IMT Mines Alès
École Mines-Télécom

LIRMM

ECOLE DOCTORALE

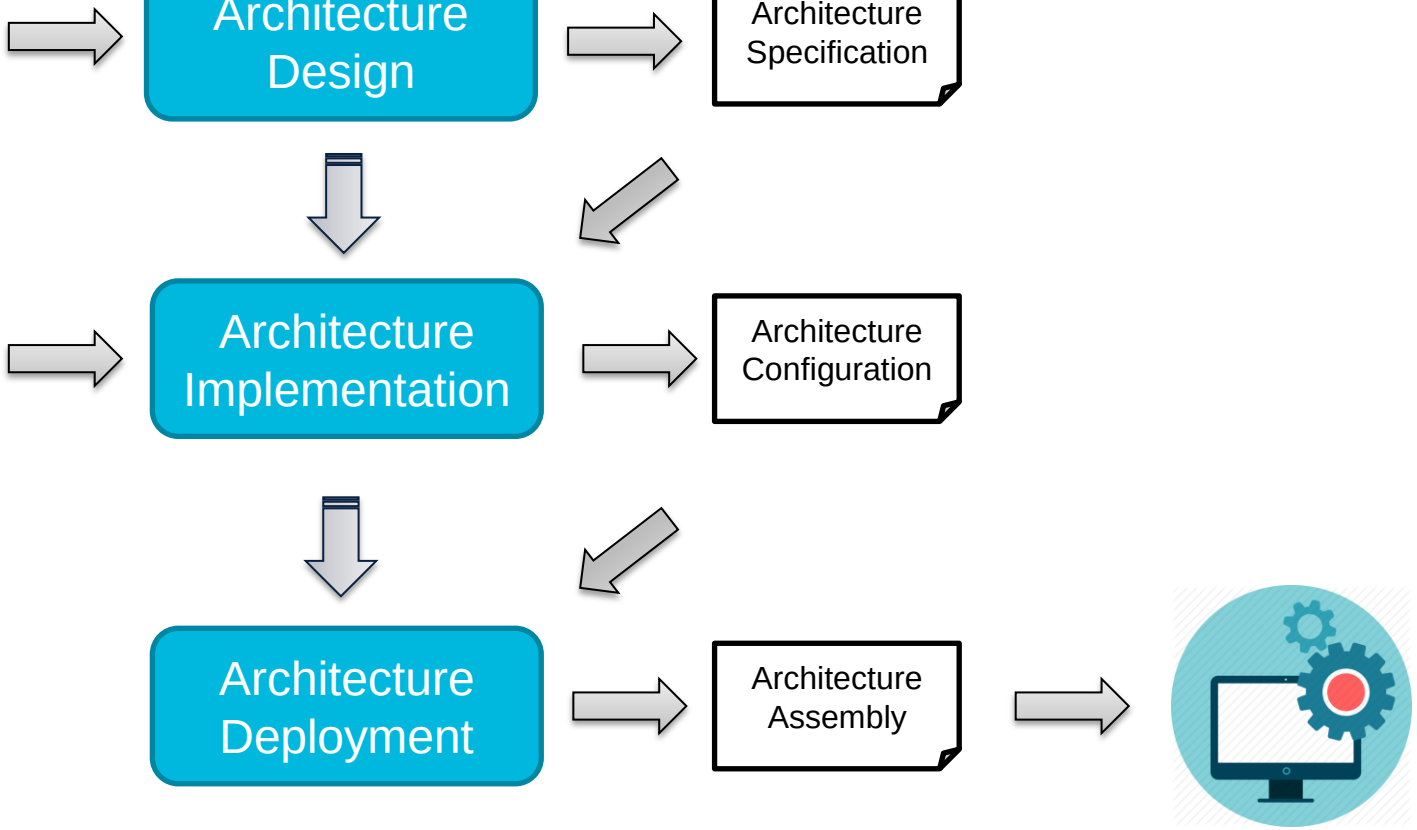
PRELIMINARY STUDY ON VERSION PROPAGATION IN THREE-LEVEL COMPONENT-BASED SOFTWARE ARCHITECTURES

Alexandre Le Borgne*,
David Delahaye+,
Marianne Huchard+,
Christelle Urtado*,
Sylvain Vauttier*

