Language Workbenches

The Killer-App for Domain Specific Languages?

An Article by Martin Fowler (2005)

Presented by Max Van Houcke

What we'll see

- Language Oriented Programming
- External vs Internal Domain Specific Languages
- Language Workbenches
- Example: Jetbrains MPS
- Conclusion

Language Oriented Programming

Language Oriented Programming

- Building software around a set of Domain Specific Languages (DSL)
- Object Reader Example

```
#12345678901234567890123456789012345678901234567890123456789012345678901234567890
SVCLFOWLER          10101MS0120050313.........................
SVCLHOHPE           10201DX0320050315.........................
SVCLTWO            x10301MRP220050329.............................
USGE10301TWO           x50214..7050329.............................
```


Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Object Reader DSL

- C#

```
ReaderStrategy result = new ReaderStrategy("USGE", typeof (Usage));
result.AddFieldExtractor(4, 8, "CustomerID");
result.AddFieldExtractor(9, 22, "CustomerName");
result.AddFieldExtractor(30, 30, "Cycle");
result.AddFieldExtractor(31, 36, "ReadDate");
return result;
```

- XML

```
<Mapping Code = "USGE" TargetClass = "dsl.Usage">
  <Field name = "CustomerID" start = "4" end = "8"/>
  <Field name = "CustomerName" start = "9" end = "22"/>
  <Field name = "Cycle" start = "30" end = "30"/>
  <Field name = "ReadDate" start = "31" end = "36"/>
</Mapping>
```


Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Object Reader DSL

- Custom DSL

```
mapping  USGE dsl.Usage
  4-8 : CustomerID
  9-22: CustomerName
  30-30: Cycle
  31-36: ReadDate
```

- Ruby

```
mapping('USGE', Usage) do
        extract 9..22, 'customer_name'
        extract 4..8, 'customer_ID'
        extract 30..30, 'cycle'
        extract 31..36, 'read_date'
end
```


Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

DSL Styles

- Unix Little Languages (e.g. awk)
- Lisp, Smalltalk, Ruby,..
- XML Config Files
- Active Data Models, Adaptive Object Models, GUI Builders,..

Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

External DSL

- Fully custom, but you have to build a parser/translator
- No symbolic integration
- No editor, lack of debugging
- Too many languages?

```
mapping  USGE dsl.Usage
  4-8 : CustomerID
  9-22: CustomerName
  30-30: Cycle
  31-36: ReadDate
```


Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Internal DSL

- Limited in syntax and structure of host language (very restricted in C type languages)
- Symbolic integration
- Full power of host language, including tools
- Too much power? Too many tools?

```
mapping('USGE', Usage) do
        extract 9..22, 'customer_name'
        extract 4..8, 'customer_ID'
        extract 30..30, 'cycle'
        extract 31..36, 'read_date'
end
```


Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Goals and Trade-off

- Increasing programmer productivity
- Involving domain experts and users, potentially writing all domain logic (although COBOL-inference looms)
- External DSLs have the most potential, but come with a cost of designing and building a parser and tool-set
- Internal DSLs can reduce cost

