

Sujet 2

AGL pour la sûreté de fonctionnement des systèmes



Introduction

- Dans le domaine de l'analyse des systèmes, deux communautés cohabitent :
 - La vérification formelle «VF» : le fonctionnel
 - la sûreté de fonctionnement «SdF» : le dysfonctionnel



Objectif

- les systèmes dont les conséquences de dysfonctionnements ne sont pas négligeables et peuvent être économiques, écologiques, ou humaines (systèmes critiques)
- avoir un certain niveau de confiance en le système.
- Envisager les raisons et les conséquences de ses dysfonctionnements pour avoir un certain niveau de confiance en le système.

Système et composants

- un ensemble d'éléments qui interagissent entre eux.
- une ou plusieurs fonctions qu'il doit accomplir
- Les éléments définissant un système sont :
 - Les fonctions assurées par le système.
 - L'architecture du système, qui comprend les divers composants mis en jeu et leurs connections.
 - Le système d'exploitation, qui représente les comportements du système durant son fonctionnement



La sûreté de fonctionnement « SdF »

- vise la maîtrise des risques
- fournit les données nécessaires pour indiquer le degré de confiance que l'utilisateur peut avoir sur le système
- utilise des concepts et des outils pour faire une analyse des systèmes.

Risque

- Le *risque* possède deux dimensions :
 - l'événement non souhaité
 - vise à diminuer (en général) la fréquence d'occurrence de la défaillance du système
 - la mesure associée à ses conséquences :
 - inutile de concevoir des systèmes trop sécurisés et, par voie de conséquence, trop onéreux par rapport aux conséquences du risque lui-même

Défaillance

- Dans le cadre de la SdF, un système connaît une défaillance lorsqu'il n'est plus en mesure de remplir sa (ou ses) fonction (s)
- une défaillance est la cessation de l'aptitude du système (ou d'un dispositif) à accomplir une fonction requise



Modes de défaillances

- Les défaillances d'un même composant peuvent avoir des conséquences différentes.
- Il est ainsi important de distinguer, pour un composant, les défaillances (modes de défaillances) qui ont des conséquences différentes et qui, donc, entraînent des comportements différents du système



Paramètres et lois de sûreté de fonctionnement

Paramètres

La fiabilité

La disponibilité

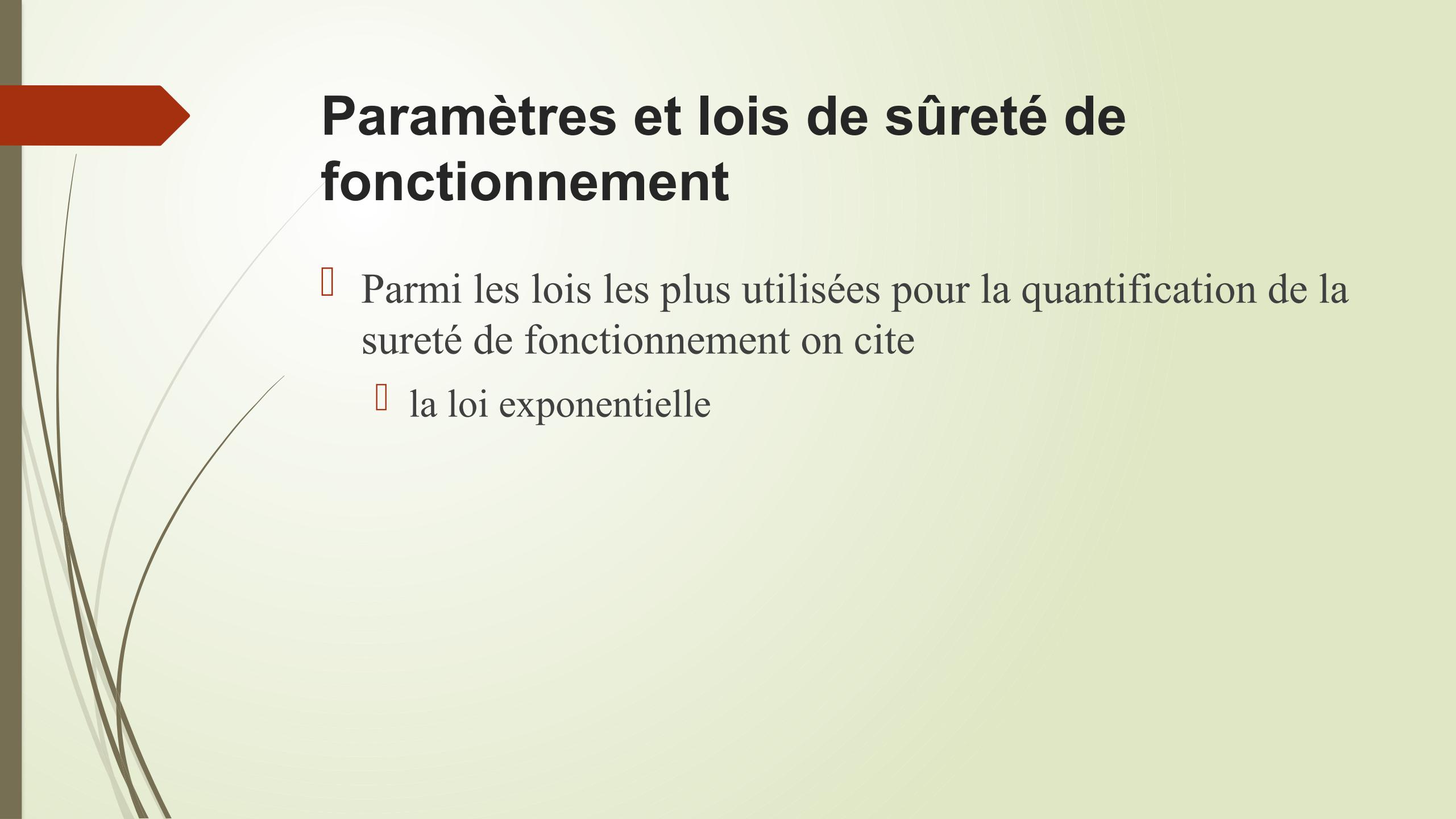
La maintenabilité

La sécurité

Le taux de défaillance

Le taux de réparation

Le taux de défaillance à la sollicitation



Paramètres et lois de sûreté de fonctionnement

- Parmi les lois les plus utilisées pour la quantification de la sûreté de fonctionnement on cite
 - la loi exponentielle

Analyses préliminaires à la sûreté de fonctionnement

- Analyse fonctionnelle :
- Pour effectuer cette analyse il est nécessaire d'identifier :
 - les caractéristiques des systèmes
 - les fonctions,
 - la structure,
 - les technologies déployées,
 - les modes de fonctionnement,
 - les conditions d'exploitation, ,,,,

Analyse des risques

- Il s'agit à la base d'une analyse qui permet **d'identifier des points critiques** devant faire l'objet d'études plus détaillées.
- Les méthodes les plus utilisées dans cette analyse sont :
 - l'Analyse Préliminaire des Risques (APR)
 - l'Analyse des Modes de Défaillances et de leurs Effets (AMDE).



