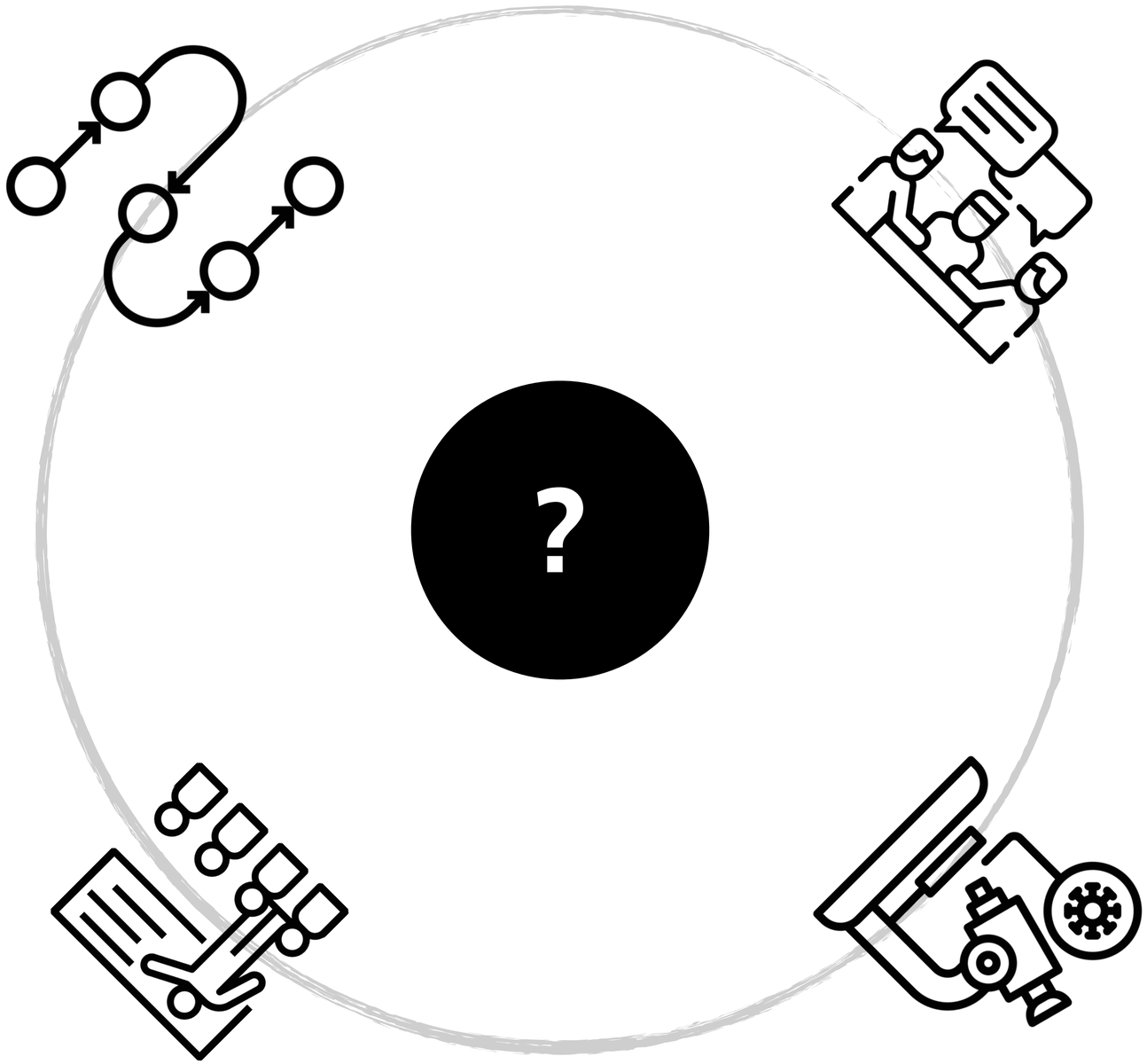
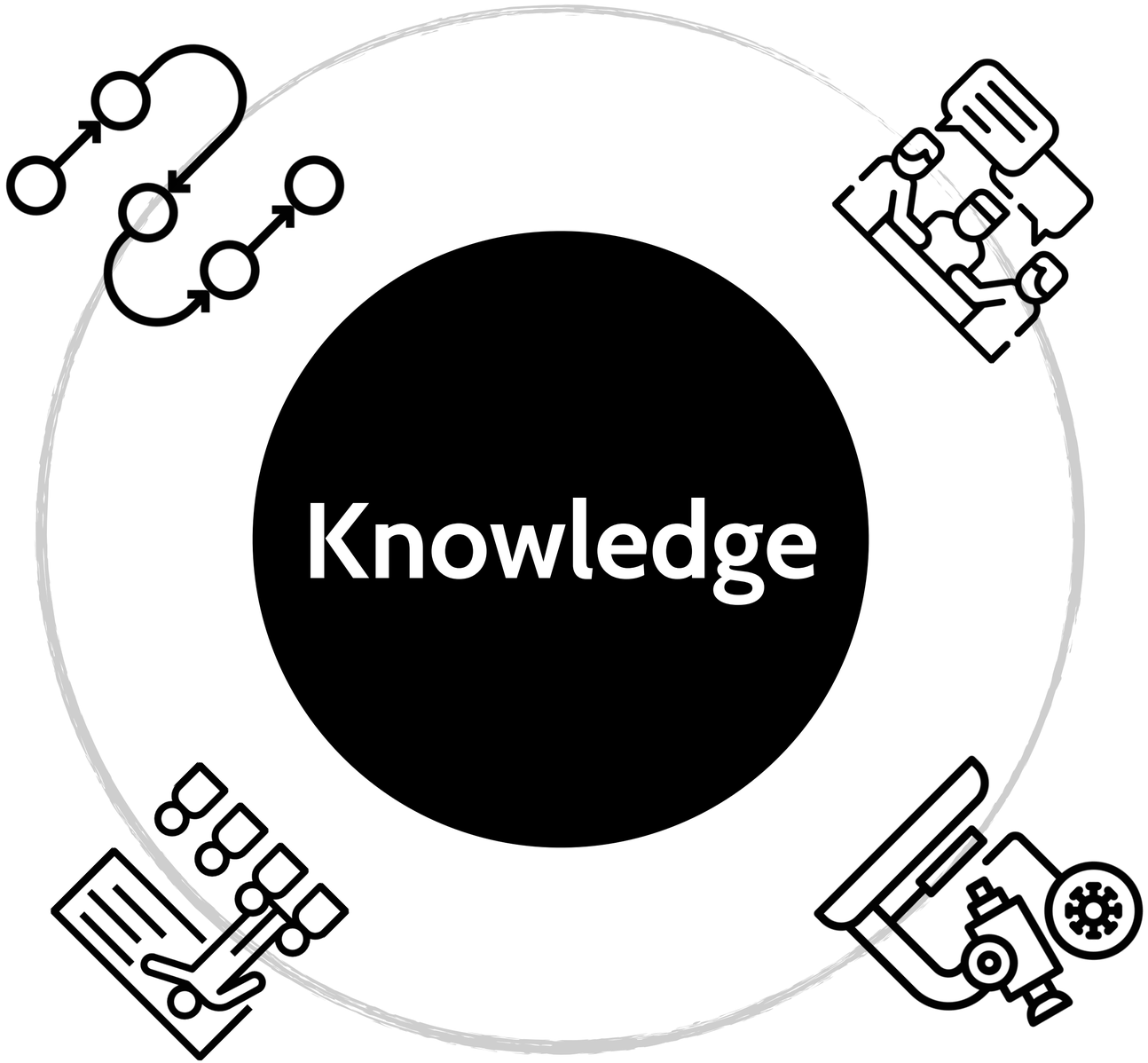
MPS Interoperability: how to put MPS at the center of an ecosystem

You will get the slides at the end!