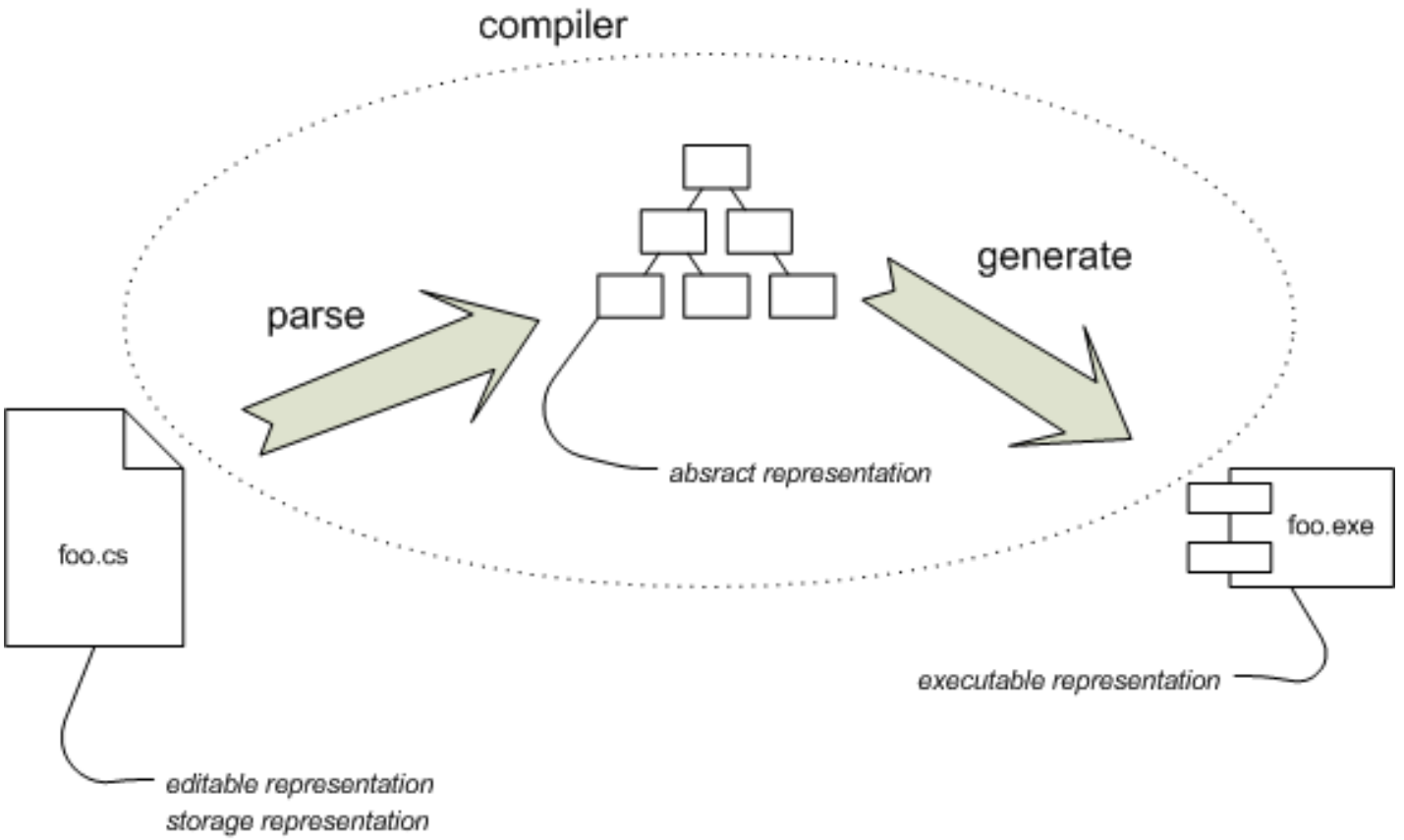
Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Language Workbenches

