IMT Mines Alès
École Mines-Télécom

LIRMM

ECOLE DOCTORALE

SUBSTITUTABILITY-BASED VERSION PROPAGATION TO MANAGE THE EVOLUTION OF THREE-LEVEL COMPONENT-BASED ARCHITECTURES

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

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

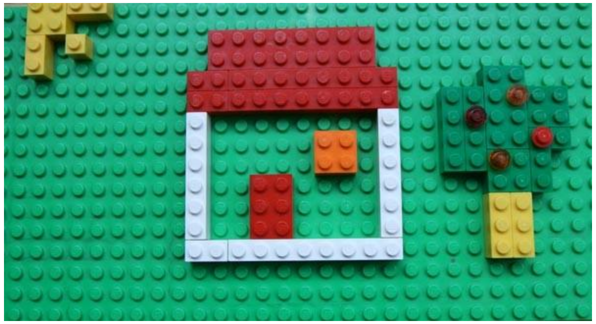
OUTLINE

IMT Mines Alès
École Mines-Télécom
LIRMM
ECOLE DOCTORALE

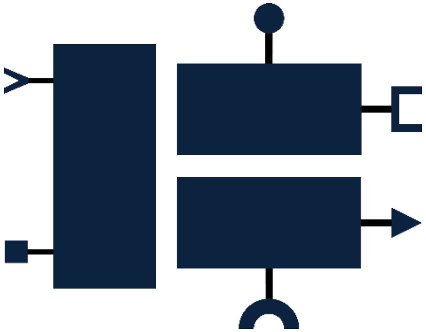
DEFINITIONS

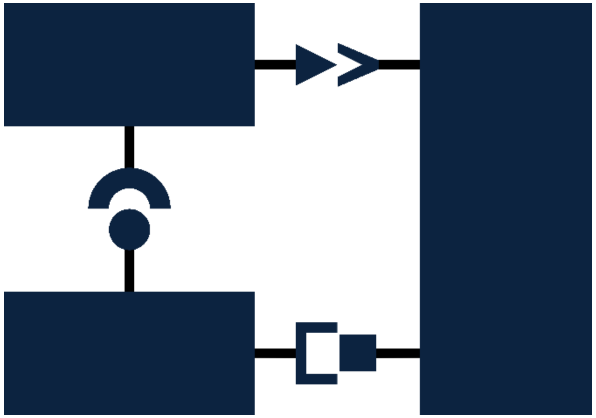
SECTION 1: DEFINITIONS

Component



Architecture





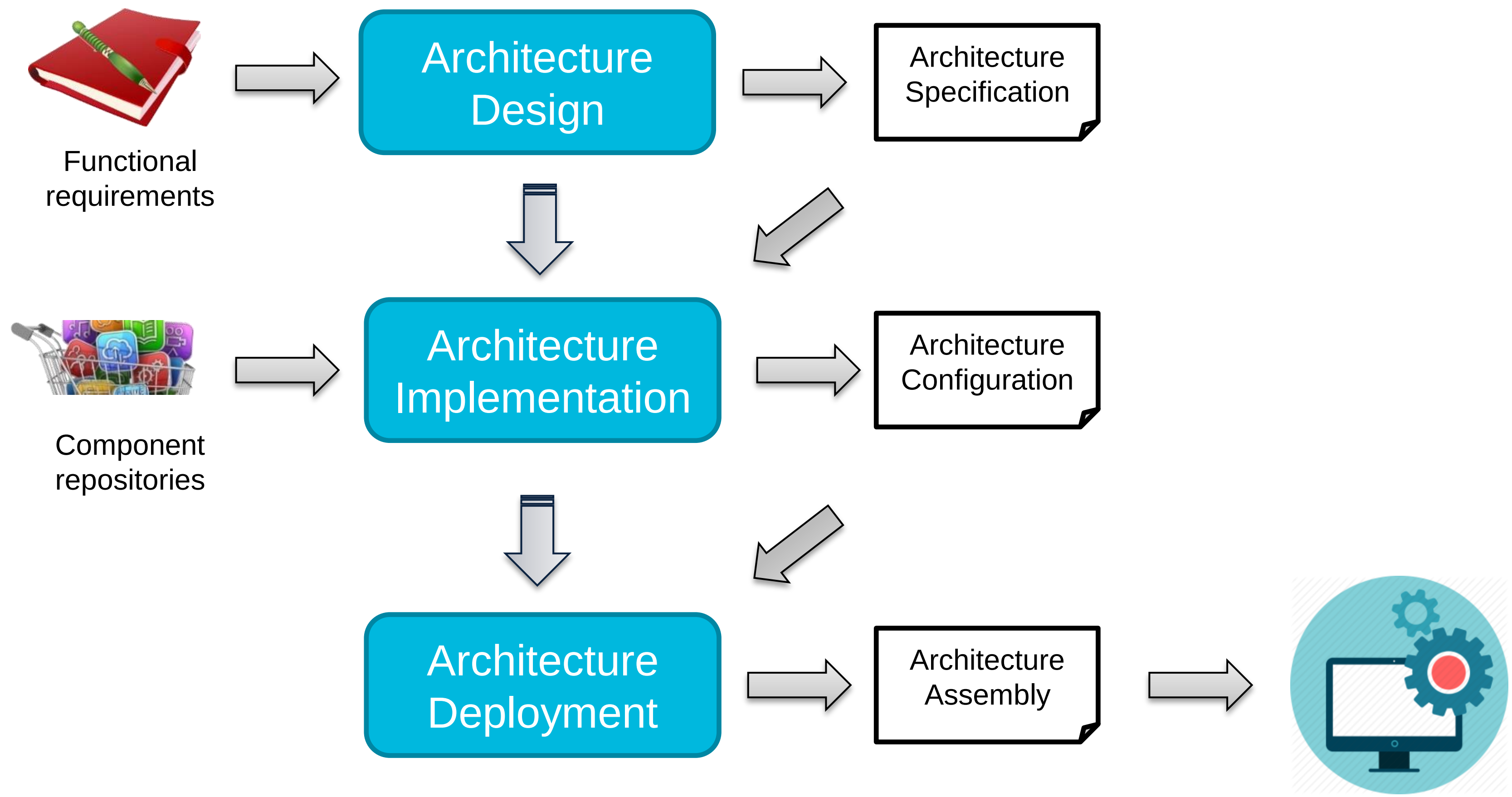
DEDAL, A THREE-LEVEL ARCHITECTURE DESCRIPTION LANGUAGE

