
Spécification d'exigences dans le contexte de lignes de produits

**Camille SALINESI
Daniel DIAZ
Raul MAZO
Alberto LORA
Olfa DJEBBI**

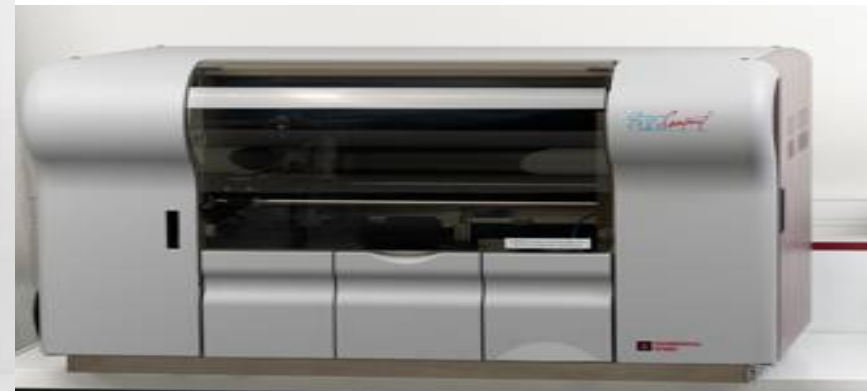
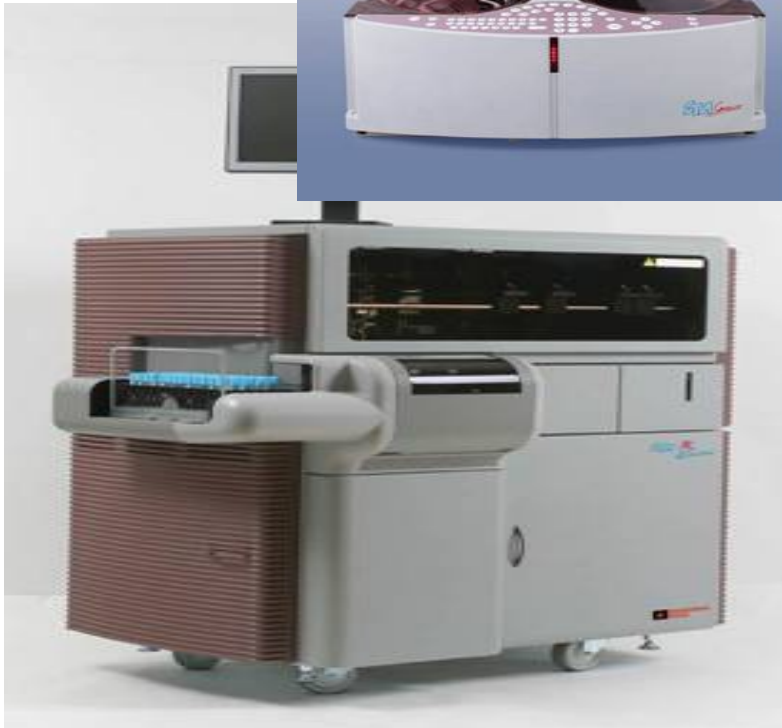
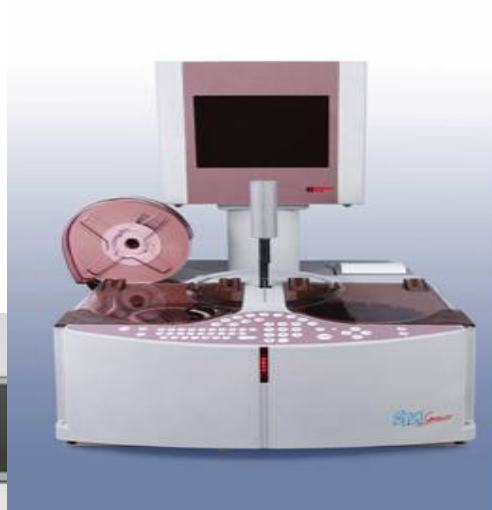
Centre de Recherche en Informatique CRI

