CENTRALE
L Y O N

Génie Logiciel :
Développement de logiciels basé sur des modèles et des processus

Méta-Modélisation

BTD/GL/MétaM 1

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modélisation

- But de la méta-modélisation
- Architecture MOF
- 4 niveaux de (méta) modélisation
- Architecture 4 niveaux généralisable en dehors du MOF
- Syntaxes abstraite et concrète
- Profils UML
- Spécialisation et définition de méta-modèles
- Cycle en Y
- Méthodes d'ingénierie : Produit et Processus

BTD/GL/MétaM 2

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Normes OMG de modélisation

- MOF : Meta-Object Facilities
  - ➔Langage de définition de méta-modèles
- UML : Unified Modelling Language
  - ➔Langage de modélisation
- CWM : Common Warehouse Metamodel
  - ➔Modélisation ressources, données, gestion d'une entreprise
- OCL : Object Constraint Language
  - ➔Langage de contraintes sur des modèles
- XMI : XML Metadata Interchange
  - ➔Standard pour échanges de modèles et méta-modèles

BTD/GL/MétaM

3

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Normes OMG de modélisation

- Plusieurs de ces normes concernent la définition et l'utilisation de méta-modèles
  - ➔MOF : but de la norme
  - ➔UML et CWM : peuvent être utilisés pour en définir
  - ➔XMI : pour échange de (méta-)modèles entre outils MOF
- MOF est un méta-méta-modèle
  - ➔Utilisé pour modéliser des méta-modèles
  - ➔Définit les concepts de base (22)
  - ➔Entité/classe, relation/association, type de données, référence, package ...
  - ➔Le MOF peut définir le MOF

BTD/GL/MétaM

4

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

4 Niveaux du MOF

- Le MOF définit 4 niveaux de modélisation
  - ➔M0 : système réel, système modélisé
  - ➔M1 : modèle du système réel défini dans un certain langage
  - ➔M2 : méta-modèle définissant ce langage
  - ➔M3 : méta-méta-modèle définissant le méta-modèle
- Le niveau M3 est le MOF
  - ➔Dernier niveau, il est méta-circulaire : il peut se définir lui même
  - ➔Le MOF est – pour l'OMG – le méta-méta-modèle unique servant de base à la définition de tous les méta-modèles

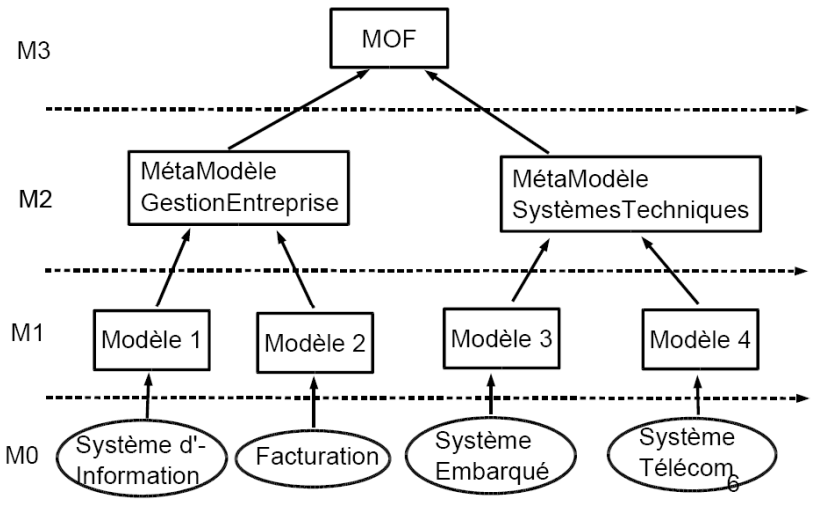
BTD/GL/MétaM

5

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

4 niveaux de modélisation



6

BTD/GL/MétaM

6

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Hiérarchie 4 Niveaux

- On retrouve cette hiérarchie à 4 niveaux en dehors du MOF et d'UML, dans d'autres espaces technologique que celui de l'OMG
- Langage de programmation
  - ➔ M0 : l'exécution d'un programme
  - ➔ M1 : le programme
  - ➔ M2 : la grammaire du langage dans lequel est écrit le programme
  - ➔ M3 : le concept de grammaire EBNF
- XML
  - ➔ M0 : données du système
  - ➔ M1 : données modélisées en XML
  - ➔ M2 : DTD XML
  - ➔ M3 : le langage XML

BTD/GL/MétaM 7

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modélisation UML

- Dans UML, on retrouve également les 4 niveaux mais avec le niveau M3 définissable en UML directement à la place du MOF
- Exemple de système à modéliser (niveau M0)
  - ➔ Une pièce possède 4 murs, 2 fenêtres et une porte
  - ➔ Un mur possède une porte ou une fenêtre mais pas les 2 à la fois
  - ➔ Deux actions sont associées à une porte ou une fenêtre : ouvrir et fermer
    - Si on ouvre une porte ou une fenêtre fermée, elle devient ouverte
    - Si on ferme une porte ou une fenêtre ouverte, elle devient fermée

BTD/GL/MétaM 8

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modélisation UML

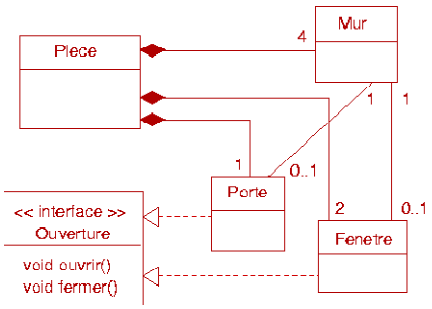
- Pour modéliser ce système, il faut définir 2 diagrammes UML : niveau M1
  - ➔ Un diagramme de classe pour représenter les relations entre les éléments (portes, murs, pièce)
  - ➔ Un diagramme d'état pour spécifier le comportement d'une porte ou d'une fenêtre (ouverte, fermée)
- On peut abstraire le comportement des portes et des fenêtres en spécifiant les opérations d'ouverture fermeture dans une interface.
- Le diagramme d'état est associé à cette interface.
- Il faut également ajouter des contraintes OCL pour préciser les contraintes entre les éléments d'une pièce.

