Using Feature Models to build Model Transformations Chains

Vincent Aranega[1], Anne Etien[1] & **Sébastien Mosser** [2,3]

(1) LIFL, Université Lille 1, France
(2) SINTEF IKT, Oslo, Norway
(3) I3S, Université Nice - Sophia Antipolis, France

Journées 2013 du GDR GPL, Session COSMAL
03.04.2013, Nancy, France

Vincent Aranega, Anne Etien, Sébastien Mosser. “Using Feature Model to build Model Transformation Chains” in Proceedings of the 15th International Conference on Model Driven Engineering Languages & Systems (MODELS), ACM/IEEE, pages 1-17, Innsbruck, Austria, 01-05 october 2012

metamodel

conforms to

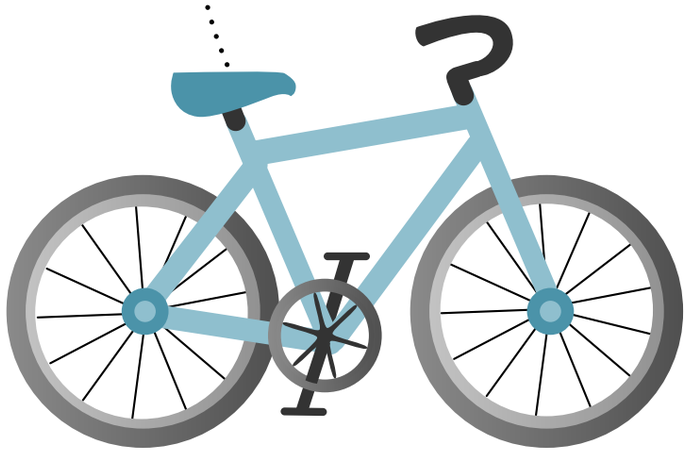
model

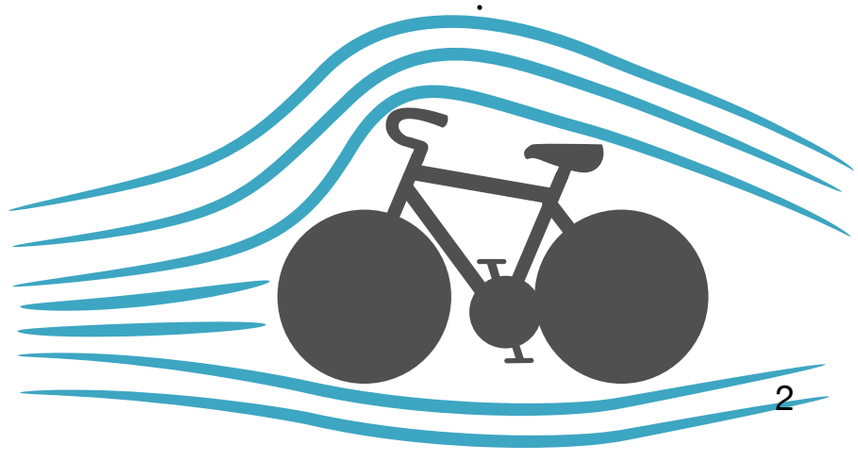
Transformation

metamodel’

conforms to

model’





metamodel

???

metamodel’