Landscape

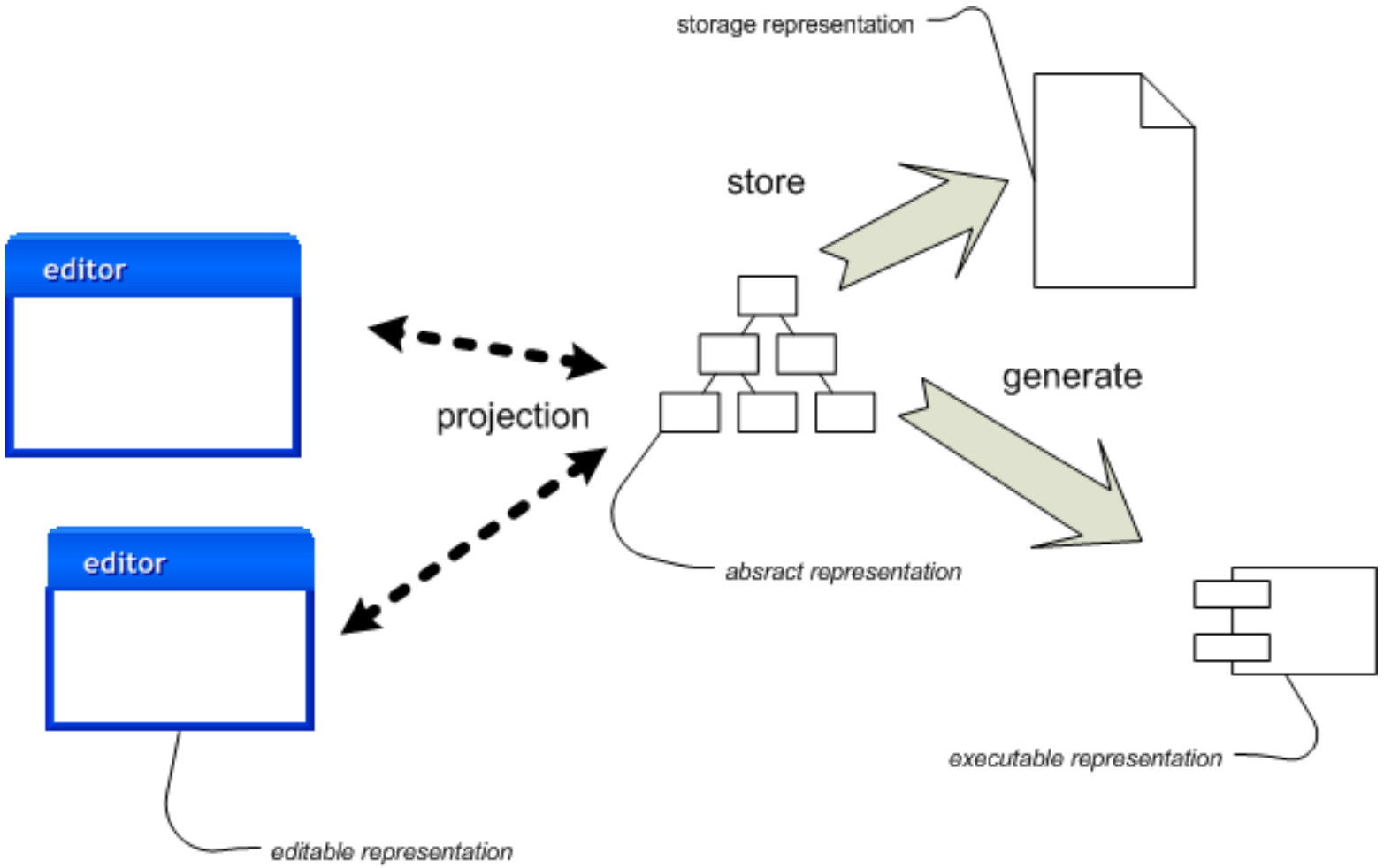
- Founder Charles Simonyi with Intentional Software (2002), acquired by Microsoft in 2017, integrated in Visual Studio
- Eclipse Xtext
- Jetbrains Meta Programming System (MPS)