IMT Mines Alès
École Mines-Télécom

LIRMM

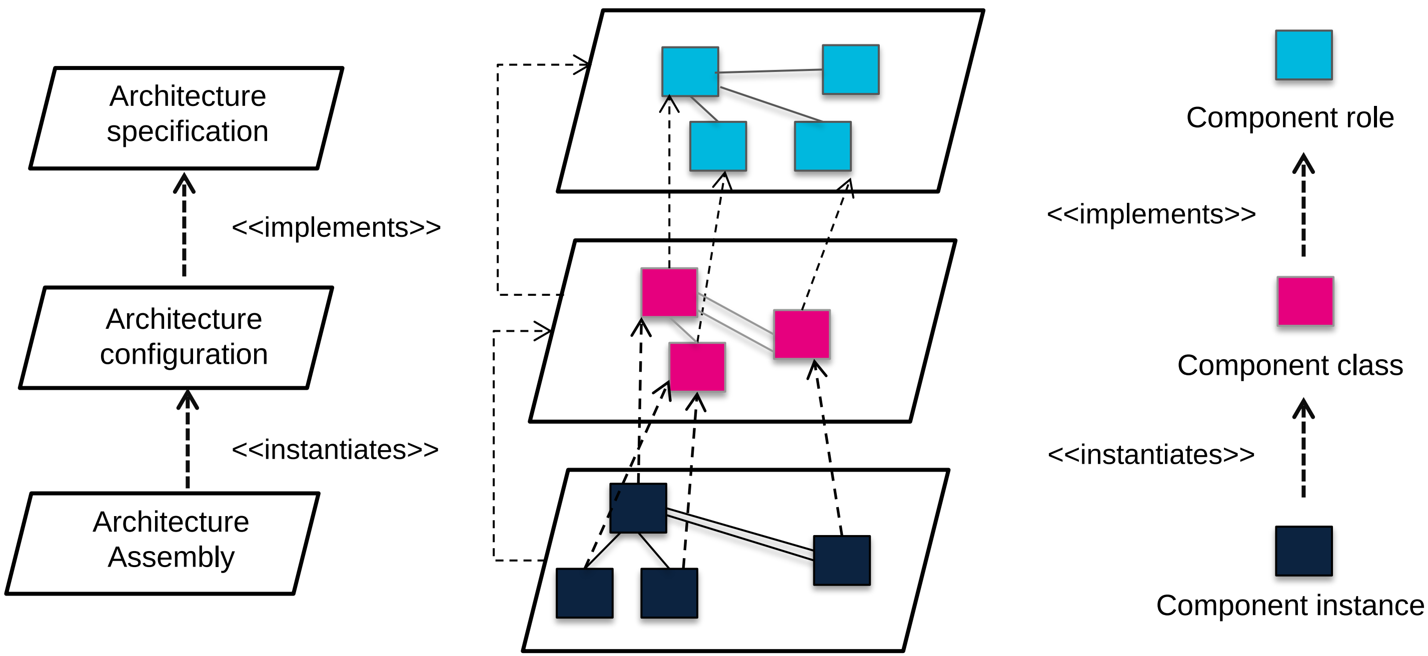
ECOLE DOCTORALE

2.1 Component-based software engineering



Dedal

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



MANAGING THE EVOLUTION OF THREE-LEVELED ARCHITECTURE DESCRIPTIONS IN DEDAL

IMT Mines Alès
École Mines-Télécom
LIRMM
ECOLE DOCTORALE I2S
Université Montpellier

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

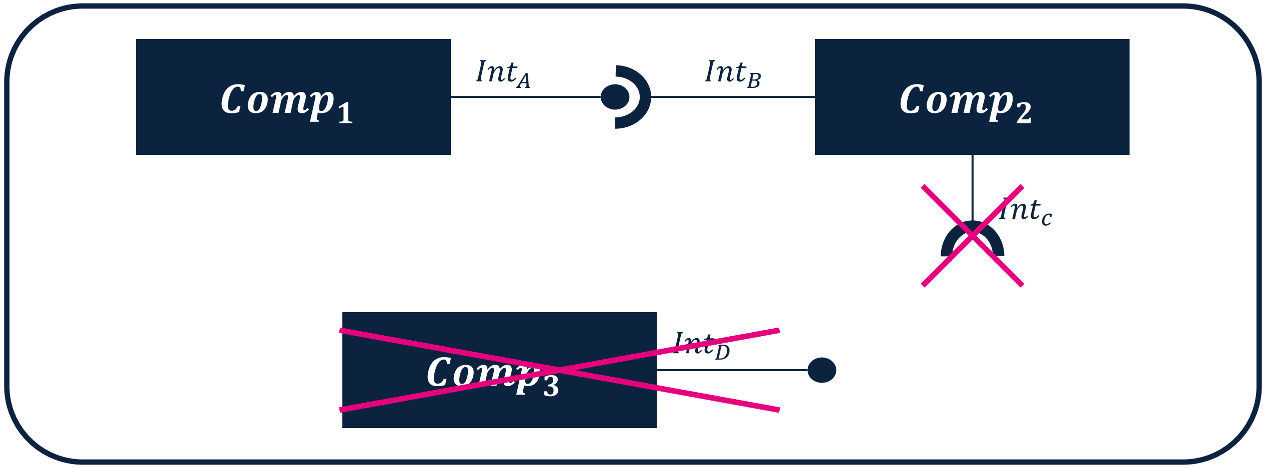
- Automatically propose an evolution plan