* IMT – Mines Alès, Nîmes, France
+ LIRMM, Montpellier, France

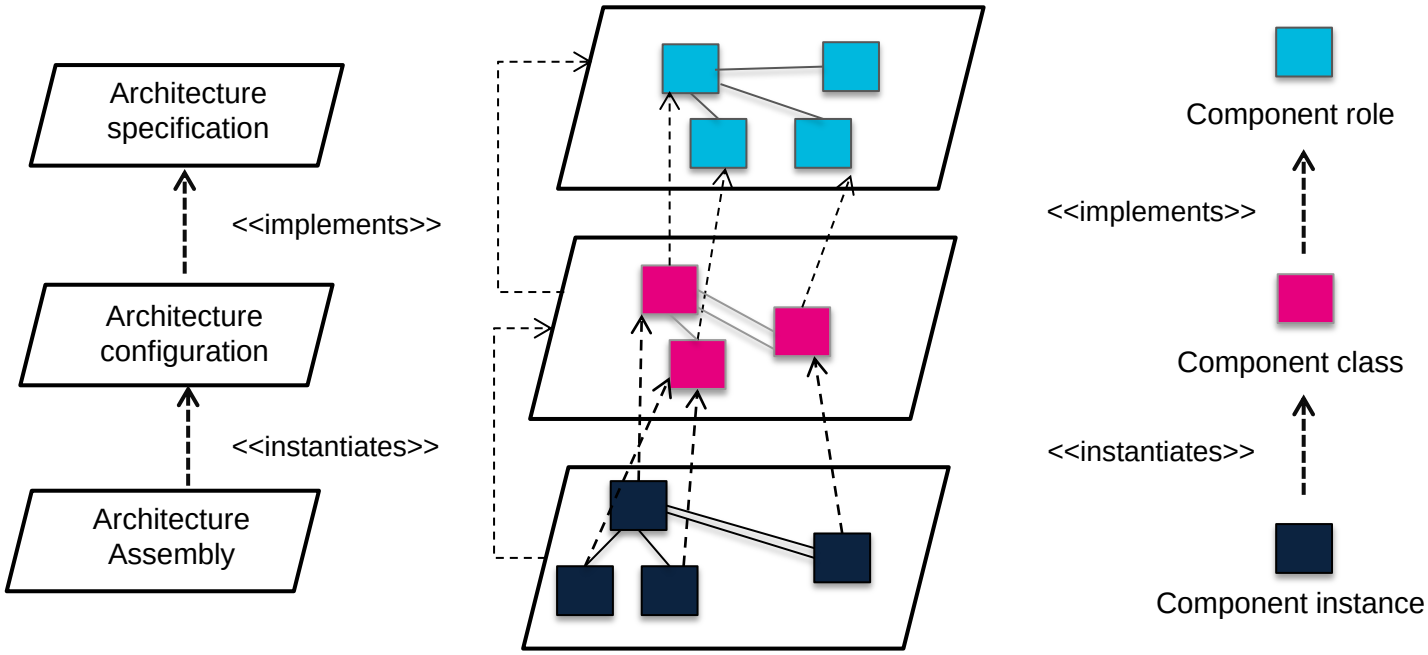
Functional requirements

Component repositories



Dedal

- A three-level architecture description language
  - Providing representations of main software engineering stages
  - Capture architectural decisions
  - foster architecture description reuse



Evolution

- Prevent obsolescence
- Derive new architectures from existing ones
- Preserve traceability
- Avoid inconsistencies (intra-level relation verification)
- Avoid loss of architectural decisions (inter-level relation enforcement)
  - Drift
  - Erosion

