Les tests logiciels : séance 3

M.Madani

ISIL - HEPL

2015-2016

Références

- Software Testing Foundations, Andreas Spillner, Tilo Linz, Hans Schaefer, Rocky Nook, 2007.
- Foundations of Software Testing, Aditya P. Mathur,Pearson Education India,2008.
- Software Testing and Quality Assurance,Kshirasagar Naik,Priyadarshi Tripathy,Wiley,2008.
- Advanced Software Testing Vol. 1, Rex Black, Rocky Nook, 2008.
- The Domain Testing Workbook, Cem Kaner,Sowmya Padmanabhan, Douglas Hoffman, 2013.

Partitionner en classes d'équivalence

- En quelques mots
  - Vise à minimiser la génération de cas de tests pas nécessaires.
  - Les entrées et les sorties attendues sont considérées.
  - Souvent repertoriée dans la catégorie des tests en boîte noire(si basé sur une spécification d'exigences).
  - Peut être utilisée à différents niveaux(system testing, integration testing et unit testing).
- Hypothèses
  - Le programme se comporte de la même manière dans chaque sous ensemble du domaine d'entrée.
- Limitations
  - Sans code, perte d'informations pour élaborer les cas de tests.

Classes d'équivalence

- Le domaine d'entrée est partitionné en un nombre fini de sous domaines qui constituent les classes d'équivalence.
- Le partitionnement est défini par une relation $\mathcal{R} : I \rightarrow I$. Normalement

$$I = \bigcup_{i=0}^{n} EC_i$$

et

$$\bigcap_{i=0}^{n} EC_i = \emptyset$$

- Le système est sensé traiter les données d'une même classe de la même manière.
- Les sous domaines serviront pour sélectionner les données en entrée.

Relation d'équivalence

- Une relation d'équivalence $\mathcal{R}$, dans un ensemble I est une relation
  - binaire : une relation binaire $\mathcal{R}$ d'un ensemble E vers un ensemble F est définie par une partie G de $E \times F$
  - réflexive : $\forall x \in I, x\mathcal{R}x$;
  - symétrique : $\forall\ (x, y) \in I^2, (x\mathcal{R}y) \Rightarrow (y\mathcal{R}x)$;
  - transitive : $\forall\ (x, y, z) \in I^3, (x\mathcal{R}y \wedge y\mathcal{R}z) \Rightarrow (x\mathcal{R}z)$.
- Si un ensemble d'éléments safisfont une relation d'équivalence, ils forment une classe d'équivalence sur cette relation.
- Exemples
  - L'égalité numérique est un exemple de relation d'équivalence.
  - Le domaine d'un chemin forme une classe d'équivalence sur la relation qui définit les chemins.

Exemple

1. Le bonus accordé pour des courtiers en assurance par leur employeur est

Nombre de contrats	Avantage
0	0%
$1 \leq 3 \leq 5$	10%
> 5	16%

2. Les classes d'équivalence sont

Paramètre	vEC	iEC	Val. représentative
Nbr de contrats x	0		0
	$1 \leq x \leq 5$		4
	$x > 5$		10
		$x < 0$	-2
		$x > 100$	101

Les principales étapes à réaliser

1. Identifier dans les documents (spécifications si existantes) les entrées et les sorties ainsi que les restrictions et les conditions associées.
   - Identifier les variables
   - Déterminer leur type et l'échelle sur lesquelles elles se trouvent.
   - Déterminer des dépendances avec d'autres variables.
2. Construire une table de classes d'équivalence.
3. Choisir les meilleurs valeurs représentatives.
4. Créer les tests en combinant les valeurs représentatives.

Identifier les variables

- Au départ d'un fragment de spécification (exemple slide 6)
  - En entrées
    - Nombre de contrats
    - Salaire (pas explicite)
    - Avantage
  - En sortie
    - Salaire
- Sur base d'une boîte de dialogue ou d'une IHM.
- Sur base des interfaces des unités de code.

Les échelles

Echelle

- nominale
  - Ensemble fini de valeurs, pas de comparaison possible, pas d'ordre des éléments.
  - Opérations : $=$, $\neq$
  - Qualitative
  - Mesure de la tendance centrale : le mode.
  - Exemple : {red,blue,yellow}
- ordinale
  - Relation d'ordre faible entre éléments.
  - Opérations : $<$, $>$, $=$,$\neq$.
  - Qualitative.
  - Mesure de la tendance centrale : le mode et la médiane.
  - Pas de notion de magnitude de la distance entre éléments.
  - Exemple : {A,B,C,D}. $A < C$ et $A < D$.

Les échelles - suite

Echelle

- intervalle
  - Relation d'ordre faible et notion de distance entre éléments.
  - Opérations : $+$ et $-$, $<$ et $>$, $=$ et $\neq$.
  - Bien que représentant des données quantitatives, les opérations de multiplication et division n'ont pas de sens.
  - Mesure de la tendance centrale : le mode et la médiane et la moyenne arithmétique.
  - Le zéro est arbitraire
- Exemple 1 :
  - $\{2,3,4,5\}$. $2 < 3$ et $2 < 4$
  - 4 est plus éloigné de 2 que 3.
  - Dans quelle proportion 4 est-il plus grand par rapport à 2 ?
- Exemple 2 :
  - 5°Celsius = 41°Fahrenheit, 10°Celsius = 50°Fahrenheit, 0°Celsuis $\neq$ 0°Fahrenheit. (equal-interval ratio scale ?)
  - (1900 - 1800 = 100 ans) $<$ (1500-1300 = 200 ans).