  - Co-evolution
  - Propagation of changes within three architecture levels

Formalization of Dedal's concepts

- Langage 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)

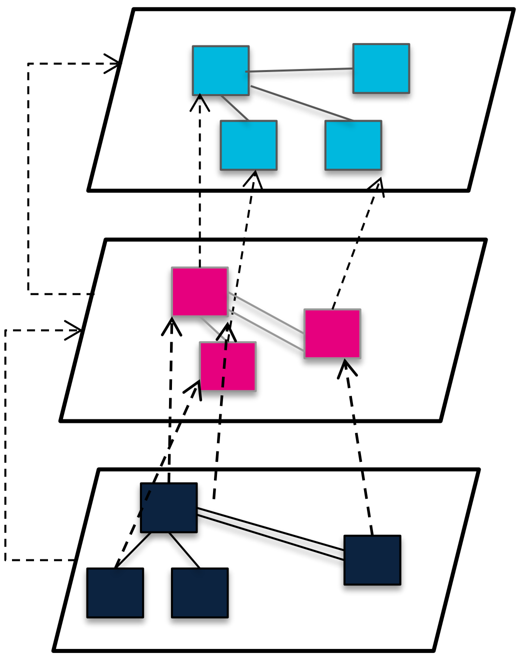
Intra-level consistency

- 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 are realized by component classes
- 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
- Each connected provider in the assembly is an instance of a provided interface from the configuration.