Automated evolution

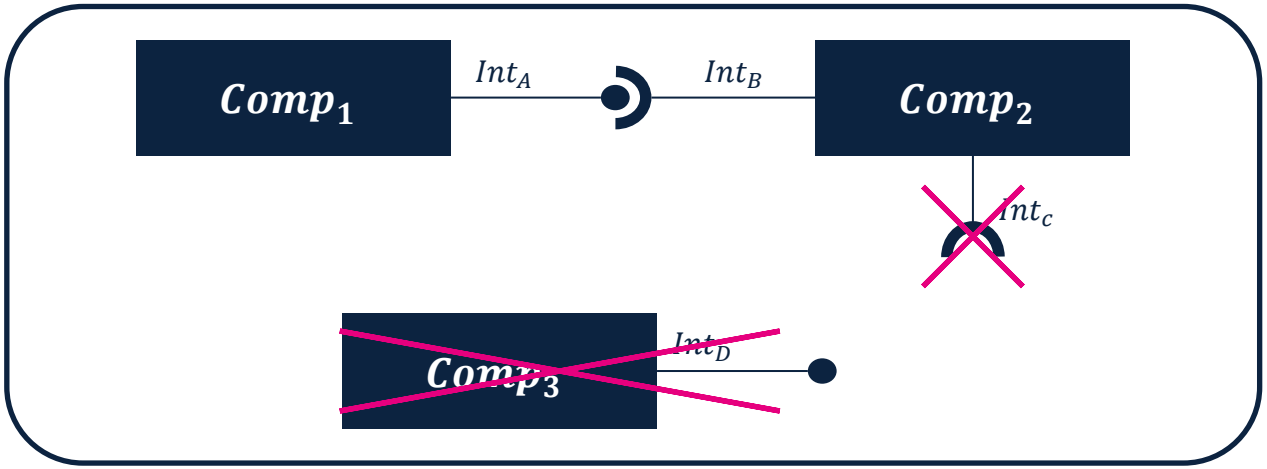
- Automatically propose an evolution plan
  - Co-evolution
  - Propagation of changes within three architecture levels

Formalization of Dedal

- Language B (first-order logic, set theory based formal language)
- formal definition of the relations between components on each architecture description level (intra-level relations)
  - connection, specialization (substitution)
- formal definition of the relations between the different architecture description levels (inter-level relations)
  - implementation, instantiation
- Derived from object type theory (*Liskov* 1993)

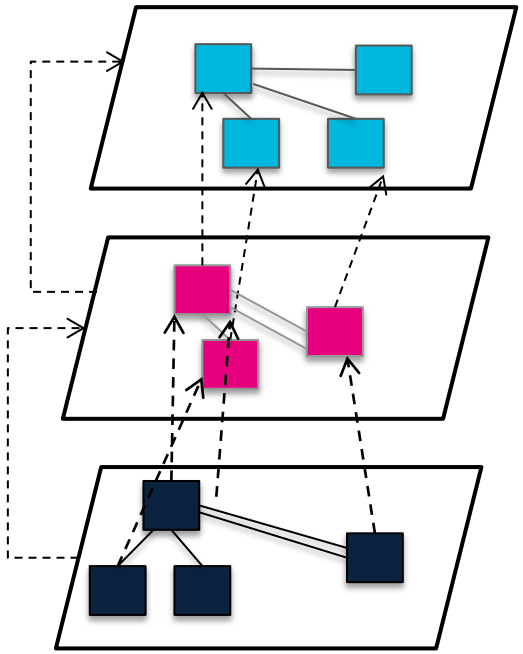
Intra-level consistency

- Name consistency
  - Unique name
- Interface consistency
  - Connected interfaces are compatible
- Interaction consistency
  - Functional objectives are realized (all the required interfaces are connected to compatible provided ones)
  - Architecture definition = connected graph



Inter-level coherence

- All component roles realized by component classes (realize relation)
  & Each connected provided interface in the configuration is included in the specification.
- Every component class from the configuration is instantiated at least once by a component instance in the assembly (instantiate relation)
  & Each connected provider in the assembly is an instance of a provided interface from the configuration.

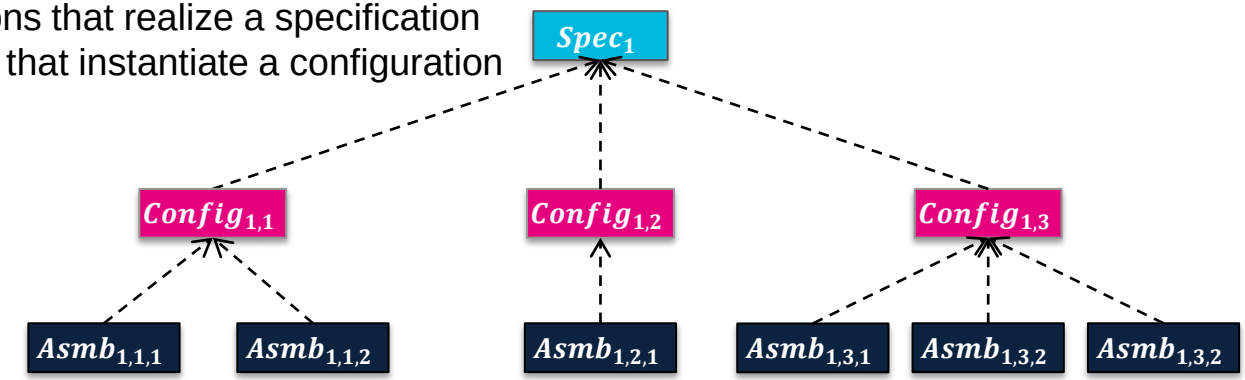


Keeping an history of the whole software life-cycle

- Individual component history
- Architecture levels history
  - Specification
  - Configuration
  - Assembly
- Whole architecture description history

