Designing and Evaluating a Configuration DSL for Ocean Models

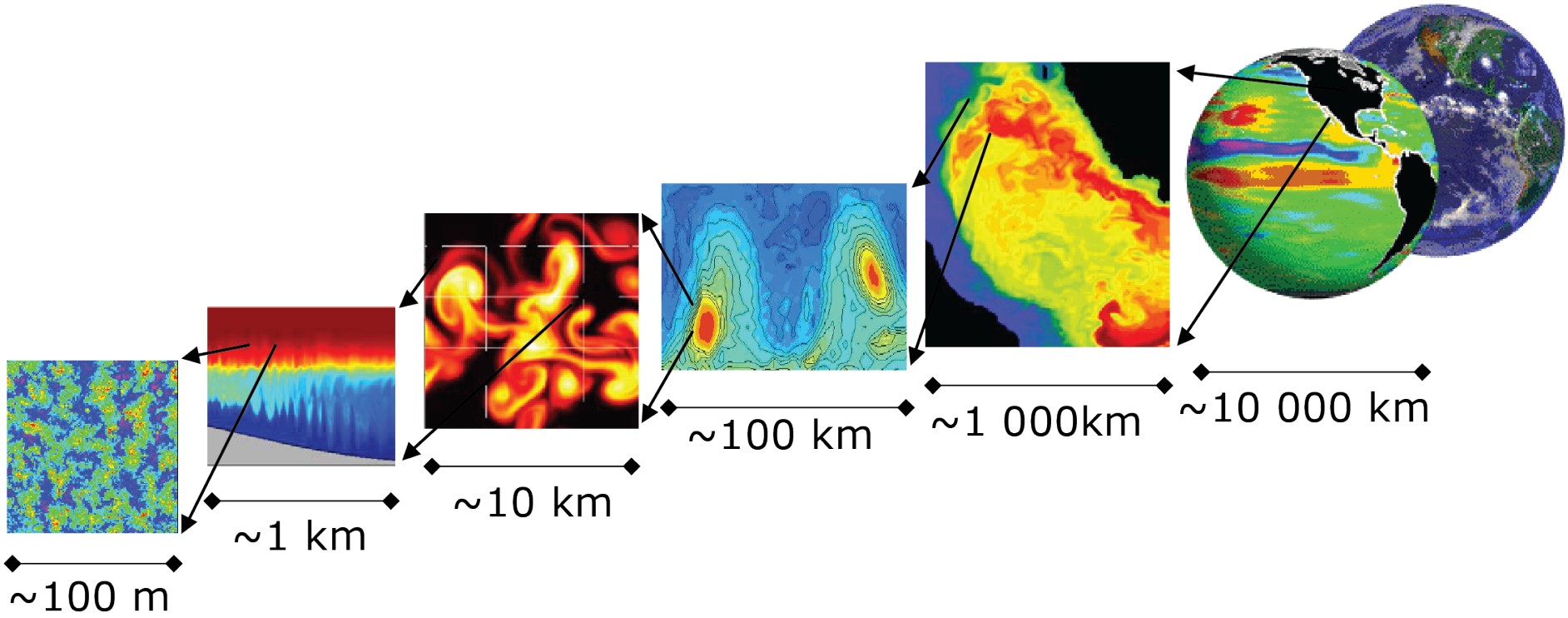
Serafim Simonov

November 17, 2020

- Simulations are based on mathematical models
- Development requires knowledge in different domains
  - Mathematics
  - Natural sciences
  - Software development
  - Parallelism
- Solution: DSL

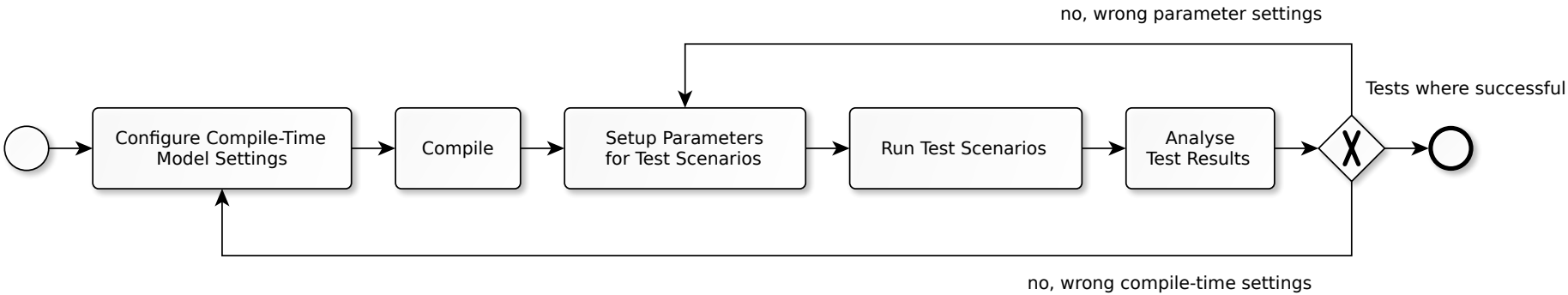
- Analyze Domain Problem
- DSL Development
- Evaluation

- DSL Development Approach Mernik et al. [2005]
- GECO Approach [Jung 2016]
- DSL Framework XText [Efftinge et al. 2012]
- Language Server Protocol