L'Ingénierie des Exigences – Application aux Lignes de Produits

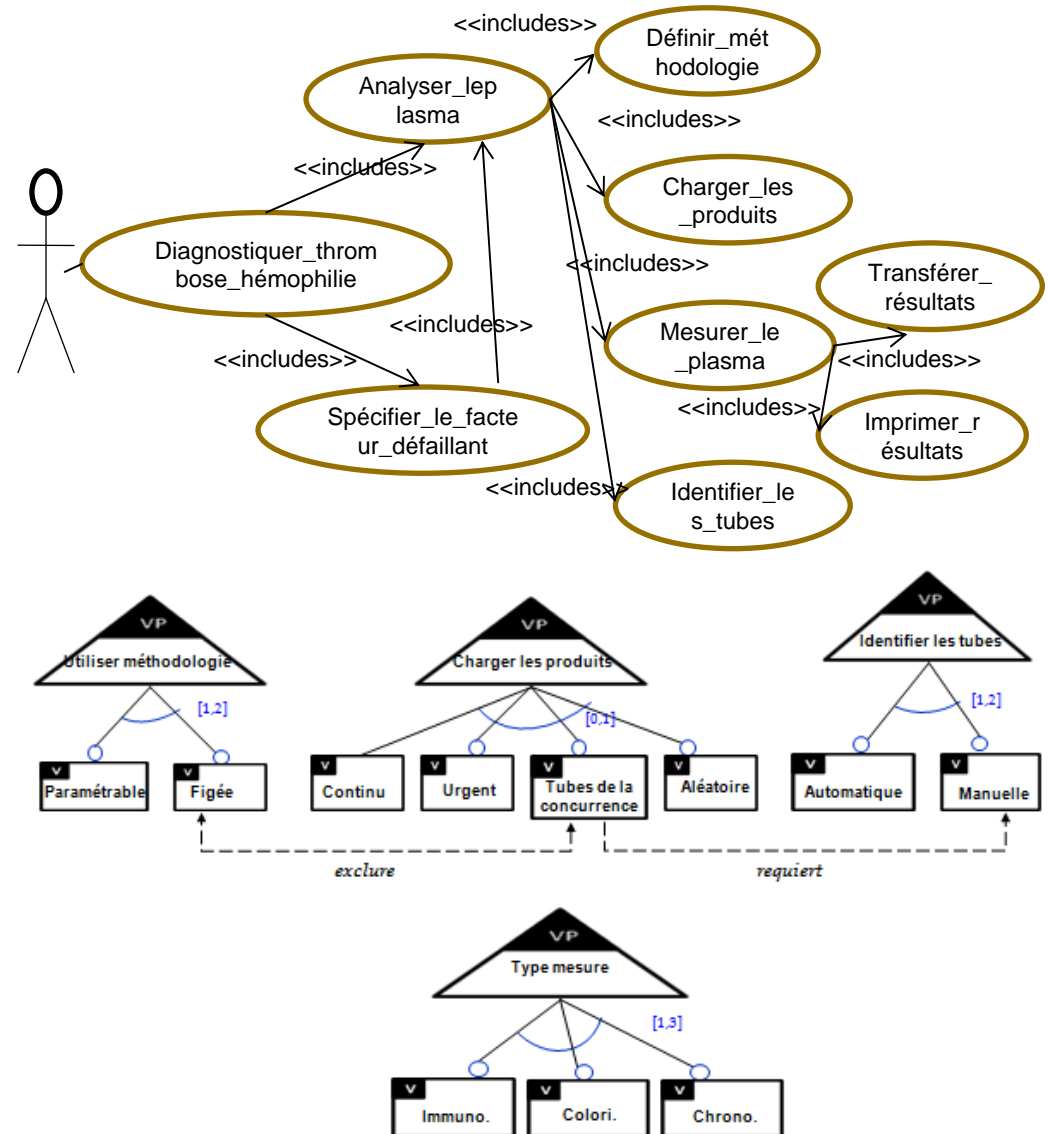
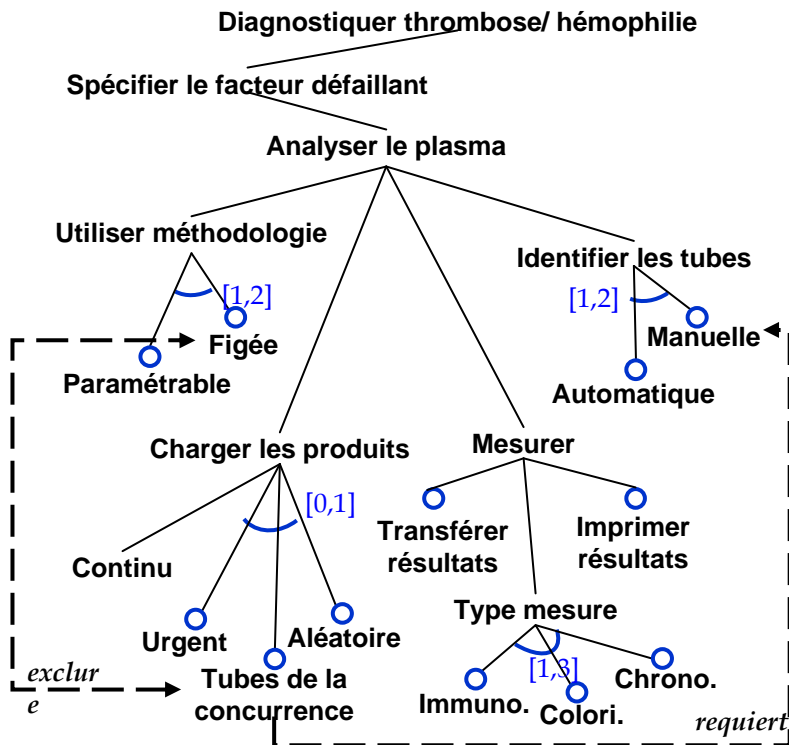
- **Lignes de produits:** un ensemble de systèmes qui:
 - Partagent un ensemble **commun** de caractéristiques
 - Satisfont les besoins d'une **mission** particulière
 - Sont développés à partir d'un noyau commun d'une manière **prescrite**