Federico Tomassetti



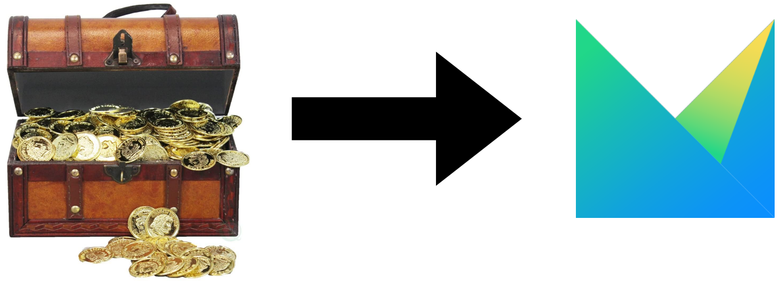

Knowledge

What we do with MPS?

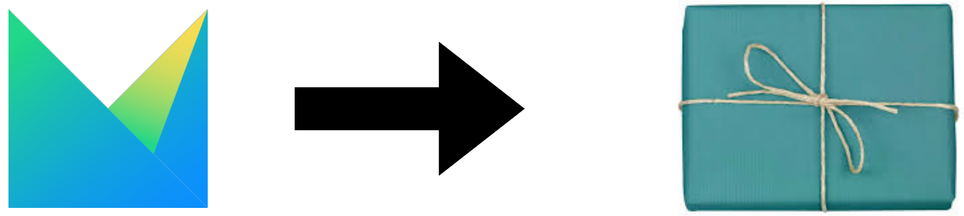
We pour what we know into MPS Models

and... ?

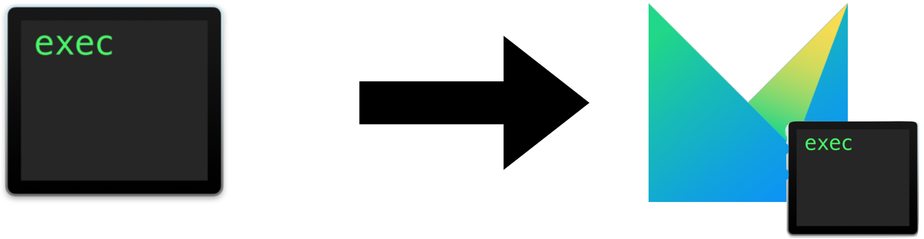
How would MPS interoperability would look like?



Bring existing knowledge into MPS



Generate stuff from knowledge to use it outside MPS



Make other tools working in MPS from within

Make other tools working with MPS from outside

Bringing stuff in: how?

One-time import or continuous import?

1. One-time import: we import and then **throw away original files and editors**

2. Continuous import: we keep data in the **original format**, and optionally keep also existing editors

Why bringing stuff in?

1. If all the knowledge is in one place we can **verify overall consistency**

2. We can replace other editors, which were not as, with editors in MPS, which we can make better at **guaranteeing consistency and we can evolve**, instead of using tools controlled by vendors

Bringing stuff in and edit it

If you want to ***evolve*** inside MPS stuff that stays outside you can do that:

1. Using custom persistence (https://www.jetbrains.com/help/mps/custom-persistence-cookbook.html)

2.Synchronizing a local copy with some external source (Modelix does that)

Bringing stuff in

Which format?

1. Common interchange formats: XML, JSON, Excel

2. Textual formats

3. Binary formats

Bringing stuff in: Common formats

There are libraries

For XML and JSON there is also an MPS Language

Bringing stuff in: Textual formats

https://github.com/julianthome/inmemantlr	Permits to instantiate parsers without having to generate java classes on disk
https://github.com/antlr/grammars-v4	Tons of grammars ready to use

Bringing stuff in: Binary formats

PNG

```
Binary format PNG

signature : unsigned int 8
  constraints: = bytes 137-80-78-71-13-10-26-10
chunks                                          : Chunk[unlimited]
```

Chunk

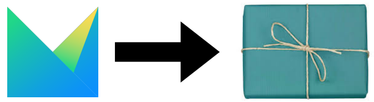
```
Polymorphic structure Chunk

fields:
  field length : unsigned int 4
  tag string ASCII 4
  payload length
  field CRC : unsigned int 4

payload alternatives:
  "IHDR" -> ihdr IHDRImageHeader                  nEntries=1
  "PLTE" -> plte PLTEPalette                      nEntries=1
  "IDAT" -> idat blob of unlimited                nEntries=1
  "IEND" -> iend IENDImageTrailer                 nEntries=1
  "tRNS" -> trns blob of unlimited                nEntries=1
  "gAMA" -> gama blob of unlimited                nEntries=1
  "cHRM" -> chrm blob of unlimited                nEntries=1
  "sRGB" -> srgb blob of unlimited                nEntries=1
  "iCCP" -> iccp ICCPEmbeddedICCProfile           nEntries=1
  "iTXt" -> itxt iTXt International textual data  nEntries=1
  "tEXT" -> text tEXt Textual data                nEntries=1
  "zTXt" -> ztxt blob of unlimited                nEntries=1
  "bKGD" -> bkgd blob of unlimited                nEntries=1
  "pHYs" -> phys blob of unlimited                nEntries=1
  "sBIT" -> sbit blob of unlimited                nEntries=1
  "sPLT" -> splt blob of unlimited                nEntries=1
  "hIST" -> hist blob of unlimited                nEntries=1
  "tIME" -> time blob of unlimited                nEntries=1
```

ClassFileFormat

```
Binary format ClassFileFormat

magic : unsigned int 4
  constraints: = 0xCA-FE-BA-BE
minor_version                 : unsigned int 2
major_version                 : unsigned int 2
constant_pool_count           : unsigned int 2
constant_pool                 : ConstantPoolElement[constant_pool_count - 1]
access_flags                  : unsigned int 2
this_class                    : unsigned int 2
super_class                   : unsigned int 2
interfaces_count              : unsigned int 2
interfaces                    : unsigned int 2[interfaces_count]
fields_count                  : unsigned int 2
fields                        : FieldInfo[fields_count]
methods_count                 : unsigned int 2
methods                       : MethodInfo[methods_count]
attributes_count              : unsigned int 1
attributes                    : AttributeInfo[attributes_count]
```

https://github.com/ftomassetti/FormatsDSL

Language to define importers (e.g., I have built importers for JVM Bytecode and PNG)

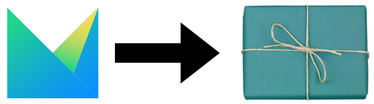


Generating stuff: why?

You can:

- Generate code, compile it, execute it
- Generate XML/JSON, interpret it to execute it
- Generate PDFs or diagrams for humans to consume them
- Generate inputs for other software to consume it

Lock-in?



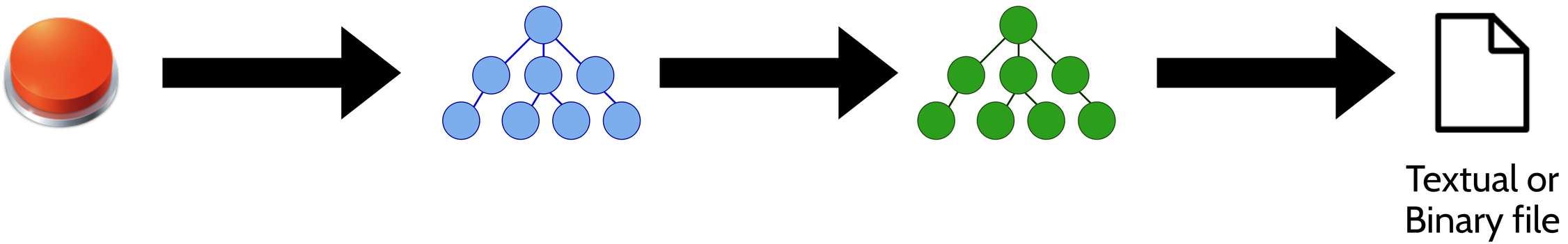
Generating stuff: how?