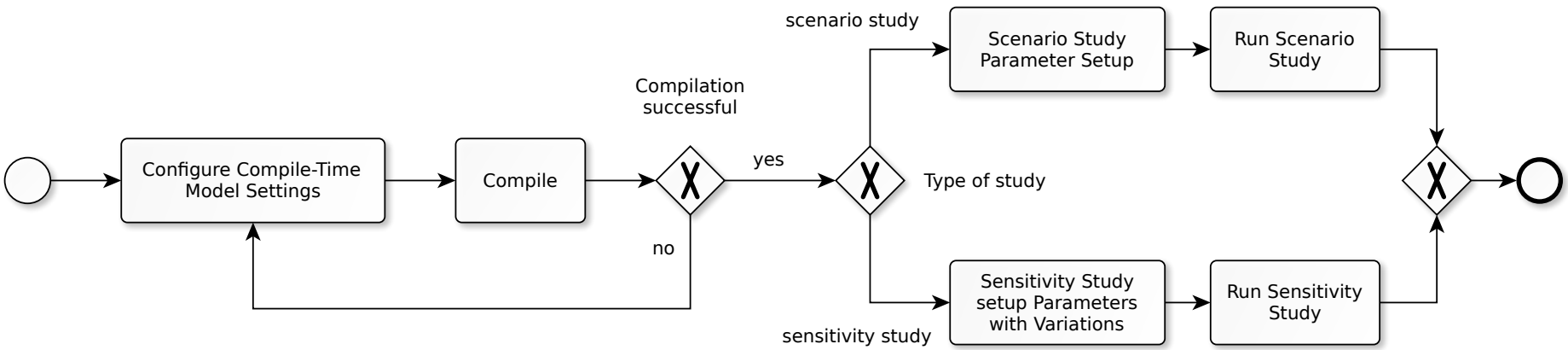


MIT General Circulation Model

- Case Study
- Docker Container
- Evaluation


no, wrong parameter settings
Tests where successful
Configure Compile-Time Model Settings
Compile
Setup Parameters for Test Scenarios
Run Test Scenarios
Analyse Test Results
no, wrong compile-time settings

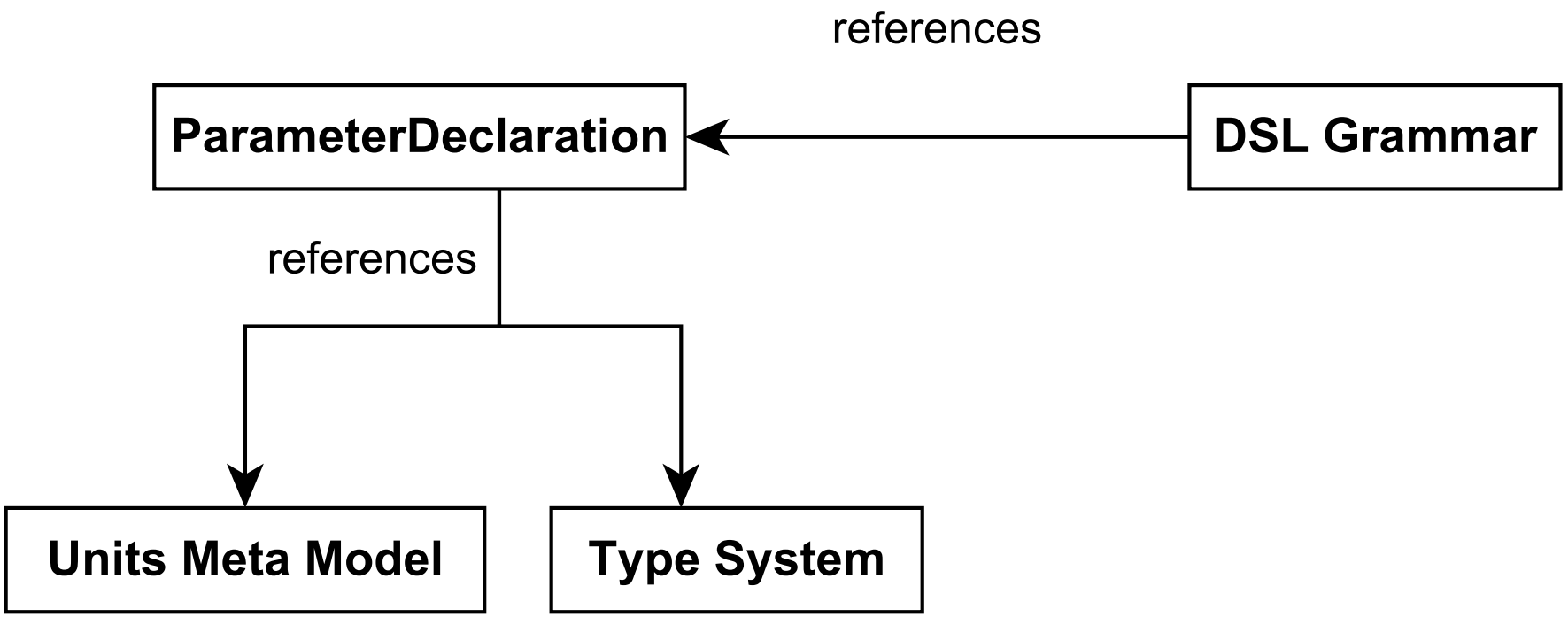
Deployment and Execution Process



- SIZE.H
- eedata
- data
- Optional Files

C A U
Christian-Albrechts-Universität zu Kiel
Technische Fakultät

- Megamodel
  - Grammar
  - Metamodels
- Unit Parser
- Validator
- Generator


references
ParameterDeclaration
DSL Grammar
references
Units Meta Model
Type System

```
Model:
    mainblock=MainBody;

MainBody:
    name=ID ':' model=[declaration::DeclarationModel|ID] '{'
    (paramGroups+=ParamGroup)*
    '}';

ParamGroup:
    name=ID '{'
    (params+=Param)*
    '}';

Param:
    declaration=[declaration::ParamDeclaration|ID] '=' value=Literal (unit=Unit)?;
```

