Towards a Dynamic Infrastructure for Playing with Systems of Systems

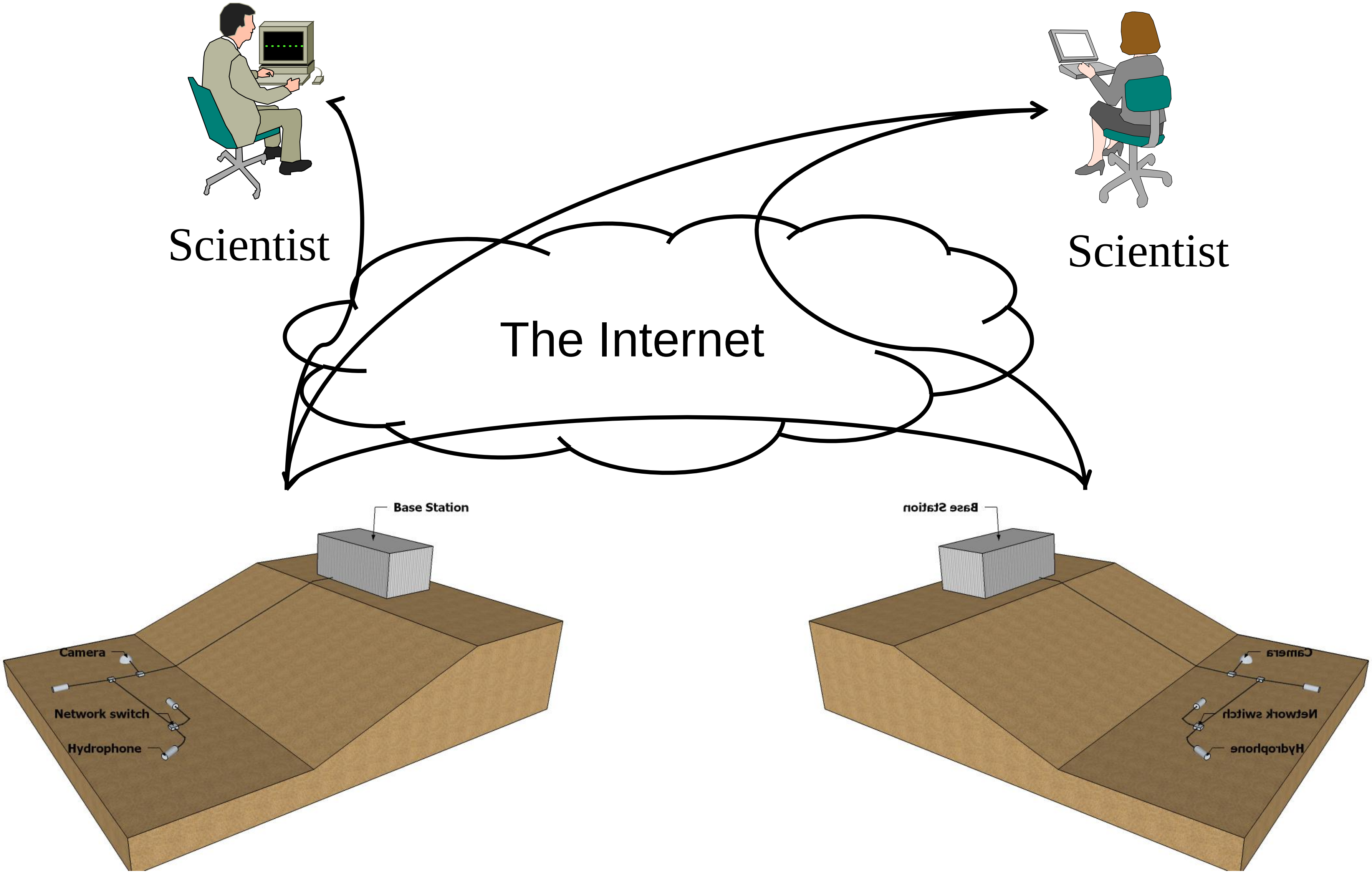
Jean-Philippe Schneider, Ciprian Teodorov, Eric Senn, Joël Champeau