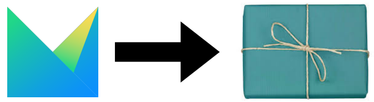
We want to answer these questions:

1. How do we trigger the generation?

2. Do we need intermediate transformation?

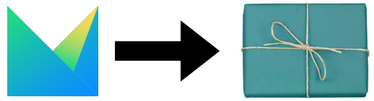
3. How do we do the last step (serialization)?





Generating stuff: trigger

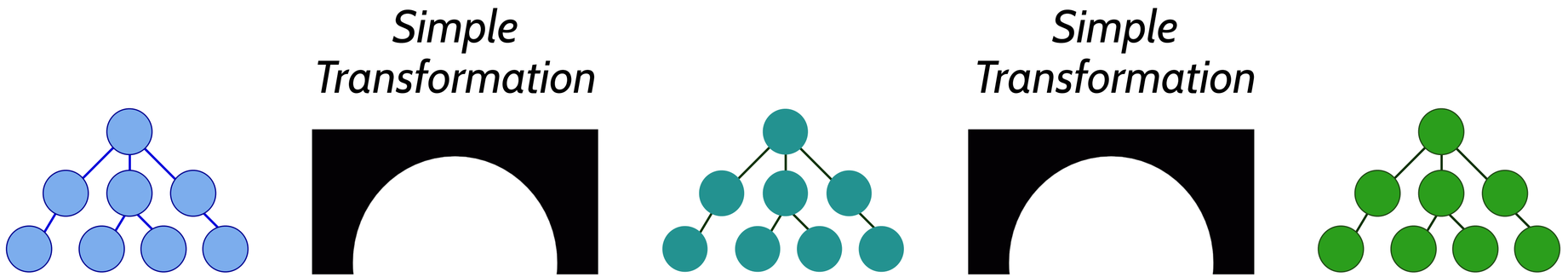
1. Automatically when building model
2. Manual (menu option, button)
3. Using MPS headless (as part of CI?)

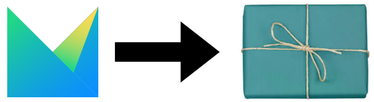


Generating stuff: transformations

Original model

Generated artifact





Generating stuff: transformations

Useful because:

1. Less complexity
2. More testability
3. Easier to absorb changes: source and target evolve!

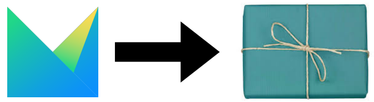
Standard Model Transformations

https://www.jetbrains.com/help/mps/generator-cookbook.html

Incremental Model Transformations, based on shadow models or Dclare

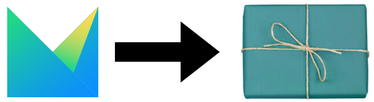
https://jetbrains.github.io/MPS-extensions/extensions/shadowmodels/

https://github.com/ModelingValueGroup/DclareForMPS

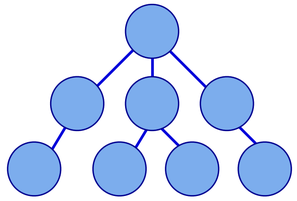
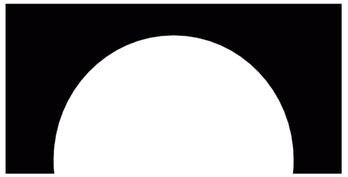
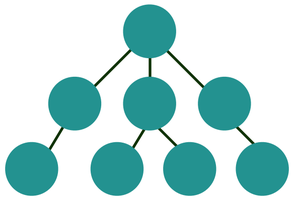


Generating stuff: serialization

1. We may generate interchange formats: same libraries as for import
2. We may generate binary formats
3. We may do a final transformation to a language which has its own textual generator, like BaseLanguage
4. We most frequently want to generate text

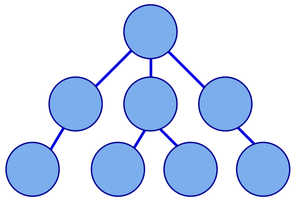


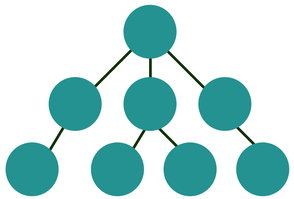
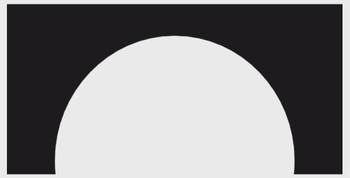
Generating stuff: serialization

Language
without an existing
text generator

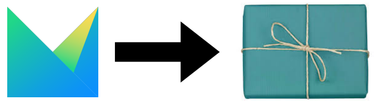
Textual file



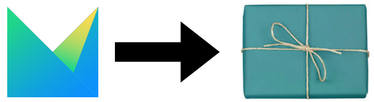

Language
with an existing
text generator

A free generator!
Java/C/Javascript/R/XML (C#, C++)



Generating stuff: textual generation

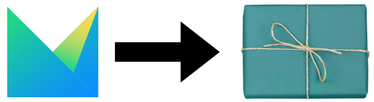
StringBuilder	Just build strings and print them on file
Text generators	Built-in support in MPS https://www.jetbrains.com/help/mps/textgen.html
JetBrains MPS Text Generator Plugin	Part of MPS Extensions https://jetbrains.github.io/MPS-extensions/extensions/plaintext-gen/



Generating stuff: resources

Resources on how to design good generators
https://coolya.github.io/maintainable-generators/

When to use the MPS Generator and when not?
https://specificlanguages.com/posts/translation-vs-export/



Generating documentation

Using Language Workbenches and Domain-Specific Languages for Safety-Critical Software Development

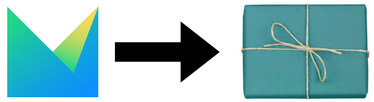
Derivation of Artifacts In critical domains, a substantial number of documents are required as evidence of the correct functioning of a CSC, to demonstrate the adherence to the development process or to make the system's behaviors understandable for non-programmer stakeholders and reviewers. When using DSLs, even though some of the documents may not strictly be necessary because the models are aligned with the domain better, reviewers or certification authorities may still require them. Generating these documents from models (to the degree possible) ensures consistency with the actual system and further reduces effort [80]. Examples include diagrams representing the structure or behavior of the system as well as trace reports.

In practice you can:

- Generate text guaranteed to be consistent with the implementation
- Generate tables summarizing results, for example test coverage, number of tests executed, number of algorithms, complexity metrics, etc.
- Diagrams describing the behavior of the application

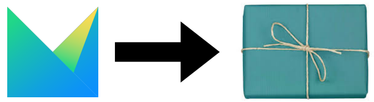
This stuff can be useful, to improve comprehension, or simply be required (regulatory agencies).

Important because you cannot always expect others to use your interface (e.g., MPS). They may need to use standard formats or archive information for later use (possibly decades from now).