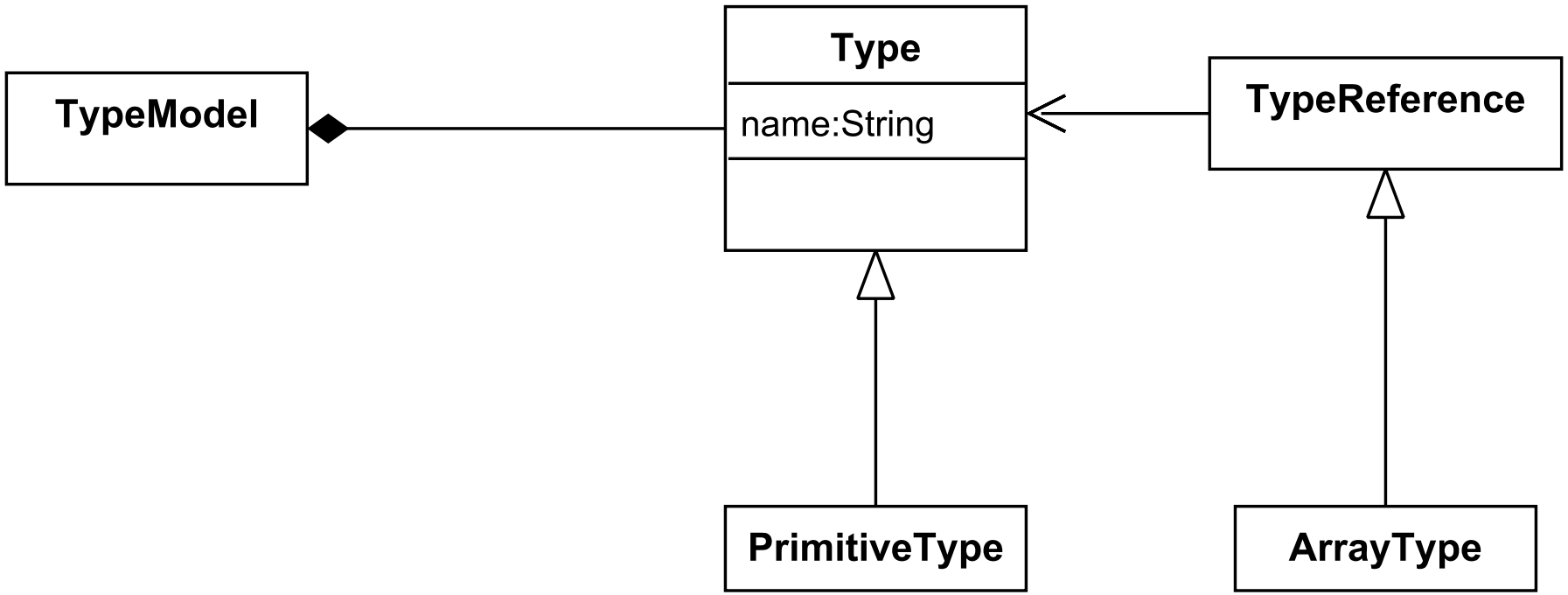
```
Model:
    mainblock=MainBody;

MainBody:
    name=ID ':' model=[declaration::DeclarationModel|ID] '{'
    (paramGroups+=ParamGroup)*
    '}';

ParamGroup:
    name=ID '{'
    (params+=Param)*
    '}';

Param:
    declaration=[declaration::ParamDeclaration|ID] '=' value=Literal (unit=Unit)?;
```