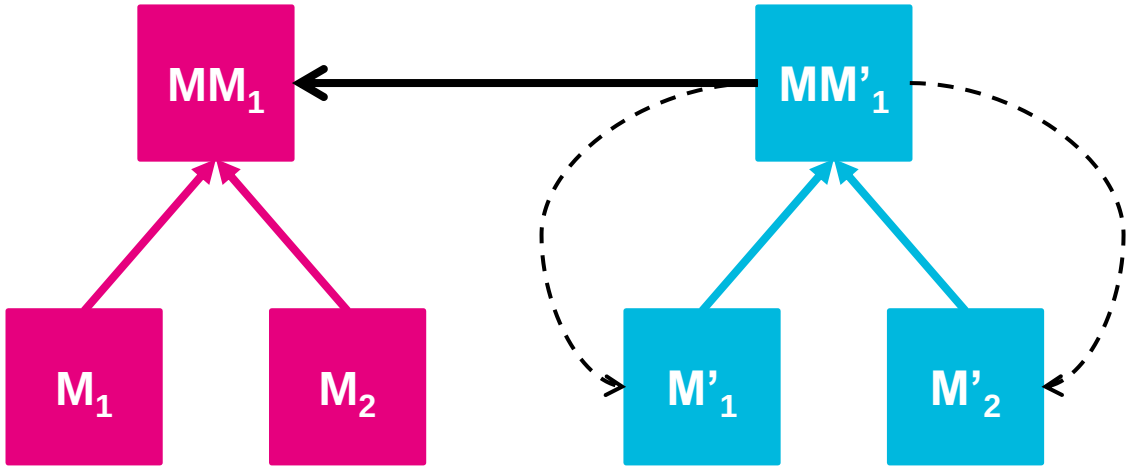
As a consequence

- History of valid configurations
  - Versions of configurations that realize a specification
  - Versions of assemblies that instantiate a configuration
  - …
- Adapting architectures



Classical approach

- Top-down approach
  - Meta-model is versioned
  - Changes are propagated to models
- Historic use of meta-models in model-driven engineering