SESoS 2014, August 26th 2014

Content

1. Context
2. Issues of SoS modeling and simulation
3. Parser combinators to transform models
4. Conclusion

Seafloor Observatory

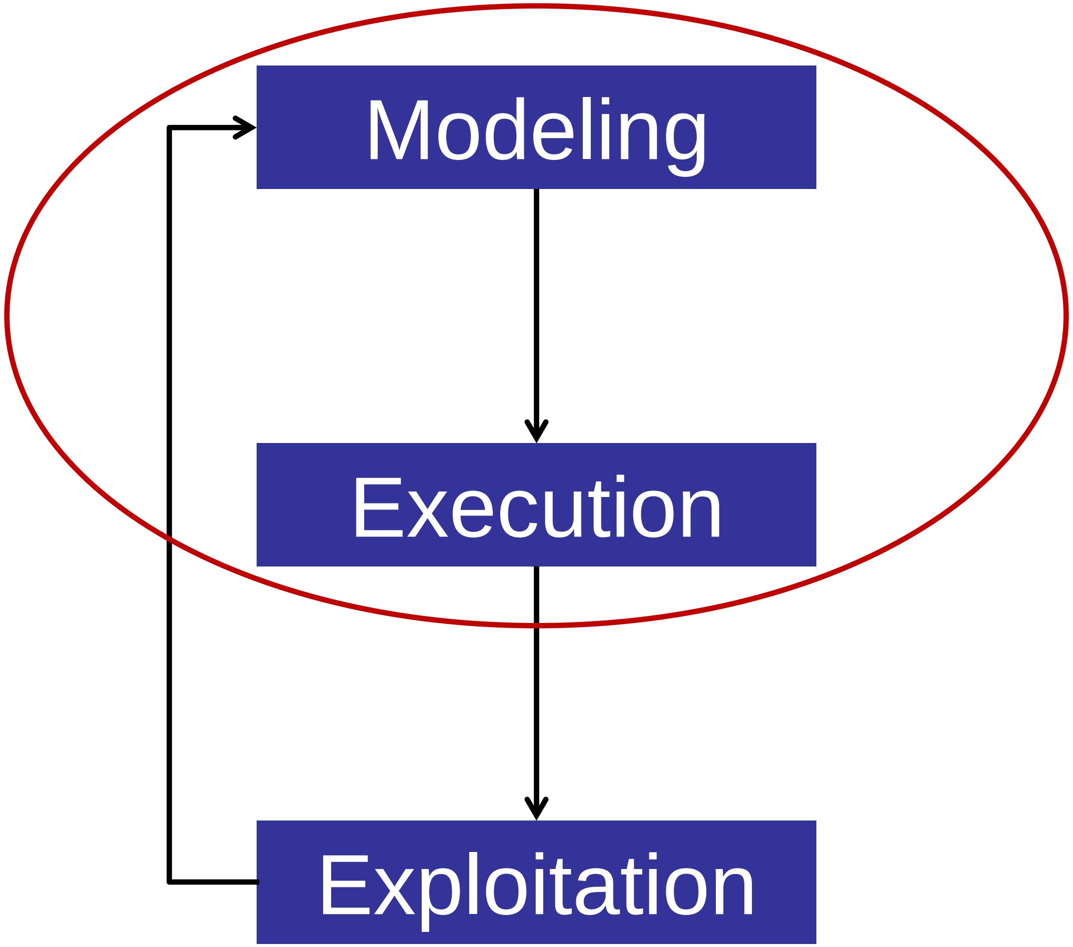


Scientist