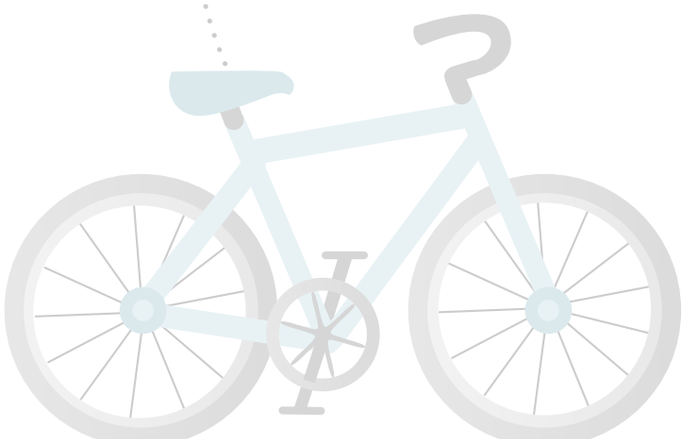
conforms to

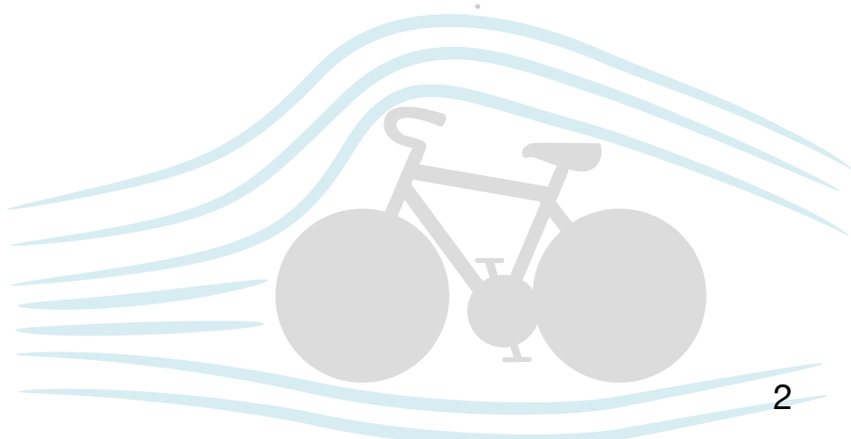
conforms to

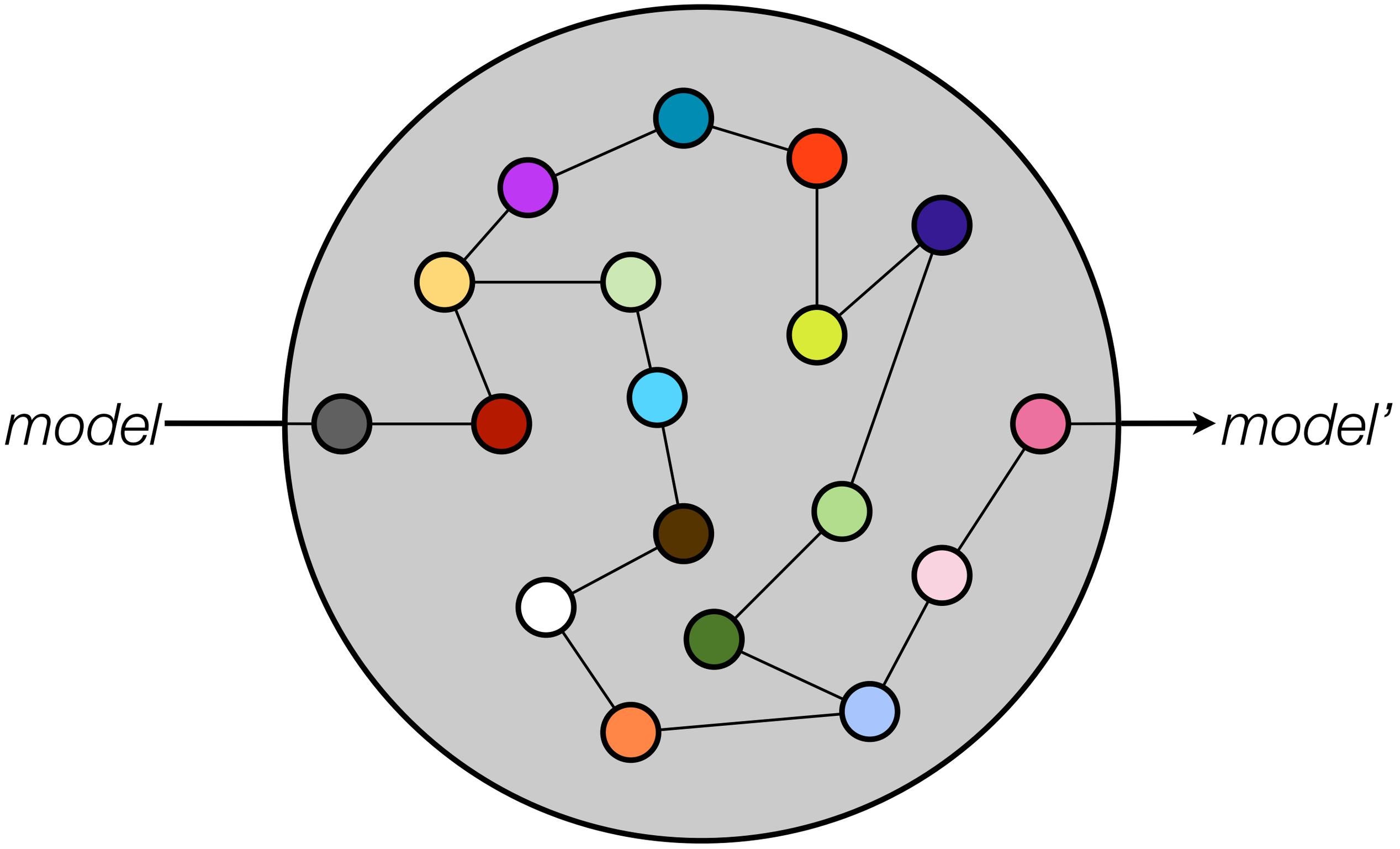
model

Transformation

model’







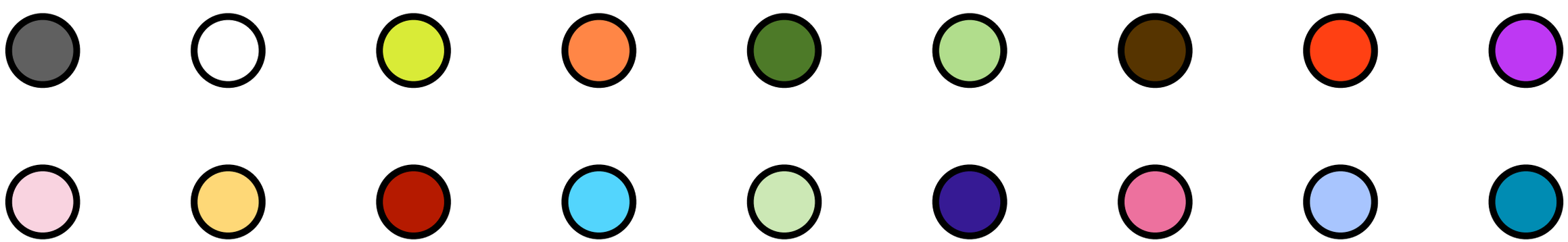
Separation of Concerns ⇒ Family of Transformations