Is a version of

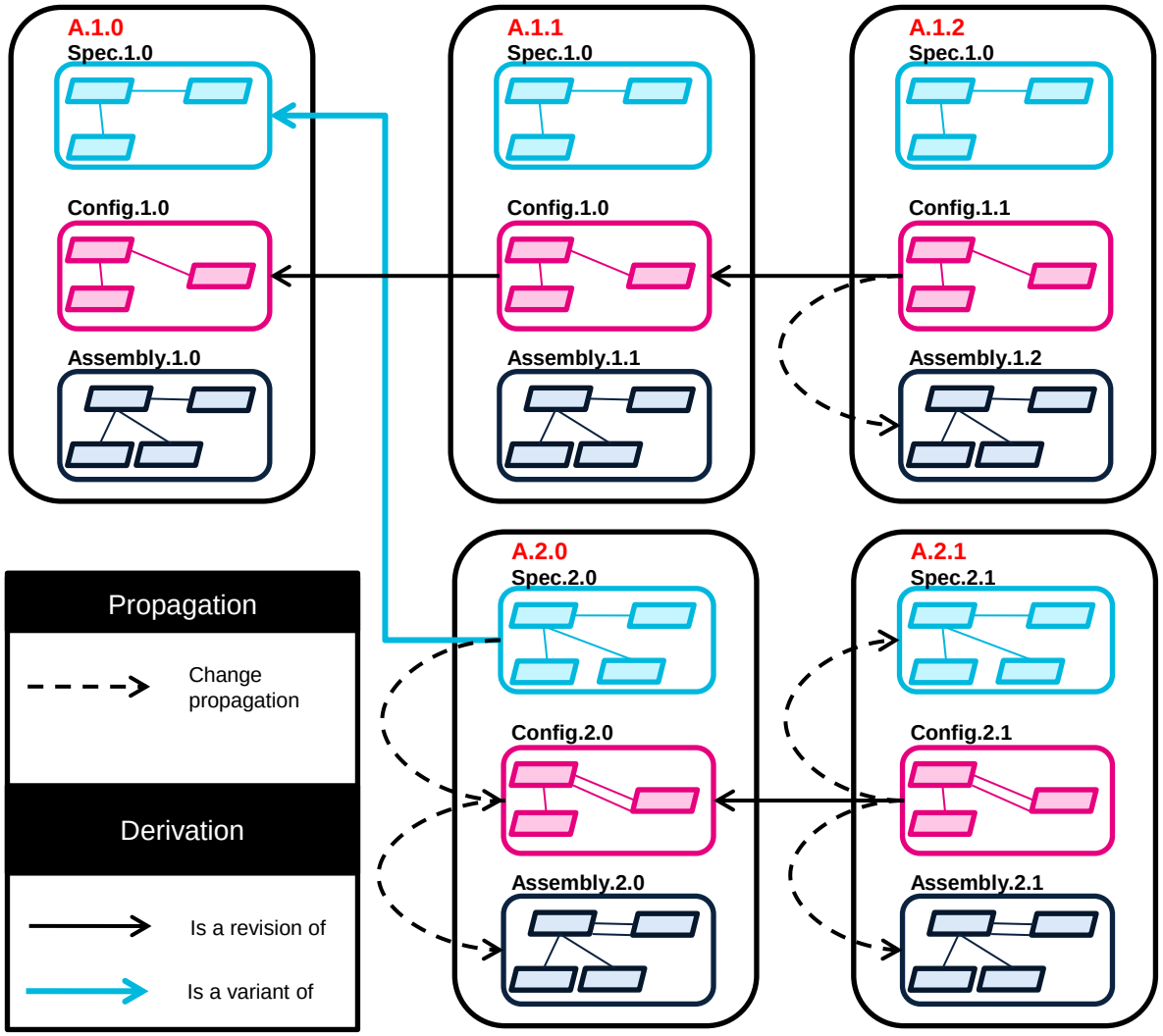
Propagation of changes

Dedal approach

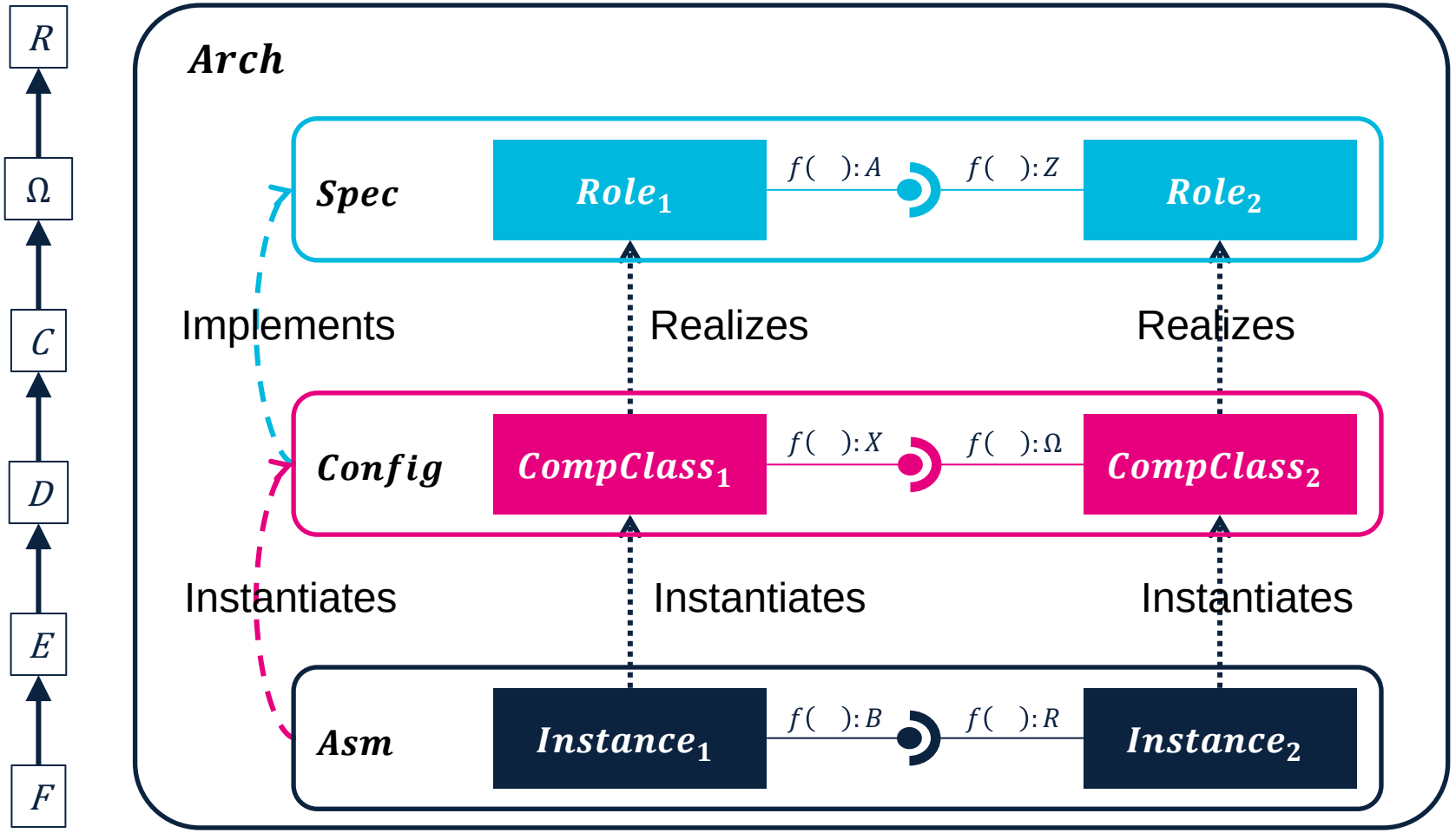
- Change may occur at any description level
- 2 kinds of version:
  - Revision (improving an existing artifact)
  - Variant (add new functionalities to an existing artifact)