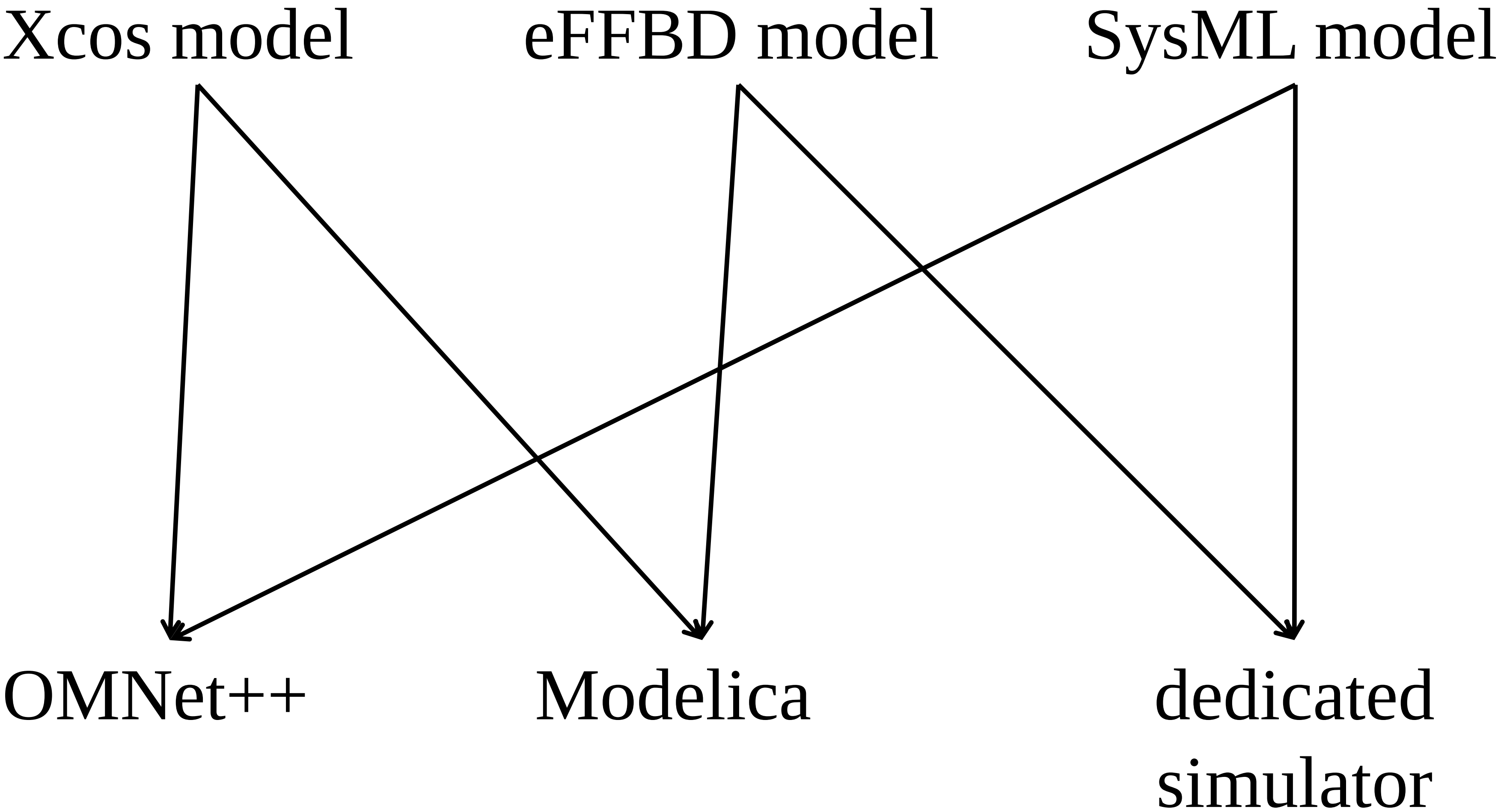
Scientist

The Internet

Simulation Process



From Model To Simulation



Xcos model

eFFBD model

SysML model

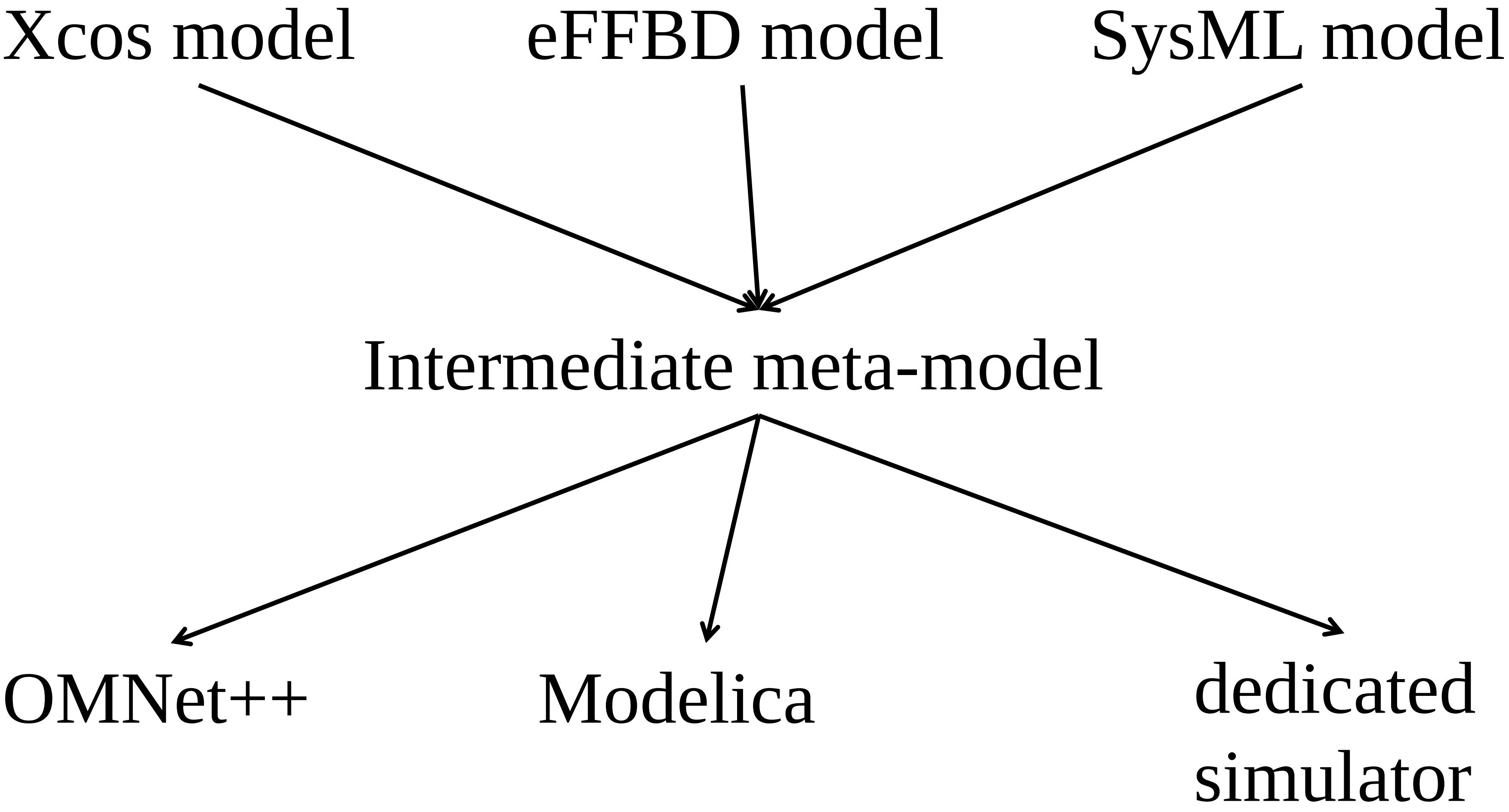
OMNet++

Modelica

dedicated simulator

Adaptation and extensibility

Through an Intermediate Representation



Xcos model

eFFBD model

SysML model

Intermediate meta-model

OMNet++

Modelica

dedicated simulator

Challenges

- How to combine several models independently from the input order?