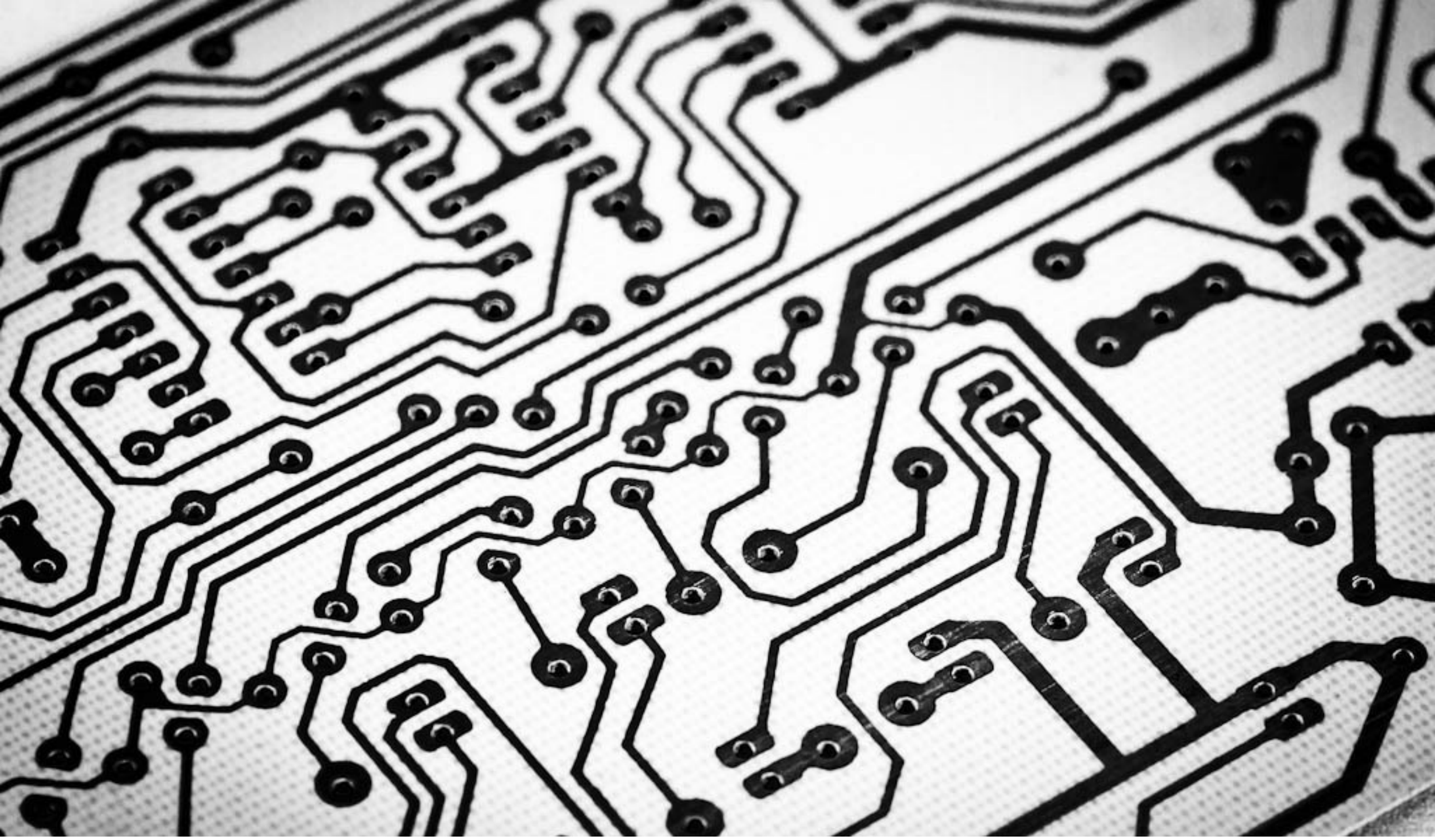
We simply shifted the problem!

How to organize the transformation library?

How to handle transformation dependencies?

How to derive executable transformation chains ?





The Gaspard2 library

Case study

Gaspard2

UML + MARTE

Gaspard Model

transformation chain

VHDL

SystemC

OpenCL

http://www.lifl.fr/west/gaspard/

Transformation	Intention
tiler2task	- Keep repetitions **hierarchy**
gpuApi	- Manage **hybrid GPU-CPU** computing
pThread	- Manage **buffered synchronisation by bloc**
sequentialC	- Generate **sequential** C code
barrier	- Manage **barrier synchronisation** for **OpenMP**
shape2loop	- **Develop repetitions** in the generated systems
scheduling	- Manage **simple** scheduling
poly_loop	- Manage **polyhedron** optimised **scheduling**
explicitAllocation	- **Explicitly** place tasks on processors
memorymapping	- Manage **absolute** memory **addresses**
tilerMapping	- Manage **tiler** (*i.e.* task distributing data) mapping on computing unit
shared	- Manage the shared **memory type**
openCL	- Generate **OpenCL** code for **scientific computation** purposes
openMP	- Generate **OpenMP** code for **scientific computation** purposes
systemcPA	- **Bind** SystemC **architecture** with SystemC **application**
systemcBind	- Manage SystemC **data exchanges**
systemcStruct	- Manage SystemC **architecture**
pthreadGen	- Generate **pthread** code for simulation purposes
functional	- Introduce **functional** abstraction

Transformation	Intention
tiler2task	- Keep repetitions **hierarchy**
gpuApi	- Manage **hybrid GPU-CPU** computing
pThread	- Manage **buffered synchronisation by bloc**
sequentialC	- Generate **sequential** C code
barrier	- Manage **barrier synchronisation** for **OpenMP**
shape2loop	- **Develop repetitions** in the generated systems
scheduling	- Manage **simple** scheduling
poly_loop	- Manage **polyhedron** optimised **scheduling**
explicitAllocation	- **Explicitly** place tasks on processors
memorymapping	- Manage **absolute** memory **addresses**
tilerMapping	- Manage **tiler** (*i.e.* task distributing data) mapping on computing unit
shared	- Manage the shared **memory type**
openCL	- Generate **OpenCL** code for **scientific computation** purposes
openMP	- Generate **OpenMP** code for **scientific computation** purposes
systemcPA	- **Bind** SystemC **architecture** with SystemC **application**
systemcBind	- Manage SystemC **data exchanges**
systemcStruct	- Manage SystemC **architecture**
pthreadGen	- Generate **pthread** code for simulation purposes
functional	- Introduce **functional** abstraction

How many transformation chains «in general» ?