Modélisation des exig. d'une LP



Modélisation des exigences d'une LP



La programmation par contraintes offre

- Contraintes arithmétiques (linéaires et non linéaires).
 $X+Y < Z$ or $X*Y <> Z$.
- Contraintes symboliques. *atmost* (2,[X,Y,Z,T],10): au moins 2 variables parmi X,Y,Z,T peuvent prendre la valeur 10.
- Contraintes booléennes: expressions booléennes avec: \wedge , \vee , \neg , \Rightarrow , \Leftrightarrow ,...
- Contraintes réifiées: pour raisonner à partir de la satisfaction/insatisfaction d'une contrainte. $X < Y \Rightarrow K=8$. Quand le solver détecte que $X < Y$ il fixe $K=8$, de la même manière s'il détecte $K < 8$ il fixe $X >= Y$. Contrainte réifiée: $A \Leftrightarrow (X < Y)$, $B \Leftrightarrow K=8$, $A \Rightarrow B$.

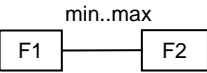
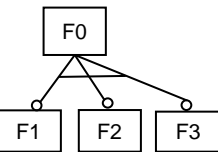

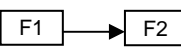
“Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.” [E. Freuder]

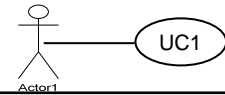
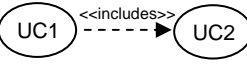
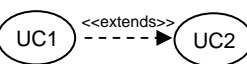
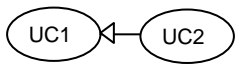
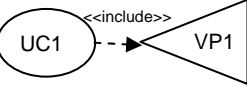

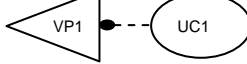
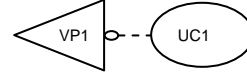
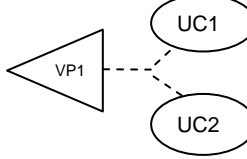
VariaMos