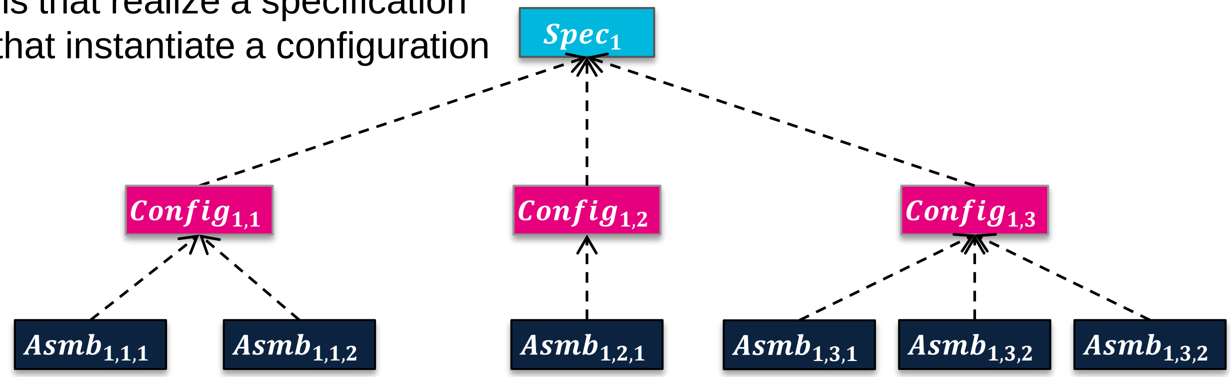
PREDICTING VERSION PROPAGATION IN DEDAL

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

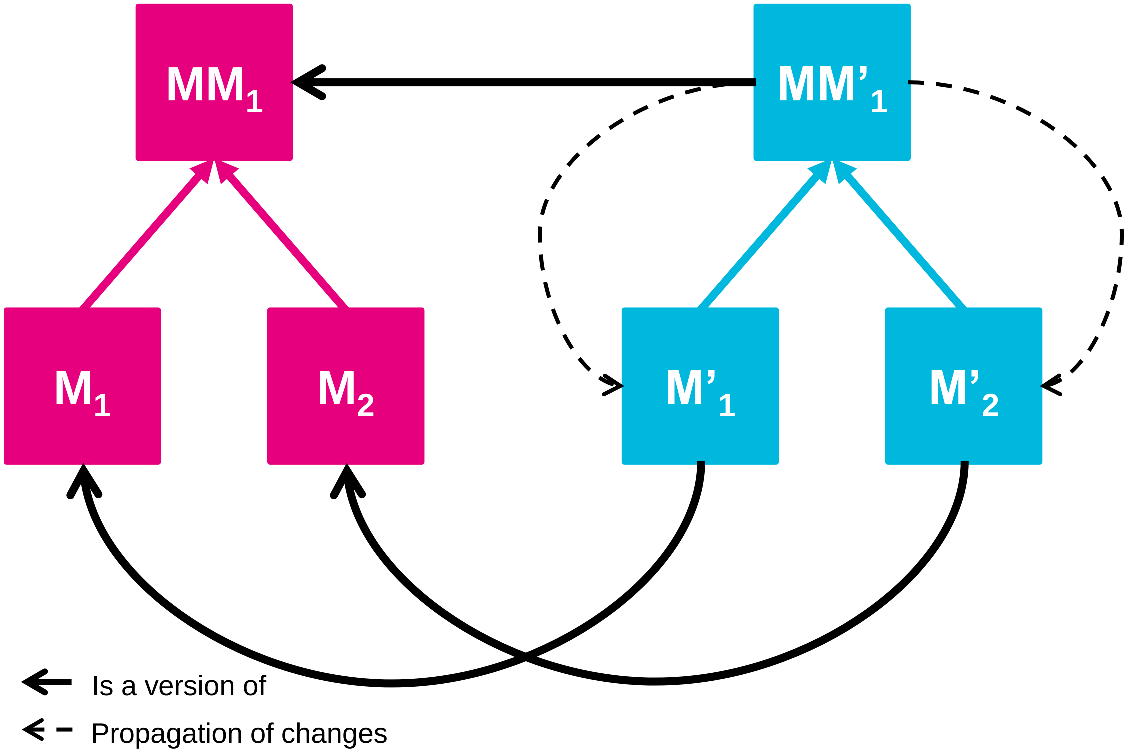
- History of valid configurations
  - Versions of configurations that realize a specification
  - Versions of assemblies that instantiate a configuration
  - …
- Adapting architectures
- Reusing architecture descriptions



Classical approach

- Top-down approach
  - Meta-model is versioned
  - Changes are propagated to models wich new version conforms the new version of the meta-model and are versions of previous models