Les échelles - fin

Echelle

- Ratio
  - Comparaison de différence par rapport à 0.
  - Reprend les caractéristiques de l'échelle intervalle.
  - Zéro absolu.
  - Données quantitatives.
  - Opérations : $+$ et $-$, $<$ et $>$, $=$ et $\neq$, $*$ et $/$.
  - Mesure de la tendance centrale : le mode et la médiane, la moyenne arithmétique et géométrique.
  - Exemple : les poids, les distances,les températures en °K ...
- Absolue : ne permet pas de transformation(vers une autre échelle) sauf $f(x) = x$.

Les principales étapes à réaliser

1. Identifier dans les documents (spécifications si existantes) les entrées et les sorties ainsi que les restrictions et les conditions associées.
   - Identifier les variables
   - Déterminer leur type et l'échelle sur lesquelles elles se trouvent.
   - Déterminer des dépendances avec d'autres variables.
2. Construire une table de classes d'équivalence.
3. Choisir les meilleurs valeurs représentatives.
4. Créer les tests en combinant les valeurs représentatives.

Déterminer les classes d'équivalence

1. Si un domain numérique et continu est spécifié pour une variable, créer
   - une vEC valide ;
   - deux iEC.
2. Si une suite de n valeurs doivent être entrées
   - une $vEC \ni$ n valeurs ;
   - 2 $iEC \ni -/+$ que le nombre attendu de vals.
3. Pour chaque valeur traîtée de manière particulière dans un ensemble de valeurs, créer
   - une vEC pour chaque valeur considérée ;
   - une iEC contenant les autres valeurs possibles.
4. Pour des conditions, créer
   - une vEC(condition remplie) ;
   - une iEC(condition non remplie).

Les principales étapes à réaliser

1. Identifier dans les documents (spécifications si existantes) les entrées et les sorties ainsi que les restrictions et les conditions associées.
   - Identifier les variables
   - Déterminer leur type et l'échelle sur lesquelles elles se trouvent.
   - Déterminer des dépendances avec d'autres variables.
2. Construire une table de classes d'équivalence.
3. Choisir les meilleurs valeurs représentatives.
4. Créer les tests en combinant les valeurs représentatives.

Les principales étapes à réaliser

1. Identifier dans les documents (spécifications si existantes) les entrées et les sorties ainsi que les restrictions et les conditions associées.
   - Identifier les variables
   - Déterminer leur type et l'échelle sur lesquelles elles se trouvent.
   - Déterminer des dépendances avec d'autres variables.
2. Construire une table de classes d'équivalence.
3. Choisir les meilleurs valeurs représentatives.
4. Créer les tests en combinant les valeurs représentatives.

Combiner les valeurs représentatives

- Règles pour combiner
  - Les valeurs représentatives choisies pour les vEC doivent idéalement être toutes combinées entre elles.
  - La valeur représentative d'une iEC ne doit être combinée qu'avec des données représentatives des vEC.
  - Nombre de cas de tests valides pour n paramètres : $|vEC_{p_1}| \times |vEC_{p_2}| \times ... \times |vEC_{p_n}|$
- Règles pour restreindre le nombre de cas de tests
  - Classer les cas de tests par la fréquence de leur occurences.
  - Préférer les cas de tests avec des valeurs choisies aux frontières des classes.
  - Au lieu de tester toutes les combinaisons de vEC, s'assurer simplement que chaque vEC est représentée au moins dans un test.
  - Ne pas combiner les données représentatives des iEC avec celles des autres iEC

Classes d'équivalence simples pour une date

#	Paramètre	vEC	iEC	Valeur repr.
1	Jour j	$1 \leq j \leq 31$		20
2			$j \leq 0$	0
3			$j > 31$	32
4	Mois m	$1 \leq m \leq 12$		7
5			$m \leq 0$	0
6			$m > 12$	13
7	Année a	$1900 \leq a \leq 2050$		2010
8			$a < 1900$	1899
9			$a > 2050$	2051

Cas de tests basiques pour une date

- Jeux de tests dérivés

Cas de tests	Jour	Mois	Année	Couvre	Sortie
t_1	20	7	2010	{1,4,7}	20/07/2010
t_2	0	7	2010	{2,4,7}	Invalide
t_3	32	7	2010	{3,4,7}	Invalide
t_4	20	0	2010	{1,5,7}	Invalide
t_5	20	13	2010	{1,6,7}	Invalide
t_6	20	7	1899	{1,4,8}	Invalide
t_7	20	7	2051	{1,4,9}	Invalide

- Présentent plusieurs faiblesses : ne tiennent pas compte de certaines contraintes entre les variables.
- Chaque classe est représentée dans au moins un test.

Raffinement des classes d'équivalence

- Jour
  1. < 1
  2. > 31
  3. {1,...,28,30}
  4. {29}
  5. {31}.

- Année
  1. < 1900
  2. > 2050
  3. {Année bissextile}
  4. {Année non bissextile}

- Mois
  1. < 1
  2. > 12
  3. {Février}
  4. {Mois 30 jours}
  5. {Mois 31 jours}.