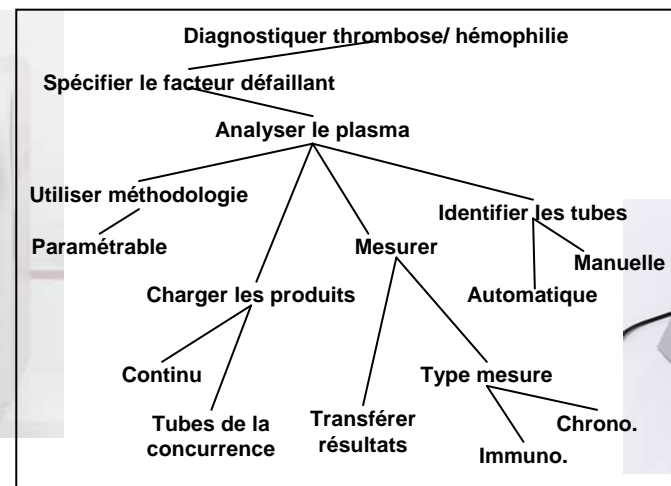
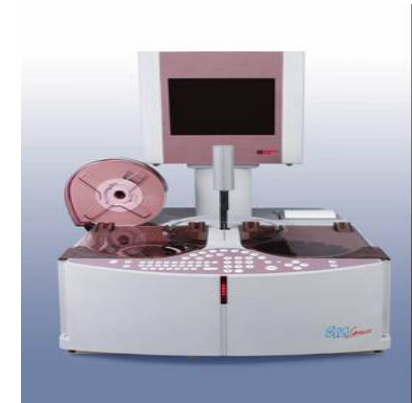
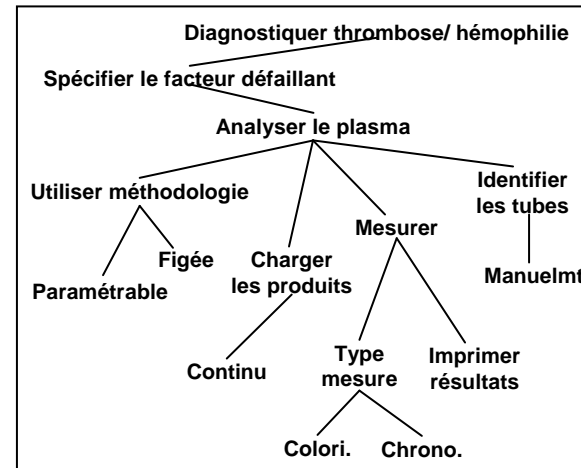
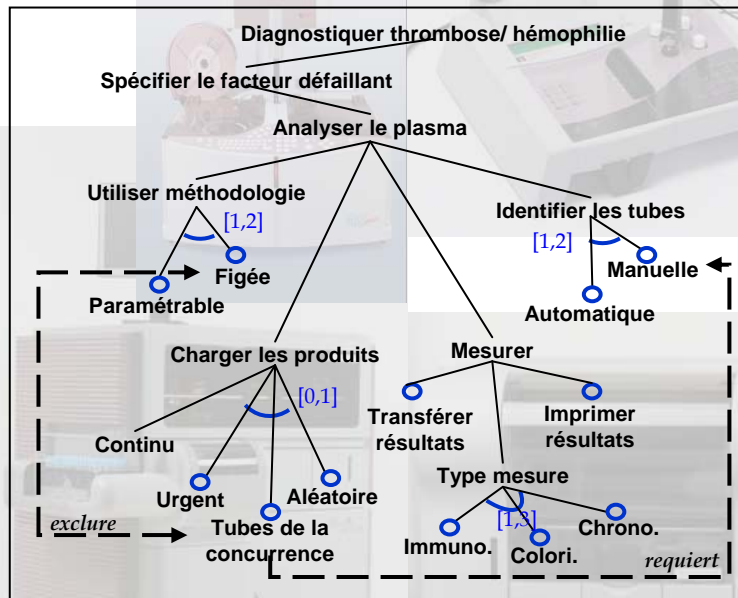
GNU Prolog

Processus de représentation de MLPs

Name and Semantic	Cardinality-based Representation	CP Representation
Feature cardinality A feature cardinality is a sequence of intervals of the form $[n1..n1'] \dots [nk..nk']$. If $F2$ is selected at least one time, the father feature ($F1$) must be selected as well.	$\text{min}..max$ 	$\text{domain}([F1], 0,1),$ $\text{domain}([F2], \text{min}, \text{max}),$ $\text{min} \geq 0,$ $F1 > 0 \implies \text{min} \leq F2 \leq \text{max}$ $F1 = 0 \implies F2 = 0$
Group cardinality If the father feature ($F0$) is selected, the value of Min and Max specifies the permissible range of optional features to be selected from the bundle ($F1$, $F2$, and $F3$). If one of the children is selected, the father feature ($F0$) must be selected as well.		$\text{domain}([F0, F1, F2, F3], 0,1),$ $F1 \leq F0,$ $F2 \leq F0,$ $F3 \leq F0,$ $F0 \implies F1 + F2 + F3 \geq \text{Min},$ $F0 \implies F1 + F2 + F3 \leq \text{Max}$
Exclusion Indicates that both features ($F1$, $F2$) cannot be selected in one product configuration and are therefore mutually exclusive.		$\text{domain}([F1, F2], 0, 1),$ $F1 + F2 \leq 1$
Requires If one feature ($F1$) is selected the required feature ($F2$) has to be selected as well, ignoring their position in the feature tree.		$\text{domain}([F1, F2], 0, 1),$ $F1 \leq F2$