APR

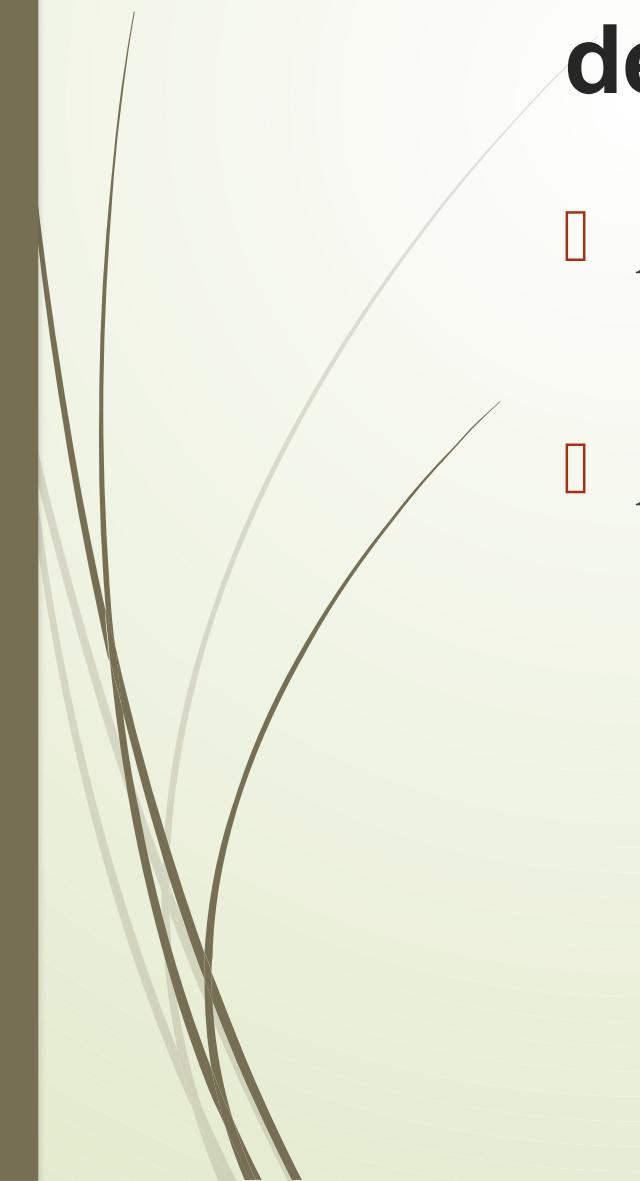
- La méthode d'Analyse Préliminaire des Risques « APR » a deux objectifs :
 - Identifier les dangers d'une installation industrielle et ses causes.
 - Evaluer la gravité des conséquences liées aux situations dangereuses et aux accidents potentiels.

AMDE et AMDE ©

- L'analyse des Modes de Défaillances et de leurs Effets AMDE ET l'AMDE(C) (complétée par une analyse de la criticité)
- l'analyse exhaustive de tous les modes possibles de pannes reliés à leurs causes et leurs effets
- La méthode AMDE(C) est une méthode qualitative et inductive visant à recenser les défaillances, puis à en estimer les risques

AMDE

Composant	Mode de défaillance	Origine de la défaillance	Conséquences de la défaillance
1ier composant	1er mode de défaillance		
	...		
....			

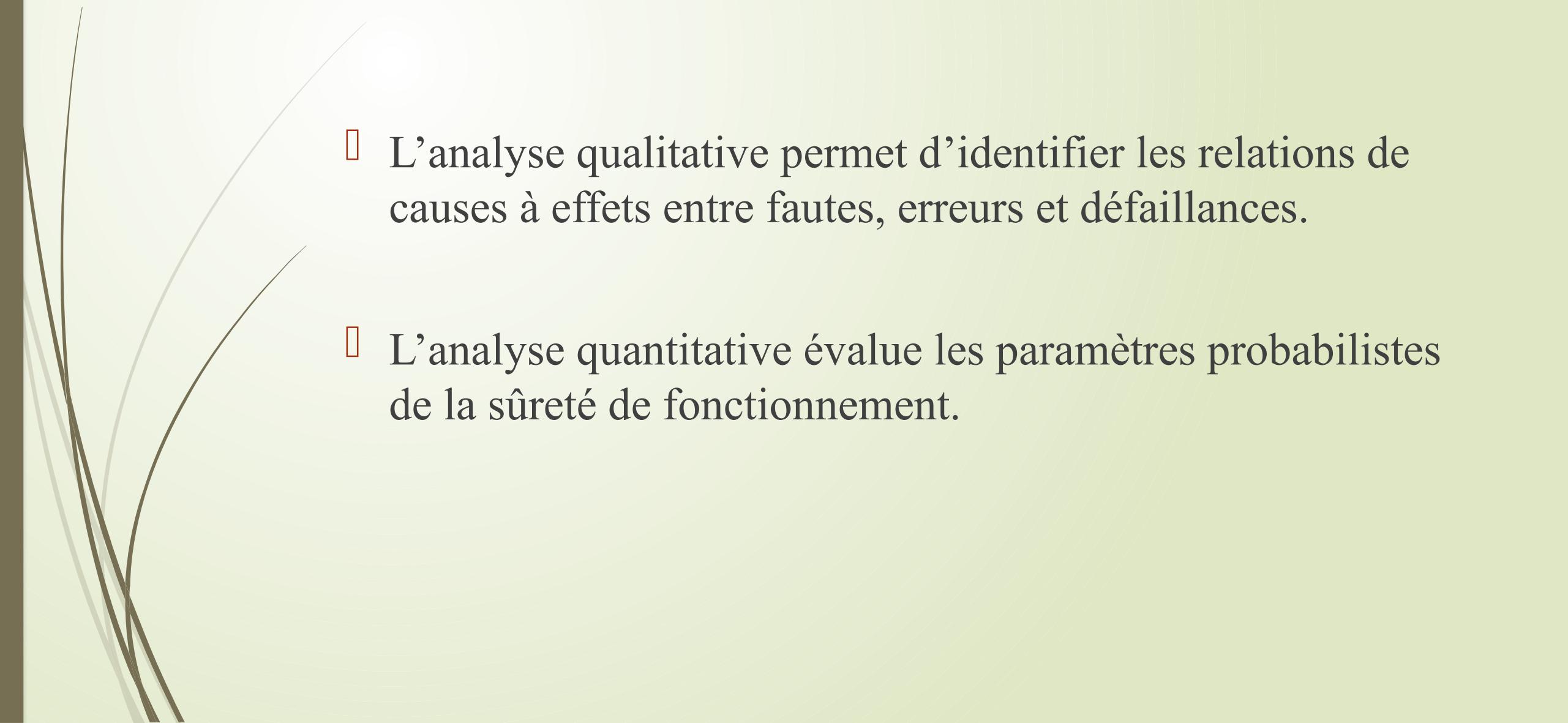


Classification des méthodes de sûreté de fonctionnement

- Analyse qualitative et quantitative
- Analyse statique et dynamique



Analyse qualitative et quantitative

- L'analyse qualitative permet d'identifier les relations de causes à effets entre fautes, erreurs et défaillances.
 - L'analyse quantitative évalue les paramètres probabilistes de la sûreté de fonctionnement.
- 