|chains| |model → text| |model → model|

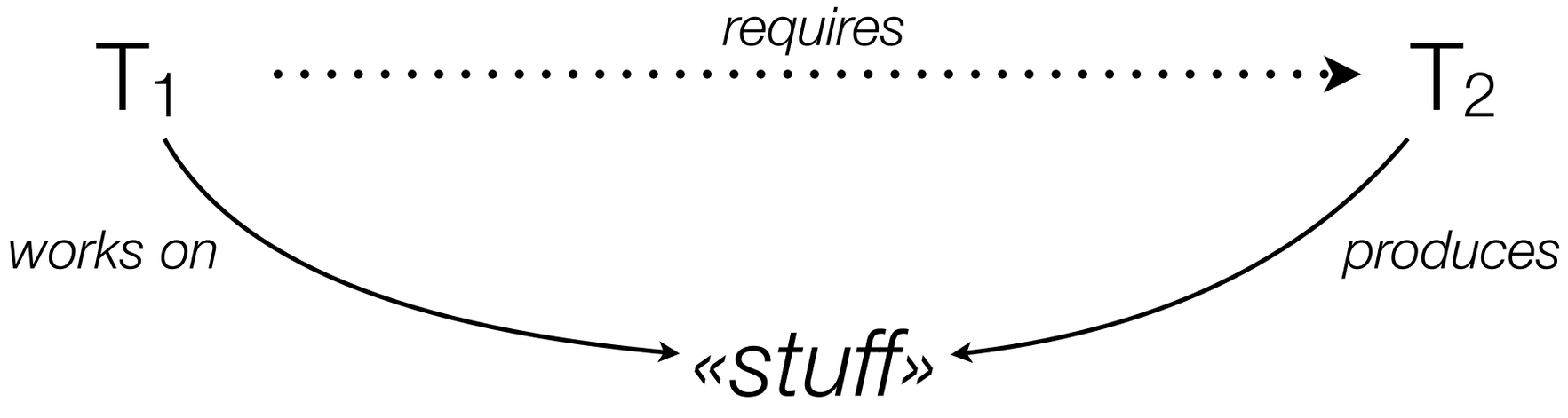
$$N_{T \cup M} = m + (m + 1) \sum_{k=1}^{n} P(k, n)$$

- A model to text transformation directly
- With or without a model to text transformation:
  - A sequence of model to model transformation

m = 5, n=14

≫ 6,5.10¹²

Problem: Not all of these chains make sense!