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

M1 : spécification du Système

- **context** Mur **inv**:
  - ➔ fenetre -> union(porte) -> size() <= 1 -- un mur a soit une fenêtre soit une porte (soit rien)
- **context** Piece **inv**:
  - ➔ mur.fenetre -> size() = 2 -- 2 murs de la pièce ont une fenêtre
  - ➔ mur.porte -> size() = 1 -- 1 mur de la pièce a une porte



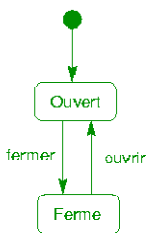


diagramme d'etat associe a l'interface Ouverture

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modélisation UML

- Les 2 diagrammes de ce modèle de niveau M1 sont des diagrammes UML valides
- Les contraintes sur les éléments des diagrammes UML et leurs relations sont définies dans le méta-modèle UML : niveau M2
  - ➔Un diagramme UML (classe, état ...) doit être conforme au méta-modèle UML
- Méta-modèle UML
  - ➔Diagramme de classe spécifiant la structure de tous types de diagrammes UML
  - ➔Avec contraintes OCL pour spécification précise

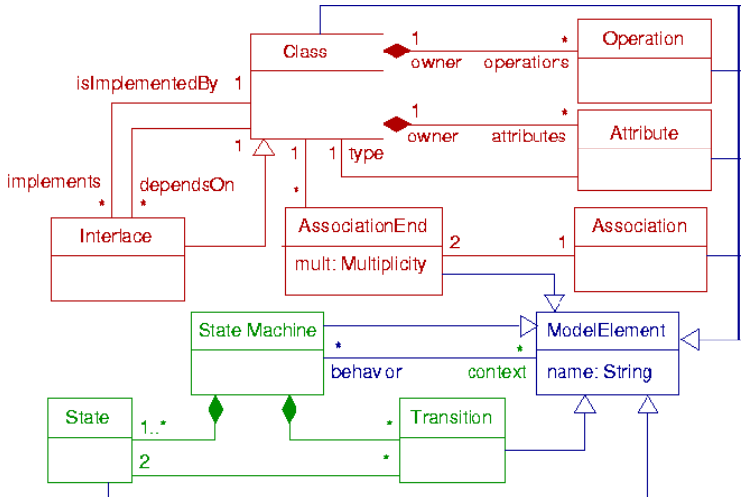
BTD/GL/MétaM

11

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

M2 : Méta-modèle UML (simplifié)



BTD/GL/MétaM

12

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

M2 : Méta-modèle UML (simplifié)

- Contraintes OCL, quelques exemples
  - ➔ **context** Interface **inv**: attributes -> isEmpty()
    - *Une interface est une classe sans attribut*
  - ➔ **context** Class **inv**: attributes -> forAll (a1, a2 | a1 <> a2 **implies** a1.name <> a2.name)
    - *2 attributs d'une même classe n'ont pas le même nom*
  - ➔ **context** StateMachine **inv**: transition -> forAll (t | self.state -> includesAll(t.state))
    - *Une transition d'un diagramme d'état connecte 2 états de ce diagramme d'état*

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

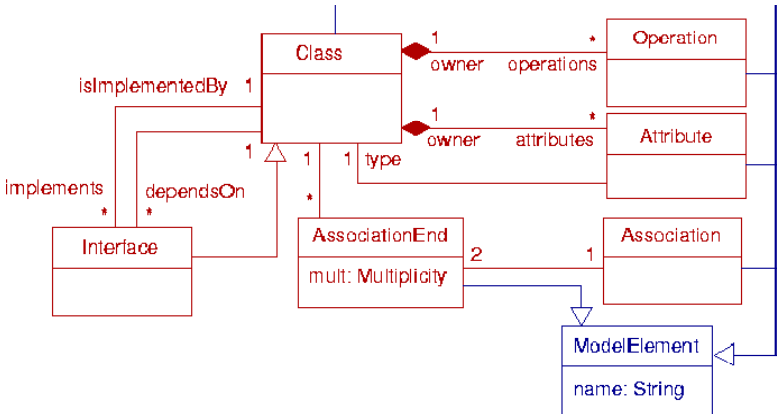
Méta-modélisation UML

- Le méta-modèle UML doit aussi être précisément défini
  - ➔ Il doit être conforme à un méta-modèle
  - ➔ C'est le méta-méta-modèle UML
- Qu'est ce que le méta-modèle UML ?
  - ➔ Un diagramme de classe UML (avec contraintes OCL)
- Comment spécifier les contraintes d'un diagramme de classe ?
  - ➔ Via le méta-modèle UML
  - ➔ Ou plus précisément : via la partie du méta-modèle UML spécifiant les diagrammes de classes
- Méta-méta-modèle UML = copie partielle du méta-modèle UML : niveau M3

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

M3 : Méta-méta-modèle UML (simplifié)



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modélisation UML

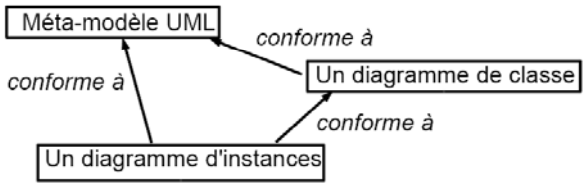
- Méta-méta-modèle UML doit aussi être clairement défini
  - ➔ Il doit être conforme à un méta-modèle
  - ➔ Qu'est ce que le méta-méta-modèle UML ?
  - ➔ Un diagramme de classe UML
- Comment spécifier les contraintes d'un diagramme de classe ?
  - ➔ Via la partie du méta-modèle UML spécifiant les diagrammes de classe
  - ➔ Via le méta-méta-modèle UML
  - ➔ Le méta-méta-modèle UML peut donc se définir lui même
  - ➔ Méta-circulaire
  - ➔ Pas besoin de niveau méta supplémentaire

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Diagrammes d'instances UML

- Un diagramme d'instances est particulier car
  - ➔ Il doit être conforme au méta-modèle UML,
  - ➔ qui définit la structure générale des diagrammes d'instances,
  - ➔ Il doit aussi être conforme à un diagramme de classe.
- Diagramme de classe est un méta-modèle
  - ➔ qui doit être conforme également au méta-modèle UML



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Syntaxe

- Un langage est défini par un méta-modèle
- Un langage possède une syntaxe respectant le méta-modèle
- Syntaxe textuelle
  - ➔ Ensemble de mots-clés et de mots respectant des contraintes définies selon des règles précises
  - ➔ Notions de syntaxe et de grammaire dans les langages
- Exemple pour langage Java
  - ➔ *public class MaClasse implements MonInterface { ... }*
- Grammaire Java pour déclaration de classe
  - ➔ *class_declaration ::= { modifier } "class" identifier ["extends« class_name] ["implements" interface_name { ",« interface_name }] "{" { field_declaration } "}"*