- Analyse statique et dynamique

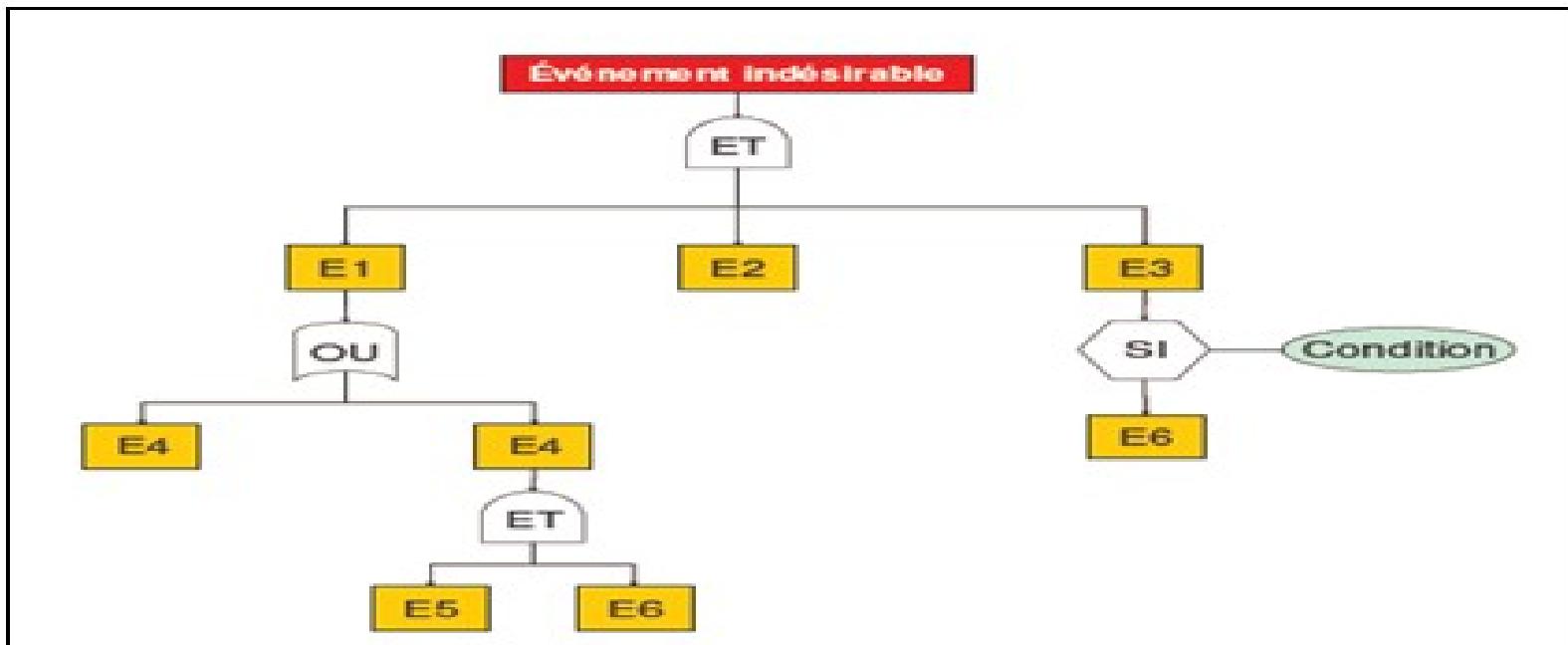
- *A- Analyse statique* : Un composant, un sous-système ou le système lui-même possède uniquement deux états: marche et panne.
- Les modèles statiques décrivent des formules booléennes (modèles booléens)
- l'étude est faite sur un état bien défini du système.
- Les méthodes les plus utilisées dans cette analyse sont : les Arbre de Défaillance (AdD).

Analyse dynamique

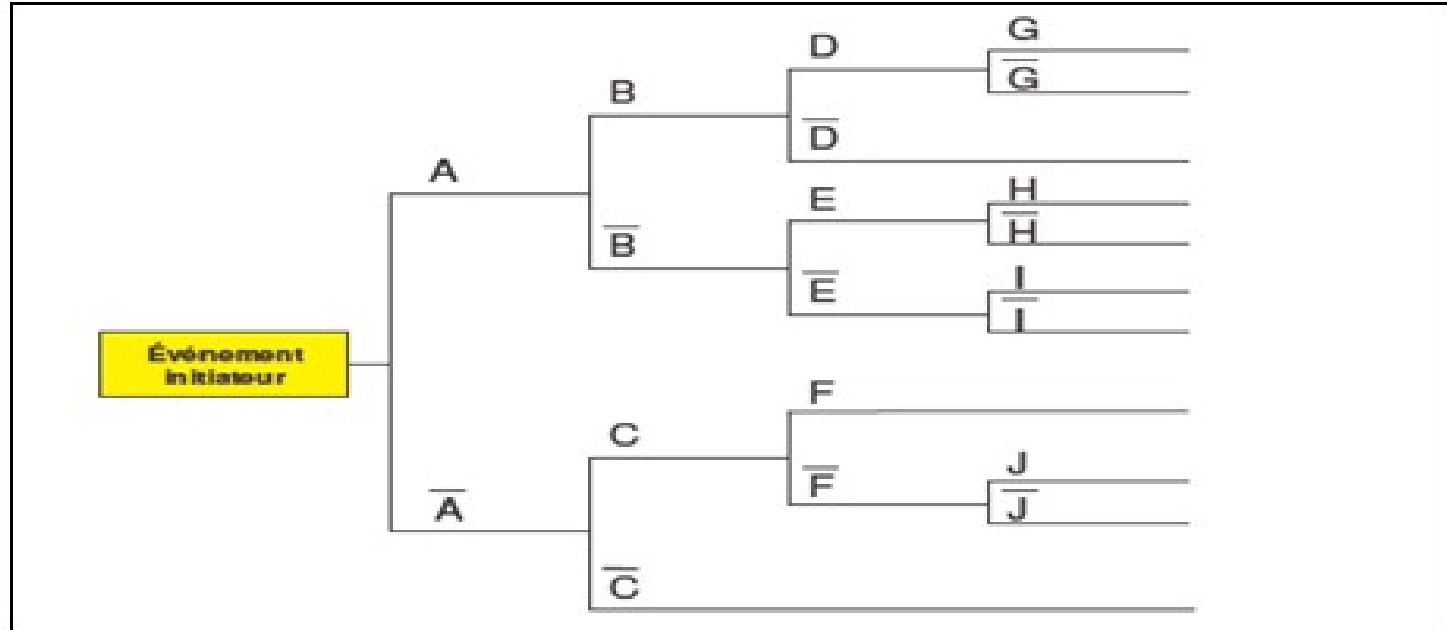
- Analyse dynamique : prend en compte la reconfiguration, les redondances, les modes de fonctionnement, les réparations, la maintenance
- utilise des modèles comportementaux ou dynamiques.
- Parmi les méthodes les plus utilisées dans cette analyse on cite les Graphes de Markov (GM), et les arbres d'événements.