Christian-Albrechts-Universität zu Kiel

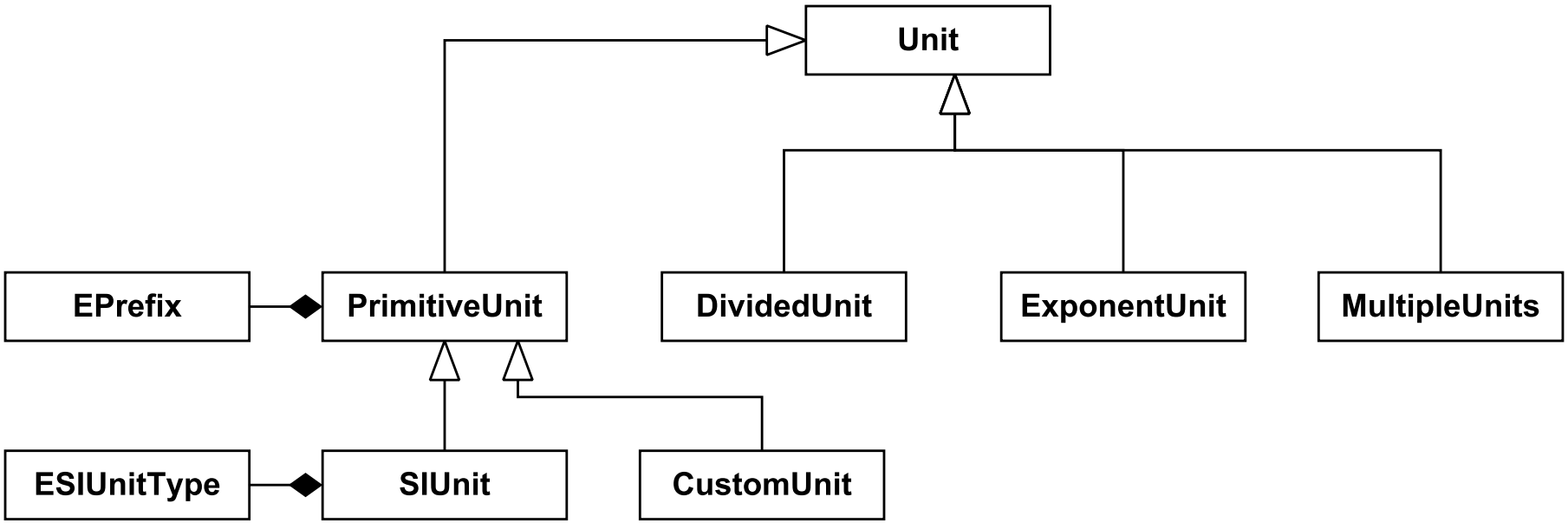
Technische Fakultät

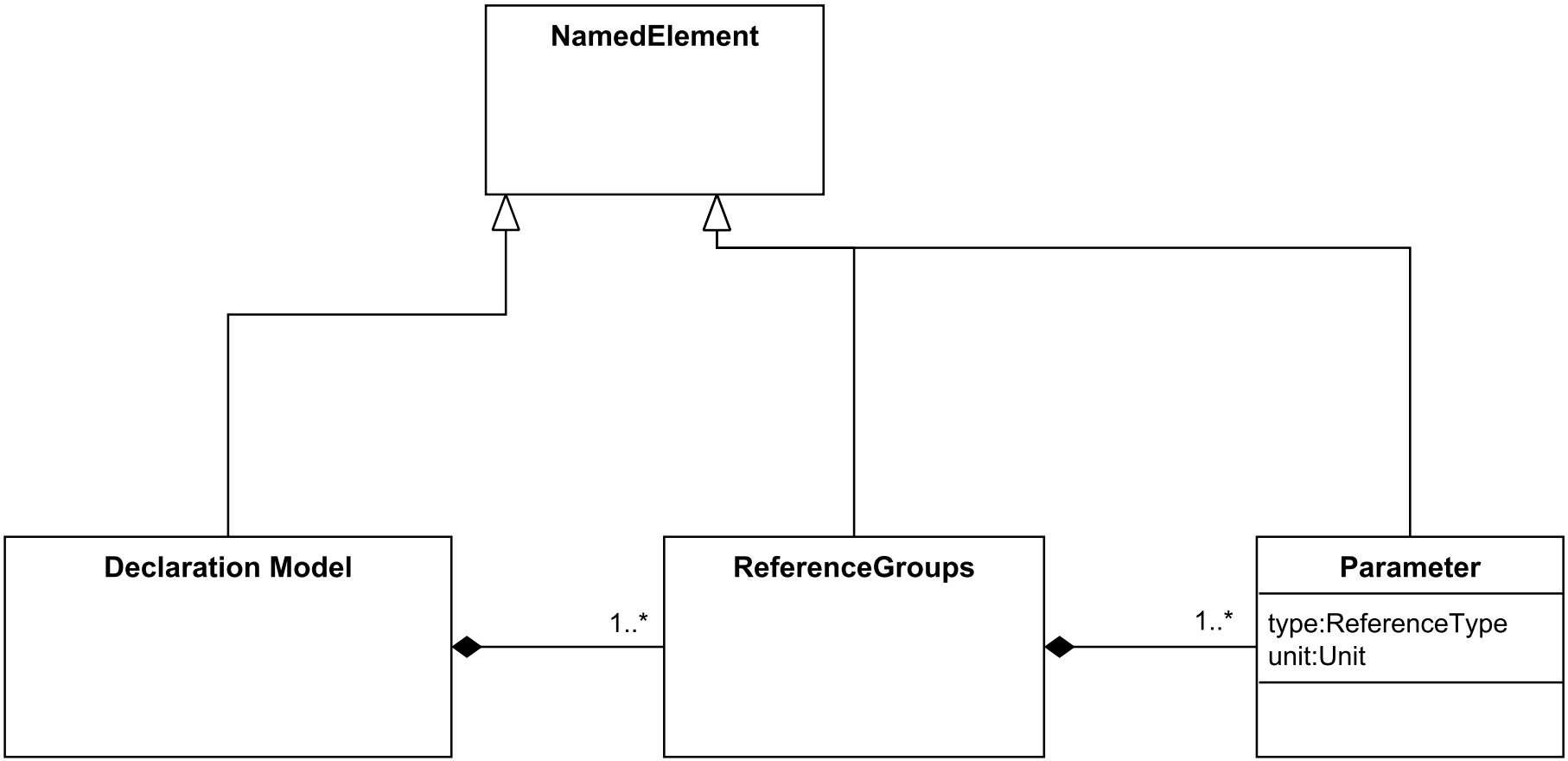
```
Model:
    mainblock=MainBody;

MainBody:
    name=ID ':' model=[declaration::DeclarationModel|ID] '{'
    (paramGroups+=ParamGroup)*
    '}';

ParamGroup:
    name=ID '{'
    (params+=Param)*
    '}';

Param:
    declaration=[declaration::ParamDeclaration|ID] '=' value=Literal (unit=Unit)?;
```