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Syntaxe

- Syntaxe graphique
  - ➔ Notation graphique, chaque élément a une forme graphique particulière
- Exemple : associations entre classes/interfaces sur les diagrammes de classe UML
  - ➔ Trait normal : association
  - ➔ Flèche, trait pointillé : dépendance
  - ➔ Flèche en forme de triangle, trait en pointillé : implémentation
  - ➔ Flèche en forme de triangle, trait plein : spécialisation

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Syntaxe

- Syntaxe abstraite/concrète
- Abstraite
  - ➔ Les éléments et leurs relations sans une notation spécialisée
  - ➔ Correspond à ce qui est défini au niveau du méta-modèle
- Concrète
  - ➔ Syntaxe graphique ou textuelle définie pour un type de modèle
  - ➔ Plusieurs syntaxes concrètes possibles pour une même syntaxe abstraite
  - ➔ Un modèle peut être défini via n'importe quelle syntaxe
    - L'abstraite
    - Une des concrètes
- MOF : langage pour définir des méta-modèles
  - ➔ Pas de syntaxe concrète définie

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

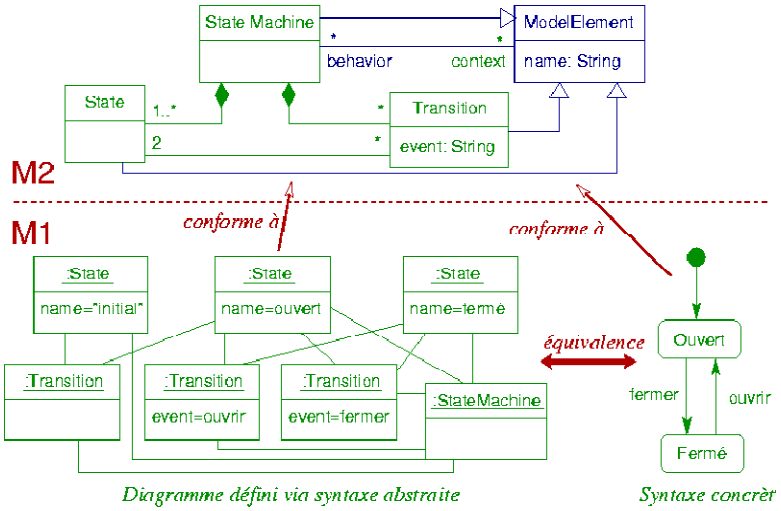
Syntaxe

- Exemple de la modélisation de la pièce
- Syntaxe concrète
  - ➔ 2 diagrammes UML (classe et états) avec syntaxe spécifique à ces types de diagrammes
- Via la syntaxe abstraite
  - ➔ Diagramme d'instances (conforme au méta-modèle) précisant les instances particulières de classes, d'associations, d'états...
  - ➔ Pour la partie diagramme d'états
  - ➔ Diagramme défini via syntaxe concrète : diagramme d'états de l'exemple
  - ➔ Diagramme défini via syntaxe abstraite : diagramme d'instance conforme au méta-modèle UML

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Syntaxe : exemple diagramme état



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Spécification de méta-modèles

- But : définir un type de modèle avec tous ces types d'éléments et leurs contraintes
- Trois approches possibles
  - ➔ Définir un méta-modèle nouveau à partir de rien
  - ➔ Modifier un méta-modèle existant : ajout, suppression, modification d'éléments et des contraintes sur leurs relations
  - ➔ Correspond au MOF, décomposé en 2 parties
    - E-MOF : essential MOF, les méta-éléments de base réutilisés tels quels dans tous les méta-modèles
    - MOF : un méta-modèle particulier défini via E-MOF
- Spécialiser un méta-modèle existant en rajoutant des éléments et des contraintes (sans en enlever)
  - ➔ Correspond aux profils UML

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Profils UML

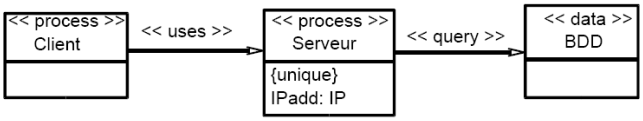
- Un profil est une spécialisation du méta-modèle UML
  - ➔ Ajouts de nouveaux types d'éléments et des contraintes sur leurs relations entre eux et avec les éléments d'UML
  - ➔ Ajouts de contraintes sur éléments existants d'UML
  - ➔ Ajouts de contraintes sur relations existantes entre les éléments d'UML
  - ➔ Aucune suppression de contraintes ou d'éléments
- Profil : mécanisme d'extension d'UML pour l'adapter à un contexte métier ou technique particulier
  - ➔ Profil pour composants EJB
  - ➔ Profil pour gestion bancaire
  - ➔ Profil pour architecture logicielle

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Profils UML

- Stéréotype : extension, spécialisation d'un élément du méta-modèle
  - ➔ Classe, association, attribut, opération ...
  - ➔ Le nom d'un stéréotype est marqué entre << ... >>
  - ➔ Il existe déjà des stéréotypes dans UML
  - ➔ << interface >> : une interface est une classe particulière (sans attribut)
  - ➔ On peut marquer des attributs d'une classe pour préciser une contrainte ou un rôle particulier : tagged value
- Exemple {unique} id: int



3TD/GL/MétaM

25

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Profils UML

- Profil UML est composé de 3 types d'éléments
  - ➔ Des stéréotypes
  - ➔ Des tagged value
  - ➔ Des contraintes (exprimables en OCL)
    - ➢ Sur ces stéréotypes, tagged value
    - ➢ Sur des éléments du méta-modèle existant
    - ➢ Sur les relations entre les éléments
- Un profil UML est défini sous la forme d'un package stéréotypé << profile >>
- Exemple de profil : architecture logicielle
  - ➔ Des composants clients et serveur
  - ➔ Un client est associé à un serveur via une interface de service par l'intermédiaire d'un proxy