- Historic use of meta-models in model-driven engineering



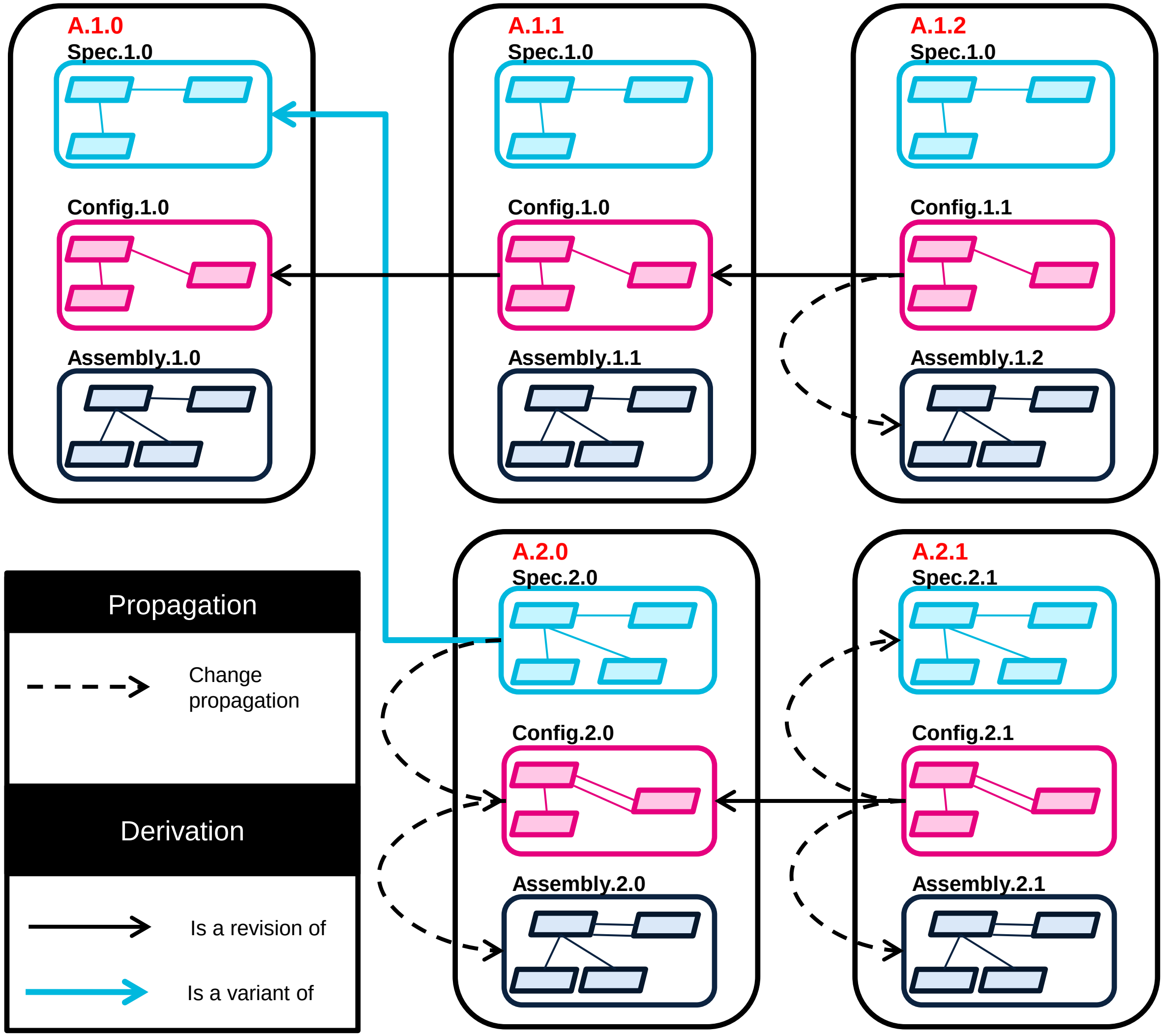
4.3 Versioning three-leveled architecture descriptions

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)

