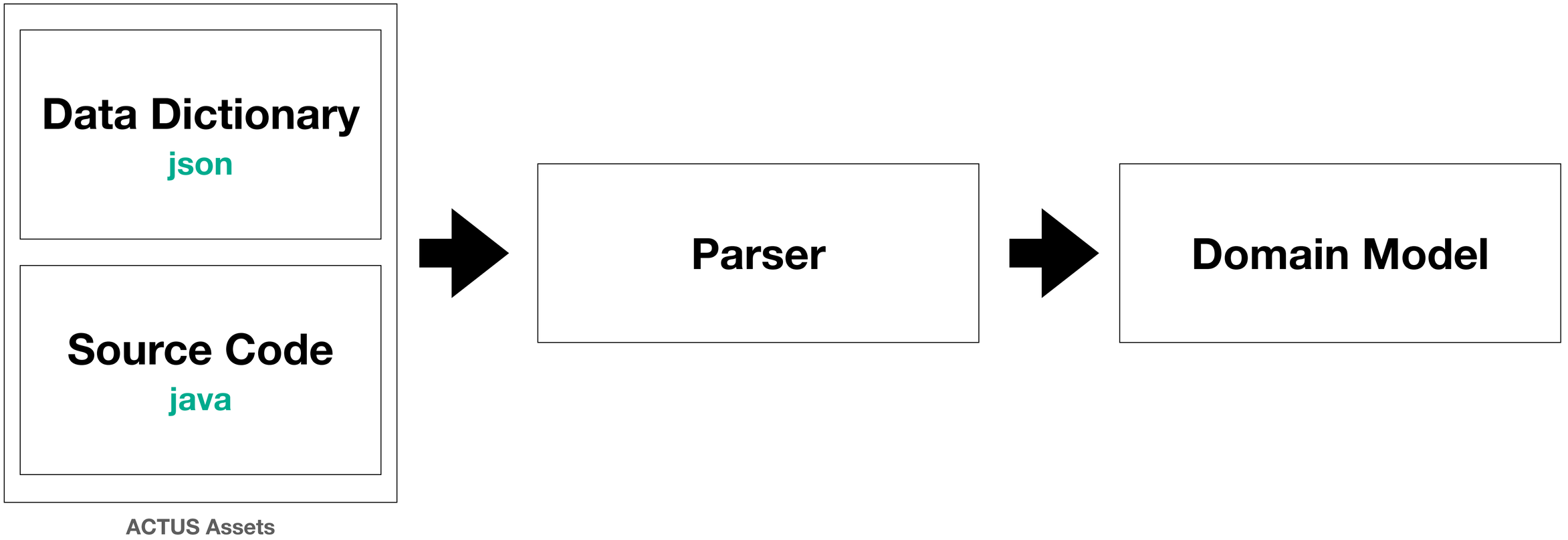
Problem Space

- A single reference implementation is limiting
- Java is top ten language - others need to be supported
- Coding alternative implementations is error prone
- Applicable termsets -> lots of boilerplate
- Aim = initialise an idiomatic implementation per target language
- Leverage ACTUS dictionary plus reference Java implementation

Solution Space

- Step 1: Parse ACTUS dictionary & Java source code
- Step 2: Instantiate domain model from parsed output
- Step 3: Execute language specific code generators
  - Input = domain model instance
  - Output = code artefacts
- Step 4: Move generated code artefacts into target libraries