Names	Use-case Representations	CP Representations
Relationship		$\text{domain}([UC1], 0, 1),$
Include		$\text{domain}([UC1, UC2], 0, 1),$ $UC1 \leq UC2$
Extend		$\text{domain}([UC1, UC2], 0, 1),$ $UC1 \leq UC2$
Generalization		$\text{domain}([UC1, UC2], 0, 1),$ $UC2 \leq UC1$
Include UC-VP		$\text{domain}([UC1, VP1], 0, 1),$ $UC1 \leq VP1$
Extend UC-VP		$\text{domain}([UC1, VP1], 0, 1),$ $VP1 \leq UC1$
Mandatory		$\text{domain}([UC1, VP1], 0, 1),$ $VP1 = UC1$
Optional		$\text{domain}([UC1, VP1], 0, 1),$ $UC1 \leq VP1$
Alternative choice cardinality		$\text{domain}([UC1, UC2, VP1], 0, 1),$ $UC1 \leq VP1,$ $UC2 \leq VP1,$ $VP1 \implies UC1 + UC2 \geq \text{Min},$ $VP1 \implies UC1 + UC2 \leq \text{Max}$

Processus de configuration – Modèle à Caractéristiques



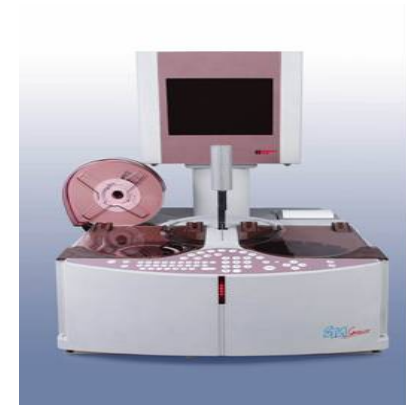
Processus de configuration – Modèle à Contraintes



```

p(L):-
L=...
fd_domain(...).
Spécifier_le_facteur_défaillant #= Diagnostiquer_thrombose_hémophilie,
Analyser_le_plasma #= Spécifier_le_facteur_défaillant,
Utiliser_méthodologie #= Analyser_le_plasma,
Paramétrable #=< utiliser_méthodologie,
Figée #=< utiliser_méthodologie,
Charger_les_produits #= Analyser_le_plasma,
Continu #= Charger_les_produits,
Urgent #=< Charger_les_produits,
Tubes_de_la_concurrence #=< Charger_les_produits,
Aléatoire #=< Charger_les_produits,
Mesurer #= Analyser_le_plasma,
Transférer_résultats #=< Mesurer,
Type_mesure #=< Mesurer,
Immuno. #=< Type_mesure,
Colori. #=< Type_mesure,
Chrono. #=< Type_mesure,
Imprimer_résultats #=< Mesurer,
Identifier_les_tubes #= Analyser_le_plasma,
Automatique #=< Identifier_les_tubes,
Manuelle #=< Identifier_les_tubes,
Utiliser_méthodologie #==> 1 #=> (Paramétrable + Figée) #=< 2,
Charger_les_produits #==> 0 #=> (Urgent + Tubes_de_la_concurrence + Aléatoire) #=< 1,
Type_mesure #==> 1 #=> (Immuno. + Colori. + Chrono.) #=< 3,
Identifier_les_tubes #==> 1 #=> (Automatique + Manuelle) #=< 2,
Figée + Tubes_de_la_concurrence #=< 1,
Tubes_de_la_concurrence #=< Manuelle,
fd_labeling(L).
    
```

| ?- L=[1,1,1,1,1,1,1,1,1,
0,0,1,0,1,0,1,1,
1,1,0,1], p(L).



| ?- L=[1,1,1,1,1,0,1,1,
1,0,1,1,1,1,0,1,
0,1,1,1], p(L).