Méthodes pour la sûreté de fonctionnement Exemple

Arbre de défaillances



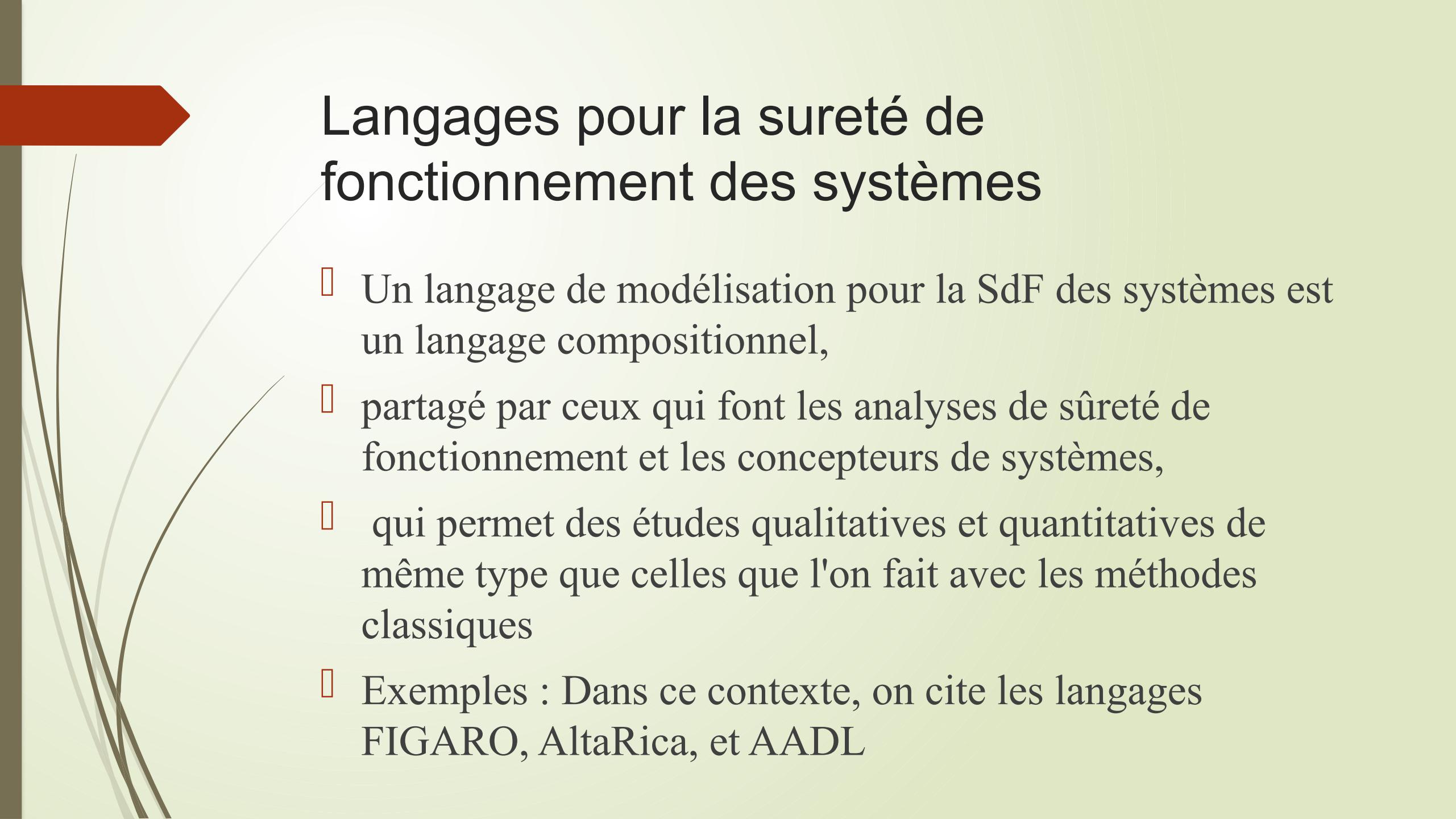
Méthodes pour la sûreté de fonctionnement Exemple





D'autres méthodes

- *Diagramme de succès*
- *Graphe de Markov*
- *Réseau de Petri*
- *Simulation de Monte-Carlo*
- „„

