What’s software language engineering?

Ralf Lämmel
Software Languages Team
University of Koblenz
http://www.softlang.org/
What’s **software language engineering**?
What’s a **software language**?
How to **engineer** such languages?
A finite state machine for a revolving door

(Concrete visual syntax)

The FSM also identifies possible transitions between states triggered by “events,” possibly causing “actions”; see the edges in the visual notation.

These are these states in the turnstile FSM:

• **locked**: The turnstile is locked. No passenger is allowed to pass.
• **unlocked**: The turnstile is unlocked. A passenger may pass.
• **exception**: A problem has occurred and metro personnel need to intervene.

There are input symbols which correspond to the events that a user or the environment may trigger. There are output symbols which correspond to the actions that the state machine should perform upon a transition. These are some of the events and actions of the turnstile FSM:

• **Event ticket**: A passenger enters a ticket into the card reader.
• **Event pass**: A passenger passes through the turnstile, as noticed by a sensor.
• **Action collect**: The ticket is collected by the card reader.
• **Action alarm**: An alarm is turned on, thereby requesting metro personnel.

The meanings of the various transitions should be clear. Consider, for example, the transition from the source state “locked” to the target state “unlocked”, which is annotated by “ticket/collect” to mean that the transition is triggered by entering a ticket and the transition causes ticket collection to happen.

FSML is a domain-specific modeling language (DSML). FSML supports state-based modeling of systems. The specification can be executed to simulate possible behaviors of a turnstile. The specification could also be used to generate a code skeleton for controlling an actual turnstile, as part of an actual metro system. FSML is a trivial language that can be used to discuss basic aspects of domain-specific language definition and implementation. For what it matters, languages for state-based behavior are widely established in software and systems engineering. For instance, the established modeling language UML consists, in fact, of several modeling languages; UML's state machine diagrams are more general than FSML. We will discuss FSML in detail in Chapter 2.

A DSL instead of a “general purpose” PL
A finite state machine for a revolving door

(Concrete textual syntax)

```plaintext
initial state locked {
  ticket/collect -> unlocked;
  pass/alarm    -> exception;
}
state unlocked {
  ticket/eject;
  pass   -> locked;
}
state exception {
  ticket/eject;
  pass;
  mute;
  release   -> locked;
}
```

A DSL instead of a “general purpose” PL
A finite state machine for a revolving door

(Abstract JSON-based syntax)

```json
"exception": [[]
    "initial": false,
    "transitions": {
        "release": [[null, "locked"]],
        "ticket": [["eject", "exception"]],
        "mute": [[null, "exception"]],
        "pass": [[null, "exception"]]
    }
"locked": [[]
    "initial": true,
    "transitions": {
        "ticket": [["collect", "unlocked"]],
        "pass": [["alarm", "exception"]]
    }
"unlocked": [[]
    "initial": false,
    "transitions": {
        "ticket": [["eject", "unlocked"]],
        "pass": [[null, "locked"]]
    }
```
Some real world software languages

- Haskell: The functional programming language Haskell
- Java: The Java programming language
- Python: The dynamic programming language Python
- ANTLR: The grammar notation of the ANTLR technology
- JSON: The JavaScript Object Notation
- JSON Schema: The JSON Schema language
- XML: Extensible Markup Language
- XSD: XML Schema Definition
- Alloy: The Alloy specification language
- CIL: Bytecode of .NET's CLR
- Common Log Format: The NCSA Common log format
- DocBook: The DocBook semantic markup language for documentation
- FOAF: The friend of a friend ontology
- INI file: The INI file format
- Java bytecode: Bytecode of the JVM
- make: The make tool and its language
- OWL: Web Ontology Language
- Prolog: The logic programming language Prolog
- QTFF: QuickTime File Format
- RDF: Resource Description Framework
- RDFS: RDF Schema
- Scala: The functional OO programming language Scala
- Smalltalk: The OO reflective programming language Smalltalk
- SPARQL: SPARQL Protocol and RDF Query Language
- UML: Unified Modeling Language
- XPath: The XML path language for querying
- XSLT: Extensible Stylesheet Language Transformations
Some types of software languages

- data-modeling languages;
- markup languages;
- programming languages;
- specification languages;
- stylesheet languages;
- transformation languages.
# Language classification by purpose