Processus de construction

```
Spécifier_le_facteur_défaillant #=  
Diagnostiquer_thrombose_hémophilie,  
Analyser_le_plasma #= Spécifier_le_facteur_défaillant,  
Utiliser_méthodologie #= Analyser_le_plasma,  
Paramétrable #= utiliser_méthodologie,  
Figée #= utiliser_méthodologie,  
Charger_les_produits #= Analyser_le_plasma,  
Continu #= Charger_les_produits,  
Mesurer #= Analyser_le_plasma,  
Type_mesure #= Mesurer,  
Colori. #= Type_mesure,  
Chrono. #= Type_mesure,  
Imprimer_résultats #= Mesurer,  
Identifier_les_tubes #= Analyser_le_plasma,  
Manuelle #=< Identifier_les_tubes
```

```
Spécifier_le_facteur_défaillant #=  
Diagnostiquer_thrombose_hémophilie,  
Analyser_le_plasma #= Spécifier_le_facteur_défaillant,  
Utiliser_méthodologie #= Analyser_le_plasma,  
Paramétrable #= utiliser_méthodologie,  
Charger_les_produits #= Analyser_le_plasma,  
Continu #= Charger_les_produits,  
Tubes_de_la_concurrence #= Charger_les_produits,  
Mesurer #= Analyser_le_plasma,  
Transférer_résultats #= Mesurer,  
Type_mesure #= Mesurer,  
Immuno. #= Type_mesure,  
Chrono. #= Type_mesure,  
Identifier_les_tubes #= Analyser_le_plasma,  
Automatique #= Identifier_les_tubes,  
Manuelle #= Identifier_les_tubes,
```

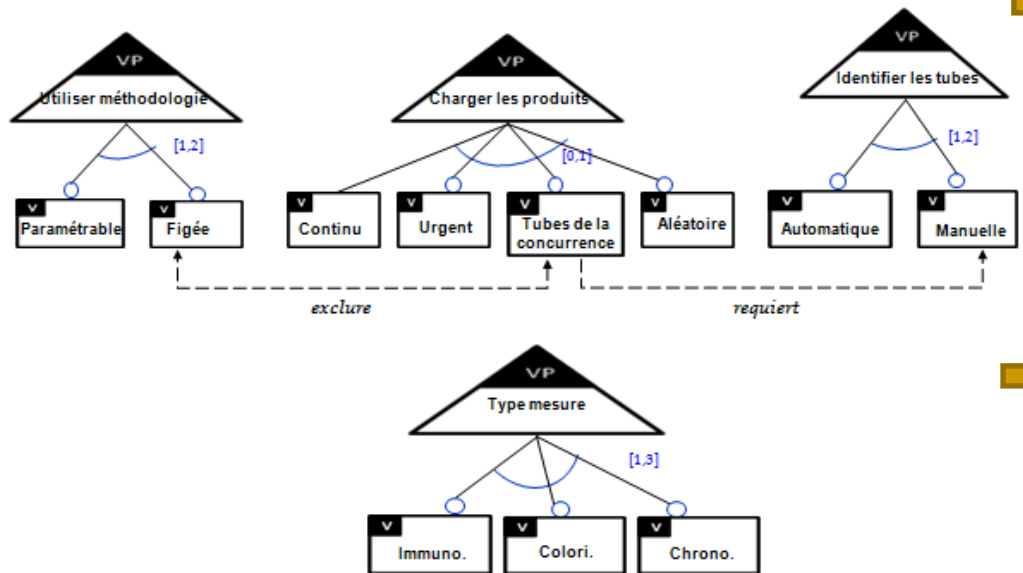
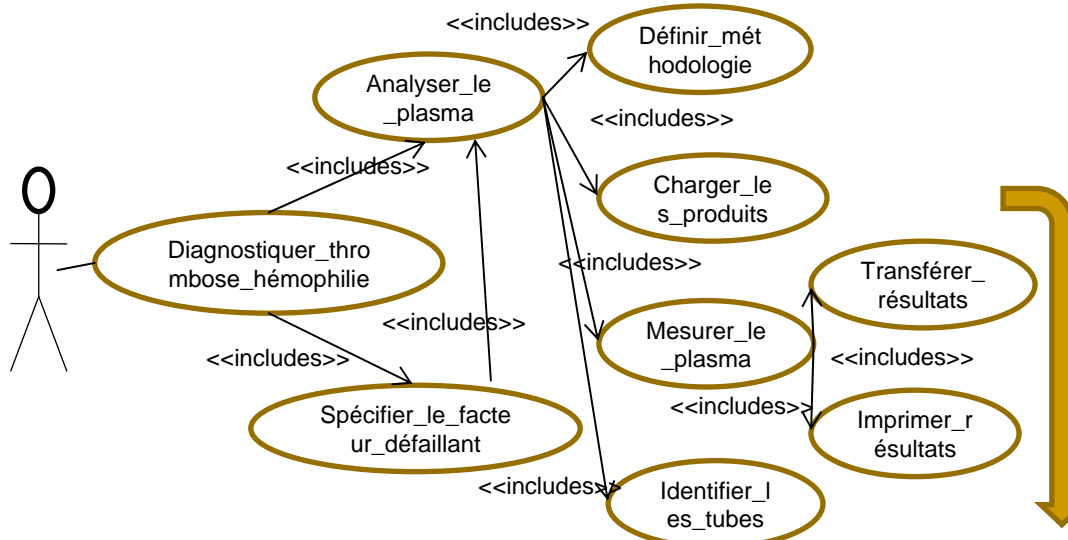
```
p(L):-  
L = [Diagnostiquer_thrombose_hémophilie, Spécifier_le_facteur_défaillant,  
Analyser_le_plasma, Utiliser_méthodologie, Paramétrable, Figée,  