Tenir compte des contraintes

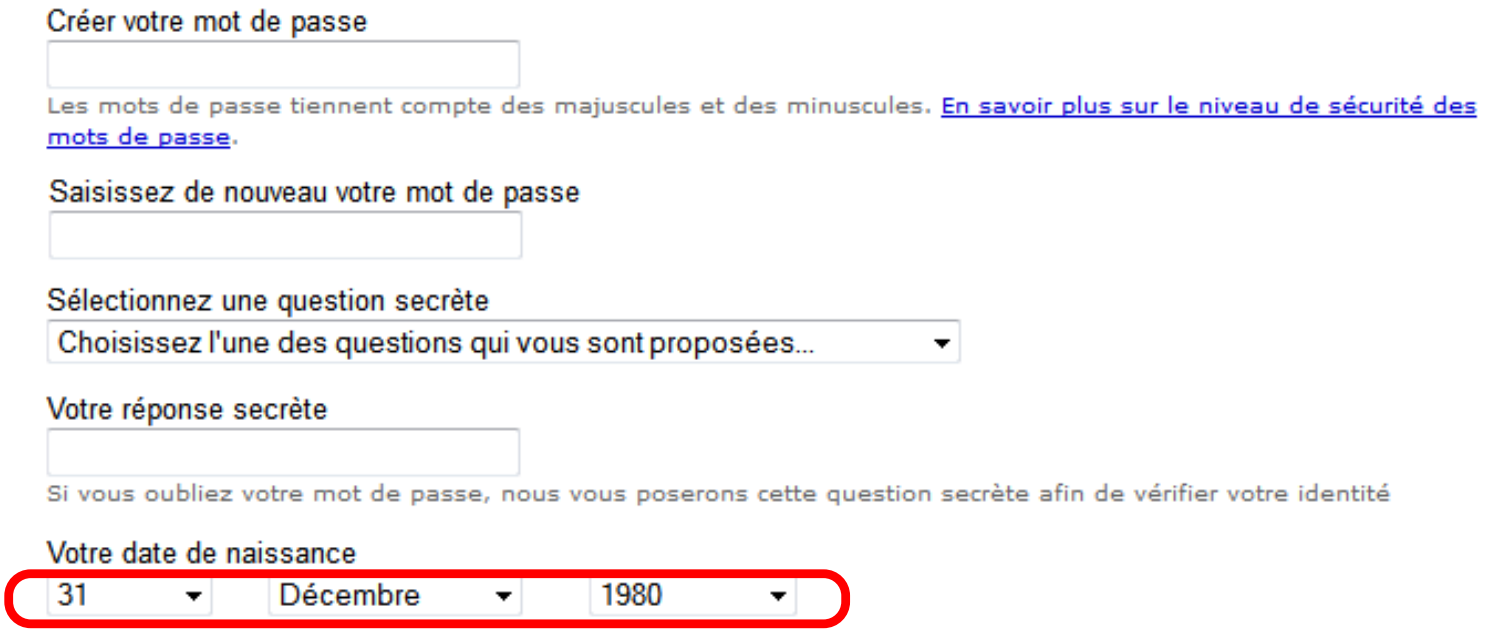
- Partitionnement multidimentionnel : consiste à tenir compte de plusieurs variables à la fois pour déterminer les classes d'équivalence.
  - Les EC ne tiennent pas tjrs compte des contraintes.
  - Nombre de EC svt plus élevé.
  - Exemple :considérer les triplets(j,m,a) pour des dates $\rightarrow$ $5 \times 5 \times 4 = 100 EC$.
  - Quelles classes préconisez vous de retirer ?
- Les conditions particulières aboutissent à de nouvelles EC. Tenir compte des contraintes pour les combiner.

Relations entre variables

- Deux variables peuvent
  - dépendre l'une de l'autre
    - x modifié $\rightarrow$ y modifié ;
    - x limite l'ensemble des valeurs de y.
  - avoir un effet conjointement(impact différent que si prises séparément)
- Si elles n'ont aucun lien mais qu'on pense que par une mauvaise conception du programme, il y a en un alors il faut tester leurs combinaisons.
- Si elle sont dépendantes ou liées directement ou indirectement, alors tenir compte de leur relation pour déterminer les EC.

Entrée non faisable

- Des cas de tests peuvent ne pas être réalisable.
- Exemple : présence d'un GUI, limitant certaines valeurs d'entrée.



Tenir compte de la sortie

- La technique doit être également appliquée pour les sorties.
- Exemple : table de classes d'équivalence pour $z = x \times y$ (Entiers non signés).

#	Var.	vEC	iEC	Val. lim.	x	y
1	z	$0 \leq z \leq MAXINT$		0	0	0
					0	$MAXINT$
					$MAXINT$	0
				$MAXINT$	1	$MAXINT$
					$MAXINT$	1
2			< 0(pas faisable)			
3			$> MAXINT$	$MAXINT + 1$	$2^{N/2}$	$2^{N/2}$
				$MAXINT \times MAXINT$	$MAXINT$	$MAXINT$

Boundary value analysis

- En quelques mots
  - Les valeurs aux frontières des classes d'équivalence sont analysées et servent pour construire les cas de tests.
  - Technique en boîte noire complémentaire du partitionnement en classes d'équivalence.
- Hypothèses
  - Les erreurs sont plus fréquentes près des valeurs extrêmes d'une variable.
  - Les variables sont supposées être indépendantes.
- Limitations
  - Ne marche pas pour n'importe quel type de variables :i.e. variables booléennes.