3TD/GL/MétaM

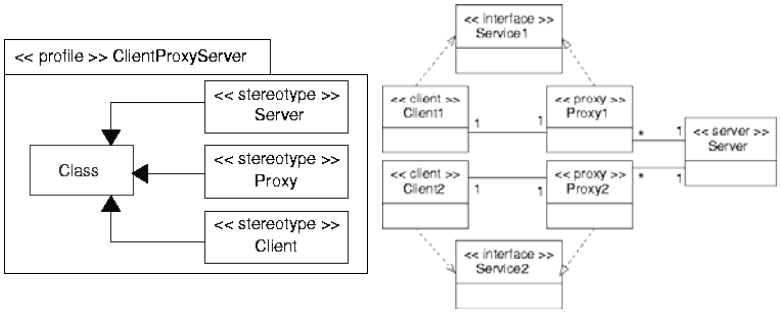
26

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Exemple Profil

- Profil nommé ClientProxyServer
- Définit trois stéréotypes : Trois classes jouant un rôle particulier : extensions de la méta-class Class du méta-modèle UML
  - ➔ Server, Proxy, Client



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

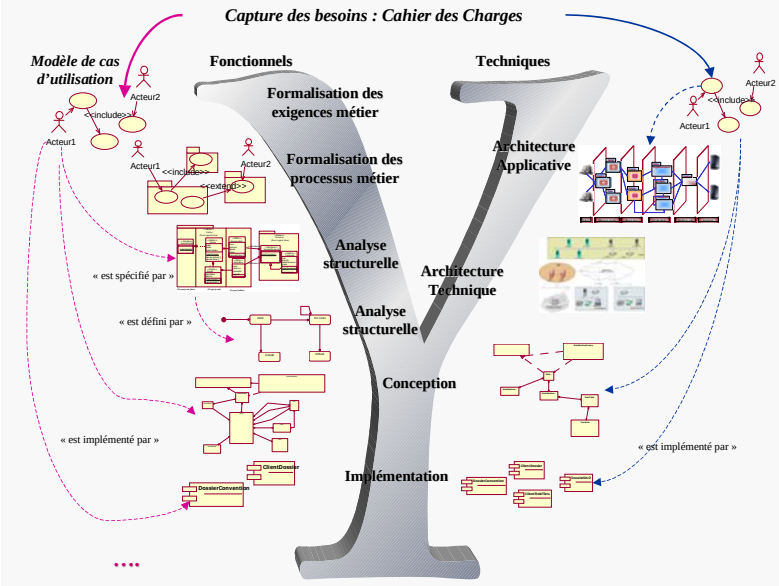
Exemple Profil

- Pour compléter le profil, ajout de contraintes OCL
  - ➔ Navigation sur le méta-modèle (simplifié) en considérant que la méta-class Class a trois spécialisations (Server, Client, Proxy)
    - **context** Client **inv**:**let** proxies = self.associationEnd.association.associationEnd.class -> select (c | c.oclIsTypeOf(Proxy)) **in** **let** interfaces = self.dependsOn **in** interfaces -> forAll (i | proxies.implements -> includes (i) **and** proxies -> forAll (p | p.implements -> includes (i) **implies** p.hasClassRefWith(self)))
    - **context** Class **def**: hasClassRefWith(cl : Class) : Boolean= self.associationEnd.association.associationEnd.class -> exists (c | c = cl)
  - ➔ Un proxy associé à un client doit implémenter une des interfaces dont dépend le client et un proxy implémentant une interface d'un client doit avoir une association avec ce client

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méthodes d'ingénierie

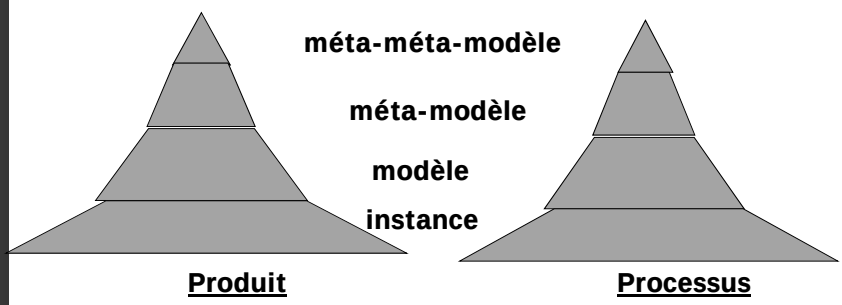


CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méthodes d'ingénierie

Modèles, démarches et outils pour comprendre et représenter le Système d'Information (SI) dans sa globalité en vue de développer et faire évoluer des SI informatisés adaptés, adaptables, adaptatifs

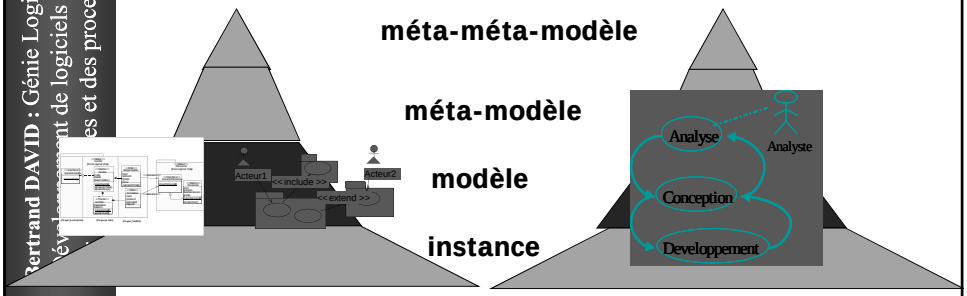


CENTRALE LYON

Bertrand DAVID : Génie Logiciel : développement de logiciels basé ... des modèles et des processus

Méthodes d'ingénierie

modèles, démarches et outils pour comprendre et représenter le SI dans sa globalité en vue de développer et faire évoluer des SI informatisés adaptés, adaptables, adaptatifs



Produit **Processus**

3TD/GL/MétaM

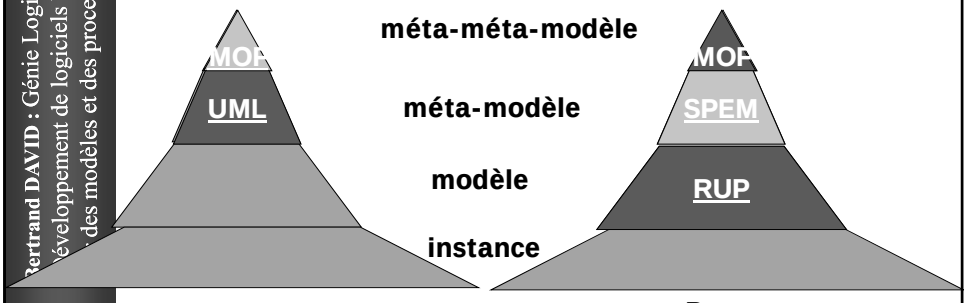
31

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : développement de logiciels basé ... des modèles et des processus