Generating documentation

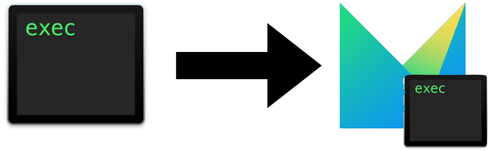
http://mbeddr.com/files/documentationdocumentation.pdf	Language to build documents with live code
https://github.com/vjramirez/PlantUML	Language for PlantUML models, also useful to generate PlantUML diagrams from custom DSL.
https://www.jetbrains.com/help/mps/dive-into-plugins-the-plantmps-plugin.html	PlantMPS, integration with PlantUML to visualize (but also export) models, derived from Mbeddr
GraphViz	I use it every time I must generate a diagram and not controlling the layout is ok



Generating documentation

```
section 1.3 workingWithMbeddr.embddingCode: Embedding Code {

  Header: Embed as Image
  [ You have already seen in the previous paragraph how to embed mbeddr code as an image
  into the document. In that example, @fig(calculator) embedded a complete top level
  construct, an interface in this case. But what if you wanted to embed only a smaller
  section, such as a state in a state machine or a single operation in an interface?
  @fig(addOp) shows an example of embedding only an operation. The code to do that is
  shown in @fig(embedding1); essentially you mention the \code(add) operation after
  the slash in the \code(embed  image) tag. ]

  ---------------------------------------------------------------------------
  embed image ExampleCode.Calculator/add as addOp
    location: imgTemp:/
    scaling:  smallCodeShot
  [ An example of how to embed only a part of a module content as an image. ]
  ---------------------------------------------------------------------------

  Header: Embedding as Text
  [ You can also embed mbeddr code as text. This is interesting in particular for Latex
  export, since you can configure the \code(listings) package to provide syntax
  highlighting for your code. The following paragraph shows how to embed the interface
  as text; not that this is not a floating entity and cannot be referenced, it is
  inlined with the text. Also note that in the inspector for the \code(embed  as text)
  tag you can specify the language name used for highlighting. By default, it is
  \code(mbeddr). ]

  embed as text ExampleCode.Calculator/

  ---------------------------------------------------------------------------
  embed doc section embddingCode as embedding1
    location: imgTemp
    scaling:  width is 100 % of page
  [ An example of how to embed only the \code(add) operation as an image. ]
  ---------------------------------------------------------------------------

}
```

Mbeddr doc



Make other tools work within MPS

Add Configuration... Git:

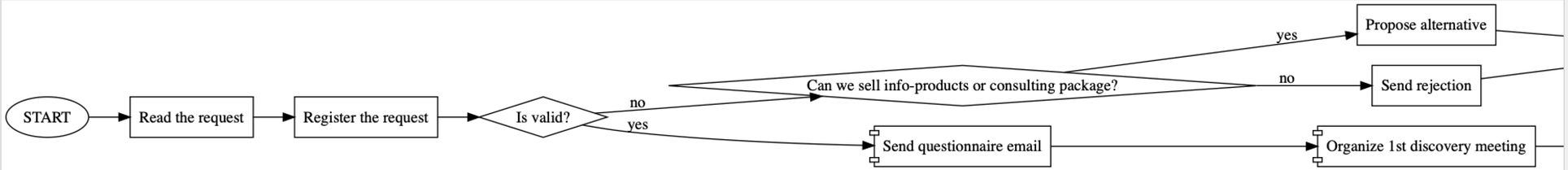
Logical View

ProcessDSL (/Users/federico/repos/Proces
com.strumenta.ide
com.strumenta.processes.sandbox (gen
com.strumenta.processes
Modules Pool
checkpoints

Offer preparation | ProcessDiagramGeneration | GraphvizFacade | GraphvizGeneration

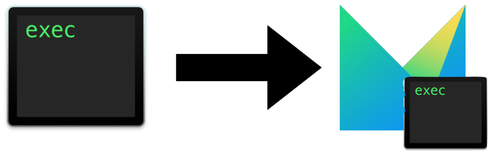
```
Process Offer preparation

responsible: Sales Manager

triggers:
  -> we receive a request for our services by email or LinkedIN

steps:

  (Sales Manager)
  [Read the request]

  (Sales Manager)
  [Register the request]

  (Sales Manager)
  <Is valid?>
    yes -> [[Send questionnaire email]]
    no -> <Can we sell info-products or consulting package?>
```

Process View



Process View | Documentation | Console | Terminal | 3: Usages | 9: Git | 0: Messages | Event Log | 2: Inspector

1: Project | 7: Structure | Favorites | Context Actions | Context Actions 2



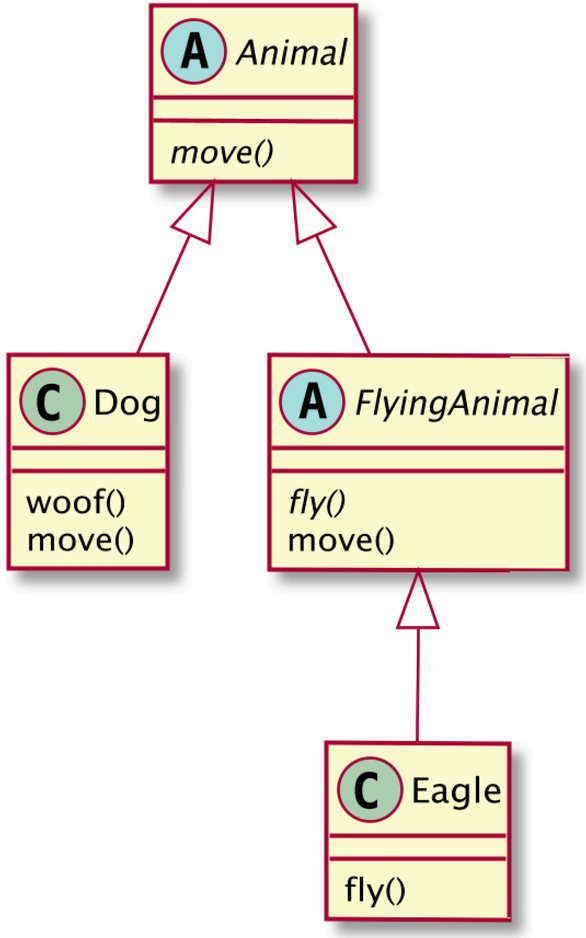
Make other tools work within MPS

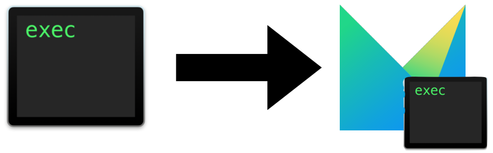
Simple | Animal | Eagle

```
public class Eagle extends FlyingAnimal {
  @Override
  public void fly() {
    // TO BE IMPLEMENTED
  }
}
```