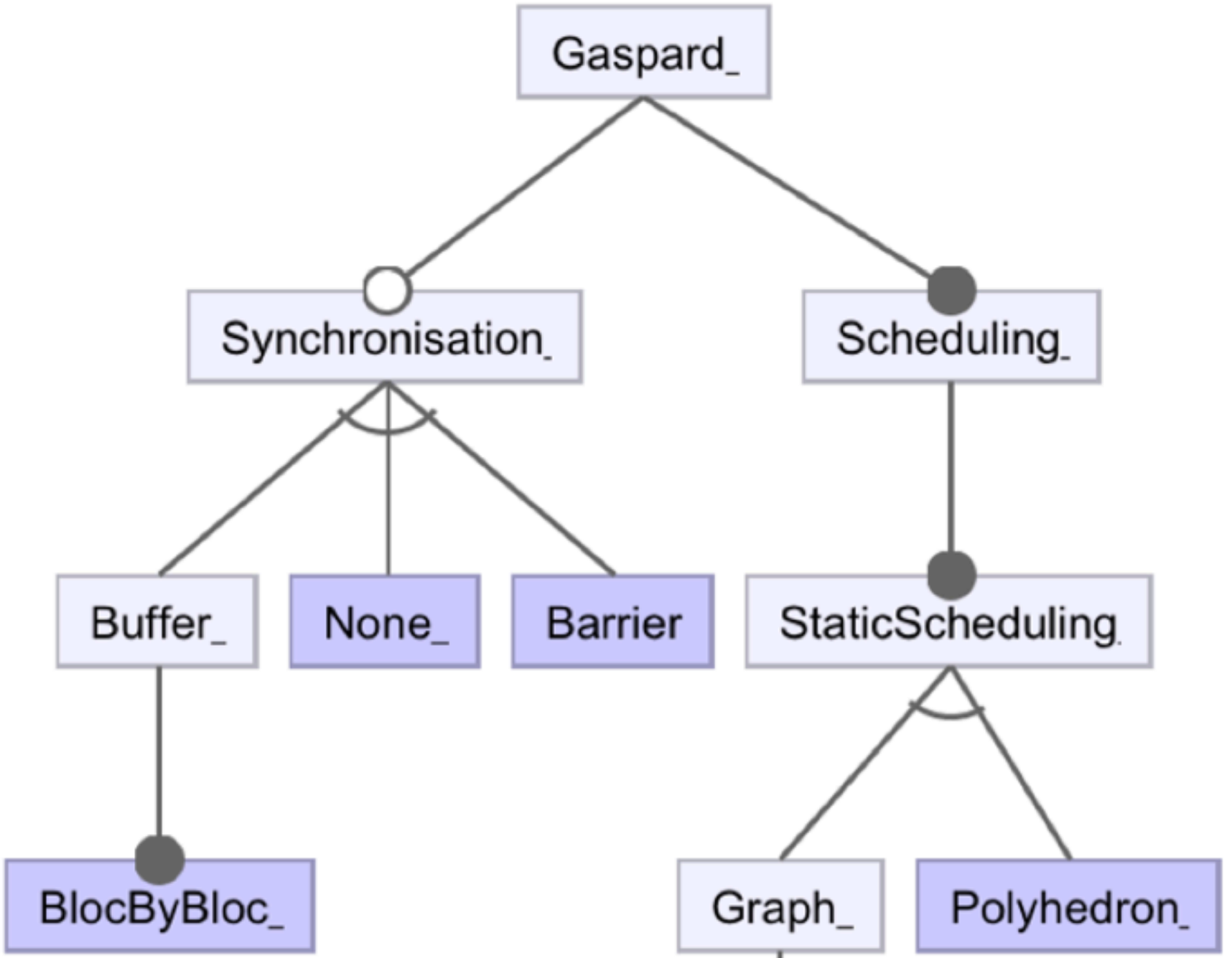


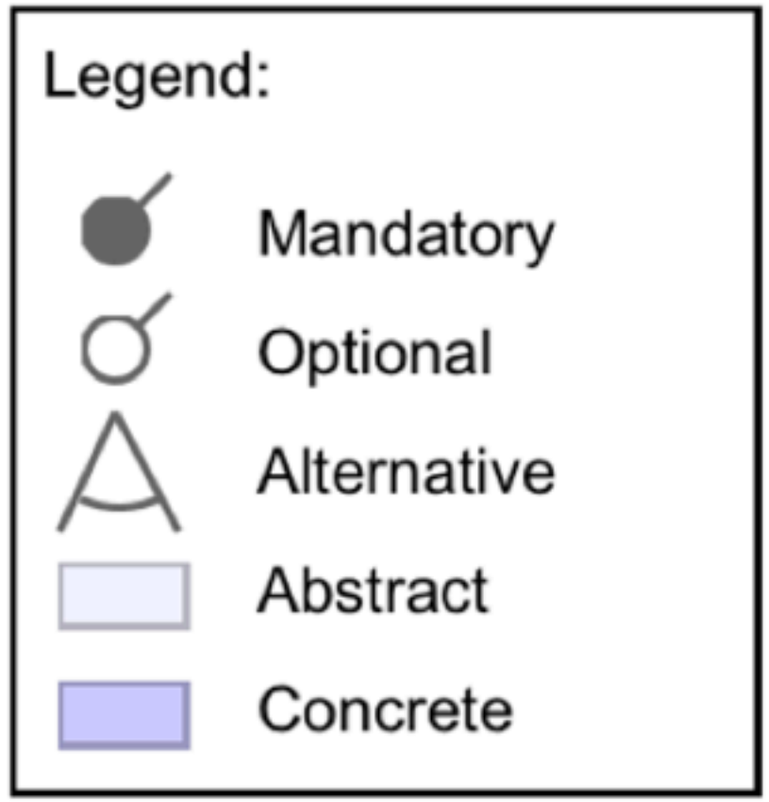
AbsoluteComputation

Develop

Repetition

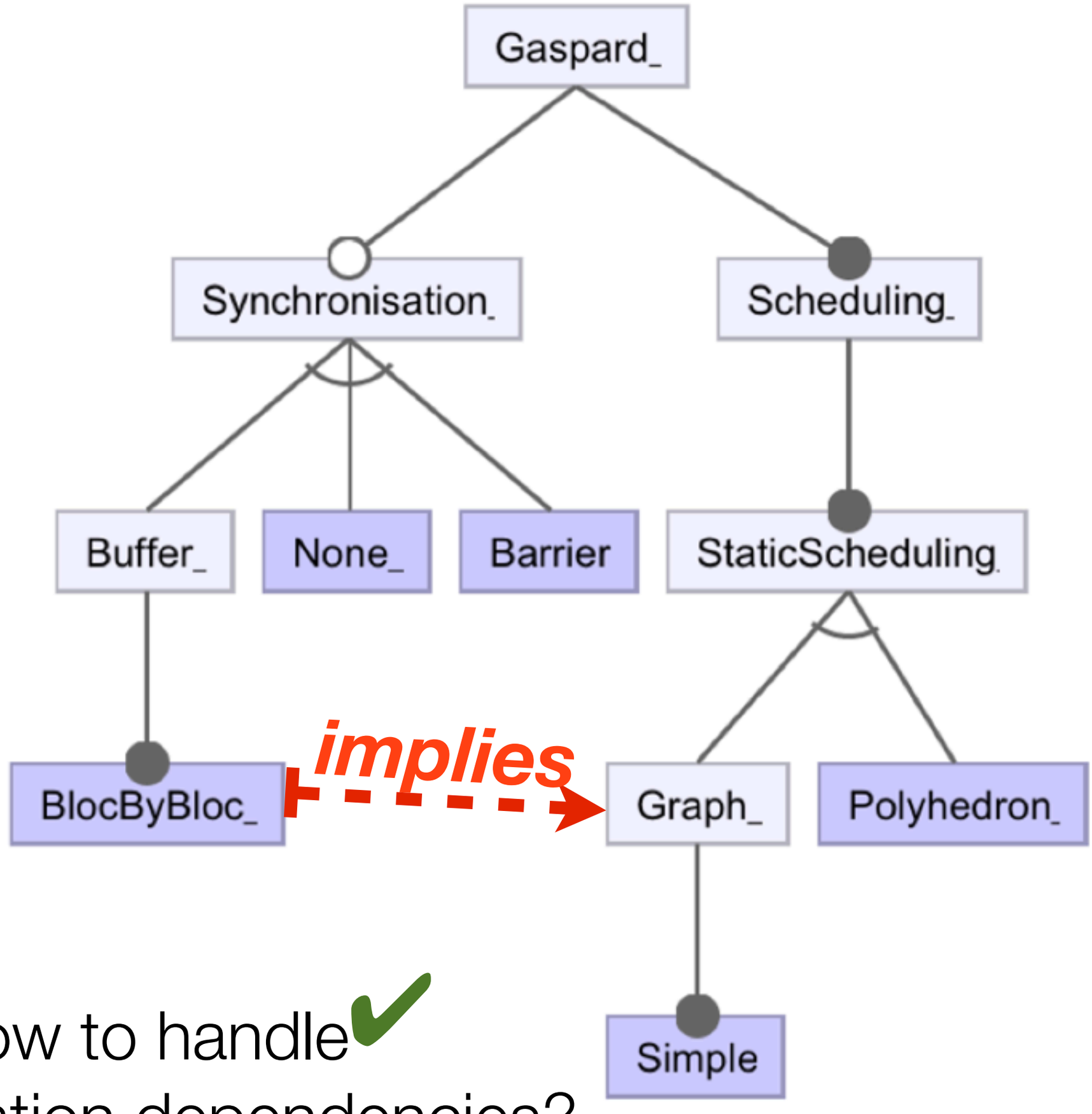
Contribution: Using Feature Models





Variability Modeling

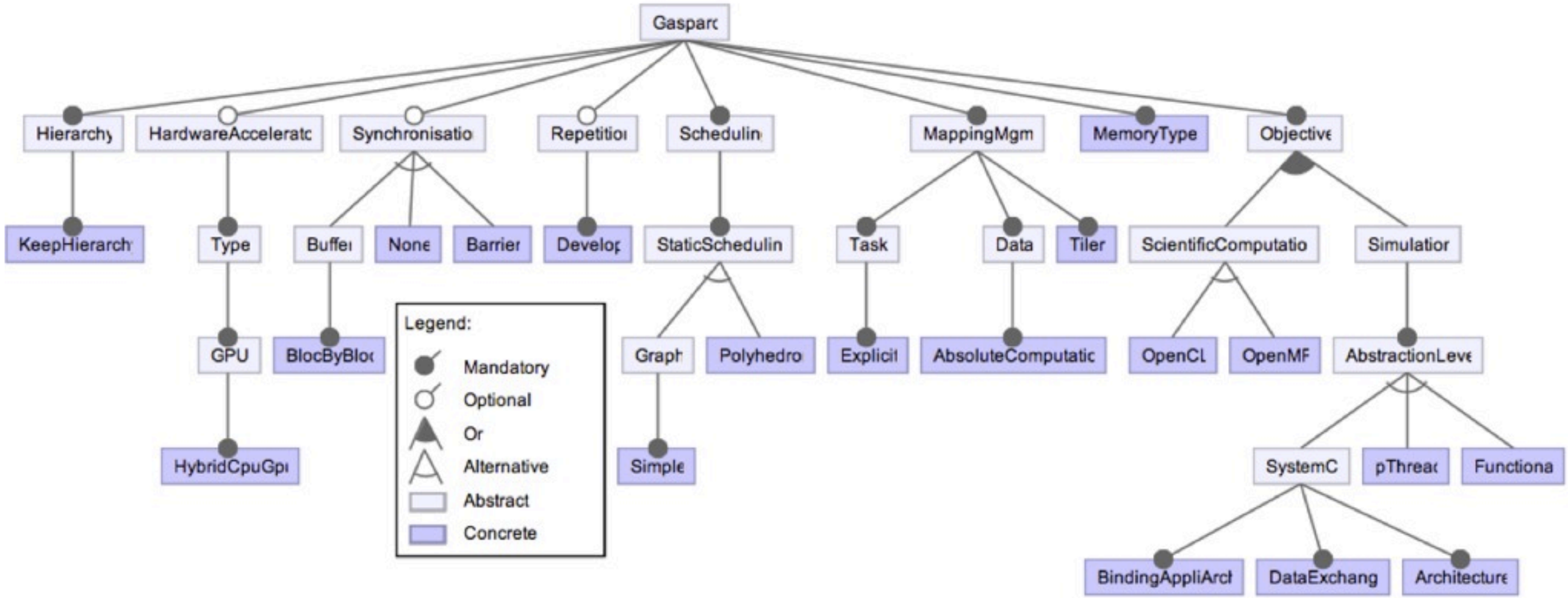
How to organize the transformation library? ✔



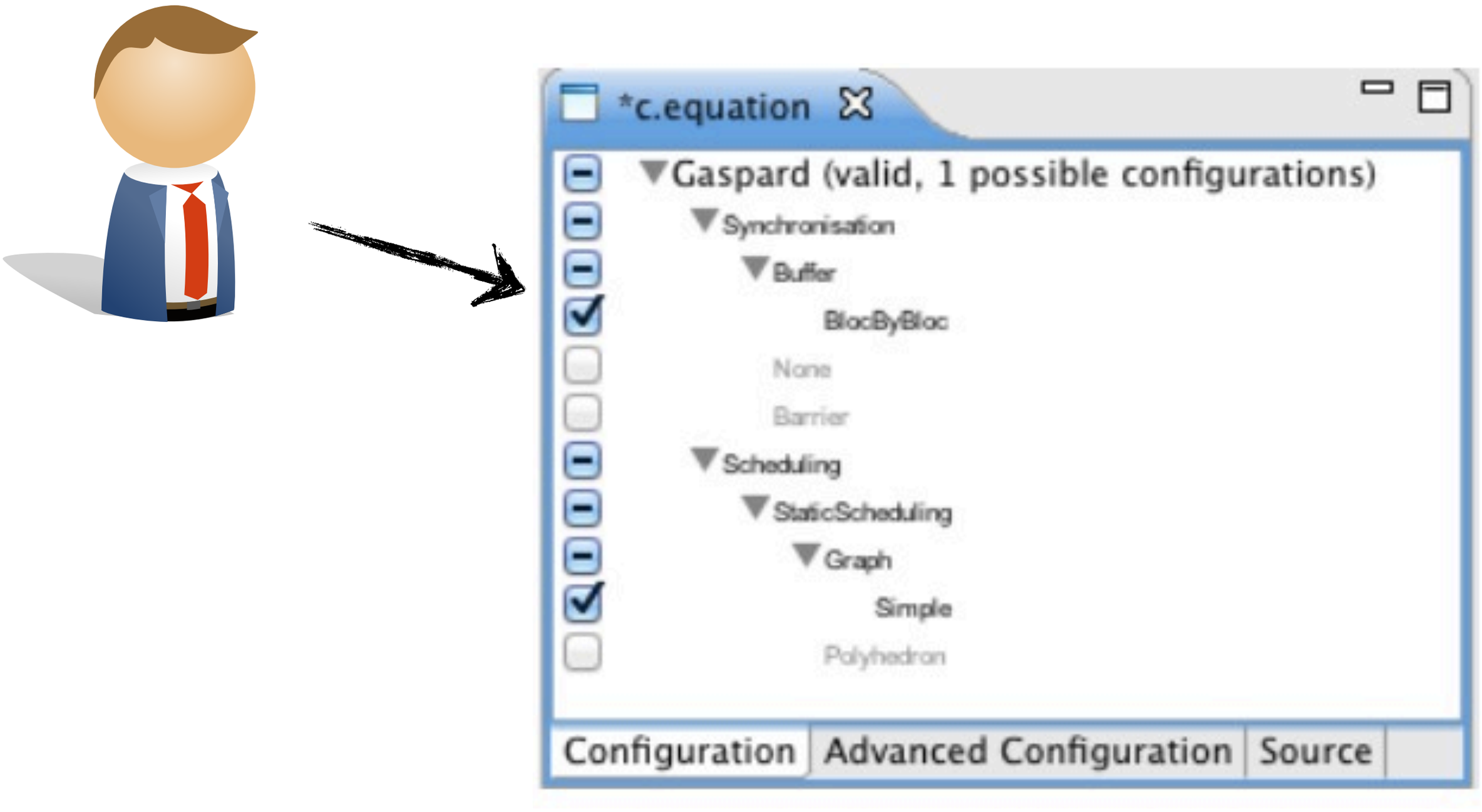
How to handle ✔ transformation dependencies?

```
AbsoluteComputation -> Develop
AbsoluteComputation -> KeepHierarchy
AbsoluteComputation -> Polyhedron
BindingAppliArchi ->
      AbsoluteComputation
BindingAppliArchi -> Architecture
BindingAppliArchi -> BlocByBloc
BindingAppliArchi -> MemoryType
BlocByBloc -> AbsoluteComputation
```