Méthodes d'ingénierie

modèles, démarches et outils pour comprendre et représenter le SI dans sa globalité en vue de développer et faire évoluer des SI informatisés adaptés, adaptables, adaptatifs



Produit **Processus**

minimum :

formalisée :

3TD/GL/MétaM

32

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

SPEM - Software Process Engineering Metamodel

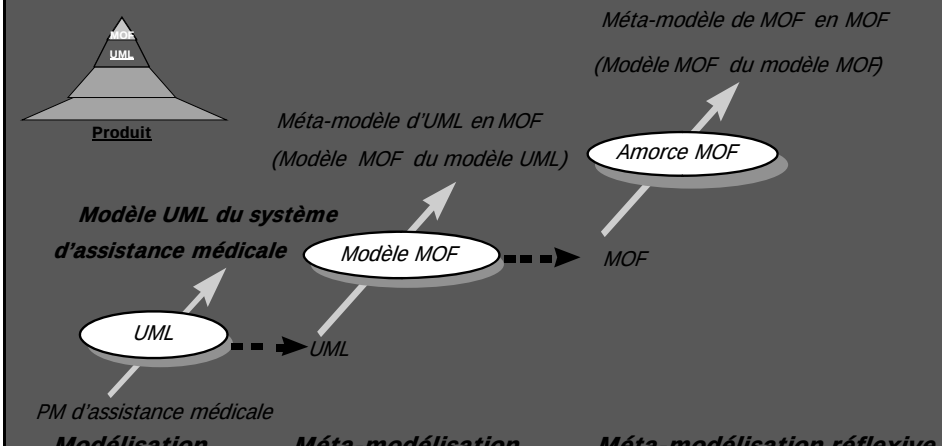
- **SPEM Software Process Engineering Metamodel - que l'on peut traduire par méta-modèle d'ingénierie des procédés logiciels - est un métamodèle (ou modèle décrivant les concepts) visant à décrire le processus de production de logiciels pour répondre à ces problématiques.**
- **C'est autant un méta-modèle de conception de processus qu'un framework conceptuel qui met à disposition les outils et les concepts pour modéliser, documenter, présenter, gérer et rendre concret le développement.**
- **La mise en œuvre de ce méta-modèle sera généralement effectuée par un ingénieur processus, une direction projet ou un gestionnaire de programme ; plus généralement toute personne en charge de l'organisation des projets de développement ou même d'un référentiel processus au sens large.**

BTD/GL/MétaM

33

CENTRALE LYON

Ingénierie des produits



Recherche :
-> langages de modélisation (méta-modèles)
-> modèles réutilisables

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits

- **Variété de langages de modélisation**
- **des langages différents pour l'ingénierie des besoins et celle des systèmes logiciels**

Produit

Intentionnel	**acteur**
	but/scénario
Opérationnel	**dynamique**
	statique

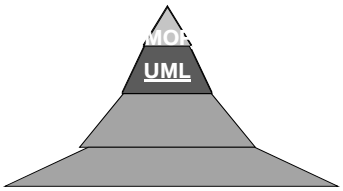
- Justifier le SI / besoins organisationnels et stratégiques de l'organisation
- Aligner ce que le système doit faire et comment il doit le faire avec pourquoi il doit le faire

Difficulté : Faire émerger les obstacles, les conflits, les situations d'accidents, les cas d'exception où le but ne sera pas atteint…

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits



Produit

Objectifs :

- Proposer et formaliser des langages de modélisation adaptés (nouveaux besoins, nouvelles technologies)
- Organiser les modèles (cohérence, transformation…)
- Proposer des modèles génériques (personnalisation, réutilisation…)

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits

Objectifs des langages de modélisation adaptés (nouveaux besoins et technologies) :

- Proposer ou Adapter des méta-modèles (pour le temps réel, le web, l'AOP - aspect-oriented programming ….)
- Combiner au sein d'une même démarche des modèles de produits hétérogènes (reposant sur des méta-modèles différents) mais complémentaires

Exemples :
Décrire des architectures réutilisables
-> UML + ACME : compréhension, formalisation, mécanisme de raffinement et réutilisation de modèles d'architecture

Méthodes de développement pour les systèmes de réalité augmentée
-> UML + modèles IHM : co-conception (SI, IHM)

3TD/GL/MétaM

37

CENTRALE LYON

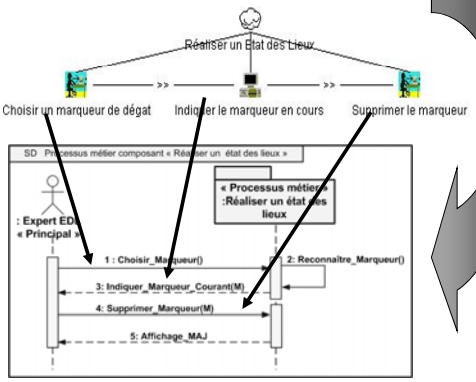
Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Co-conception : des modèles hétérogènes

Activité de coopération

Scénario

Scénario textuel, représentant un cas concret facile à comprendre par tous



Activité de coordination

Développement des systèmes de réalité augmentée : spécialiste SI + spécialiste IHM + ergonome

3TD/GL/MétaM

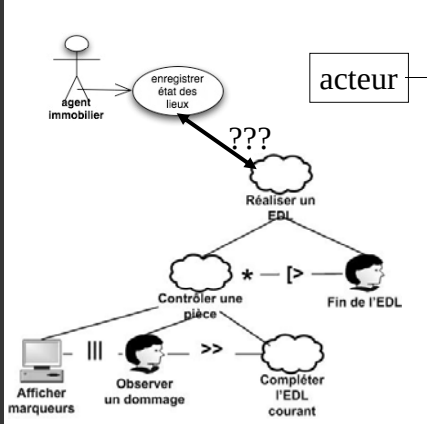
38

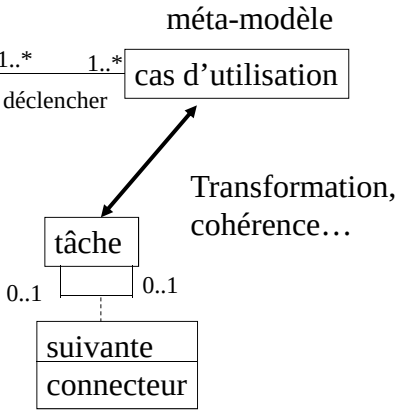
CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits