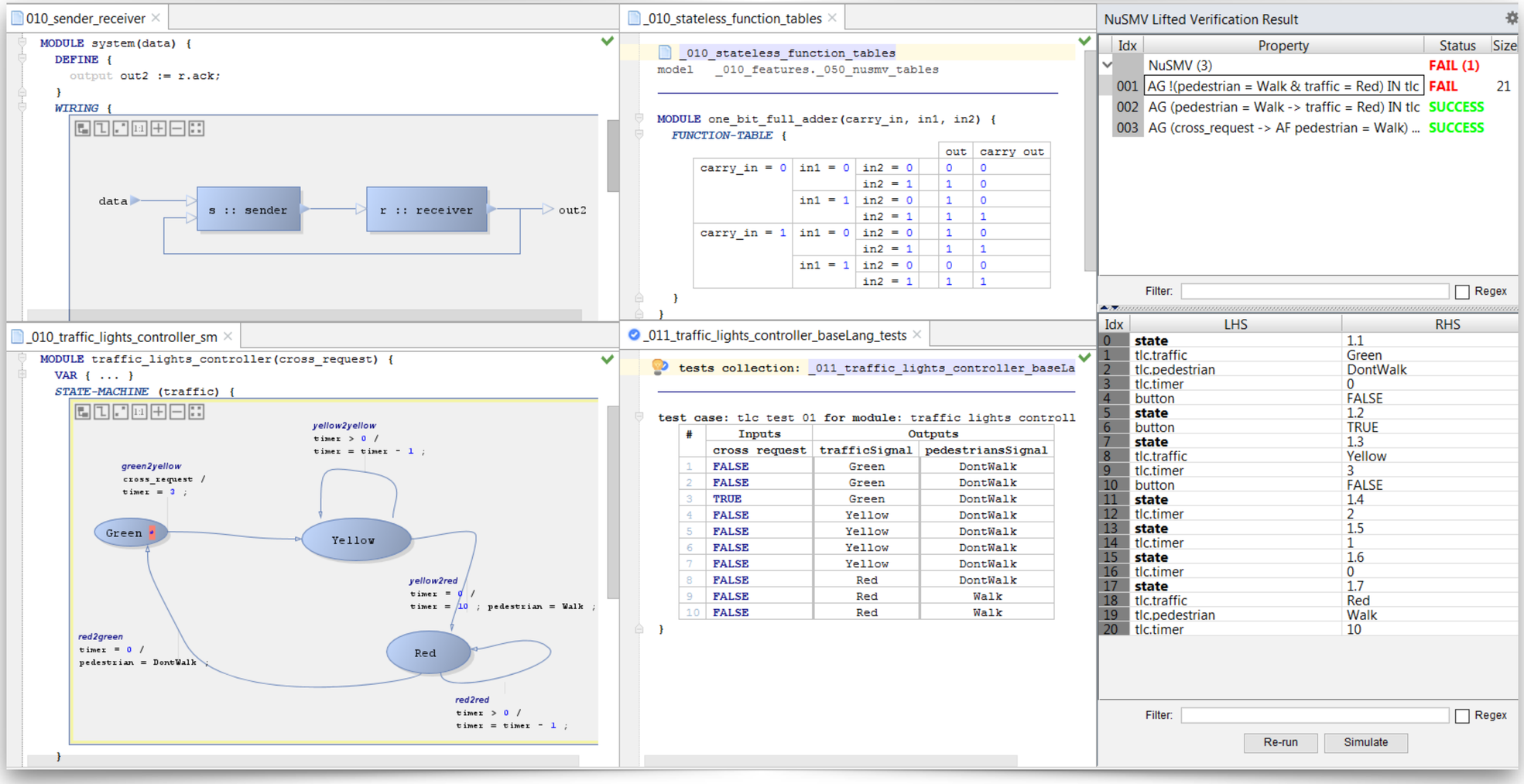
Visualization | Class Diagram Simple

Overview | Classes





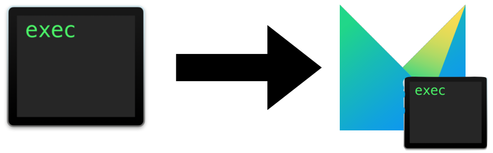
Make other tools work within MPS



FASTEN: FormAl SpecificaTion ENvironment

https://sites.google.com/site/fastenroot/

Integrate NuSMV for model checking



Build servers

https://github.com/mbeddr/mps-gradle-plugin	Miscellaneous tasks that were found useful when building MPS-based projects with Gradle.
https://github.com/mbeddr/mbeddr.build.docker	Basically a TeamCity build agent with some additional packages for compiling the generated Java and C code.
https://github.com/specificlanguages/mps-gradle-plugin	Useful to package and publish MPS languages as maven package

Make other tools work with MPS

Http, WebSockets

MPSServer can run in headless MPS and can be used to read and write models, to trigger intentions, to calculate types, to perform model validation, etc.

Basically it permits to integrate with all the runtime capabilities of MPS.

https://github.com/strumenta/mpsserver

Make other tools work with MPS

Global routes

GET /: return the message `MPS Server up and running.`. It can be used to verify that the MPS Server is up.

GET /server/extensions: return a list of the extensions that were loaded. The list is composed by the names of such extensions.

GET /languages: return a list of `LanguageInfo`, including all languages which are part of the LanguageRegistry. `LanguageInfo` contains two fields: `qualifiedName` and `sourceModuleName`.

GET /modules: return a list of `ModuleInfo`. It traverses the modules which are part of the repository. It consider the flags `includeReadOnly` and `includePackaged` (both default to false) to decide how to filter the modules. including all languages which are part of the LanguageRegistry. `ModuleInfo` contains these fields: `name`, `uuid`, `foreignName`, `packaged`, `readOnly`.

- query parameter *includeReadOnly*, flag, default false. If set include read-only modules in the list
- query parameter *includePackaged*, flag, default false. If set include packaged modules in the list

Make other tools work with MPS

GET http://localhost:2904/languages Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 196 ms Size: 16.36 KB Save Respons

Pretty Raw Preview Visualize HTML


```
{
"success": true,
"message": "ok",
"value": [
{
"qualifiedName": "jetbrains.mps.baseLanguage.constructors",
"sourceModuleName": "jetbrains.mps.baseLanguage.constructors"
},
{
"qualifiedName": "jetbrains.mps.debugger.api.lang",
"sourceModuleName": "jetbrains.mps.debugger.api.lang"
},
{
"qualifiedName": "jetbrains.mps.lang.extension",
"sourceModuleName": "jetbrains.mps.lang.extension"
},
{
"qualifiedName": "jetbrains.mps.lang.editor.tooltips",
"sourceModuleName": "jetbrains.mps.lang.editor.tooltips"
},
{
"qualifiedName": "jetbrains.mps.lang.feedback.problem.failingRule",
"sourceModuleName": "jetbrains.mps.lang.feedback.problem.failingRule"
},
```

Work with MPS Models outside MPS

Raison d'être

The reason for this tool's existence is wanting to have some way to in(tro)spect MPS projects outside of MPS itself. Instead of trying to shoehorn such functionality into an MPS plugin, the idea is to run a separate process that analyses, and potentially auto-fixes MPS XML files. Also, running MPS headless is typically somewhat problematic to set up, and get/keep working. As this tool proves, it's not always necessary to actually load and run MPS.

https://github.com/dslmeinte/mps-open-source/blob/master/mps-analyser/README.md

Similar to mps-analyser, but it works also with binary models (a lot of libraries shipped with MPS are in this format).

It generates JSON files from MPS models and resolve concepts.

Long term plan to generate Kotlin classes and perform more advanced analysis.

https://github.com/strumenta/mpsinterop

Make other tools work with MPS

```java
public class AddChild extends RequestMessage {
  public NodeReference container;
  public string containmentName;
  public string conceptToInstantiate;
  public int index = -1;
  @Nullable()
  public RegularNodeIDInfo smartRefNodeId;
}
```

```typescript
export interface AddChild extends RequestMessage {
  type: 'addChild';
  container: NodeReference;
  containmentName: string;
  conceptToInstantiate: string;
  index: number;
  smartRefNodeId?: NodeId;
}
```

I need the generator to evolve outside MPS: from MPS I need the data

Make other tools work with MPS

```
public class AddChild extends RequestMessage {
  public NodeReference container;
  public string containmentName;
  public string conceptToInstantiate;
  public int index = -1;
  @Nullable()
  public RegularNodeIDInfo smartRefNodeId;
}
```

```
21018     <node concept="312cEg" id="5p1VBoMcmdg" role="jymVt">
21019       <property role="TrG5h" value="containmentName" />
21020       <node concept="3Tm1VV" id="5p1VBoMcmdh" role="1B3o_S" />
21021       <node concept="17QB3L" id="5p1VBoMcmdi" role="1tU5fm" />
21022     </node>
```