Preserve architectural integrity

- Propagation of change / version

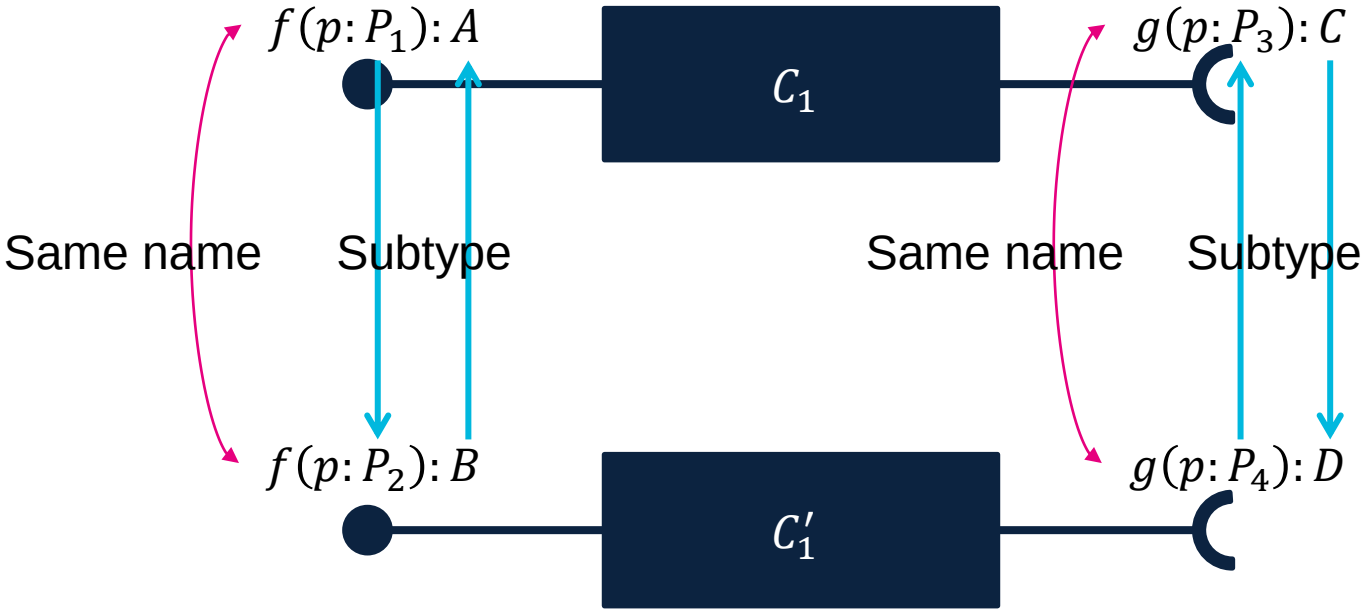


3.4 Base case


R
Ω
C
D
E
F
Arch
Spec
Role1
f( ):A
f( ):Z
Role2
Implements
Realizes
Realizes
Config
CompClass1
f( ):X
f( ):Ω
CompClass2
Instantiates
Instantiates
Instantiates
Asm
Instance1
f( ):B
f( ):R
Instance2