Parsing Assets



Domain Model

Dictionary

Taxonomy

Contract

Term

Term Set

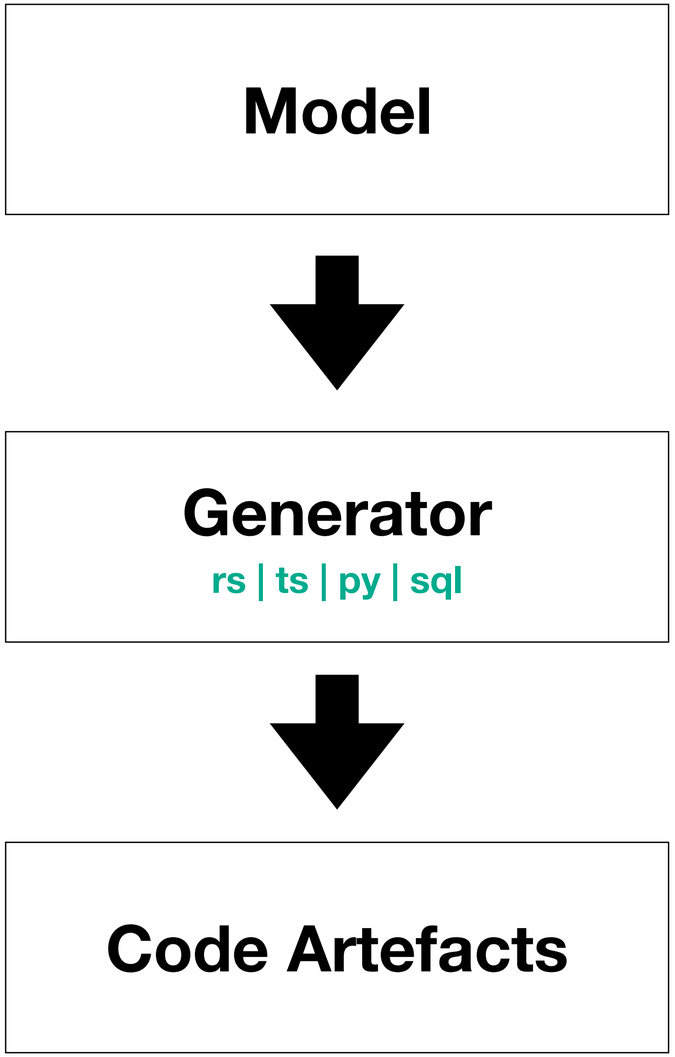
Applicability

Enum

Scalar Type

Function Type

Code Generation



- Domain model instance is passed into code generator
- Language specific jinja2 templates are loaded into memory
- Engine hands off domain model to each template
- One code artefact per template
- Generated dode artefacts are handed to writer
- Writer persists artefacts to file system

How To Run

```
import argparse
import pathlib
import actusmp

# CLI argument parser.
_ARGS = argparse.ArgumentParser("Writes code generated from ACTUS dictionary to file
system.")

# Set CLI argument: target programming language.
_ARGS.add_argument(
    "--lang",
    choices=[i for i in actusmp.TargetLanguage],
    dest="lang",
    help="Target programming language status.",
    type=lambda x: actusmp.TargetLanguage[x]
    )

# Set CLI argument: output directory.
_ARGS.add_argument(
    "--dest",
    dest="dest",
    help="Target file system directory into which code will be written.",
    type=pathlib.Path
    )

# Set CLI argument: path to JAVA reference implementation repo (actus-core).
_ARGS.add_argument(
    "--core",
    dest="path_to_core",
    help="path to JAVA reference implementation repo (actus-core).",
    type=pathlib.Path
    )


def _main(args: argparse.Namespace):
    """Main entry point.

    :param args: Parsed command line arguments.

    """
    actusmp.write(args.lang, args.dest, args.path_to_core)


# Entry point.
if __name__ == "__main__":
    _main(_ARGS.parse_args())
```

- **Step 0**
  - **Clone repos**
    - **actus-core**
    - **actus-mp**
- **Step 1**
  - **Prepare python environment**
- **Step 3**
  - **Invoke generator**

Seeded Libraries

actus-core-rs

actus-core-ts

actus-core-py

actus-sql (wip)

Next Steps

- Hand off seeded libraries to implementors
- Each library to use actus-tests
- Once library tests succeed then notify ACTUS foundation
- ACTUS foundation to review library & certify
- ACTUS certification types: deprecated | supported | reference
- Certification to also depend upon documentation !

ACTUS-MP

Meta-Programming Framework

Mark Greenslade June ‘23