Make other tools work with MPS

```
{
  "properties": {
    "name": "containmentName"
  },
  "children": [
    {
      "properties": {},
      "children": [],
      "references": [],
      "conceptName": "jetbrains.mps.baseLanguage.structure.PublicVisibility",
      "id": "62155111521636442 1",
      "containmentLinkName": "visibility"
    },
    {
      "properties": {},
      "children": [],
      "references": [],
      "conceptName": "jetbrains.mps.baseLanguage.structure.StringType",
      "id": "6215511152163644242",
      "containmentLinkName": "type"
    }
  ],
  "references": [],
  "conceptName": "jetbrains.mps.baseLanguage.structure.FieldDeclaration",
  "id": "6215511152163644240",
  "containmentLinkName": "member"
},
```

https://github.com/strumenta/mpsinterop

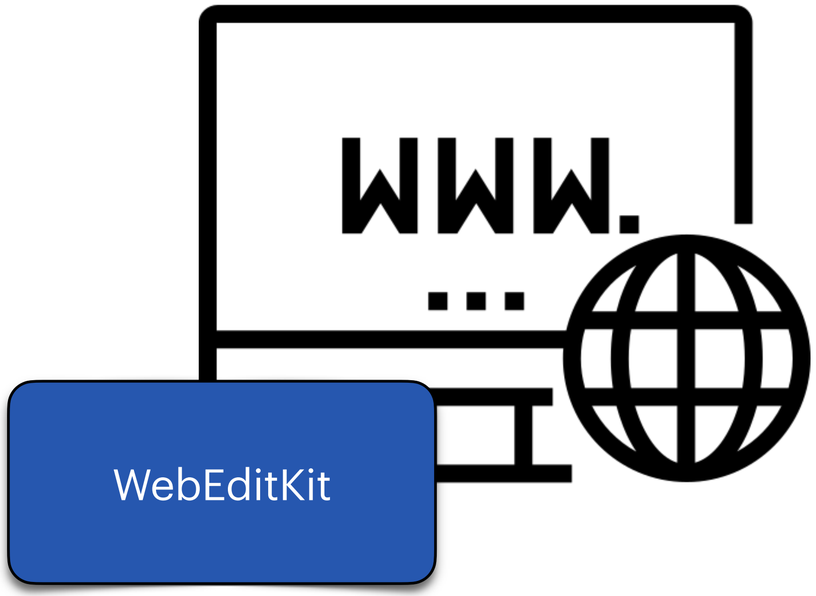
Make other tools work with MPS

```
1442  // tslint:disable-next-line:max-classes-per-file
1443  export class BinaryExpression extends Expression {
1444    static CONCEPT_NAME = 'org.iets3.core.expr.base.structure.BinaryExpression';
1445    static LINK_LEFT = 'left';
1446    static LINK_RIGHT = 'right';
1447
1448    constructor(data: NodeData) {
1449      super(data);
1450    }
1451
1452    left(): Expression {
1453      return this.childByLinkName( linkName: 'left') as Expression;
1454    }
1455
1456    right(): Expression {
1457      return this.childByLinkName( linkName: 'right') as Expression;
1458    }
1459
1460    leftChildCell(): VNode {
1461      return childCell(this, BinaryExpression.LINK_LEFT);
1462    }
1463
1464    rightChildCell(): VNode {
1465      return childCell(this, BinaryExpression.LINK_RIGHT);
1466    }
1467  }
```

https://github.com/strumenta/mpsinterop

Make other tools work with MPS

Http, WebSockets

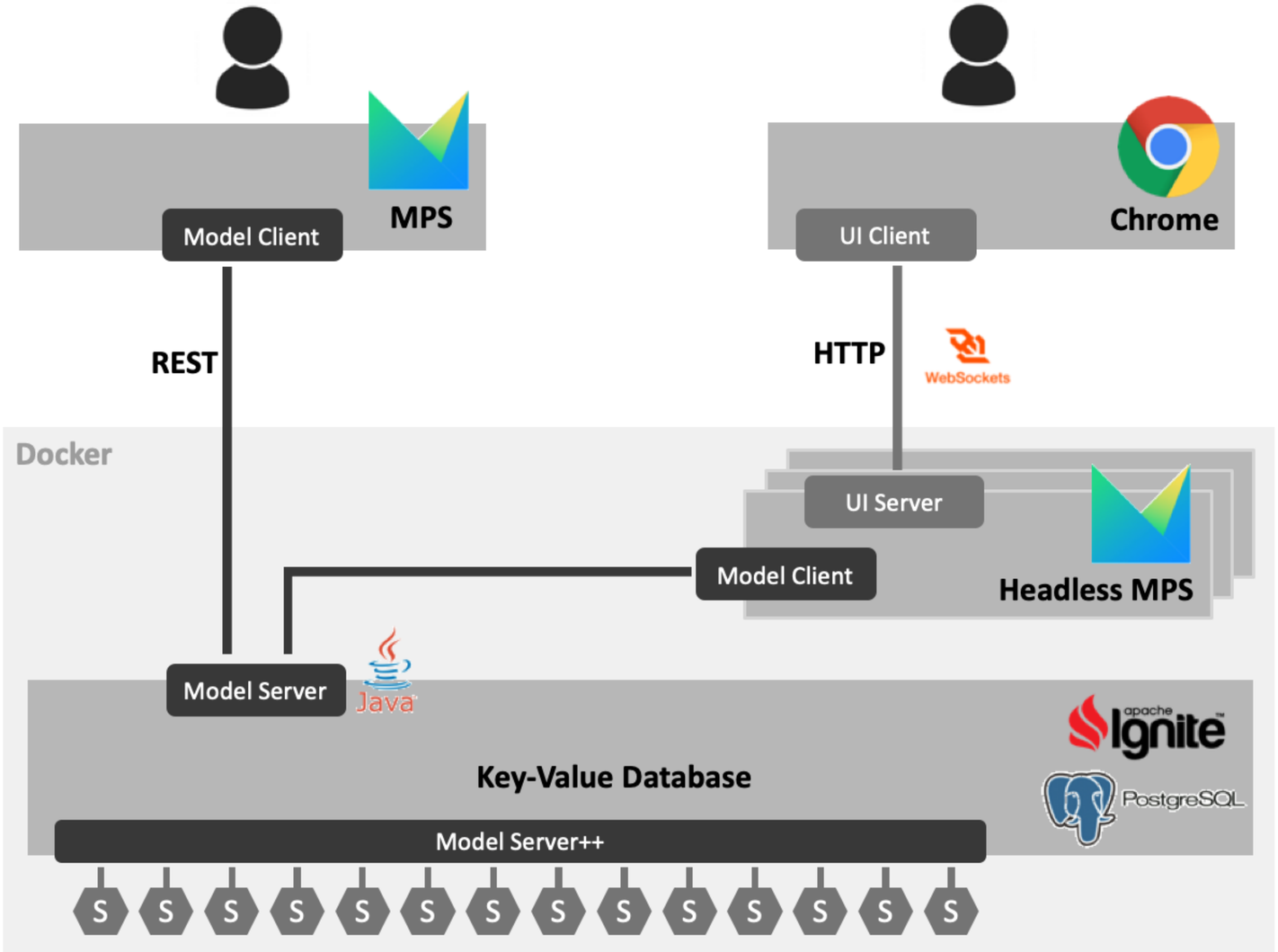


WebEditKit is a framework to create editors for MPS models that run in the browser.

We talk in more details about it in a presentation on the 4th of February.