Frontières et valeurs adjacentes

- A chaque valeur frontière, on considère un incrément à l'intérieur et à l'extérieur de la EC. L'incrément peut être fixé
  - dans une spécification ;
  - par le type de la variable.
- La frontière peut appartenir à une EC à part.
- Lorsque l'on cherche les valeurs frontières, on cherche les changements de comportement.
- Les frontières sont aussi matérielles $\rightarrow$ tests de robustesse !

Les variables entières

- Les entiers
  - Les valeurs sont comprises entre `MININT` et `MAXINT`.
  - Si l'entier est non signé, `MININT=0`.
  - Si l'entier est signé, `MININT=-MAXINT-1`(svt).
  - `MAXINT+1` passe à `MININT`. Le warning d'overflow peut être transformé en erreur.
  - La division peut donner un résultat tronqué.
- Exemples
  - limits.h(C ANSI)

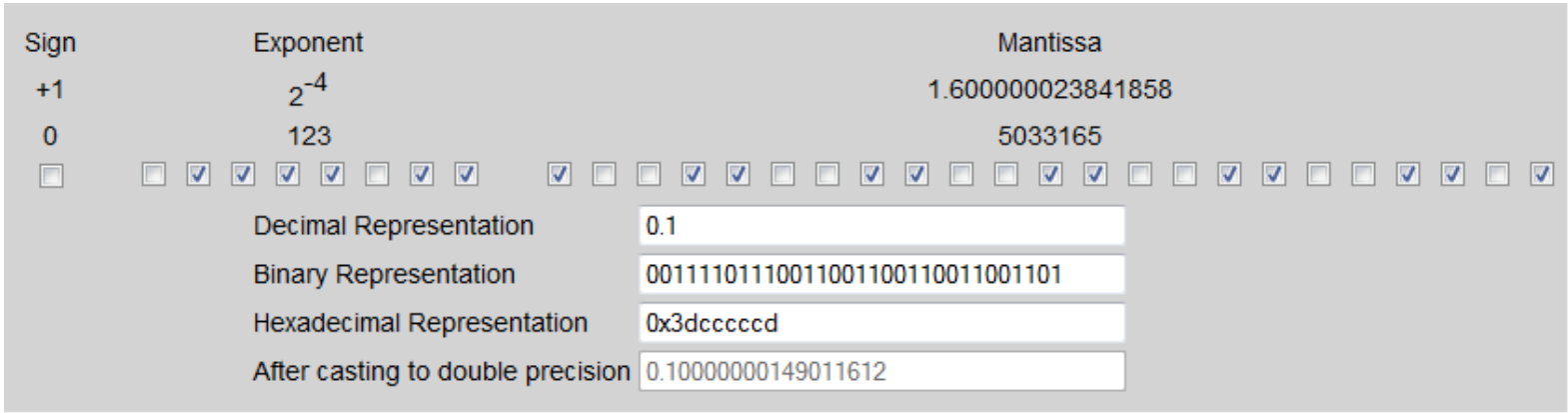
```
1 #define INT_MIN      (-2147483647 - 1)
2 #define INT_MAX        2147483647
3 #define UINT_MAX       0xffffffff
```

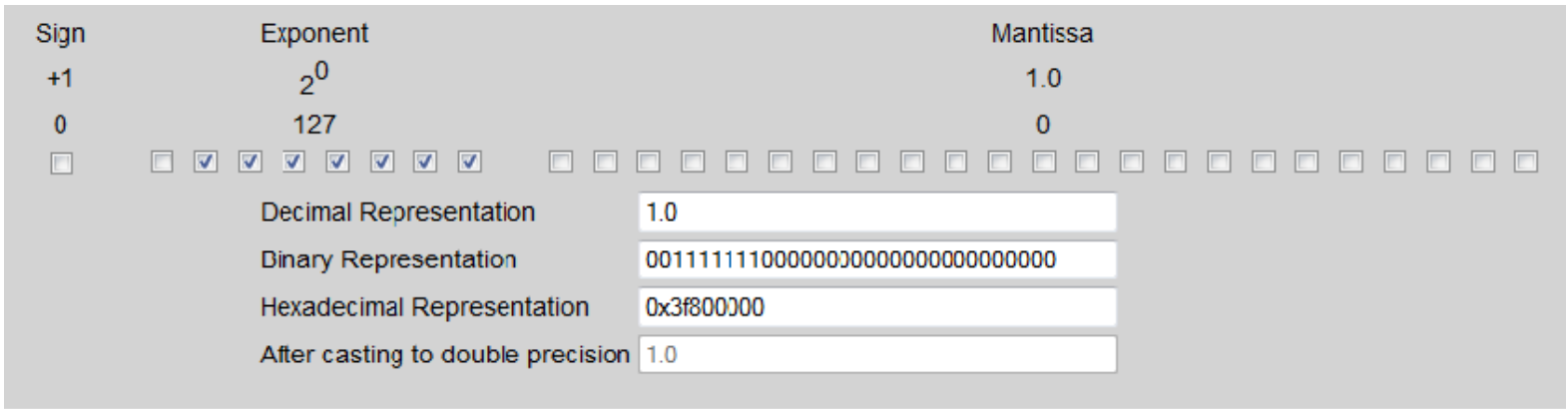
  - En Java, il existe BigInteger qui permet de représenter des entiers bien plus grands (pas de limites en théorie mais bien toujours en pratique à cause de l'implémentation).