Objectifs des modèles organisés : cohérence, transformation…





Développement des systèmes de réalité augmentée : spécialiste SI + spécialiste IHM + ergonome

BTD/GL/MétaM 39

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits

Objectifs des modèles génériques (personnalisation, réutilisation…)

Personnalisation : des modèles permettant de produire des logiciels personnalisables dynamiquement (par les utilisateurs finaux ou le système)
Exemple : lignes de produit

Réutilisation : des modèles et/ou des logiciels réutilisables pour produire de nouveaux modèles ou logiciels (par les concepteurs)
Exemple : composants métier, patrons, framework…

Une des techniques de modélisation **: la variabilité**

BTD/GL/MétaM 40

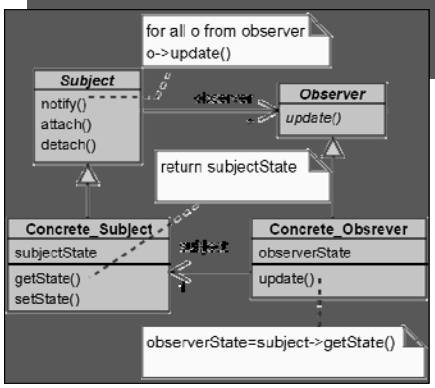
CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Réutilisation et variabilité

Exemple : design pattern (patron observateur)

- **Intention :** ***Définir une dépendance entre les observateurs d'un même sujet telle que, quand le sujet change d'état, tous ces observateurs soient informés et mis à jour .***



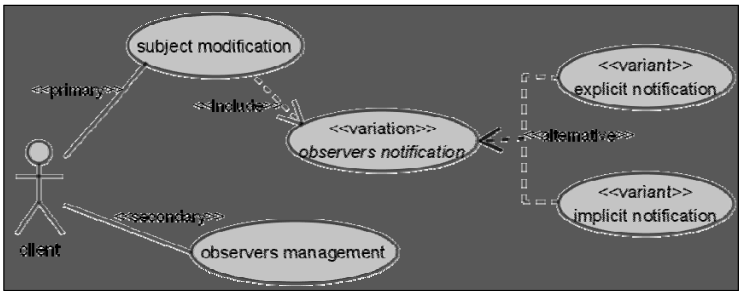
- Des spécifications incomplètes
- Ne met pas en évidence les variantes possibles
- Ne met pas en évidence le caractère obligatoire ou facultatif des propriétés

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Variantes fonctionnelles

Extensions d'UML pour l'expression de la variabilité fonctionnelle

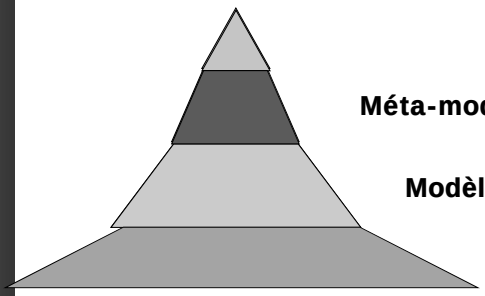


Extensions similaires proposées pour la personnalisation mais aussi pour la représentation de modèles de domaine.

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des produits



Méta-modèles adaptés et adaptables

Modèles génériques

Produit

Au minimum :

Bien formalisée :

Offrant des modèles génériques :

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des processus

Variété de modèles et méta-modèles de processus orientés activité, produit, contexte, décision, but

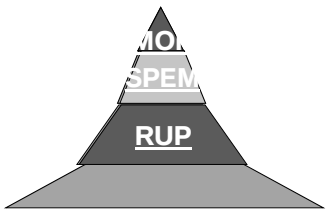
Processus

Décisionnel	**contexte** **décision**
Contrôle	**produit** **activité**

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des processus



Processus

Objectifs :

- Combiner des modèles multi-vues et de niveaux d'abstraction différents
- Réutiliser et Adapter des modèles de processus
- Proposer des moteurs d'exécution, de guidage, de trace

BTD/GL/MétaM

45

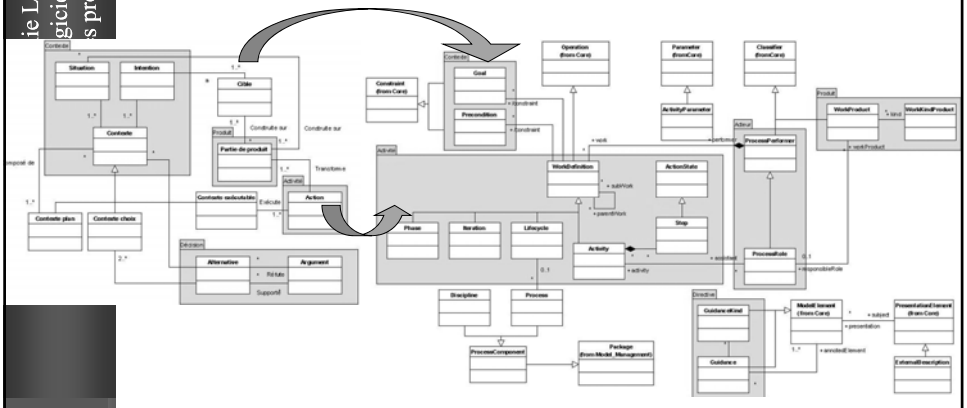
CENTRALE
L Y O N

ie Logiciel : giciels basé s processus

Ingénierie des processus

Objectifs Combiner des modèles multi-vues et de niveaux d'abstraction différents

Etablir des correspondances, des cohérences, entre leur méta-modèle.
ex : NATURE (C. Rolland) SPEM (OMG)