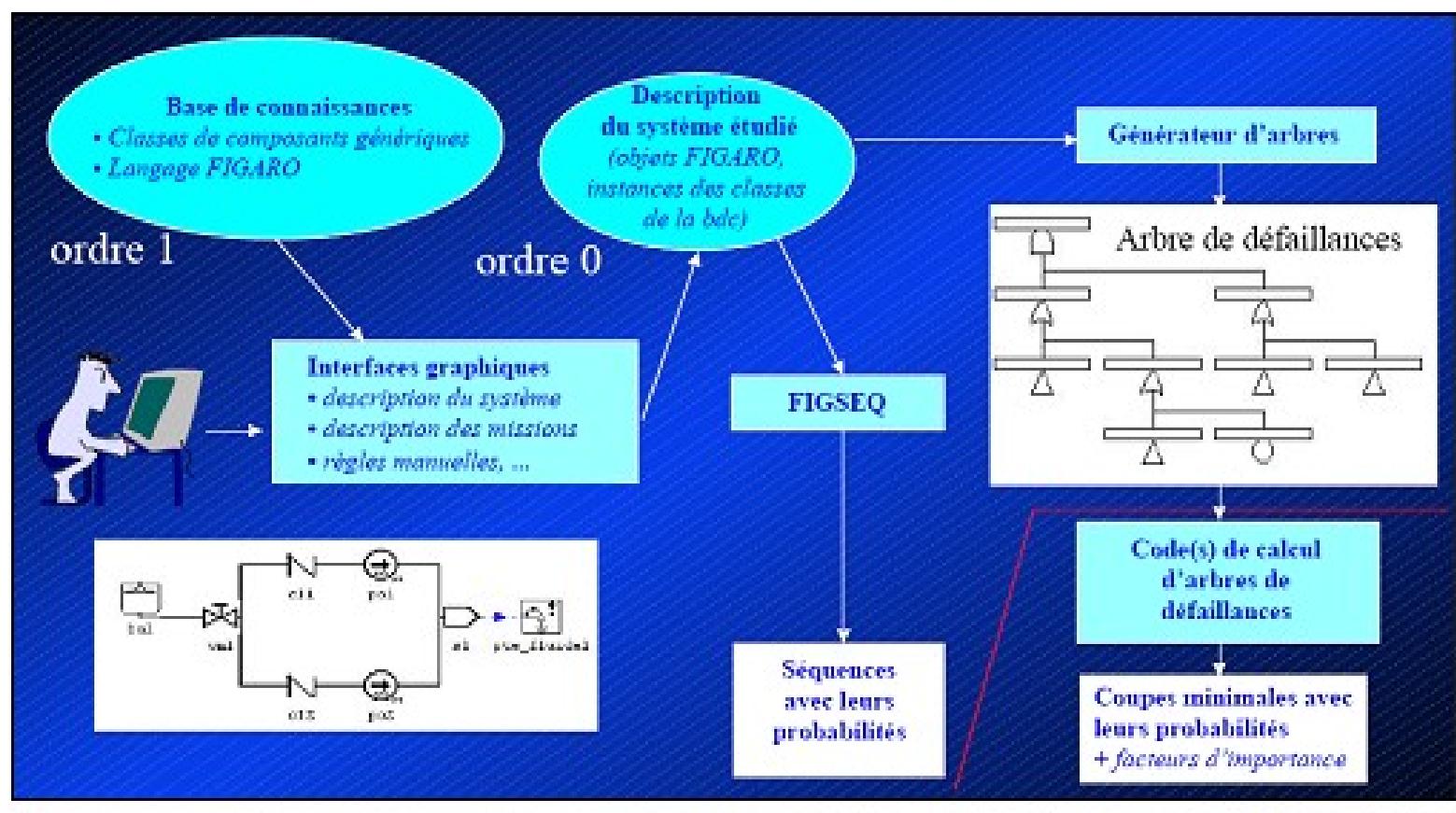


Langages pour la sûreté de fonctionnement des systèmes

- Un langage de modélisation pour la SdF des systèmes est un langage compositionnel,
- partagé par ceux qui font les analyses de sûreté de fonctionnement et les concepteurs de systèmes,
- qui permet des études qualitatives et quantitatives de même type que celles que l'on fait avec les méthodes classiques
- Exemples : Dans ce contexte, on cite les langages FIGARO, AltaRica, et AADL

Langage FIGARO

- Figaro1
- Figaro0
- **VisualFigaro**
- **KB3**



Langage AltaRica

- le LaBRI conçoit l'outil AltaRica Checker « + » (Bernard, 2009), basé à la fois sur les Altatools et sur MecV
- Cecilia OCAS



AADL

- D'éditer des modèles AADL (outils Stood, ADELE, OSATE, TOPCASED,etc).
- De générer le logiciel du système à partir d'un modèle AADL (Stood, Ocarina,etc).
- De conduire diverses analyses (OSATE, Versa/-Furness, ADAPT, Cheddar,etc).



Autres langages

- **SyReIAn**
 - **UML**
 - **EAST-ADL2**
- 

Exemple de critères de comparaisons

	FIGARO	AltaRica	AADL
Principaux objectifs	langage pour la SdF des systèmes		Langage de description d'architecture
Principales caractéristiques	permet d'écrire des modèles probabilistes quantifiables pouvant être simplifiés en modèles qualitatifs	orienté vers la création de modèles qualitatifs pouvant être enrichis par des informations de nature probabiliste	associé à l'annexe de modèle d'erreur ; AADL permet l'analyse qualitative pouvant être enrichie par des informations de nature probabiliste
	la déclaration des types et leurs instantiations dans deux niveaux de langage	la déclaration des types et leurs instantiations dans un seul niveau de langage	la déclaration des types, leurs implémentations et leurs instantiations dans un seul niveau de langage
	possède deux niveaux : l'ordre 1 (déclaration d'objets) et l'ordre 0 (instanciation d'objets)	plusieurs variantes du langage: AltaRica LaBRI, AltaRica Data-Flow	Le langage AADL est associé à différentes annexes (annexe de traduction, annexe comportemental, etc.)

	FIGARO	AltaRica	AADL
Concepts de base	un seul type d'objets	un seul type de composants	3 familles de composants : matériel, logiciel, hybride
	<p>un type d'objet est défini par :</p> <p>ses variables (attribut, constante, effet, panne), les interfaces, les règles d'interactions et les règles d'occurrences</p>	un composant est défini par son état interne, son interface, les assertions et les transitions	un type d'un composant AADL est constitué de trois parties : les éléments d'interface, les flots et les propriétés
	les règles d'occurrences définissent le changement d'état de l'objet	les transitions définissent le changement d'état du composant.	les transitions (définies dans l'implémentation des composants du modèle d'erreur définissent le changement d'état du composant.
	permet héritage, mais pas l'organisation hiérarchique entre les objets	Permet l'assemblage hiérarchique des composants sans conditions, mais pas d'héritage entre les composants	permet héritage, et l'organisation hiérarchique entre les composants.