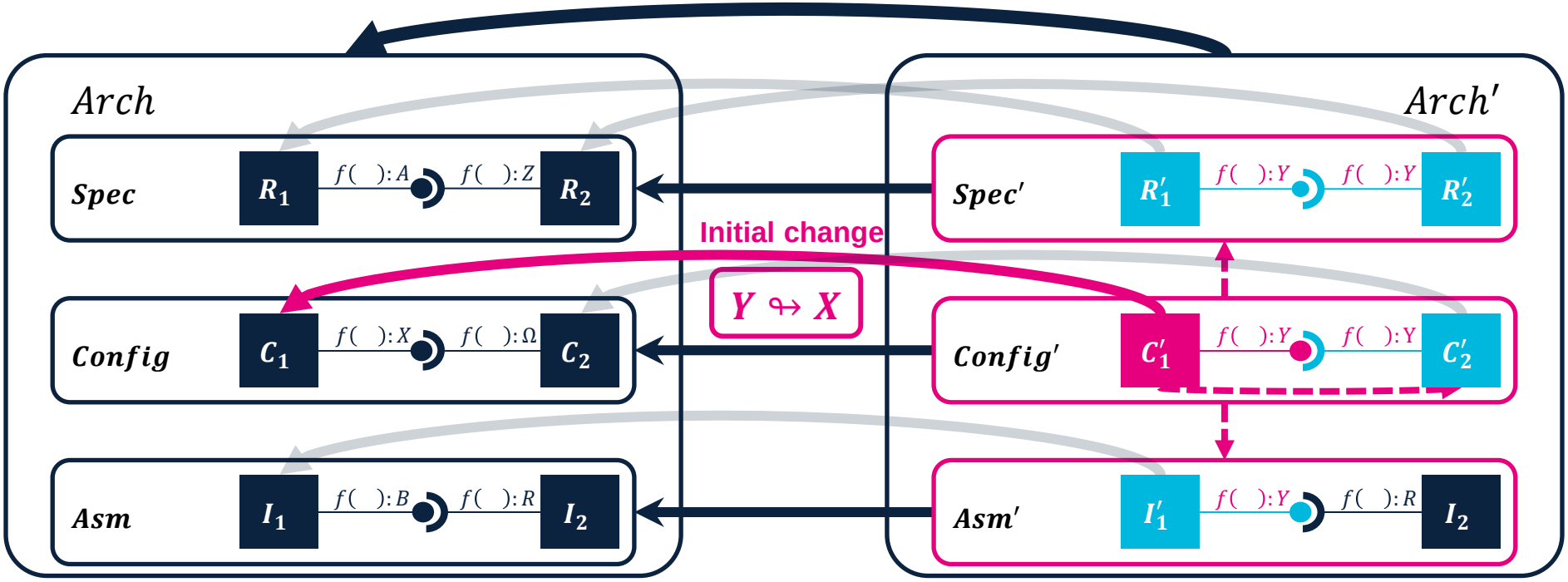
Substitutable provided functionality

Substitutable required functionality



C_1' is substitutable for C_1

3.6 Example of version propagation



$$B \preccurlyeq X \preccurlyeq A \preccurlyeq Z \preccurlyeq \Omega \preccurlyeq R$$
$$(Y \preccurlyeq R) \wedge (Y \parallel \Omega)$$

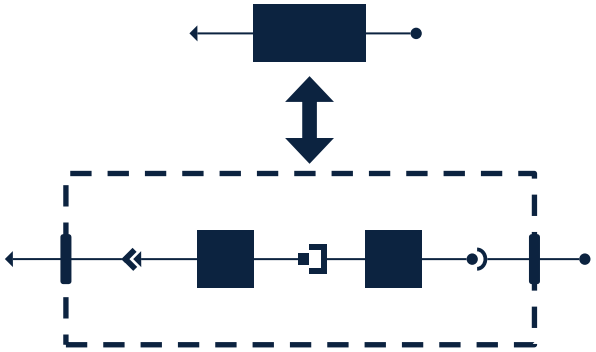
$$[(\neg(Y \preccurlyeq A)) \vee (\neg(Y \succcurlyeq B))] \wedge [\neg(Y \preccurlyeq \Omega)]$$