Charger_les_produits, Continu, Urgent, Tubes_de_la_concurrence,  
Aléatoire, Mesurer, Transférer_résultats, Type_mesure, Immuno., Colori.,  
Chrono., Imprimer_résultats, Identifier_les_tubes, Automatique, Manuelle],
```

```
fd_domain([...]...)
```

```
Spécifier_le_facteur_défaillant #= Diagnostiquer_thrombose_hémophilie,  
Analyser_le_plasma #= Spécifier_le_facteur_défaillant,  
Utiliser_méthodologie #= Analyser_le_plasma,  
Paramétrable #=< utiliser_méthodologie,  
Figée #=< utiliser_méthodologie,  
Charger_les_produits #= Analyser_le_plasma,  
Continu #= Charger_les_produits,  
Urgent #=< Charger_les_produits,  
Tubes_de_la_concurrence #=< Charger_les_produits,  
Aléatoire #=< Charger_les_produits,  
Mesurer #= Analyser_le_plasma,  
Transférer_résultats #=< Mesurer,  
Type_mesure #=< Mesurer,  
Immuno. #=< Type_mesure,  
Colori. #=< Type_mesure,  
Chrono. #=< Type_mesure,  
Imprimer_résultats #=< Mesurer,  
Identifier_les_tubes #= Analyser_le_plasma,  
Automatique #=< Identifier_les_tubes,  
Manuelle #=< Identifier_les_tubes,  
Utiliser_méthodologie #==> 1 #=> (Paramétrable + Figée) #=< 2,  
Charger_les_produits #==> 0 #=> (Urgent+Tubes_de_la_concurrence+Aléatoire) #=< 1,  
Type_mesure #==> 1 #=> (Immuno. + Colori. + Chrono.) #=< 3,  
Identifier_les_tubes #==> 1 #=> (Automatique + Manuelle) #=< 2,  
Figée + Tubes_de_la_concurrence #=< 1,  
Tubes_de_la_concurrence #=< Manuelle,
```

```
fd_labeling(L).
```