Les variables floats

- Les floats $nombre = s \times m \times b^e$
  - s : le bit de signe
  - m : la mantisse
  - e : l'exposant
- Différentes précisions
  - Simple précision : 23 bits et $-126 < e < 127$
  - Double précision : 52 bits et $-1022 < e < 1023$
  - Quadruple précision : 112 bits et $-16382 < e < 16383$
- Math.nextUp $\rightarrow$ retourne la valeur float la plus proche dans la direction de $+\infty$

Exemples en virgule flottante





BVA et floats

- Problèmes avec les floats
  - Les décimales en virgule flottante n'ont en général pas une représentation binaire exacte.
  - Le passage d'un type à un autre, ainsi que l'utilisation des plusieurs types dans une même expression.

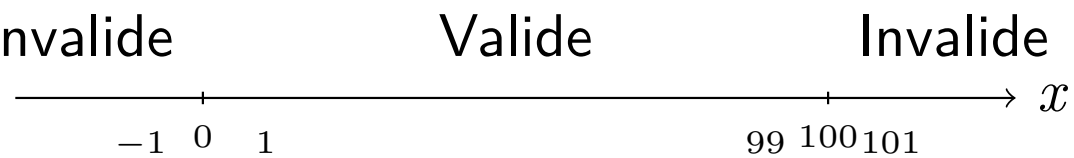
a) Définir la précision souhaitée : nombre de décimales. On parle aussi de ϵ, la plus petite différence reconnaissable.

b) Chercher le changement de comportement
   - Exemple : ≤ 0 :valide, $0 <$: invalide.Changement à 0.01.

c) Ne pas tester des égalités avec de floats mais se servir d'epsilon.

Choisir les valeurs pour élaborer les cas de tests

- Choisir des valeurs au mileu des EC est redondant.
- Il y a au moins deux valeurs frontières dans chaque EC.
- On doit décider si on teste avec deux ou trois valeurs.
  - $1 \leq x \leq 99$: 1 $\rightarrow$ 0,1,2 et 99 $\rightarrow$ 98,99,100.
  - $0 < x < 100$: 0 $\rightarrow$ -1,0,1 et 100 $\rightarrow$ 99,100,101.



Exemple

- Test d'une application relative à l'octroi de crédits hypothécaires.
  1. €20 000 $\leq mc \leq$ €500 000
  2. €50 000 $\leq vm \leq$ €1 000 000
  3. si $mc >$ €300 000 alors la décision d'octroi est reportée sur un manager de plus haut niveau.
  4. L'application arrondi les euros à la centaine d'euros la plus proche.

Exemple : partitions pour le montant du crédit

#	Paramètre	vEC	iEC	B.V.
1	mc		Non chiffres	-
2			null	-
3			Avec décimales	-
4			<0	$-MAXINT$
				-1
5			$=0$	0
				49
6			trop peu	50
				19 949
7		min val		19 950
		max sans accord		300 049
8		min avec accord		300 050
		max avec accord		500 049
9			trop élevé	500 050
				$MAXINT$

Exemple : partitions pour la valeur de la maison

#	Paramètre	vEC	iEC	B.V.
1	vm		Non chiffres	-
2			null	-
3			Avec décimales	-
4			<0	$-MAXINT$
				-1
5			$=0$	0
				49
6			trop peu	50
				49 949
7		min val		49 950
		max val		1 000 049
8			trop élevé	1 000 050
				$MAXINT$

Exemple : partitions pour le transfert vers un responsable

#	Donnée	vEC	iEC	B.V.
1	$transfert$	true		-
2		false		-

Cas de tests

Cas de tests	mc	vm	$transfert$	Accepter ?
t_1	19 950	49 950	N	O
t_2	500 049	1 000 049	O	O
t_3	300 049	1 000 049	N	O
t_4	300 050	49 950	O	O
t_5	"ab-c"	49 950	-	N
t_6	null	100 000	-	N
t_7	20 000,01	200 000	-	N
t_8	$-MAXINT$	300 000	-	N
t_9	-1	400 000	-	N
t_{10}	0	500 000	-	N
t_{11}	49	600 000	-	N
t_{12}	50	700 000	-	N
t_{13}	19 949	800 000	-	N
t_{14}	500 050	900 000	-	N
t_{15}	$MAXINT$	1 000 049	-	N

Cas de tests - suite

Cas de tests	mc	vm	$transfert$	Accepter ?
t_{16}	19 950	"ab-c"	-	N
t_{17}	20 000	200 000,01	-	N
t_{18}	20 000	null	-	N
t_{19}	50 000	$-MAXINT$	-	N
t_{20}	100 000	-1	-	N
t_{21}	200 000	0	-	N
t_{22}	300 049	49	-	N
t_{23}	300 050	50	-	N
t_{24}	400 000	49949	-	N
t_{25}	500 000	1 000 050	-	N
t_{26}	500 049	$MAXINT$	-	N

Remarques sur l'exemple

- Les autres dimensions pour une variable sont considérées(succinctement).
- Il existe des dépendances entre les variables(le transfert vers un responsable selon le mc). Certaines combinaisons n'ont dès lors pas de sens. D'autres techniques sont alors plus adaptées pour traiter ce genre de situation, comme les tables de décision et les graphes causes-effets.
- Les règles de combinaisons des vEC et iEC sont respectées.
- L'exemple ne traite que les entrées(transfert n'est pas une entrée directe).
- La couverture des cas de tests en termes de valeurs frontières est complète.