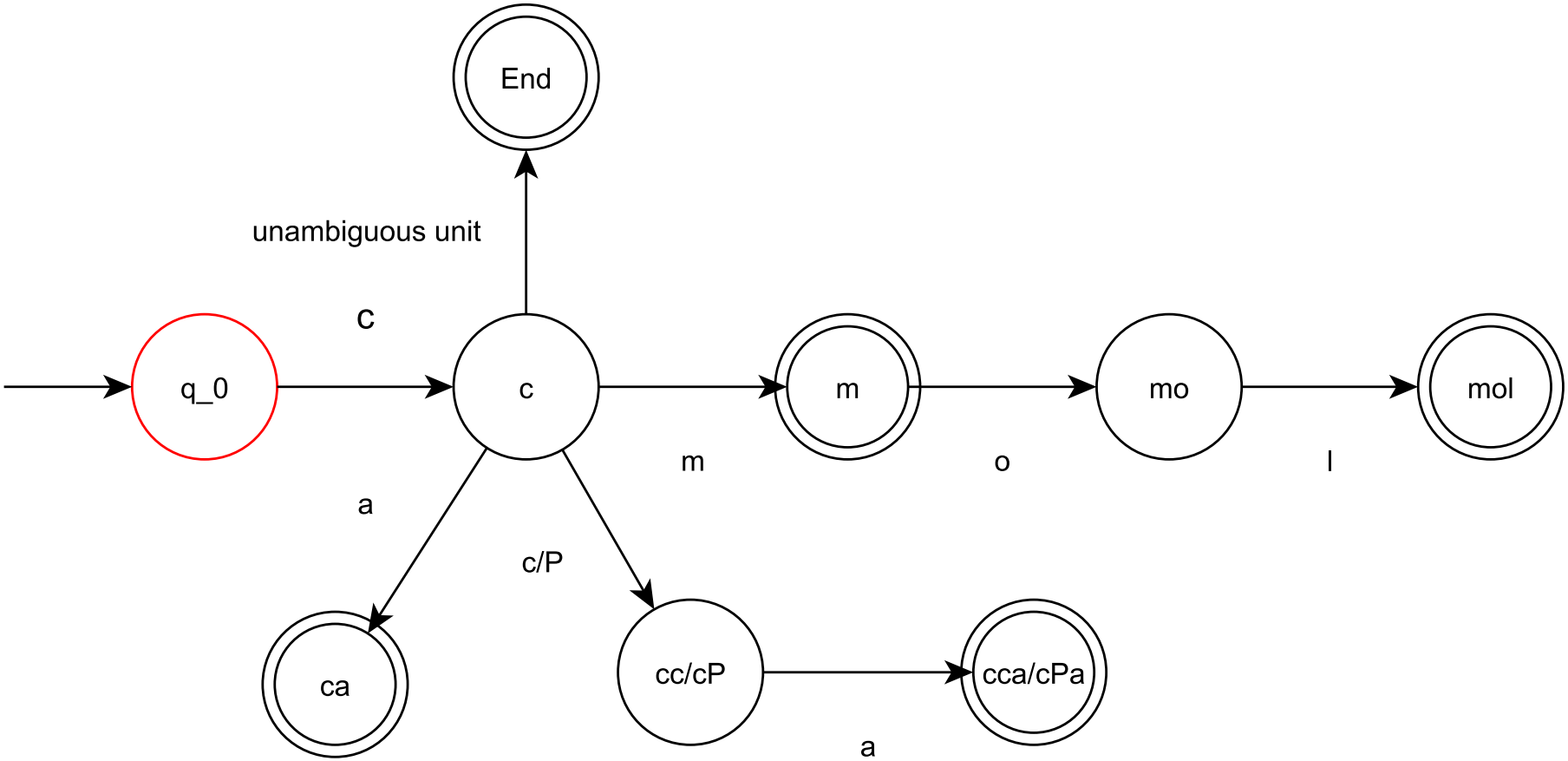
```
Model:
    mainblock=MainBody;

MainBody:
    name=ID ':' model=[declaration::DeclarationModel|ID] '{'
    (paramGroups+=ParamGroup)*
    '}';

ParamGroup:
    name=ID '{'
    (params+=Param)*
    '}';

Param:
    declaration=[declaration::ParamDeclaration|ID] '=' value=Literal (unit=Unit)?;
```