BTD/GL/MétaM

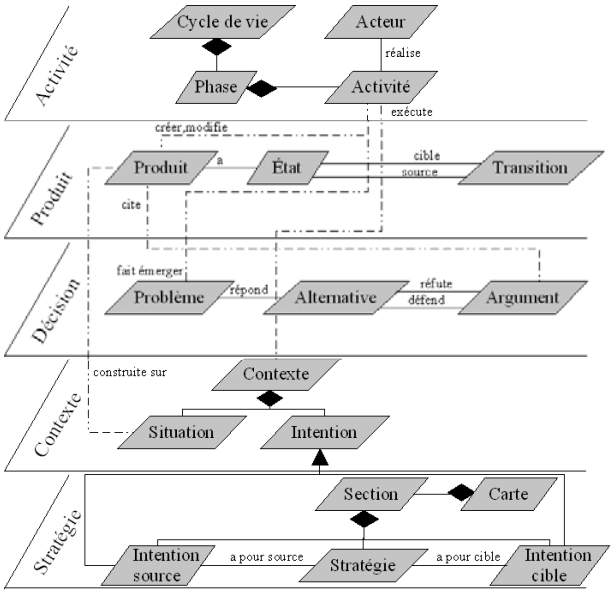
INCREMENT

46

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Ingénierie des processus



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Synthèse : Ingénierie des Produits et des Processus

	Produit	Processus	
Intentionnel	acteur	contextuel	Décisionnel
	but/scénario	décision	
Opérationnel	dynamique	produit	Contrôle
	statique	activité	

Intégration

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Synthèse : Ingénierie des Produits et des Processus

	Produit	Processus	
Intentionnel	acteur	contextuel	Décisionnel
	but/scénario	décision	
Opérationnel	dynamique	produit	Contrôle
	statique	activité	

Modularité, Réutilisation, Personnalisation, Cohérence, Raffinement, Transformation.

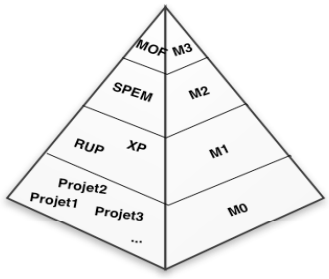
Avec des langages de modélisation et des outils de plus en plus consensuels
UML, MOF, OCL, QVT (Query/Views/Transformations), JMI (Java Meda Interface)…

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Modélisation de processus

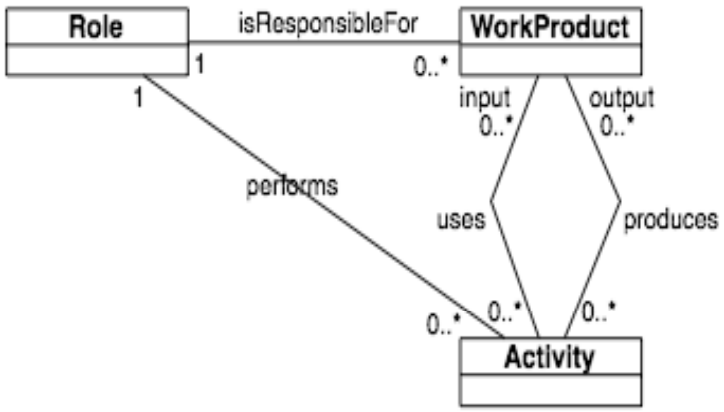
- **Contexte**
  - ➔ **Modélisation de processus**
  - ➔ **SPEM v1.1**
  - ➔ **Standard OMG**
- **Problématique**
  - ➔ **Ambiguïtés**
  - ➔ **Insuffisances**



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

SPEM : modèle conceptuel

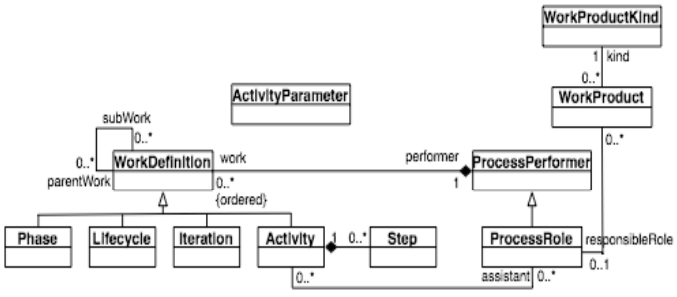


BTD/GL/MétaM 51

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

SPEM : méta-modèle

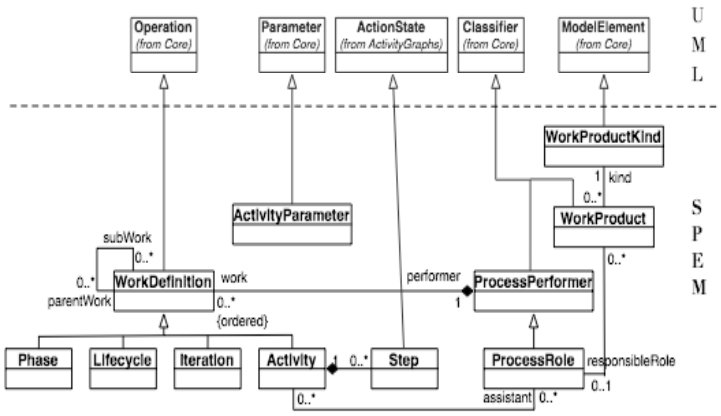


BTD/GL/MétaM 52

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

SPEM : profil UML



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

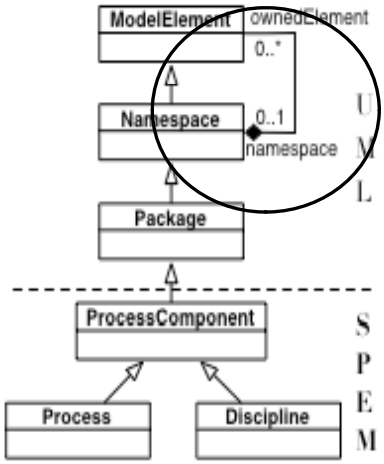
Méta-modèle MOF vs profil UML

	Méta-modèle	*Profil UML*
Concepts	domaine uniquement	définis par des stéréotypes
Contraintes OCL	ajout de propriétés	ajout de propriétés restriction de la sémantique UML
Outillage	spécifique	réutilisation d'outils UML

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modèle vs profil *(exemple)*