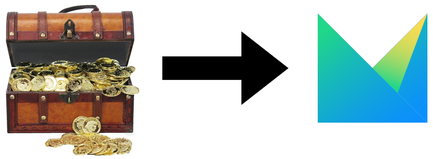
https://github.com/strumenta/webeditkit

Make other tools work with MPS

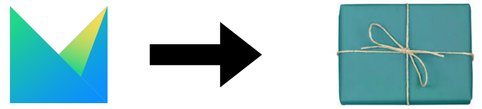


https://github.com/modelix/modelix

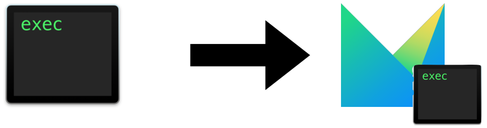
Recap



One-time import or continuous import?
Binary, Textual, and Common formats



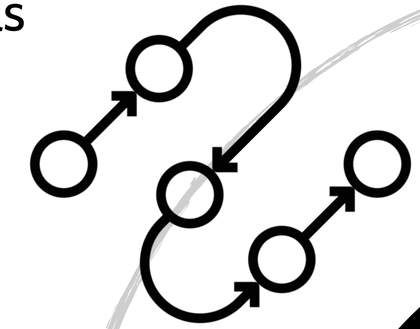
Triggers, Transformations,
Different mechanisms for text generation
Generation of documentation



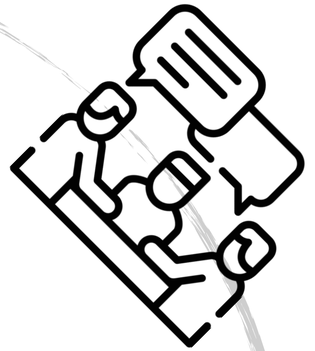
Graphviz
PlantUML
Solvers

MPSInterop,
MPSServer,
WebEditKit
Modelix

Integration with other tools necessary for other steps of the process



Generation of documentation, Possibility to visualize models outside MPS



Generation of documentation, Possibility to visualize models outside MPS

Import of external knowledge, integration with model-checkers

Summary

MPS provides a lot of value when used alone.

But making it the core element of a larger ecosystem:

- Permits to leverage investements made in building knowledge outside it or in other tools
- Leverage the investment in MPS by benefitting in areas in which MPS is not useful
- Reduce lock-in fear (motivated or not)

We have seen that there are many different ways to integrate with MPS, taking more advantage of it.

Next steps

Generation works already very well, but we can get better runtime integration.

For further integration I believe in **Modelix and MPSServer**, to reuse not just the knowledge captured in MPS but also its capabilities

This is a community effort!

I have cited the work of different great contributors to this community.

Markus Völter
Sergej Koščejev
Sascha Lißon
Kolja Dummann
Marco Lombardo
Fabien Campagne
Přemek Vysoký
Alessio Stalla
Meinte Boersma
Víctor Julio Ramírez-Durán
Daniel Ratiu
Wim Bast
(and myself)
and of course the JetBrains' people!

You can and should contribute too!

Provide feedback, discuss your issues, report bugs, send pull requests.

To move things forward we need the help of everyone!

This is a community effort!

Resources:

http://mps.rocks/

MPS Slack: http://slack-mps.jetbrains.com/

https://strumenta.community

Let’s discuss!

Link to the slides:
https://strumenta.com/mps-interoperability

Federico Tomassetti

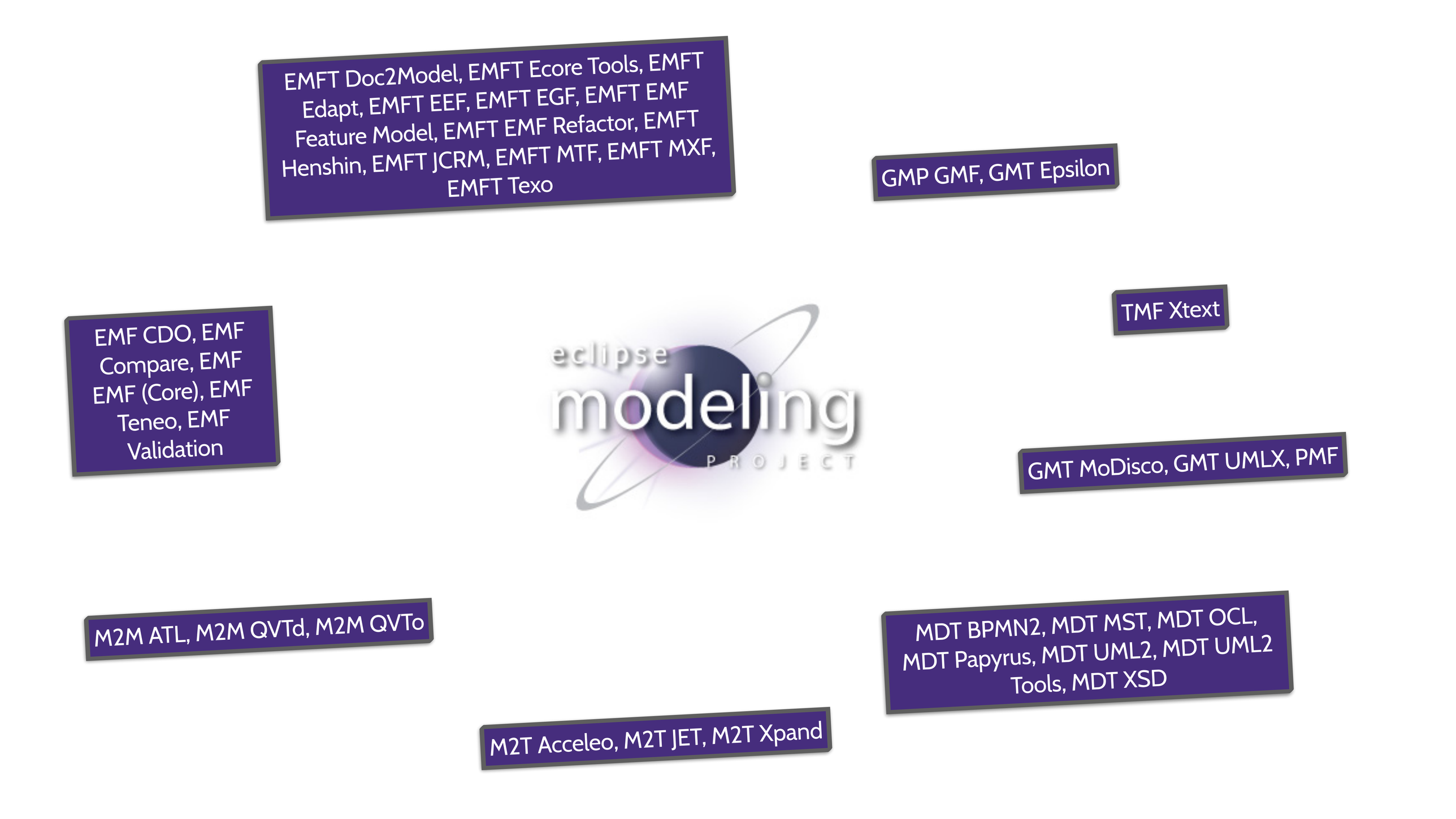
STRU
MEN
TA

Is MPS unique in this need for interoperability?

No.