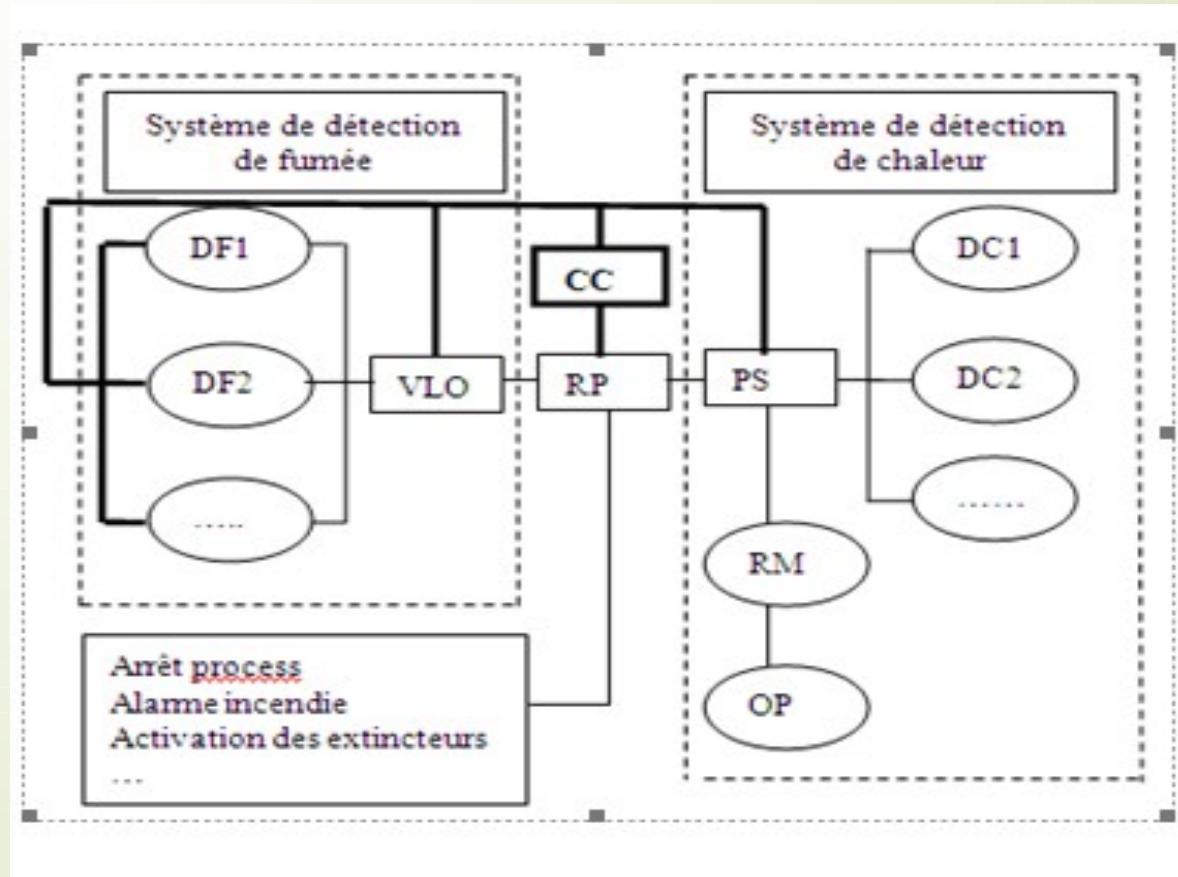
	FIGARO	AltaRica	AADL
Syntaxe et mots-clés	la syntaxe est proche du langage naturel	la syntaxe rappelle celle d'un langage de programmation	
	grand nombre de concepts et mots	nombre réduit de concepts et de mots-clés	grand nombre de concepts et mots clefs,

	FIGARO	AltaRica	AADL
Syntaxe et mots-clés	la syntaxe est proche du langage naturel	la syntaxe rappelle celle d'un langage de programmation	
Sémantique	grand nombre de concepts et mots	nombre réduit de concepts et de mots-clés	grand nombre de concepts et mots clefs,
Robustesse		le comportement qualitatif du modèle équivaut au fonctionnement d'un automate qui génère les états atteignables à partir des états initiaux	permet la détection des erreurs syntaxiques et les incohérences sémantiques, la vérification formelle, et les études de sûreté de fonctionnement du modèle avec différentes analyses et divers outils (à condition de se limiter à la variante AltaRica Data-Flow pour le langage AltaRica)

	FIGARO	AltaRica	AADL
Puissance de modélisation	le système est étudié dans sa globalité	Le système est étudié avec plusieurs niveaux hiérarchiques	
	l'héritage permet de réduire le nombre de déclarations de caractéristiques dans les types, mais on a plus de types	pas d'héritage donc répétition des caractéristiques communes entre les types, mais moins de déclarations de types	l'héritage permet de réduire le nombre de déclarations de caractéristiques dans les types, mais on a plus de types
	les interactions complexes entre les objets sont faciles à modéliser (accès direct)	les interactions complexes entre les composants sont plus difficiles à modéliser il faut passer par la hiérarchie	
Réutilisation composants	les objets réutilisables se trouvent dans une base de connaissance écrite en Figaro1	les composants réutilisables se trouvent dans les bibliothèques de composants	
	la généralisation et l'héritage favorisent la réutilisation des objets Figaro	<u>l'encapsulation favorise la réutilisation des composants</u>	<u>l'encapsulation et l'extension favorisent la réutilisation des composants</u>
Représentation graphique	l'interface graphique permet de représenter le modèle physique du système à partir du modèle du langage		

	FIGARO	AltaRica	AADL
Aptitude à la traduction	la traduction en langage AltaRica suppose le respect de certaines restrictions par le modèle FIGARO de départ, il n'est pas possible par exemple de traduire des modèles Figaros avec des boucles, et des modèles comportementaux probabilistes complexes en modèles AltaRica	Il est possible de traduire automatiquement tout modèle AltaRica en modèle FIGARO	des travaux récents ont permis de transformer les modèles AADL en modèles AltaRica
	pas de traduction en langage de programmation	permet la traduction vers des langages de programmation tels que C ou Ada	
	difficile à traduire vers d'autres modèles de vérification	peut être mis en relation avec des modèles de vérification tels que Esterel ou Lustre.	
Résultats obtenus	<ul style="list-style-type: none"> - avec les outils qui ont été spécifiquement développés pour interpréter les langages AltaRica (la version DataFlow), FIGARO, et AADL il est possible à partir de ces types de modèles, de faire les traitements suivants : - génération d'AdD (sous réserve que le modèle satisfasse certaines conditions d'indépendance des événements associés aux pannes) - simulation interactive, simulation Monte-Carlo - génération de graphe de tous les états atteignables à partir de l'état initial - quantification probabiliste du graphe si toutes ses transitions correspondent à des événements associés à des lois exponentielles ou instantanées - recherches et éventuellement quantification de séquences menant à des états possédant certaines caractéristiques, etc. 		

Exemple : Analyse fonctionnelle



Analyse fonctionnelle

- Le système de détection de chaleur comprend des détecteurs de chaleur (DC1, DC2, etc.) composés de cellules fusibles à 72 °C mises en série et reliés à un circuit sous pression.
- En cas de dépassement de 72 °C les fusibles fondent, la pression dans le circuit baisse et déclenche le pressostat PS qui actionne le relais principal RP.
- Le pressostat PS et le relais RP nécessitent la présence de la source de tension continue (batterie) CC.
- Il suffit qu'un seul détecteur de chaleur se déclenche pour obtenir la perte de pression.
- L'alarme à commande manuelle est sous la responsabilité d'un opérateur OP toujours présent dans l'atelier.
- En cas d'urgence, il peut déclencher le relais à commande manuelle RM qui active le pressostat PS et donc le relais RP.