<table>
<thead>
<tr>
<th>Purpose (language)</th>
<th>Purpose (element)</th>
<th>Classifier</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>Program</td>
<td>Programming language</td>
<td>Java</td>
</tr>
<tr>
<td>Querying</td>
<td>Query</td>
<td>Query language</td>
<td>XPath</td>
</tr>
<tr>
<td>Transformation</td>
<td>Transformation</td>
<td>Transformation language</td>
<td>XSLT</td>
</tr>
<tr>
<td>Modeling</td>
<td>Model</td>
<td>Modeling language</td>
<td>UML</td>
</tr>
<tr>
<td>Specification</td>
<td>Specification</td>
<td>Specification language</td>
<td>Alloy</td>
</tr>
<tr>
<td>Data representation</td>
<td>Data</td>
<td>Data format</td>
<td>QTFF (QuickTime file format)</td>
</tr>
<tr>
<td>Documentation</td>
<td>Documentation</td>
<td>Documentation language</td>
<td>DocBook</td>
</tr>
<tr>
<td>Configuration</td>
<td>Configuration</td>
<td>Configuration language</td>
<td>INI file</td>
</tr>
<tr>
<td>Logging</td>
<td>Log</td>
<td>Log format</td>
<td>Common Log Format</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The lifecycle of a software language

**Fig.** — The lifecycle of a software language. The nodes denote phases of the lifecycle. The edges denote transitions between these phases. The lifecycle starts with *domain analysis*. The lifecycle ends (theoretically) with *language retirement*. We may enter cycles owing to language evolution.
Typical software language implementations

- Compiler
- Interpreter
- Language processors
  - Parsers / unparsers / preprocessors
  - Software transformations
  - Software analyses
  - Software generators
  - Software translators
  - Software specializers
  - …
Fig. — Simplified data flow in a compiler. The rectangles with rounded edges represent logical phases of compilation. The remaining nodes (rectangles and triangles) correspond to input and output, as expressed by the direction of the arrows.
Example of re-factoring (re-engineering)

```java
void printOwing(double amount) {
    printBanner();
    System.out.println("name: " + name);
    System.out.println("amount: " + amount);
}

void printOwing(double amount) {
    printBanner();
    printDetails(amount);
}

void printDetails(double amount) {
    System.out.println("name: " + name);
    System.out.println("amount: " + amount);
}
```
**Fig.** Overall data flow for a re-engineering transformation. We have marked the phase which replaces code generation in the standard data flow for compilation.
Example of analysis (reverse engineering)

**Fig.** — An API-usage map for an open-source Java project. The complete rectangle (in terms of its size) models the references to all APIs made by all developers. The nested rectangles partition references by domain (e.g., GUI rather than Swing or AWT). The rectangles nested further partition references by API; one color is used per API. Within each such rectangle, the contributions of distinct developers (1, …, 8 for the top-eight committers and “R” for the rest) are shown. Source: [4].
Data flow for reverse engineering

Fig. — Overall data flow for fact extraction in reverse engineering. We have marked the phase which replaces code generation in the standard data flow for compilation. The resulting tables represent “projections” of the source code, for example, call relationships between functions.
Technological spaces

Grammarware  string, grammar, parsing, CST, AST, term, rewriting, ...
XMLware     XML, XML infoset, DOM, DTD, XML Schema, XPath, XQuery, XSLT, ...
JSONware    JSON, JSON Schema, ...
Modelware   UML, MOF, EMF, class diagram, modeling, metamodeling, model transformation, MDE, ...
SQLware     table, SQL, relational model, relational algebra, ...
RDFware     resource, triple, Linked Data, WWW, RDF, RDFS, OWL, SPARQL, ...
Objectware  objects, object graphs, object models, state, behavior, ...
Javaware    Java, Java bytecode, JVM, Eclipse, JUnit, ...
Technological spaces

Fig. — A few technological spaces with their instance (data) level and the schema level. The thin arrows model “conformance” such as an XML document conforming to an XML schema. The thick arrows hint at expected “bridges” between the spaces as needed for technological space travel. Inspired by [161].
Online resources

YAS (Yet Another SLR (Software Language Repository))
http://www.softlang.org/yas
YAS’ GitHub repository contains all code.

The Software Languages Book
http://www.softlang.org/book