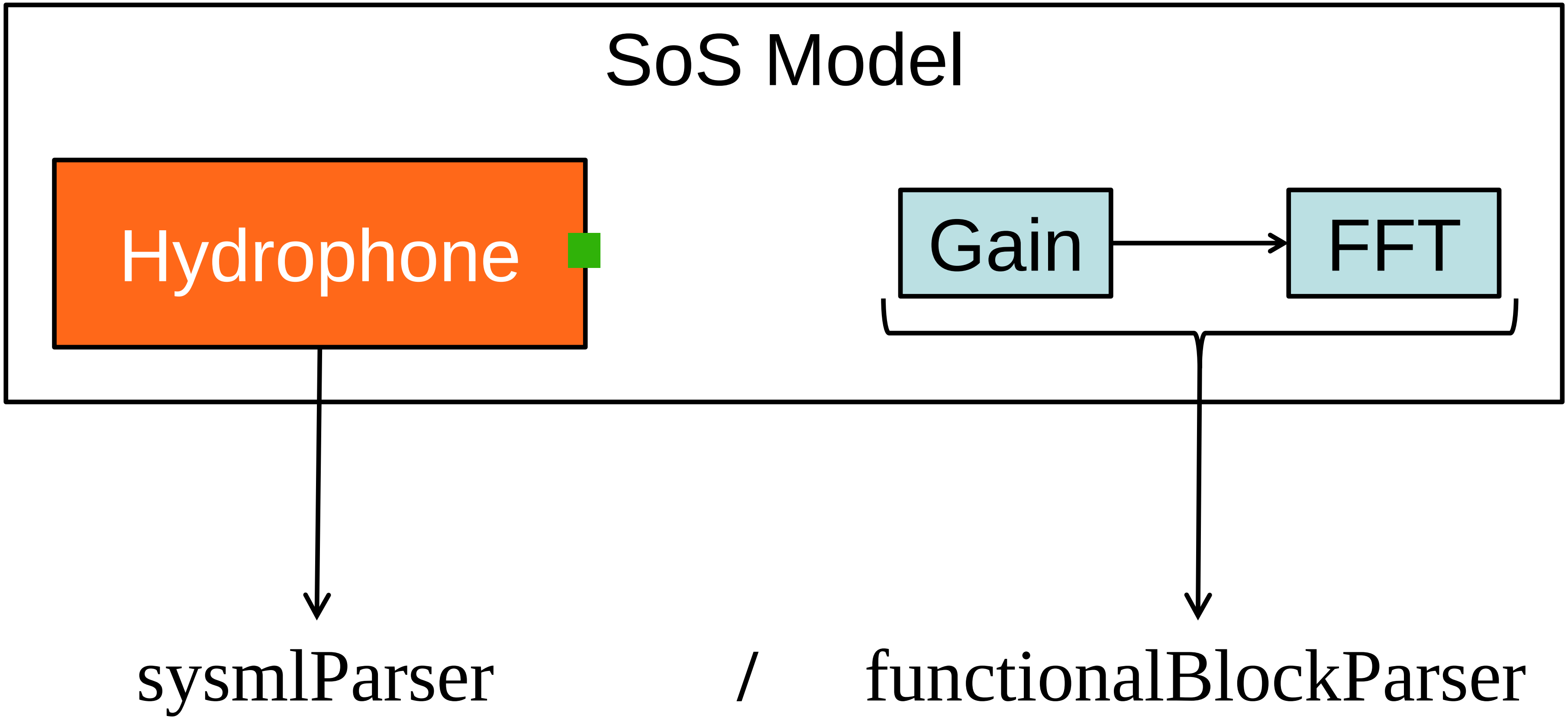
- How to extend tooling to support evolving input and output formalisms?

Parser Combinators

- **1 formalism = 1 model analyzer = 1 parser**
- **PetitParser (Grammar and Parser as Objects):**
  - ➔Associating one predicate (syntactic condition) with multiple actions.
  - ➔Using combinator operators such as sequence, choice or ordered choice.

How to use parser combinators in SoS modeling?