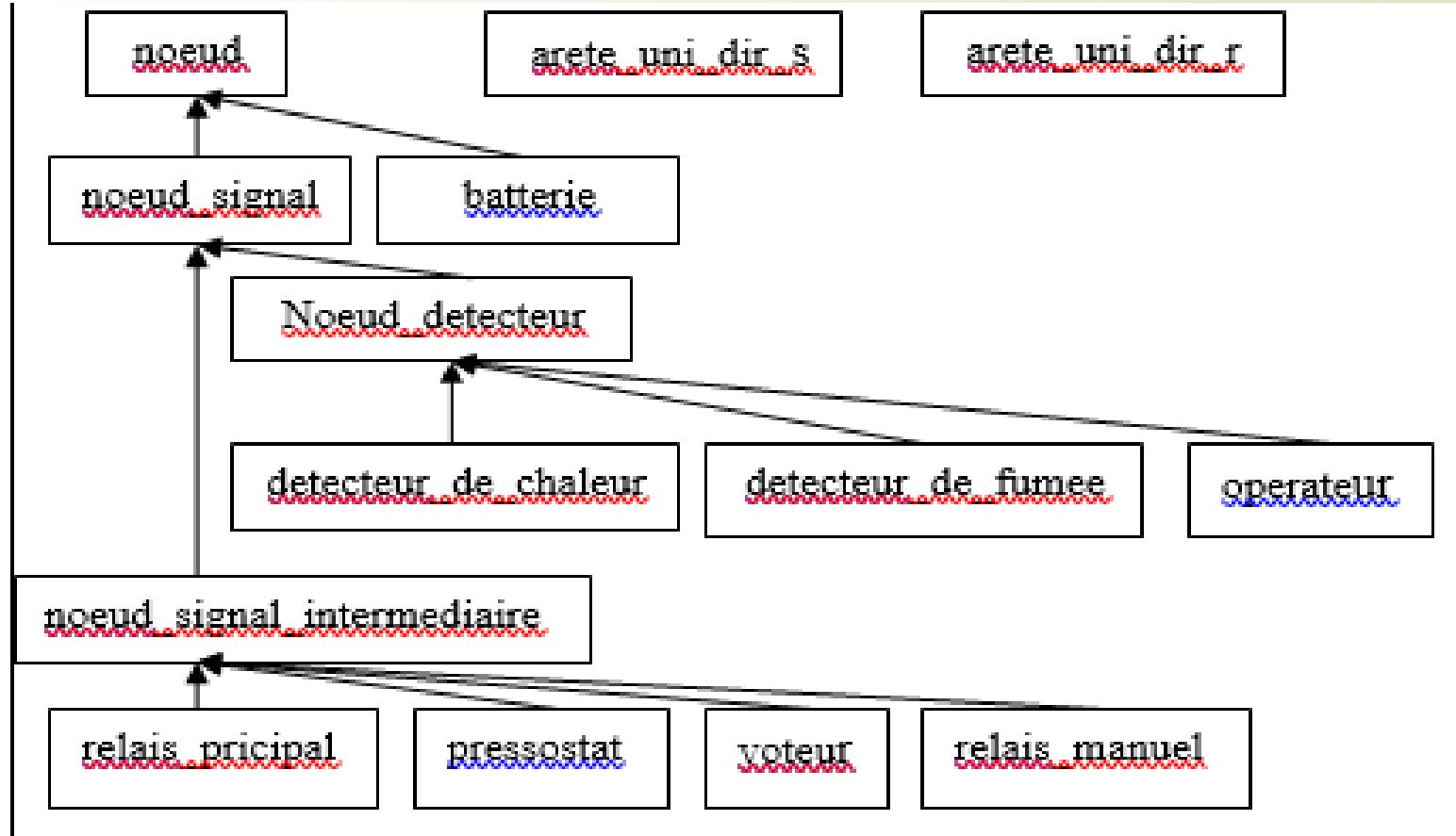
Analyse fonctionnelle

- Le système de détection de fumée comprend des détecteurs de fumée (DF1, DF2, etc.)
- reliés à un système de vote VLO.
- Si au moins un détecteur est activé, le système de vote actionne le relais principal RP (VLO nécessite la présence de la source de tension CC).
- Les détecteurs de fumée nécessitent aussi la présence de la source de tension CC.

Modes de défaillances

Composant	Mode de défaillance	Conséquences de la défaillance sur la mission du composant
DC	PANNE	Pas de signal d'incendie
RM	PANNE	Pas de transmission de signal d'incendie de l'opérateur au PS
OP	PANNE	Pas de signal d'incendie
PS	PANNE	Pas de transmission de signal du système de détection de chaleur au RP
RP	PANNE	Pas de transmission de signal du RP
DF	PANNE	Pas de signal d'incendie
VLO	PANNE	Pas de transmission de signal du système de détection de fumée au RP
CC	PANNE	Pas d'alimentation en courant continu pour : RP, et les systèmes de détection de chaleur et d'incendie.

Composants FIGARO



« main » est composé de :

- *Un système de détection de chaleur « sys_chal_1 »* : ce nœud contient deux détecteurs de chaleur « dc_1 », « dc_2 », un pressostat « ps_1 », un opérateur « opr_1 », et un relais manuel « rm_1 ».
- *Un système de détection de fumée « sys_fum_1 »* : ce nœud contient deux détecteurs de fumée « df_1 », « df_2 », et un voteur « vote_1 ».
- *Un relais principal « rp_1 »*.
- *Une batterie « btr_1 »*.

Les variables de flux déclarées dans le modèle sont

- *e_signal* : entrée du signal d'incendie.
- *s_signal* : sortie du signal d'incendie.
- *e_relie* : entrée du courant continu.
- *s_relie* : sortie du courant continu.

Les variables d'état déclarés dans le modèle sont :

- *marche* : le composant est en état de marche.
- *incendie* : le composant a détecté un incendie.

Les événements déclarés dans chaque composant sont :

- *DEFAILLANCE* : occurrence d'une défaillance.

Les transitions représentant le passage de l'état « marche » à l'état « panne » déclarés dans