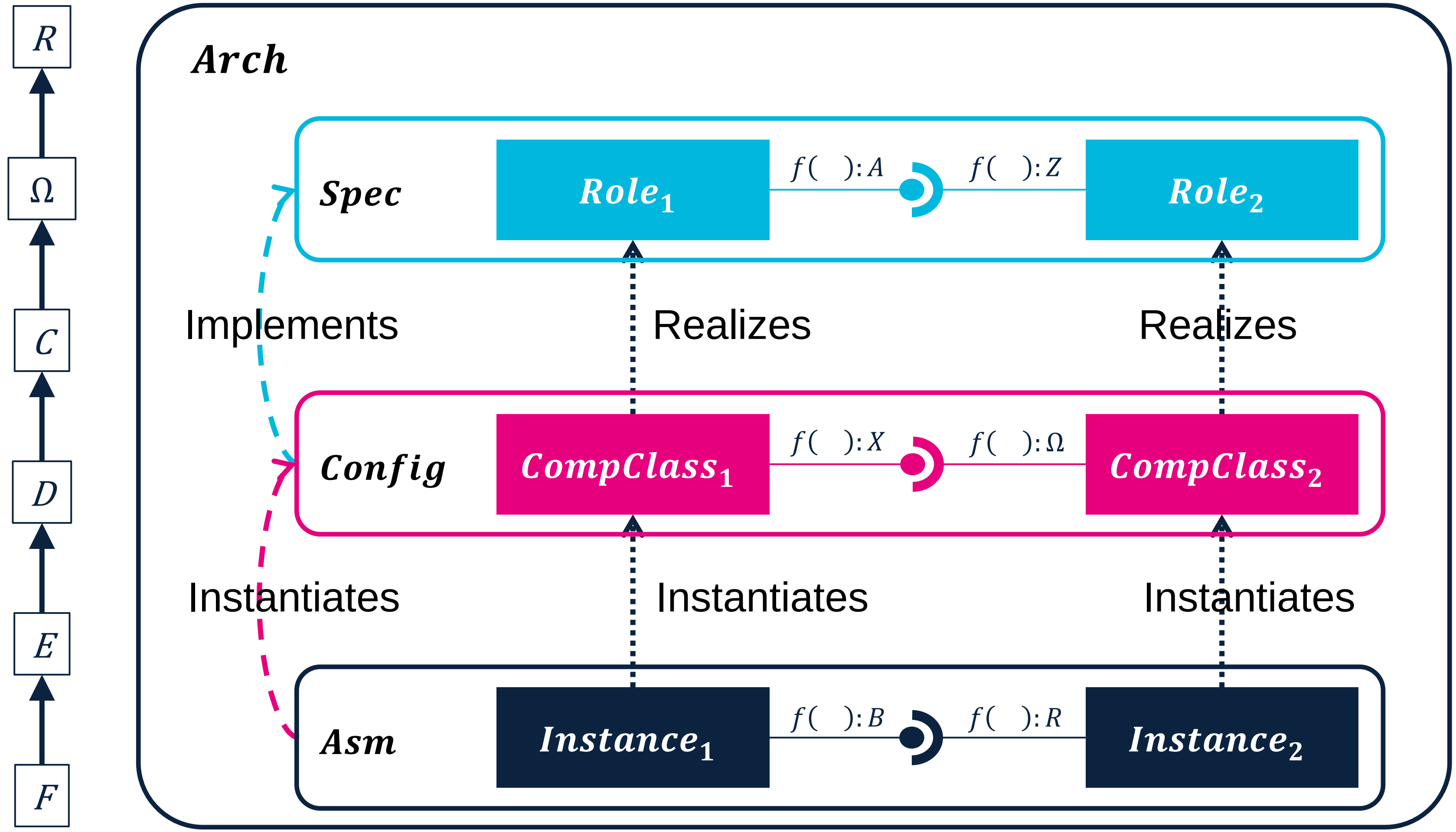
Preserve architectural integrity

- Propagation of change / version



IMT Mines Alès
École Mines-Télécom
LIRMM
ECOLE DOCTORALE

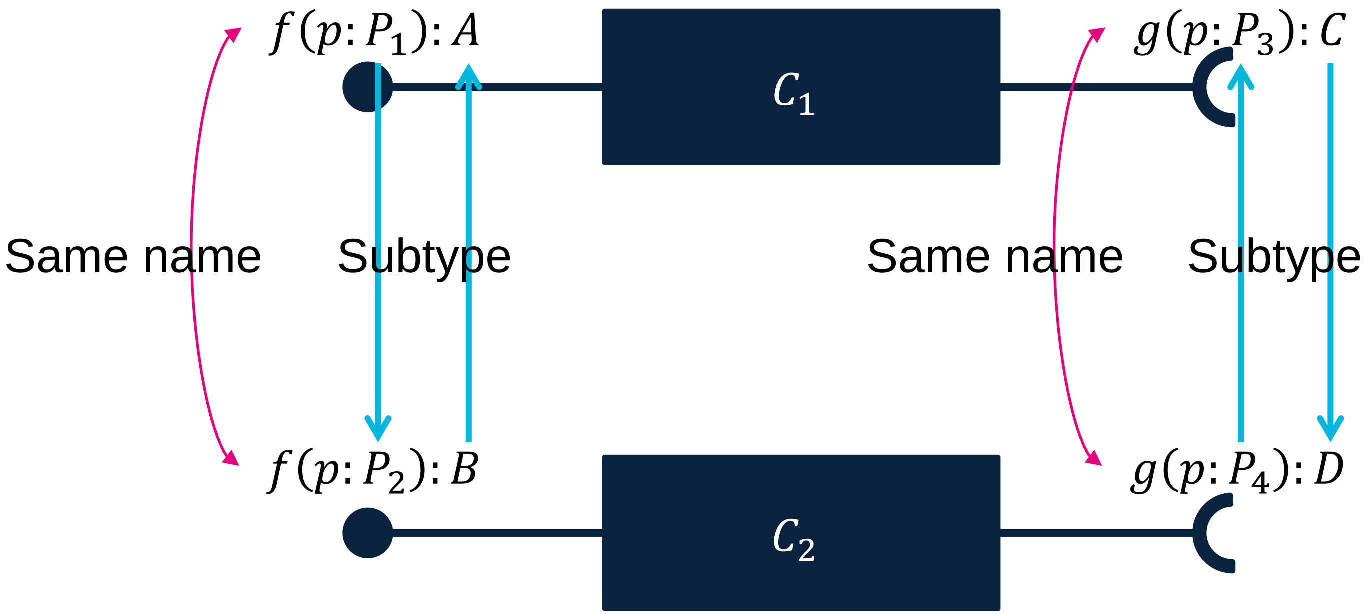
4.4 Base case


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

4.5 Substitutability-based version propagation study

Substitutable provided functionality

Substitutable required functionality



C_2 is substitutable for C_1

SECTION 4: VERSION PROPAGATION

4.6 Rules for propagating version