Combine Formalism Specific Parsers



SoS Model

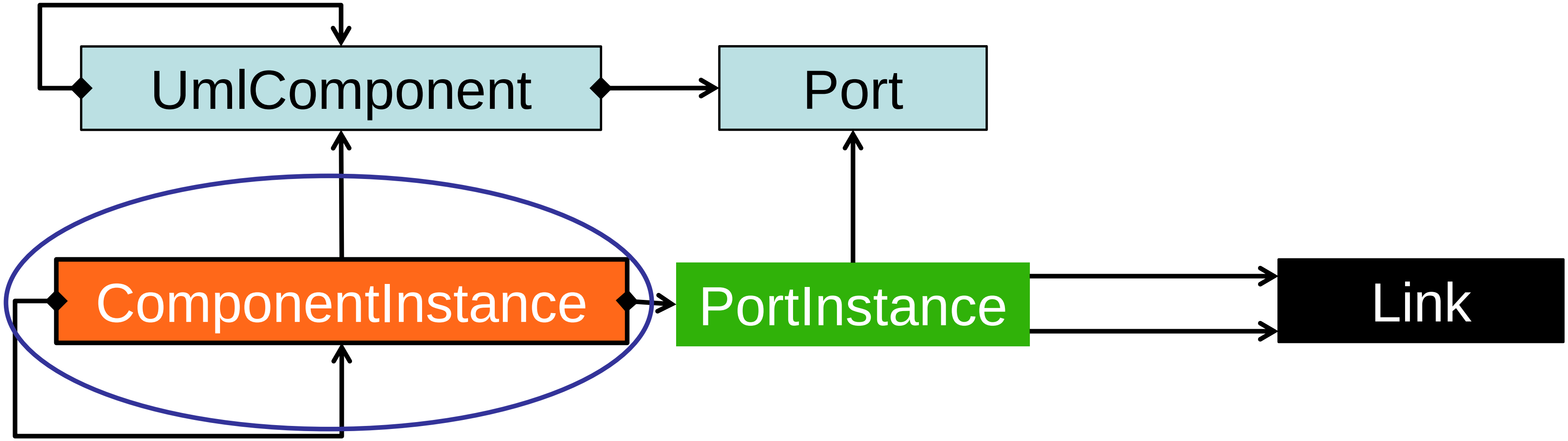
Hydrophone

Gain

FFT

sysmlParser / functionalBlockParser

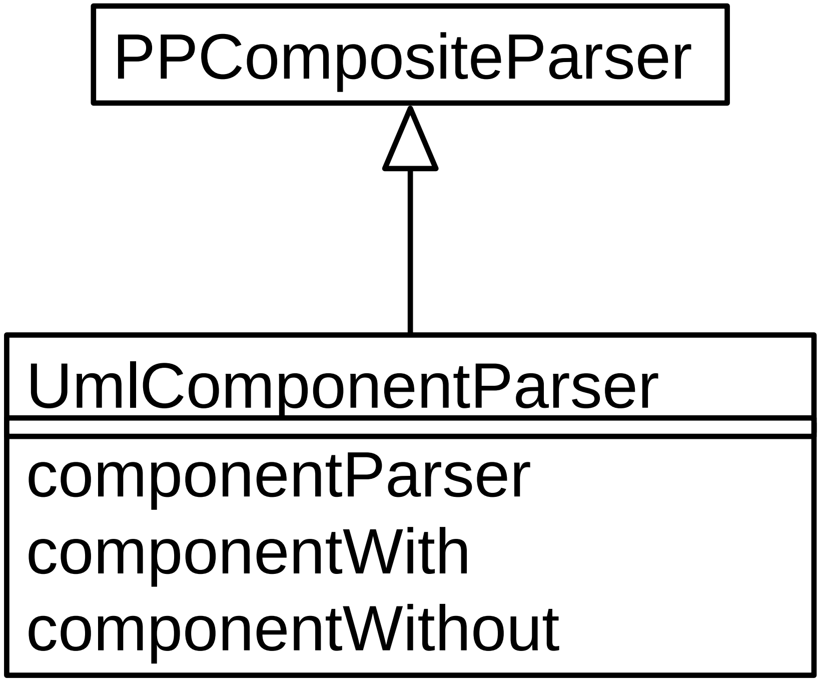
Combine Context-Specific Parsers



Distinguish cases:

- components with subcomponents
- components without subcomponents

Writing Parsers



PPCompositeParser

UmlComponentParser

componentParser
componentWith
componentWithout

Writing parsers

- **componentWith**

  Predicate

  ```
  ^[:instance | instance isComponent and: [instance
  hasSubcomponents]] asPredicateParser.
  ```