Automated Extraction


Gasparc
Hierarchy
HardwareAcceleratc
Synchronisatio
Repetitio
Schedulin
MappingMgm
MemoryType
Objective
KeepHierarch
Type
Buffe
None
Barrier
Develop
StaticSchedulin
Task
Data
Tiler
ScientificComputatio
Simulatior
GPU
BlocByBloc
Graph
Polyhedro
Explicit
AbsoluteComputatic
OpenCL
OpenMP
AbstractionLeve
HybridCpuGpi
Simple
SystemC
pThreac
Functiona
BindingAppliArcl
DataExchang
Architecture
Legend:
Mandatory
Optional
Or
Alternative
Abstract
Concrete

37 «products»

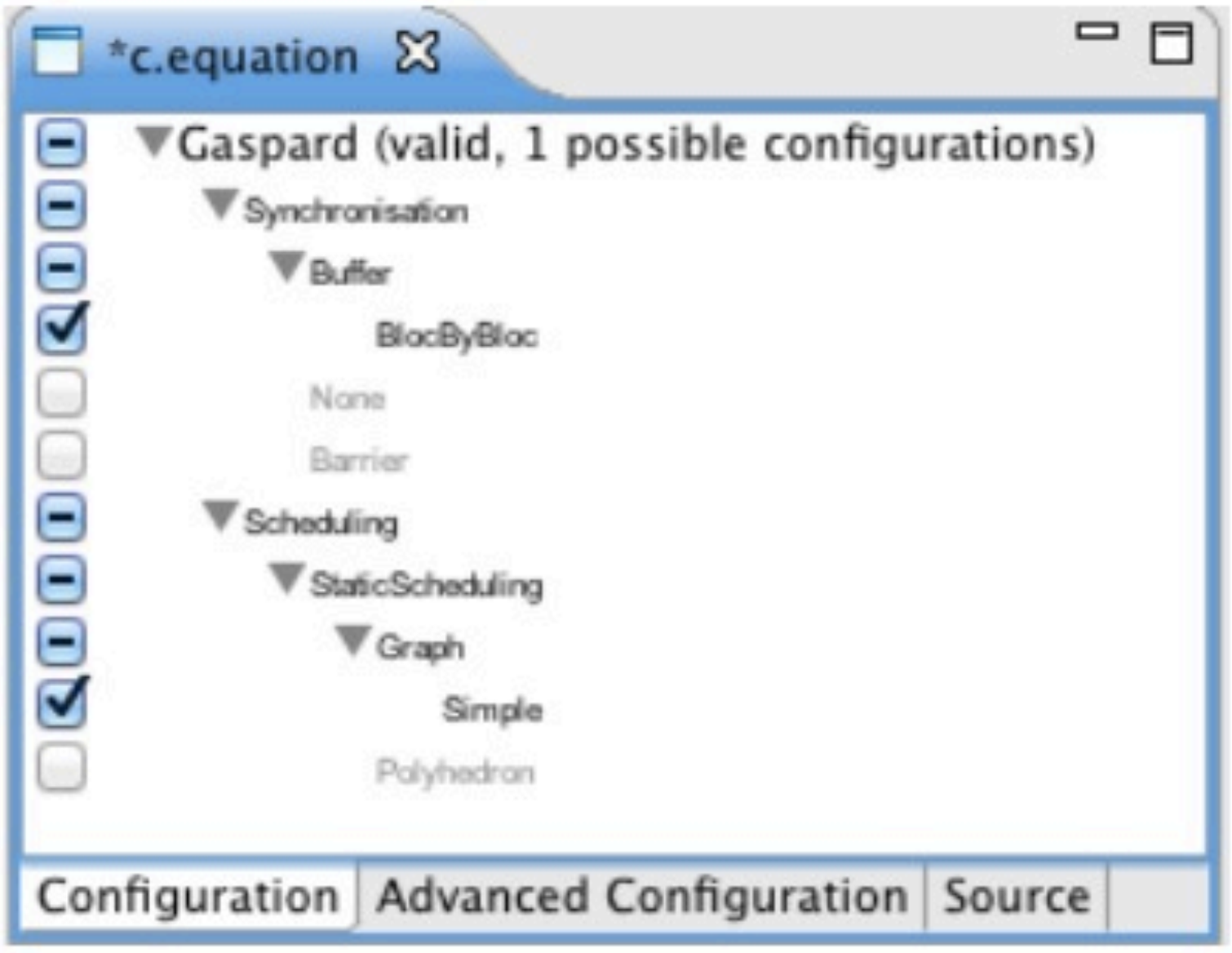


How to derive executable transformation chains? ✔

$$p = \{Gaspard, MemoryType, Polyhedron, Data, Barrier, MappingMgmt, KeepHierarchy, Hierarchy, Tiler, Develop, StaticScheduling, AbsoluteComputation, Task, Explicit, ScientificComputation, Scheduling, Objective, Repetition, Synchronisation, OpenMP\}$$