TypeModel
Type
name:String
TypeReference
PrimitiveType
ArrayType

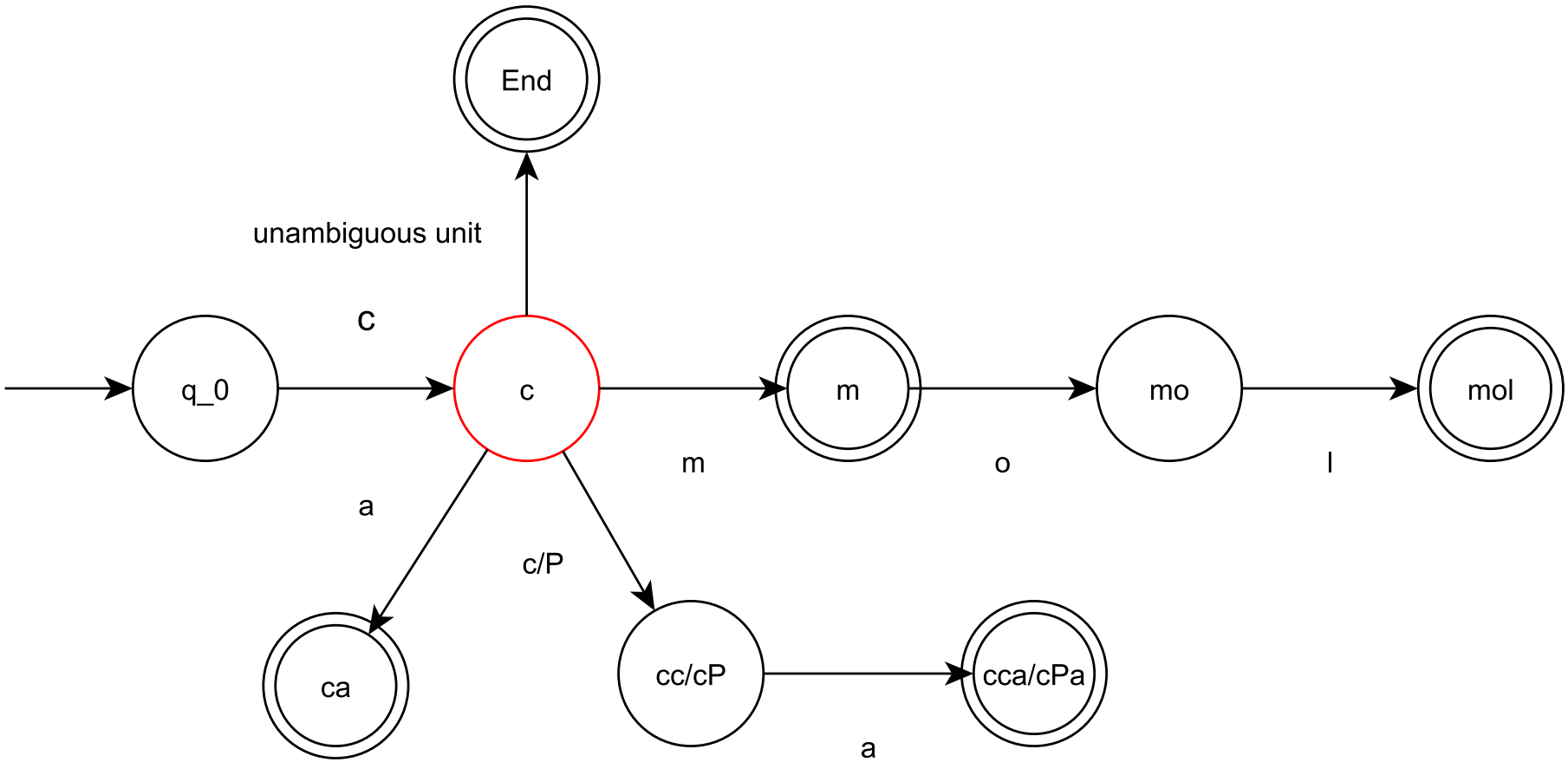
Units Metamodel



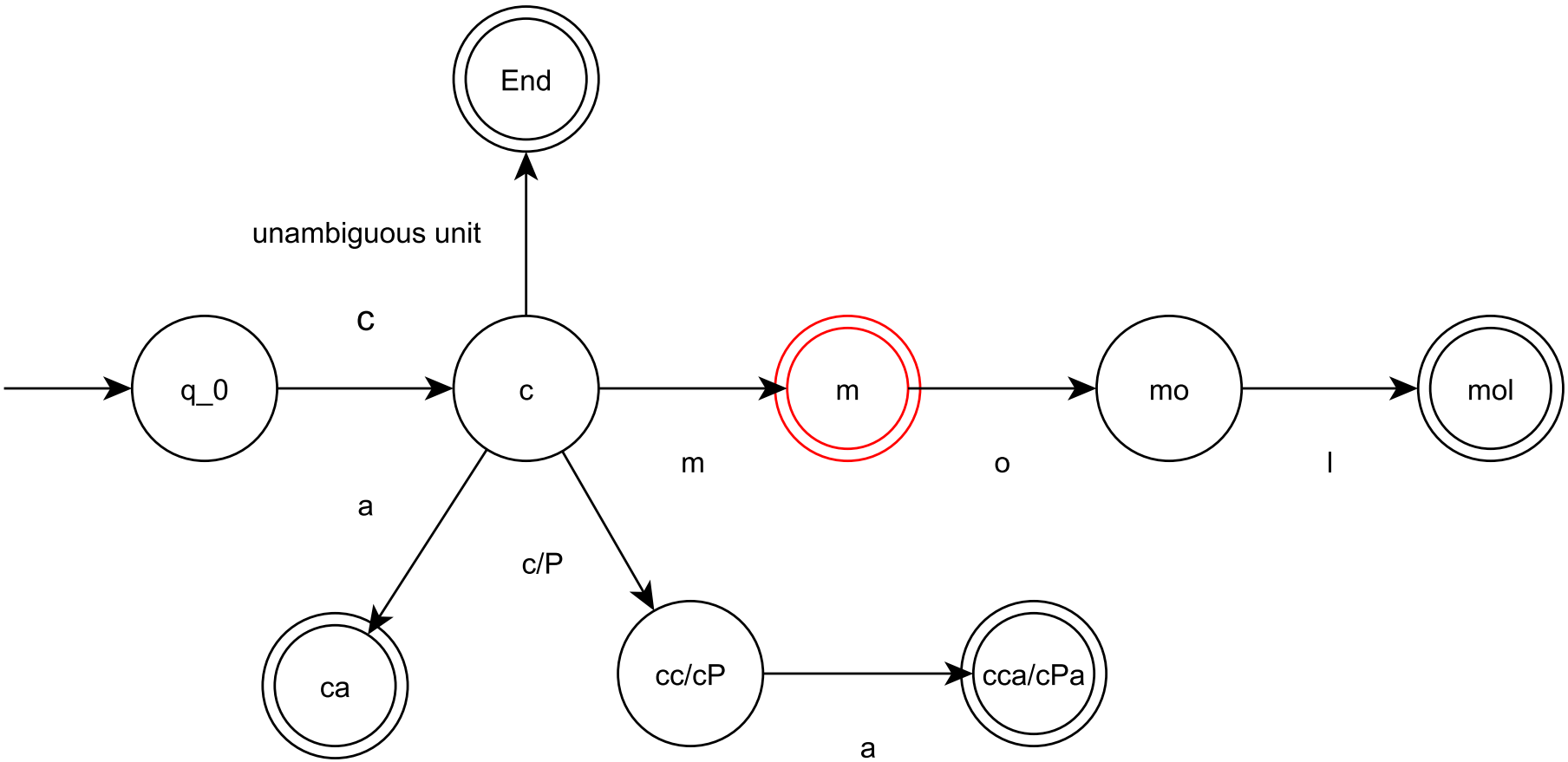

NamedElement
Declaration Model
ReferenceGroups
Parameter
type:ReferenceType
unit:Unit
1..*
1..*