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

Provided functionality

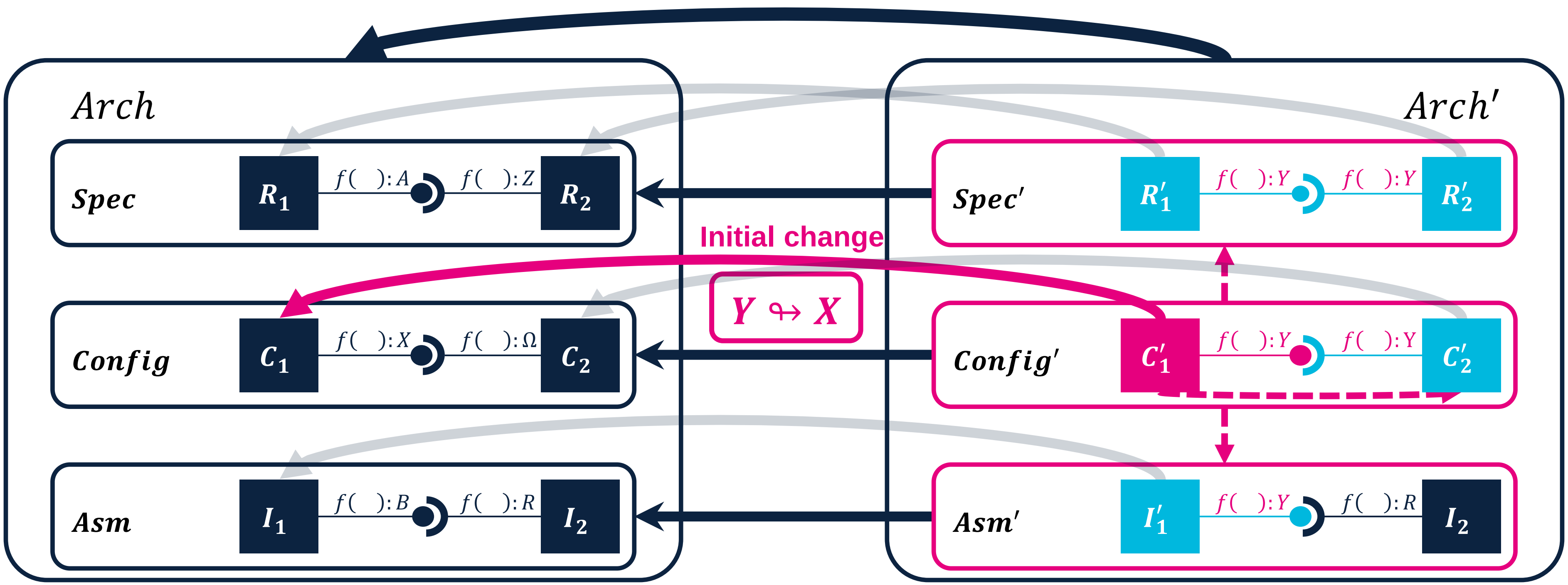
Specification		Configuration		Assembly	
$Y \looparrowright A$		$Y \looparrowright X$		$Y \looparrowright B$	
Non-propagation					
$X \preccurlyeq Y \preccurlyeq Z$		$B \preccurlyeq Y \preccurlyeq A$		$Y \preccurlyeq X$	
Propagation					
Inter-level	**Intra-level**	**Inter-level**	**Intra-level**	**Inter-level**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