Graphes causes-effets

- En quelques mots
  - Modélisation des relations et des dépendances entre des conditions d'entrée et de sortie.
  - Représentation visuelle des dépendances dans un graphe causes-effets puis dans une table de décisions comme base pour établir les cas de tests.
  - Technique de test en boîte noire.
- Hypothèses
  - Il existe des dépendances entre les différentes entrées et les effets que ces entrées engendrent.
- Limitations
  - Si le nombre de conditions à représenter est élevé, le graphe et la table de décisions perdent de leur lisibilité.
  - Compresser la table de décisions peut être source d'erreurs.

Les principales étapes à réaliser

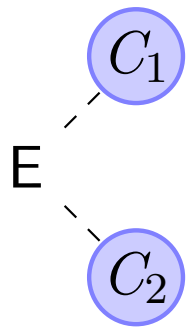
1. Identifier les causes et les effets sur base des exigences.
2. Exprimer les relation entre les causes et les effets à l'aide d'un graphe causes-effets
3. Construire une table de décisions à l'aide du graphe causes-effets.
4. Générer les cas de tests sur base de la table de décisions.

Les causes et les effets

- Tout élément, condition dans les exigences suceptible de provoquer un effet observable ou non suite à l'exécution du programme.
- Les contraintes sur les causes
  - exclusives : une cause vraie au maximum à un instant donné.
  - inclusives : au moins une cause est présente.
  - requises : une cause vraie nécessite qu'une autre soit vraie également.
  - uniques : exactement une cause parmi plusieurs doit être vraie.
- Un effet est la réponse suite à l'exécution du programme soumis à une combinaison d'entrées se rapportant à des conditions.

Causes exclusives

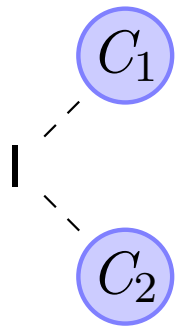
- Une seule cause ne peut être vraie à un moment donné.



Contrainte	Valeurs	
$E(C_1, C_2)$	C_1	C_2
	0	0
	1	0
	0	1

Causes inclusives

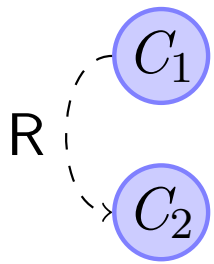
- Au moins une condition est vraie.



Contrainte	Valeurs	
$I(C_1, C_2)$	C_1	C_2
	1	0
	0	1
	1	1

Cause requise

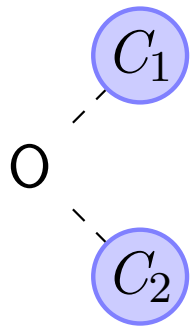
- Une cause vraie implique qu'une autre le soit également.



Contrainte	Valeurs	
$R(C_1, C_2)$	C_1	C_2
	1	1
	0	0
	0	1

Causes uniques

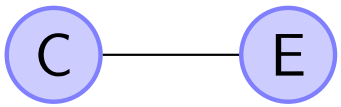
- A un instant donné, exactement une cause parmi plusieurs est toujours vérifiée.



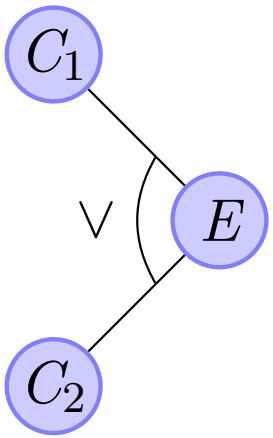
Contrainte	Valeurs	
$O(C_1, C_2)$	C_1	C_2
	1	0
	0	1