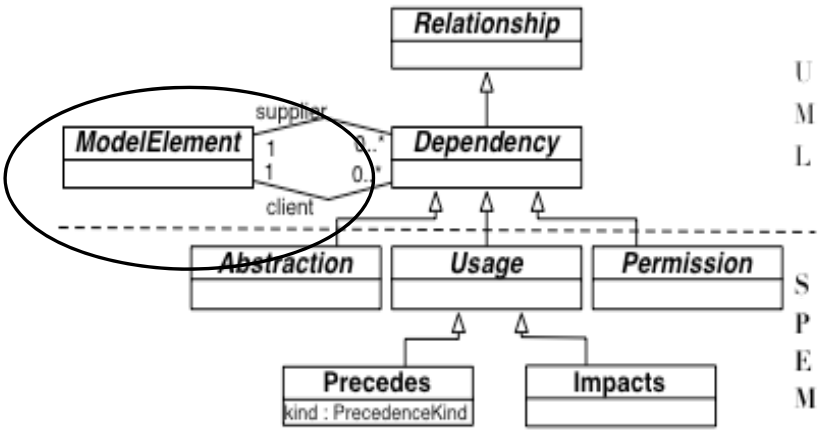


BTD/GL/MétaM 55

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Méta-modèle vs profil *(exemple)*

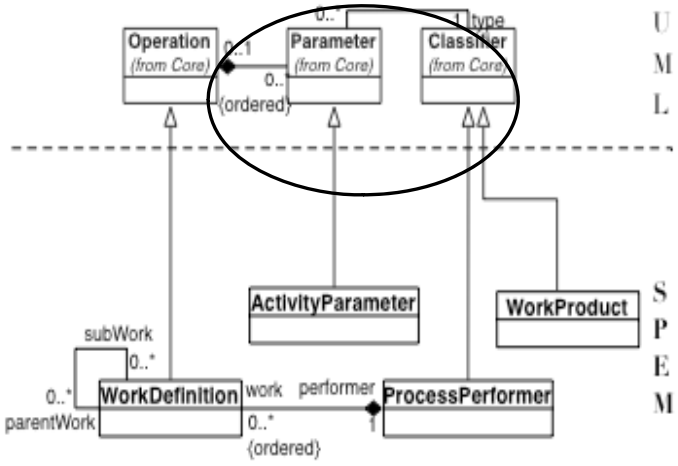


BTD/GL/MétaM 56

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

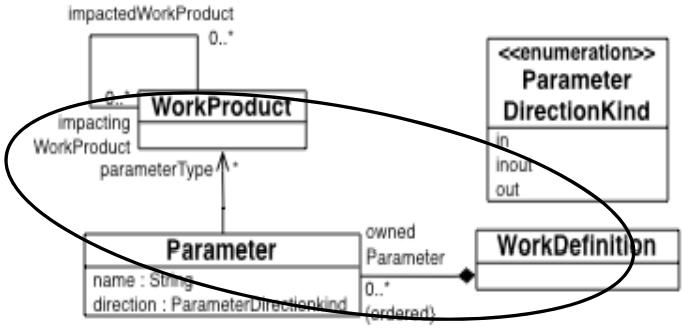
Méta-modèle vs profil *(exemple)*



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Solution 1: redéfinition de relation



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Solution 2 : contraintes OCL

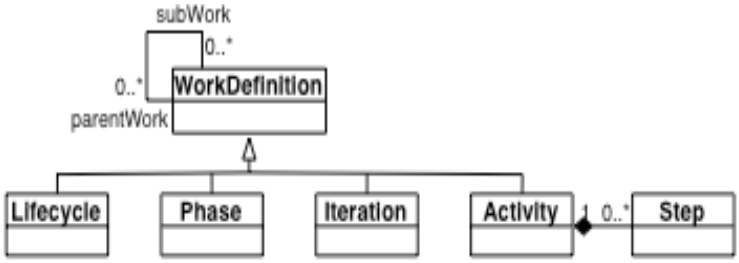
« An* Impacts *dependency acts from one WorkProduct to another »
context **Impacts** *inv* **:**
self**.supplier.isOclType(*WorkProduct*)**
and
self**.client.isOclType(*WorkProduct*)**

« A* Precedes *dependency acts from one WorkDefinition to another »
context **Precedes** *inv* **:**
self**.supplier.isOclType(*WorkDefinition*)**
and
self**.client.isOclType(*WorkDefinition*)**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

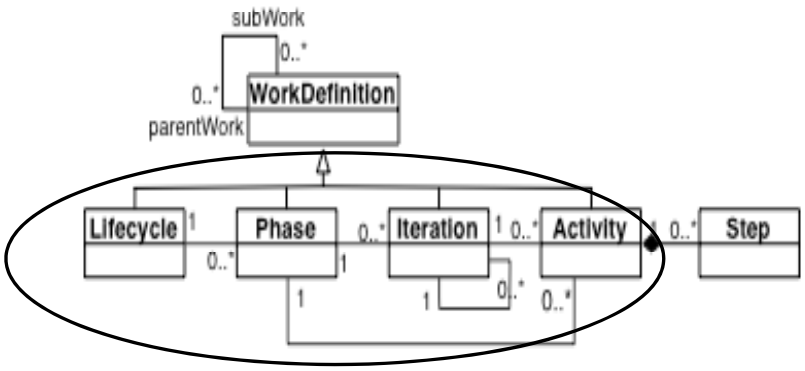
Spécialisation des *WorkDefinition*



CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Spécialisation des *WorkDefinition*

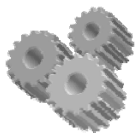
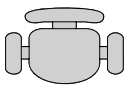
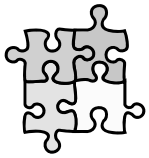


CENTRALE LYON

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Vers une ingénierie des processus

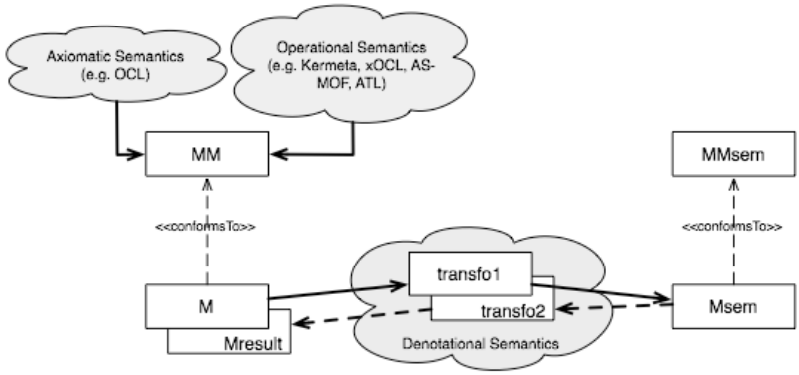
- **Méta-procédé**
- **Composants de processus**
- **Evaluation et adaptation des processus**

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel : Développement de logiciels basé sur des modèles et des processus

Vers une exécutabilité des modèles de processus



BTD/GL/MétaM

63