Required functionality

Specification		Configuration		Assembly	
$Y \looparrowright Z$		$Y \looparrowright \Omega$		$Y \looparrowright R$	
Non-propagation					
$A \preccurlyeq Y \preccurlyeq \Omega$		$Z \preccurlyeq Y \preccurlyeq R$		$Y \succcurlyeq \Omega$	
Propagation					
Inter-level	**Intra-level**	**Inter-level**	**Intra-level**	**Inter-level**	**Intra-level**
$\neg(Y \preccurlyeq \Omega)$	$\neg(Y \succcurlyeq \mathrm{A})$	$(\neg(Y \succcurlyeq Z \Rightarrow \uparrow))$ $\vee (\neg(Y \preccurlyeq R \Rightarrow \downarrow))$	$\neg(Y \succcurlyeq X)$	$\neg(Y \succcurlyeq \Omega)$	$\neg(Y \succcurlyeq B)$
$(Y \parallel \Omega) \wedge (Y \parallel A)$		$\left[\left(\neg(Y \succcurlyeq Z)\right) \vee \left(\neg(Y \preccurlyeq R)\right)\right] \wedge [\neg(Y \succcurlyeq X)]$		$\left(\neg(Y \succcurlyeq \Omega)\right) \wedge \left(\neg(Y \succcurlyeq B)\right)$	

IMT Mines Alès
École Mines-Télécom

4.7 Example of version propagation



$$B \preccurlyeq X \preccurlyeq A \preccurlyeq Z \preccurlyeq \Omega \preccurlyeq R$$

$$(Y \preccurlyeq R) \land (Y \parallel \Omega)$$

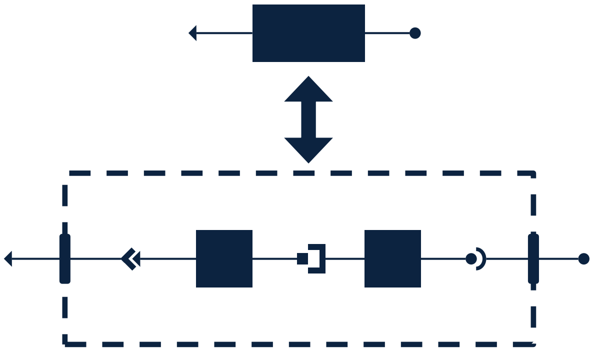
$$[(\neg(Y \preccurlyeq A)) \lor (\neg(Y \succcurlyeq B))] \land [\neg(Y \preccurlyeq \Omega)]$$

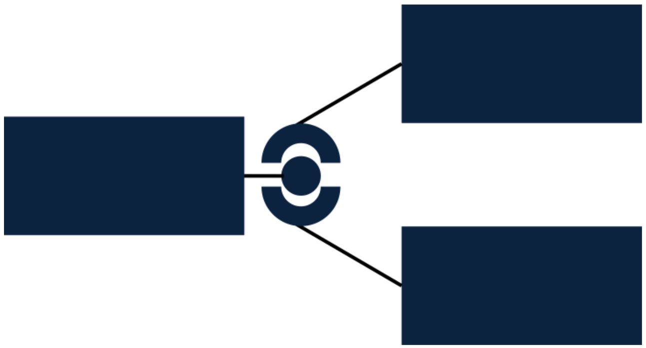
Is a version of

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

RELATED WORK

Many ADLs

C2-SADEL, Darwin, Wright, Dynamic Wright, ArchWare, SAEV, SAEM, Plastik...

ADL	3-levels (Full life-cycle coverage)	Finest grained type	Architecture version aware
SOFA 2.0	☹ (configuration and non-descriptive assembly)	😐 Interface type	☺ (Through composite components)
XADL 2.0	☹ (design-time and run-time)	😐 Interface	☺
MAE	☹ (design-time and run-time)	☺ Interface elements (signature + input parameters)	☺

CONCLUSION AND PERSPECTIVES

IMT Mines Alès
École Mines-Télécom

LIRMM

ECOLE DOCTORALE

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

- Focus on the representation of the versioning concepts
- Versioning meta-model
- Version-ready Dedal meta-model
- Formalization and automation of version propagation
- Extraction of component-based architecture models from descriptors (Spring, OSGi, Maven…)

QUESTIONS?

IMT Mines Alès
École Mines-Télécom

LIRMM

ECOLE DOCTORALE

UNIVERSITÉ MONTPELLIER