Notations des graphes causes-effets

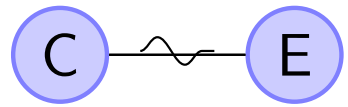
- $C \Rightarrow E$



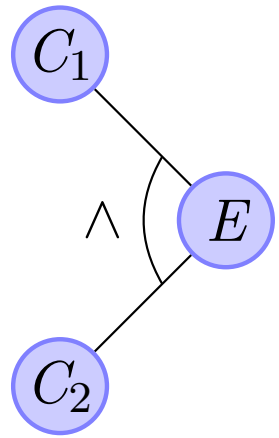
- $C_1 \vee C_2 \Rightarrow E$



- $\neg C \Rightarrow E$

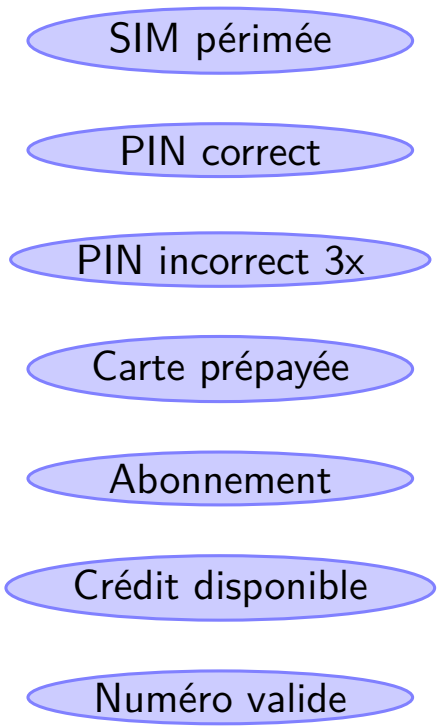


- $C_1 \wedge C_2 \Rightarrow E$

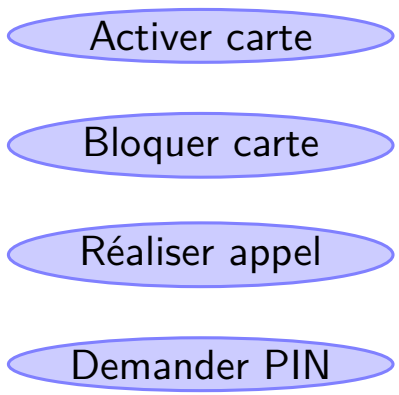


Exemple : lister les causes et les effets

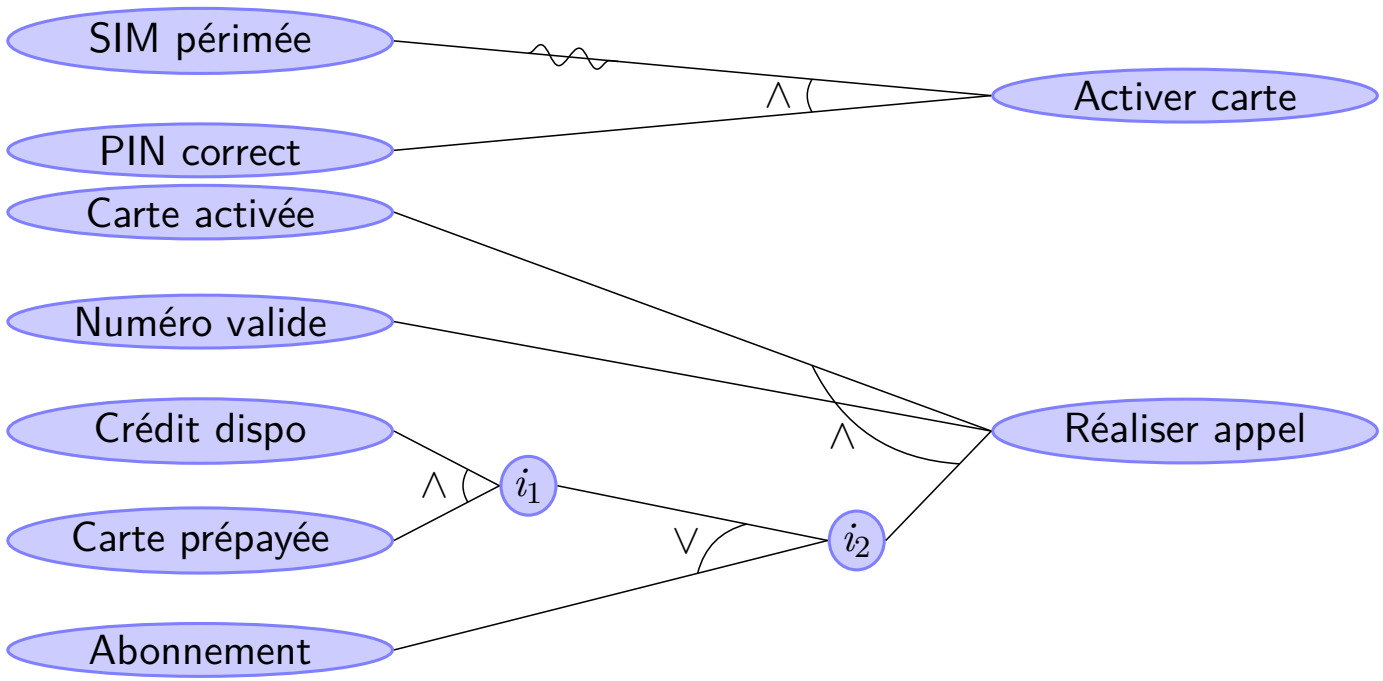
- Causes



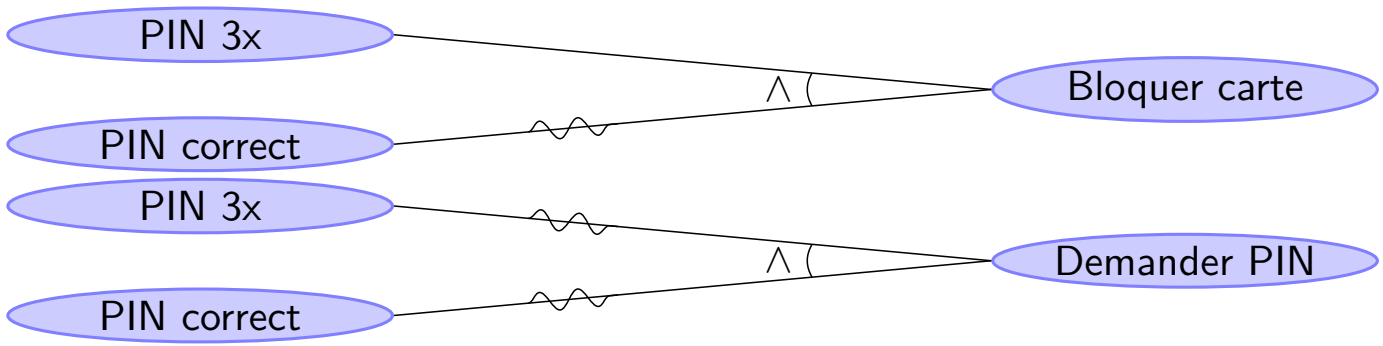
- Effets



Analyser les contraintes pour chaque effet



Analyser les contraintes pour chaque effet II



Contraintes entre les causes

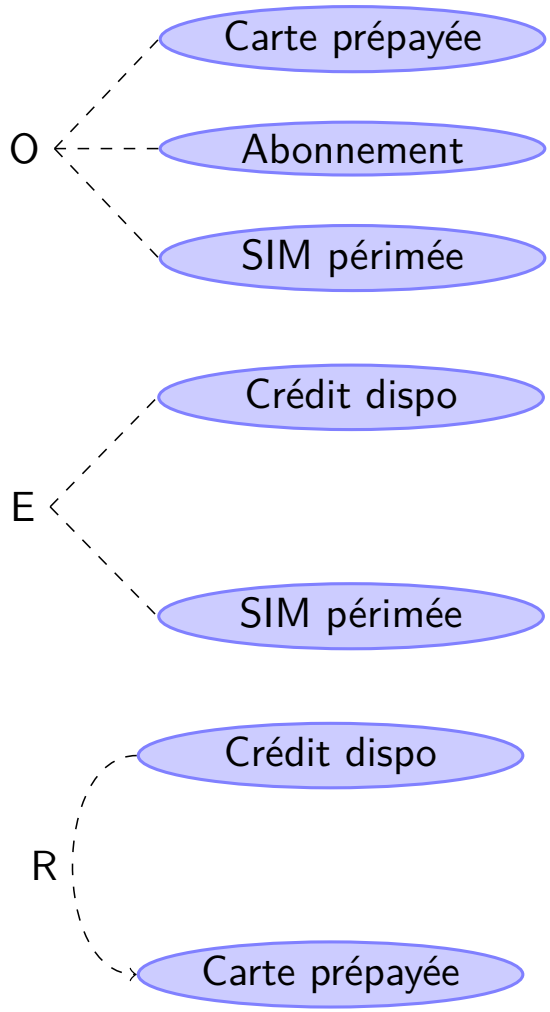


Table de décisions

- Représentation élégante de contraintes logiques.
- Les premières lignes correspondent aux causes.
- Les effets(actions) sont représentées par les lignes en dessous des causes.
- Chaque colonne représente une combinaison de causes. On parle également de règles(métiers) .

	Combinaisons			
Causes	1	2	3	4
C_1	O	N	N	O
C_2	O	N	O	N
Actions				
A_1	O	N	O	N
A_2	N	O	O	N

Détection des fautes

Les tables de décisions sont utiles pour détecter

- des actions entreprises sous une certaine combinaison de conditions alors qu'elles ne devraient pas l'être ;
- l'absence de réalisation d'une action alors qu'elle devrait être entreprise.

Remarques sur les tables de décisions

- Représentation de combinaisons faisables.
- Règles non exclusives ou exclusives entre elles.
- La table peut être compressée lorsque certaines causes n'influent pas pour des combinaisons donnant lieu aux mêmes actions.
- Le nombre de colonnes $nbcol = 2^{nb\_de\_conditions}$ si
  - la table est non compressée ;
  - les causes représentent des vérités logiques.

Réduction de deux combinaisons

	Combinaisons		