Let's consider:

- The Eclipse Modeling Project
- Más, by Meinte Boersma
- Platform for system and business engineering tools, as proposed by Markus Völter


EMFT Doc2Model, EMFT Ecore Tools, EMFT Edapt, EMFT EEF, EMFT EGF, EMFT EMF Feature Model, EMFT EMF Refactor, EMFT Henshin, EMFT JCRM, EMFT MTF, EMFT MXF, EMFT Texo
GMP GMF, GMT Epsilon
TMF Xtext
EMF CDO, EMF Compare, EMF EMF (Core), EMF Teneo, EMF Validation
eclipse
modeling
PROJECT
GMT MoDisco, GMT UMLX, PMF
M2M ATL, M2M QVTd, M2M QVTo
MDT BPMN2, MDT MST, MDT OCL, MDT Papyrus, MDT UML2, MDT UML2 Tools, MDT XSD
M2T Acceleo, M2T JET, M2T Xpand

EMFT Doc2Model, EMFT Ecore Tools, EMFT Edapt, EMFT EEF, EMFT EGF, EMFT EMF Feature Model, EMFT EMF Refactor, EMFT Henshin, EMFT JCRM, EMFT MTF, EMFT MXF, EMFT Texo

GMP GMF, GMT Epsilon

TMF Xtext

EMF CDO, EMF Compare, EMF EMF (Core), EMF Teneo, EMF Validation

GMT MoDisco, GMT UMLX, PMF

M2M ATL, M2M QVTd, M2M QVTo

MDT BPMN2, MDT MST, MDT OCL, **MDT Papyrus**, MDT UML2, MDT UML2 Tools, MDT XSD

M2T Acceleo, M2T JET, M2T Xpand

Extract from *Post-mortem for Más*

WRITTEN BY
Meinte Boersma
All-round software value creator, specialising in DSL construction and language engineering.

Follow

81

Pitfall 1 — no integration

As already advertised: it took me a long while to come out of denial on this one…

Basically, the only way to do something with something modelled in Más (in a language modelled in Más) was to download a JSON or XML representation of it. I was a long way off from allowing language builders to define generators in Más, also because of the editor framework that was holding me back. But even that would not have made enough of a difference to swing the pendulum, since I also didn't have proper versioning, etc.

I alluded to this problem in a previous blog as well, and I find Más is not the only language workbench suffering from it. My advice: envision how to make your language workbench integrate with a well-chosen subset of types of actual projects in terms of both audience (see also pitfall 4), technology (platform, IDE, versioning, etc.), as well as process (collaboration, issue tracking, releases, etc.). Let this vision drive the rest of the effort, and you just might end with a usable or even sellable product.

https://medium.com/@dslmeinte/post-mortem-for-m%C3%A1s-aeca7542c4c8

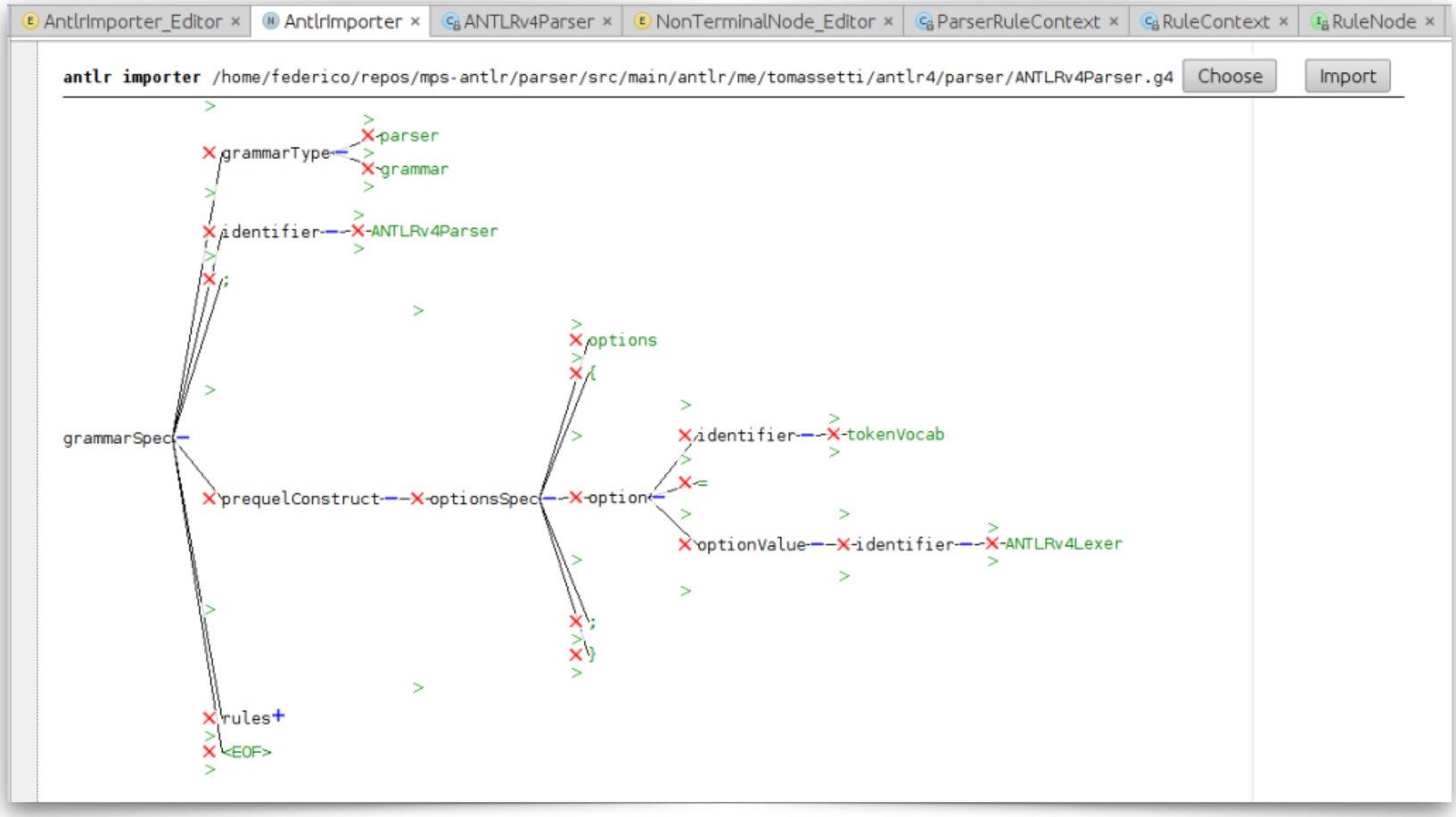
Extract from *An open platform for systems and business engineering tools*

*A platform for developing systems and business engineering tools must be unbiased regarding the languages used to define the models and the analyses that run on them. Instead it should provide the infrastructure for defining languages, for working effectively with large, multi-paradigm models in a collaborative manner, **and for integrating arbitrary model analysis and transformation services**.*

*Ideally, all modeling needs might be addressed with languages native to the tool. However, **there will likely always be a need to integrate existing modeling tools** (such as the ubiquitous Simulink or Doors); so **a meaningful way of integrating existing tools is required**. In addition, initially, existing IDEs like MPS or modeling tools such as MagicDraw might serve as the editor frontend for models*

https://voelter.de/data/pub/APlatformForSystemsAndBusinessModeling.pdf

→ Bringing stuff in: Textual formats



Article on visualizing ANTLR parse trees into MPS

https://tomassetti.me/antlr-and-jetbrains-mps-parsing-files-and-display-the-ast-usign-the-tree-notation/

Bringing stuff in: Textual formats

https://github.com/julianthome/inmemantlr	Permits to instantiate parsers without having to generate java classes on disk
https://github.com/antlr/grammars-v4	Tons of grammars ready to use
https://github.com/CampagneLaboratory/ANTLR_MPS	Support for working with ANTLR4 grammars in MPS
https://github.com/premun/ingrid	MPS Plugin to create languages from ANTLR Grammar

Knowledge Centric Development