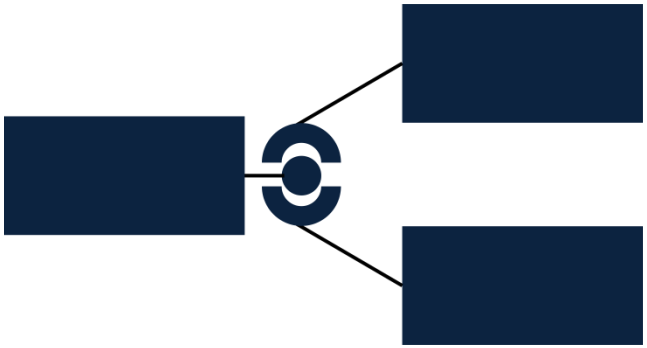
Is a version of

Version/Change propagation

1 *to* n replacement

- Cases of 1 *to* n replacement:
  - A role may be realized by n component classes
  - Many roles may be realized by one component class
  - A component class may be instanciated by n component instances





Multiple connections

- Separately study each connection

SECTION 4: CONCLUSION AND PERSPECTIVES

Substitutability-based principles for predicting version propagation in three-level component-based architectures

- Identification of component substitution scenarios
- component substitution is not a fine-grained enough criterion → parameter types into signatures

Future work

- Formalization and automation of version propagation

SECTION 3: VERSION PROPAGATION

3.6 Rules for propagating version

Hypothesis on types (Figure 1): $B \preccurlyeq X \preccurlyeq A \preccurlyeq Z \preccurlyeq \Omega \preccurlyeq R$

Provided functionality

Specification	Configuration	Assembly
$Y \nrightarrow A$	$Y \nrightarrow X$	$Y \nrightarrow B$

Non-propagation

Specification	Configuration	Assembly
$X \preccurlyeq Y \preccurlyeq Z$	$B \preccurlyeq Y \preccurlyeq A$	$Y \preccurlyeq X$

Propagation

Specification		Configuration		Assembly	
Inter-level	**Intra-level**	**Inter-level**	**Intra-level**	**Inter-level**	**Intra-level**
$(Y \parallel X) \vee (Y \prec X)$	$(Y \parallel Z) \vee (Y \succ Z)$	$(\neg(Y \preccurlyeq A \Rightarrow \uparrow)) \vee (\neg(Y \succcurlyeq B \Rightarrow \downarrow))$	$\neg(Y \preccurlyeq \Omega)$	$\neg(Y \preccurlyeq X)$	$\neg(Y \preccurlyeq R)$
$(Y \parallel X) \wedge (Y \parallel Z)$		$[(\neg(Y \preccurlyeq A)) \vee (\neg(Y \succcurlyeq B))] \wedge [\neg(Y \preccurlyeq \Omega)]$		$\neg(Y \preccurlyeq X)$	

Required functionality

Specification	Configuration	Assembly
$Y \nrightarrow Z$	$Y \nrightarrow \Omega$	$Y \nrightarrow R$

Non-propagation

Specification	Configuration	Assembly
$A \preccurlyeq Y \preccurlyeq \Omega$	$Z \preccurlyeq Y \preccurlyeq R$	$Y \succcurlyeq \Omega$

Propagation

Specification		Configuration		Assembly	
Inter-level		**Inter-level**	**Intra-level**	**Inter-level**	**Intra-level**
$\neg(Y \preccurlyeq \Omega)$		$(\neg(Y \succcurlyeq Z \Rightarrow \uparrow)) \vee (\neg(Y \preccurlyeq R \Rightarrow \downarrow))$	$\neg(Y \succcurlyeq X)$	$\neg(Y \succcurlyeq \Omega)$	$\neg(Y \succcurlyeq B)$
$(Y \parallel \Omega) \wedge (Y \parallel A)$		$[(\neg(Y \succcurlyeq Z)) \vee (\neg(Y \preccurlyeq R))] \wedge [\neg(Y \succcurlyeq X)]$		$(\neg(Y \succcurlyeq \Omega)) \wedge (\neg(Y \succcurlyeq B))$	

IMT Mines Alès
École Mines-Télécom

LIRMM

ECOLE DOCTORALE