Classic Compiler



Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Language Workbench



Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

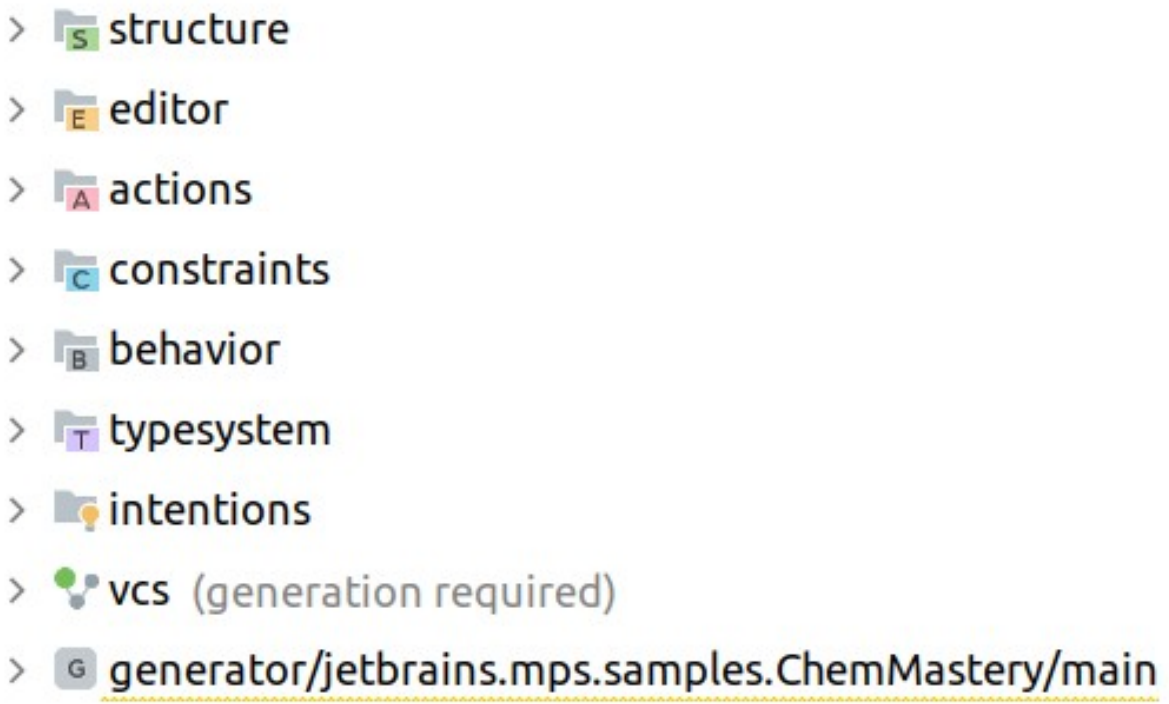
Additional Challenges

- Refactoring, evolving a DSL concurrently with its use
- Version control
- Vendor lock-in

Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Example: Jetbrains MPS

Jetbrains MPS



Structure (Abstract Syntax)

```
concept Field extends     BaseConcept
              implements INamedConcept

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  start : integer
  end   : integer

  children:
  << ... >>

  references:
  << ... >>
```

```
concept Mapping extends     BaseConcept
                implements INamedConcept

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  code        : string
  targetClass : string

  children:
  field : Field[0..n]

  references:
  << ... >>
```

Editor (Concrete Syntax)

```
<default> editor for concept Mapping
  node cell layout:
    [/
      [> mapping { code } { targetClass } <]
      [>      (/ % field %                                  /) <]
              /empty cell: press <Ctrl>-<Enter> to add field
    /]

  inspected cell layout:
    <choose cell model>
```

```
Style:
<no base style> {
  selectable : false
  background-color : none
}
```

Editor (Concrete Syntax)

```
reader configuration Config1

     mapping SVCL jetbrains.mps.samples.readerConfigLanguage.dsl.ServiceCall
          4  - 18 : CustomerName
          19 - 23 : CustomerID
          24 - 27 : CallTypeCode
          28 - 35 : DateOfCallString
     -------
     mapping USGE jetbrains.mps.samples.readerConfigLanguage.dsl.Usage
          4  - 8     : CustomerID
          9  - 22    : CustomerName
          23 - ?end? : <no name>
```

Generator (Transformations)

```
root mapping rules:
  concept          ReaderConfiguration --> readerConfigClass : class_ReaderConfiguration
  inheritors       false
  condition        <always>
  keep input root default
```

```
root template
input ReaderConfiguration

public class $class_ReaderConfiguration$ {
  public class_ReaderConfiguration() {
    <no statements>
  }
  public void Configure(Reader target) {
    $LOOP$target.AddStrategy(this._configure_mapping_());$
  }
  $LOOP$public ReaderStrategy $_configure_mapping_$() {
          ReaderStrategy result = new ReaderStrategy("$_CODE_$", ->$Usage$.class);
          $LOOP$result.AddFieldExtractor($0$, $0$, "$_field_name_$");$
          return result;
        }$
}
```

Generator (Transformations)

```
root mapping rules:
  concept          ReaderConfiguration --> readerConfigClass : _reader_configuration_
  inheritors       false
  condition        <always>
  keep input root default
```

```
root template
input ReaderConfiguration

xml $[_reader_configuration_].xml

<no prolog>
<ReaderConfiguration>
  $LOOP$ <Mapping Code="$[__CODE__]" TargetClass="$[_target_class_]">
           $LOOP$[<Field name="$[__field_name__]" start="$[0]" end="$[0]"/> ]
         </Mapping>
</ReaderConfiguration>
```

Conclusion

- Domain Specific Languages can offer an enormous increase in productivity and language workbenches make the process of creating them much easier
- Possibilities are endless, just have to be realized
- Workbenches have to overcome a lot of challenges and still have a lot of room for improvement (such as standardization)

Martin Fowler. Language Workbenches: The Killer-App for Domain Specific Languages? martinfowler.com. 2005

Thank you!