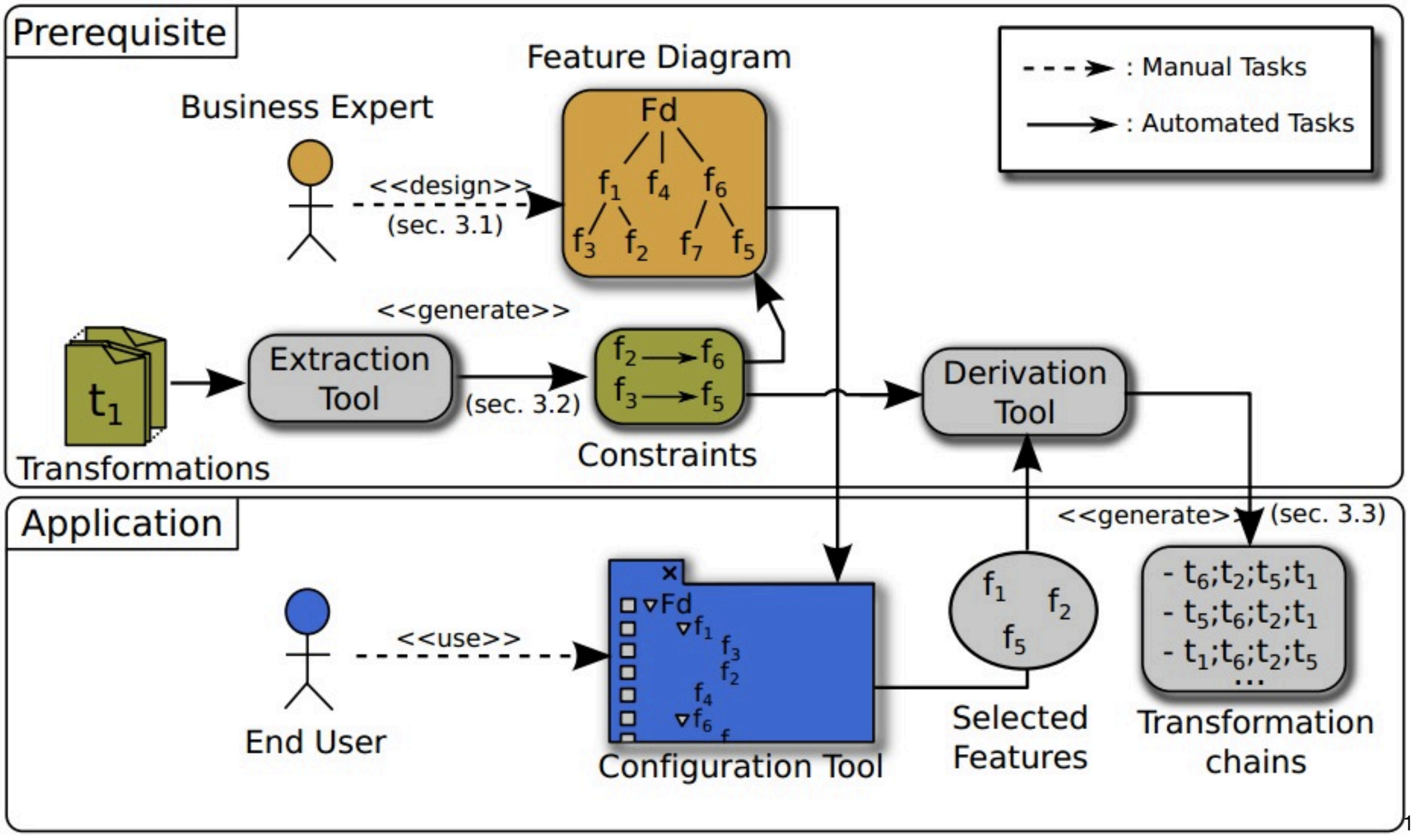
$$T_p = \{explicitAllocation, memMapping, openMP, poly\_loop, shape2loop, tilerMapping\}$$

Feature Requirement	$\rightsquigarrow$	**Transformation Ordering**
$AbsoluteComputation \rightarrow Develop$	$\rightsquigarrow$	$memMapping \rightarrow shape2loop$
$AbsoluteComputation \rightarrow Polyhedron$	$\rightsquigarrow$	$memMapping \rightarrow poly\_loop$
$MemoryType \rightarrow KeepHierarchy$	$\rightsquigarrow$	$memMapping \rightarrow tiler2task$



$$tpl_p = [\ [openMP], [explicitAllocation],$$
$$[memMapping, [shape2loop, poly\_loop, tiler2task]\ ]$$

The Altogether



Conclusions

and perspectives

Summary: Handling Families of Transformations.

How to organize the transformation library? → Feature Model

How to handle transformation dependencies? → Constraints

How to derive executable transformation chains ? → Product Derivation

Perspectives

Enhance constraints support

Typing ⇒ Business

Infer Feature Model architecture

Automated extraction

Generalizing the SPL process

Another domain than transformations

Using Feature Models to build Model Transformations Chains

Vincent Aranega[1], Anne Etien[1] & **Sébastien Mosser** [2,3]

(1) LIFL, Université Lille 1, France
(2) SINTEF IKT, Oslo, Norway
(3) I3S, Université Nice - Sophia Antipolis, France

Thanks for your attention!

Credits: C.line Design & sxc.hu