  Parser

- **componentWithout**

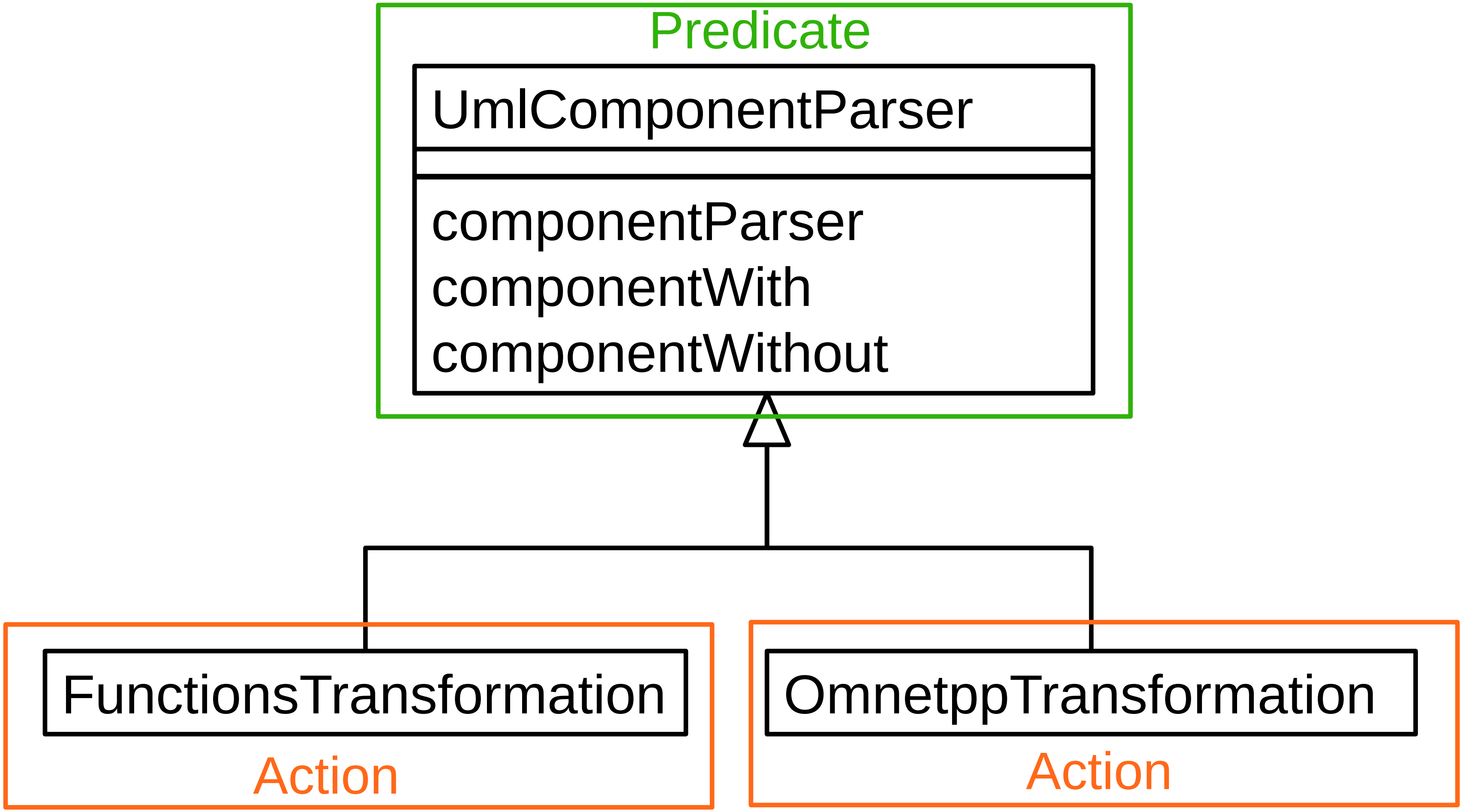
  ```
  ^[:instance | instance isComponent and: [instance
  hasSubcomponents not]] asPredicateParser.
  ```

- **componentParser**

  ```
  ^(componentWith / componentWithout).
  ```

  Composition

Associating Predicate and Actions



Parameterize parsing according to desired transformation action

Associating Predicate and Actions

- **FunctionsTransformation>>componentWith**

Predicate

```
^super componentWith ifTrueDo: [:instance |
    LogicalFunctionGroup from: instance].
```

Action

- **OmnetppTransformation>>componentWith**

```
^super componentWith ifTrueDo: [:instance |
    OmnettppComposite from: instance].
```

Conclusion

- **SoS design and analyzes need adaptative and extensible modeling tooling.**

- **Parser combinators provide an efficient support based on:**
  - Several output formalisms → one predicate / n-actions
  - Several input formalisms → Combinator operators.

- **Approach to be extended to an industrial use case.**

Thank you for your attention