Causes	1	2	3
C_1	O	N	N
C_2	N	N	O
Actions			
A_1	O	O	N
A_2	N	N	O

	Combinaisons	
Causes	1	2
C_1	-	N
C_2	N	O
Actions		
A_1	O	N
A_2	N	O

Règles non exclusives

	Combinaisons	
Causes	1	2
C_1	O	-
C_2	-	O
Actions		
A_1	O	
A_2		O

Construction d'une table de décisions

1. Identification des conditions et des actions dans les spécifications ainsi que leurs relations logiques.
2. Lister les causes dans une table de décisions.
3. Calculer le nombre de combinaisons possibles.
4. Pour chaque ligne
   - 4.1 Déterminer le facteur de répétition R en divisant le nombre restant de combinaisons par le nombre de valeurs possibles pour la condition.
   - 4.2 Ecrire $R\times$ la première valeur, puis $R\times$ la suivante et ainsi de suite jusqu'au nombre de combinaisons total.
5. Ecrire les effets pour chaque combinaisons et réduire les entrées dans la table.

Table de décisions sur base d'un graphe causes-effets

1. Créer une structure de table vide
2. Pour chaque effet e
   - 2.1 Déterminer dans le graphe et au départ de l'effet la liste des combinaisons des causes qui donnent l'effet e.
   - 2.2 Vérifier que les combinaisons sont possibles en fct des contraintes.
   - 2.3 Ajouter les combinaisons et mettre à jour le(s) effet(s).
   - 2.4 Réduire les combinaisons.

Construire des entrées pour un effet

	Combinaisons	
Causes	1	2
Carte périmée	N	N
Carte Activée	O	O
Carte prépayée	O	N
Abonnement	N	O
Crédit dispo	O	N
Numéro valide	O	O
Actions		
Réaliser Appel	O	O

Elaboration des cas de tests

- Chaque colonne de la table de décisions donne lieu à un cas de test au moins.
- Les valeurs des variables sont choisies
  - de manière à remplir les conditions d'une combinaison donnée ;
  - sur base des techniques vues précédemment(B.V.A par exemple).