End
unambiguous unit
c
q_0
c
m
mo
mol
m
o
l
a
c/P
ca
cc/cP
cca/cPa
a

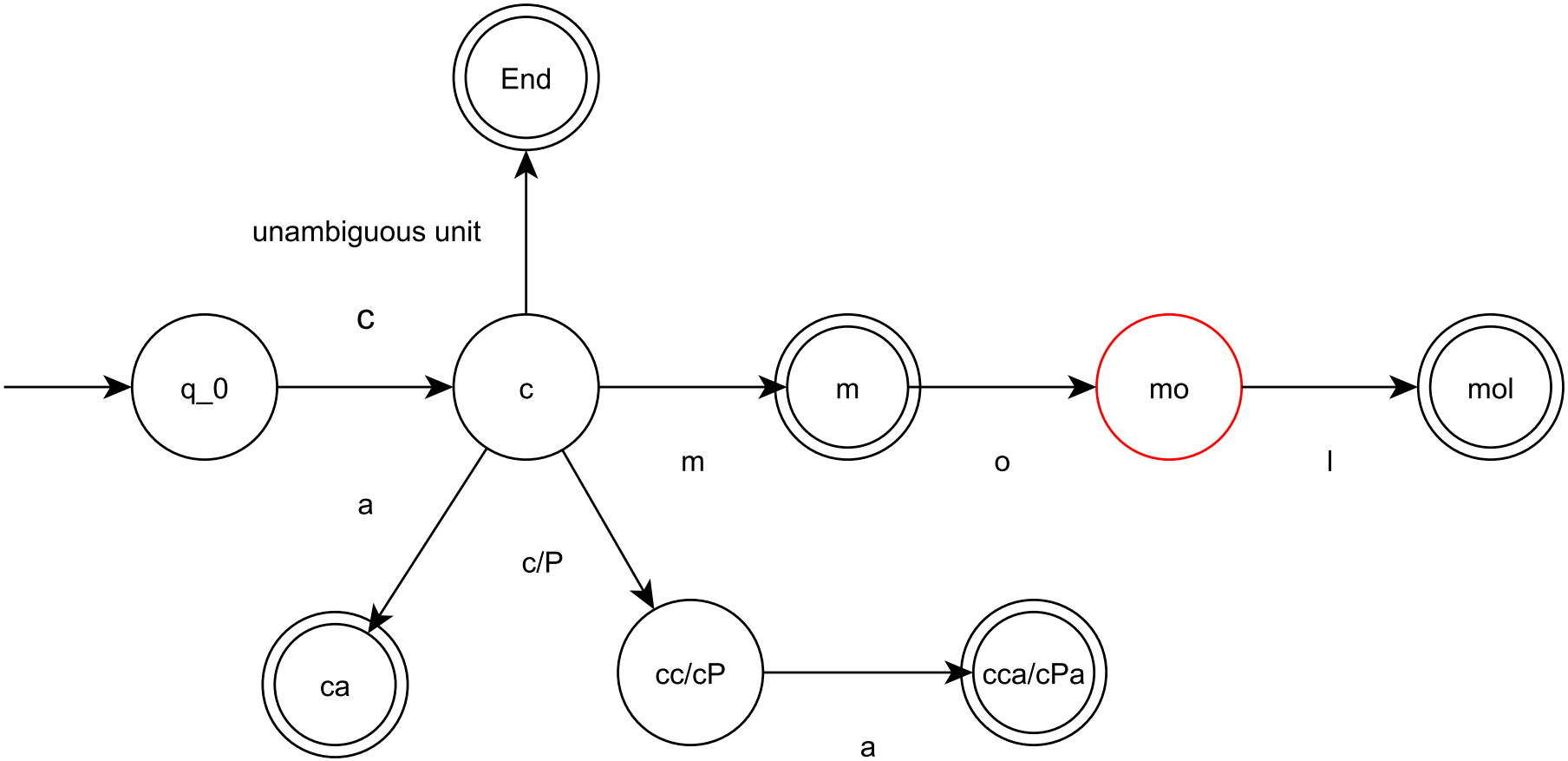
c



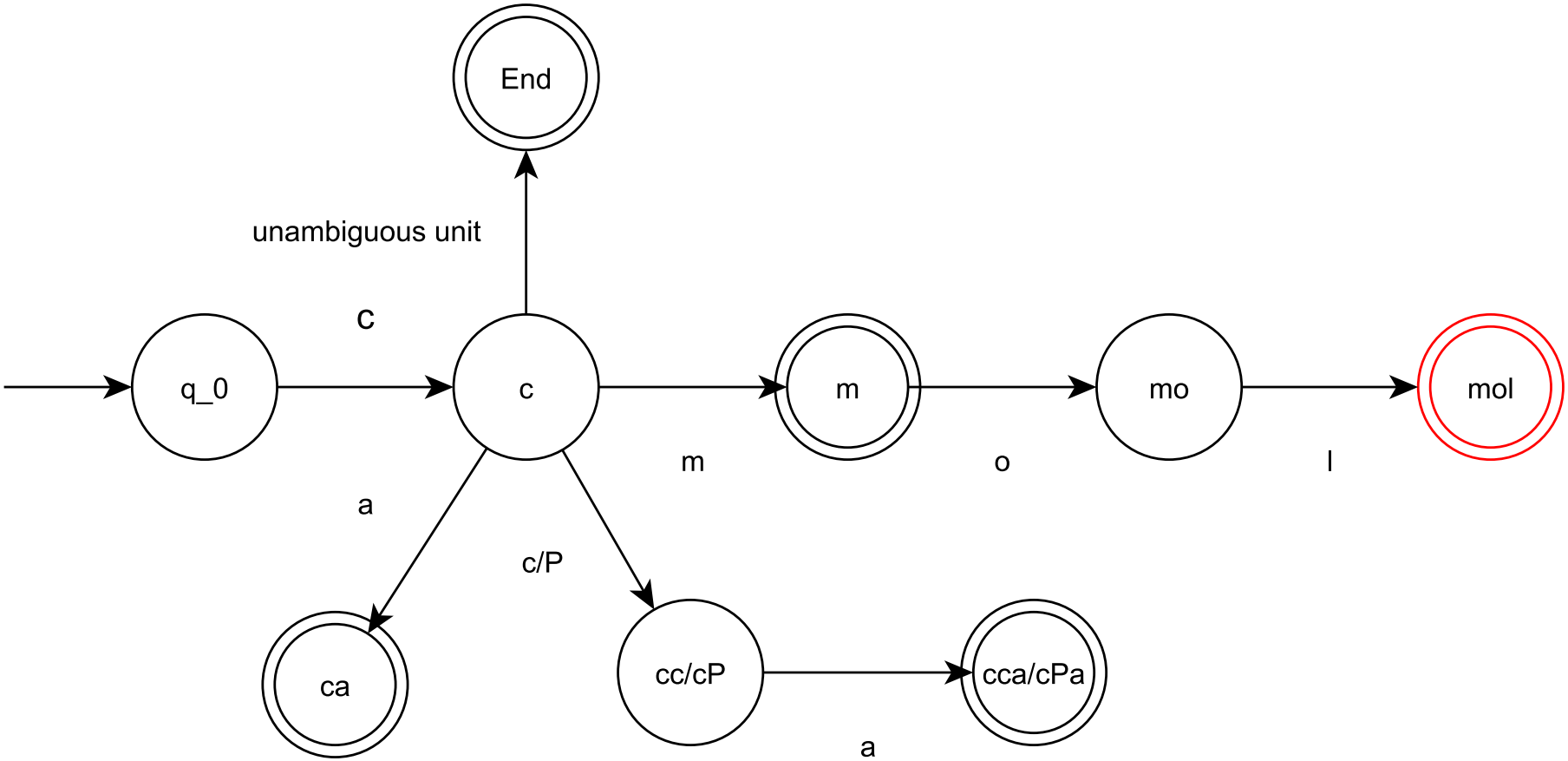
cm



cmo

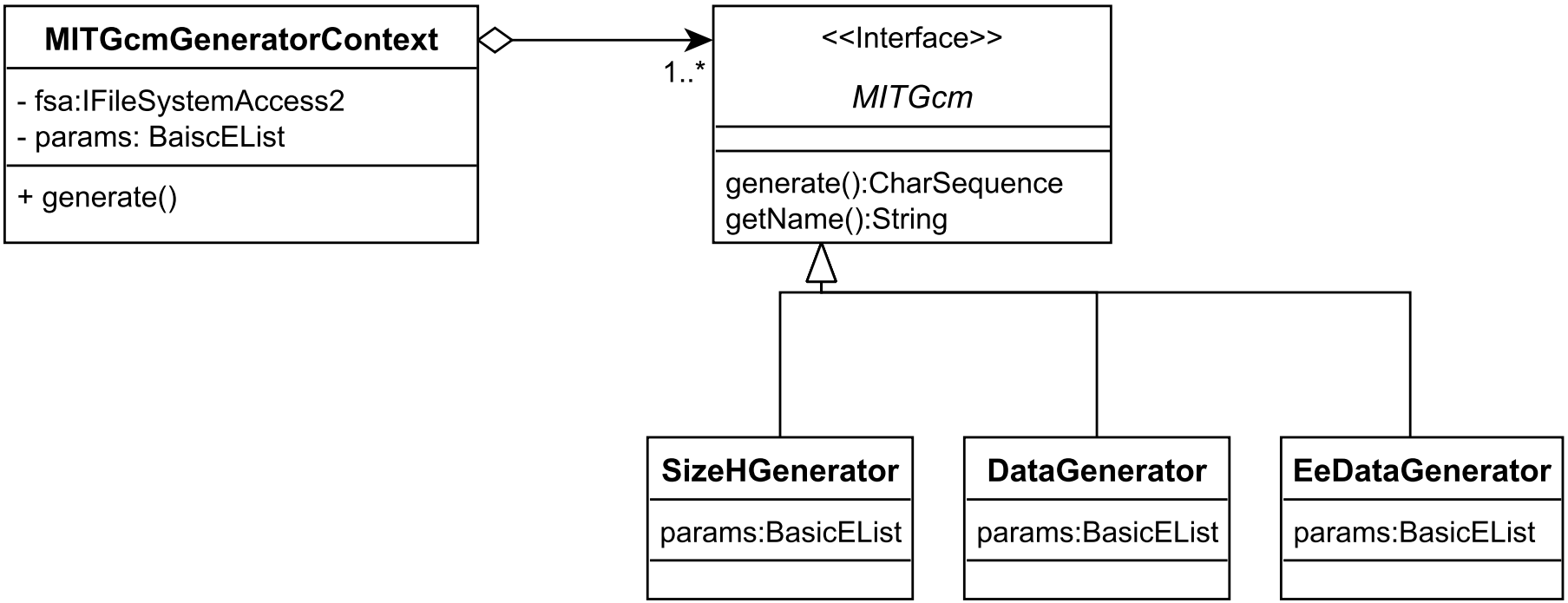


cmol



- Type check
- Unit check
- Check of parameter constraints

- Implementation provided for *data, SIZE.h, eedata*
- Strategy Pattern



Evaluation criteria

- Can we use the DSL to specify proper configurations?
- Can the generator produce MITgcm configurations?
- Is the DSL usable and understandable by a user?

Evaluation setup

- JupyterLab using LSP
- Docker Build for MITgcm
- Three example experiments

Evaluation result

- Model run produces the same output
- Some minor formatting issues
- Test users wish better context help

Conclusion

- Concept works with limitations
- Compatible configuration files
- User experience must be improved

Future Work

- Import support
- Implement generator for all configuration files
- Study another Ocean Models
- Professional advice