Processus d'intégration multi-vues

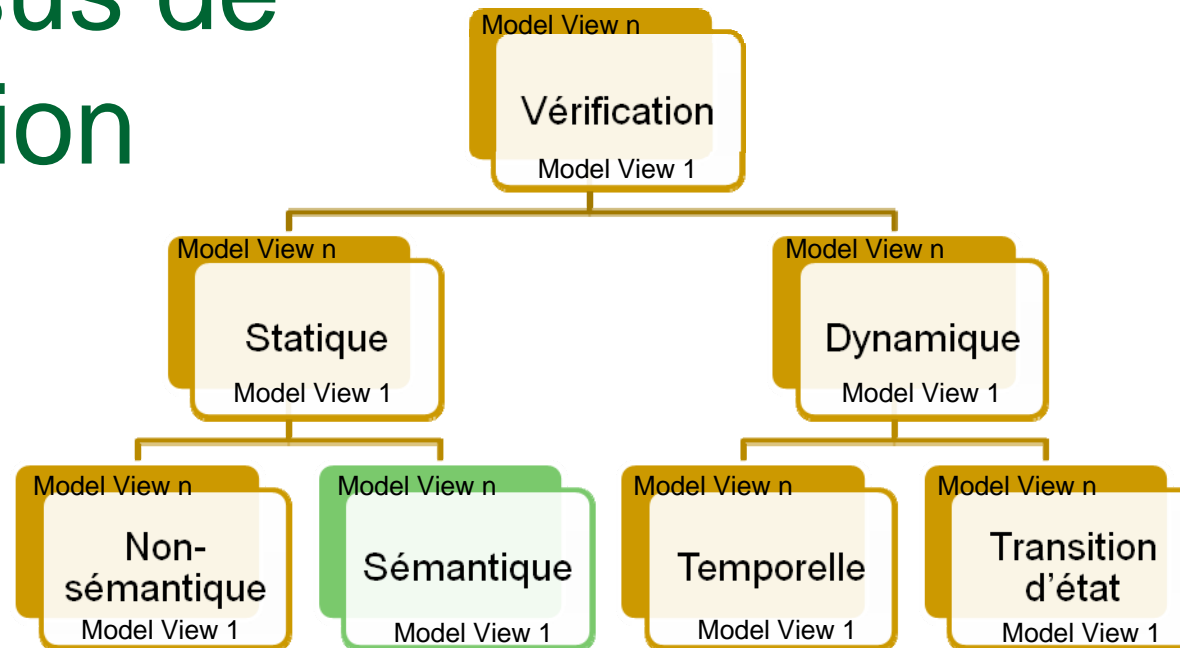


p(L):-
 L = [Diagnostiquer_thrombose_hémophilie, Spécifier_le_facteur_défaillant, Analyser_le_plasma, Utiliser_méthodologie, Paramétrable, Figée, Charger_les_produits, Continu, Urgent, Tubes_de_la_concurrence, Aléatoire, Mesurer, Transférer_résultats, Type_mesure, Immuno., Colori., Chrono., Imprimer_résultats, Identifier_les_tubes, Automatique, Manuelle],
 fd_domain([...]...)

Spécifier_le_facteur_défaillant #= Diagnostiquer_thrombose_hémophilie,
 Analyser_le_plasma #= Spécifier_le_facteur_défaillant,
 Utiliser_méthodologie #= Analyser_le_plasma,
 Paramétrable #=< utiliser_méthodologie,
 Figée #=< utiliser_méthodologie,
 Charger_les_produits #= Analyser_le_plasma,
 Continu #= Charger_les_produits,
 Urgent #=< Charger_les_produits,
 Tubes_de_la_concurrence #=< Charger_les_produits,
 Aléatoire #=< Charger_les_produits,
 Mesurer #= Analyser_le_plasma,
 Transférer_résultats #=< Mesurer,
 Type_mesure #=< Mesurer,
 Immuno. #=< Type_mesure,
 Colori. #=< Type_mesure,
 Chrono. #=< Type_mesure,
 Imprimer_résultats #=< Mesurer,
 Identifier_les_tubes #= Analyser_le_plasma,
 Automatique #=< Identifier_les_tubes,
 Manuelle #=< Identifier_les_tubes,
 Utiliser_méthodologie #==> 1 #=> (Paramétrable + Figée) #=< 2,
 Charger_les_produits #==> 0 #=>
 (Urgent+Tubes_de_la_concurrence+Aléatoire) #=< 1,
 Type_mesure #==> 1 #=> (Immuno. + Colori. + Chrono.) #=< 3,
 Identifier_les_tubes #==> 1 #=> (Automatique + Manuelle) #=< 2,
 Figée + Tubes_de_la_concurrence #=< 1,
 Tubes_de_la_concurrence #=< Manuelle,

fd_labeling(L).

Processus de vérification



$p(L)$:-

$L = [X1, X2, X3, X4, X5],$
 $fd_domain([X1,X2,X3], 0, 1),$
 $fd_domain([X4,X5], 0, 10),$
 $X1 \#>= X2,$
 $X2 \# = X3,$
 $X5 \#> 3,$
 $X3 \#<=> X4 \#>= 5 \#V X5 \#>= 5,$
 $fd_labeling(L).$

trouver 1 sol, puis une autre,...

| ?- $p(L).$

fixer les valeurs de X1 et X3

| ?- $L=[1,_,1,_,_], p(L).$

trouver toutes les solutions

| ?- $findall(L, p(L), LSol), length(LSol, NbSol).$

Conclusions

- Notre approche peut être utilisée pour:
 - Traiter le problème de la **spécification** (permettant de représenter toutes les vues correspondantes).
 - Traiter le problème de la **construction** (permettant d'utiliser les modèles de prod. pour const. le MLP).
 - Traiter le problème de **l'intégration des vues** (permettant d'examiner la cohérence du système)
 - Traiter le problème de la **vérification et validation** (permettant d'exécuter opérations d'analyse).
-

Des remarques?
Des suggestions?
Autres propositions?

Merci de votre attention!

Références

- Dines Bjorner. *Software Engineering 3 Domains, requirements and Software Design*. Springer-Verlag 2006
 - Klaus Pohl, Gunter Bockle, and Frank van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, July 2005
 - Freuder, E.C.: *Synthesizing Constraint Expressions*, in: *Communications ACM* 21(11): 958-966, ACM, 1978
-