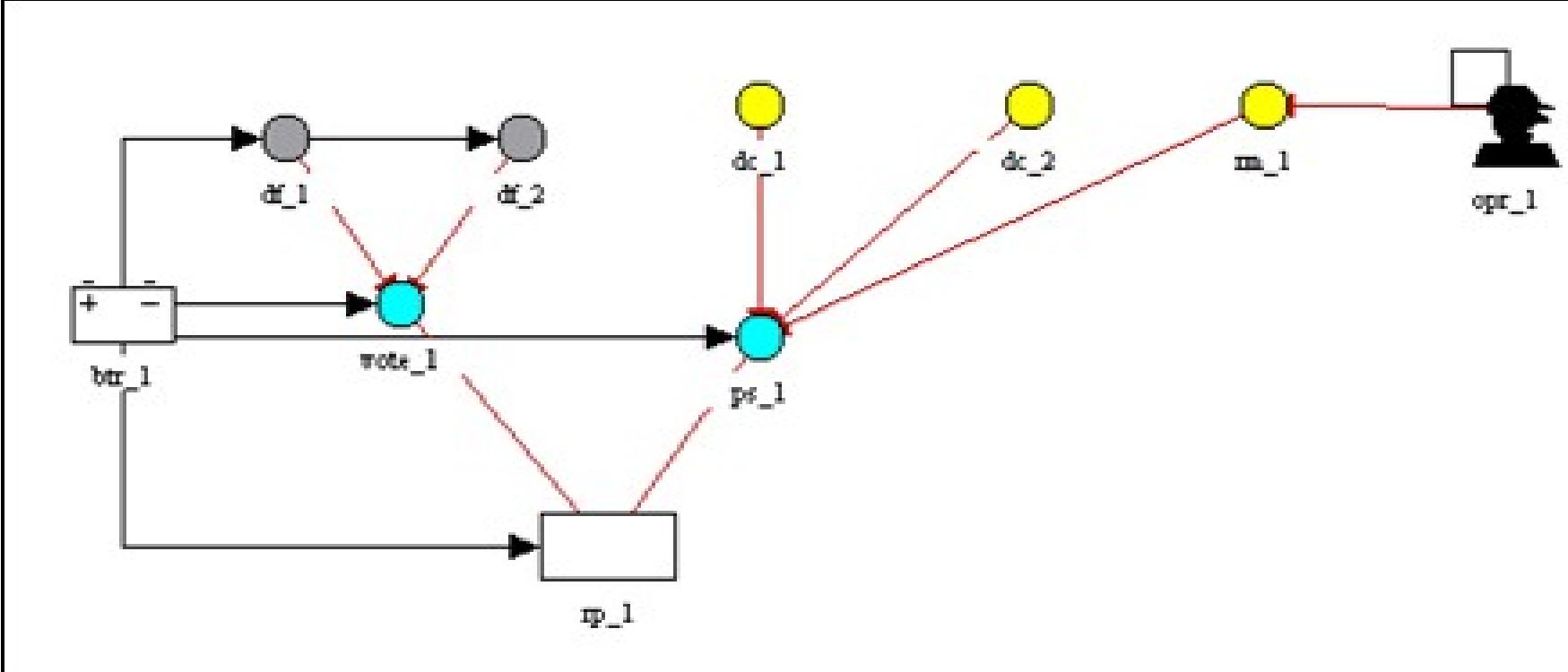
Interfaces

- Pour représenter les connexions entre les composants nous avons défini les interfaces dans les types « arete_uni_dir_s » et « arete_uni_dir_r ».
- Le type « arete_uni_dir_s » permet de transmettre le signal d'incendie,
- le type « arete_uni_dir_r » permet de transmettre le courant continu.

Déclaration des composants

```
Types
NODE : noeud
NODE : noeud_signale
NODE : detecteur_de_fumee
NODE : detecteur_de_chaleur
NODE : relai_manuelle
NODE : relai_principale
NODE : voteur
NODE : pressostat
LINK : arete_uni_dir_s
LINK : arete_uni_dir_r
NODE : batterie
NODE : operateur
NODE : noeud_detecteur
NODE : noeud_signale_intermediaire
FIGARO
```

le schéma physique complet du système de détection d'incendie KB3

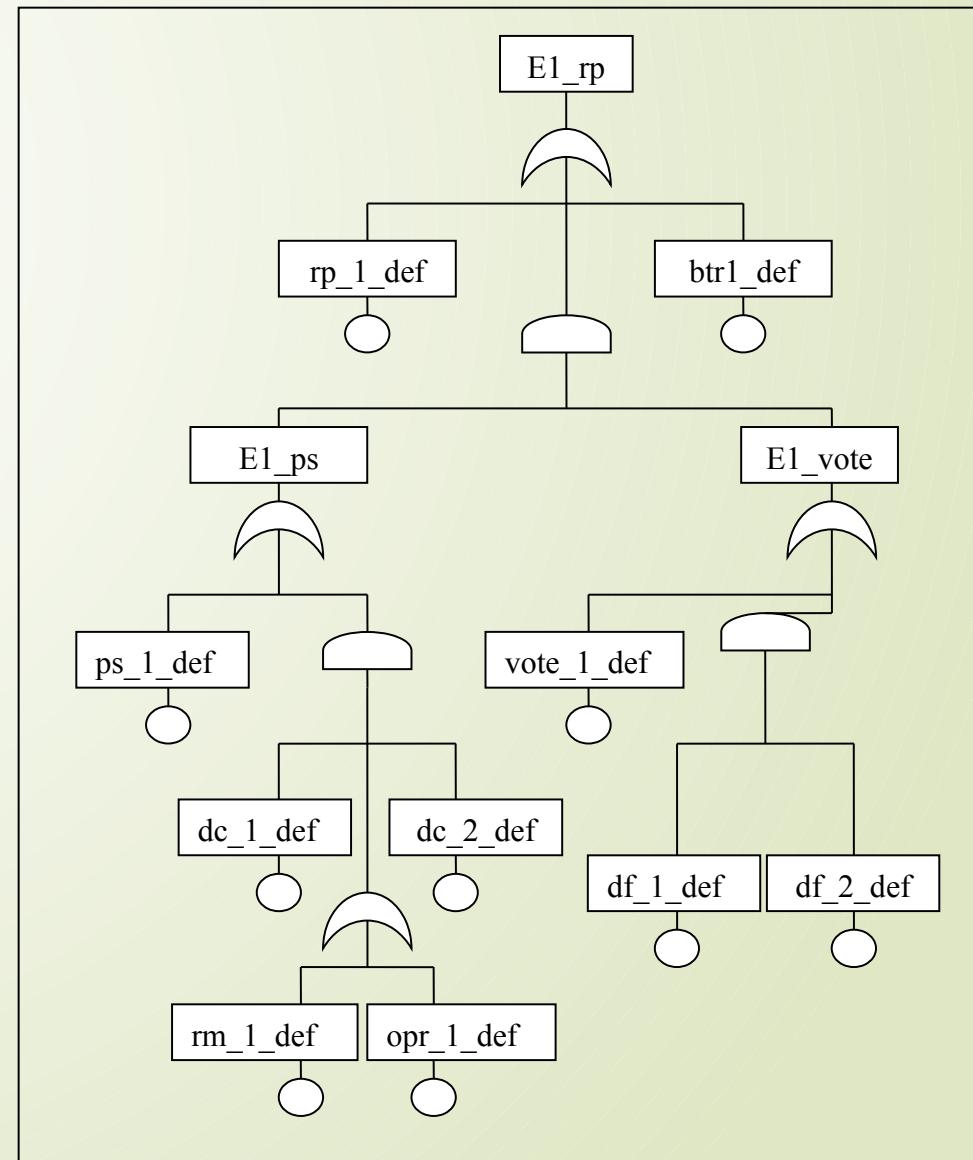




ADD

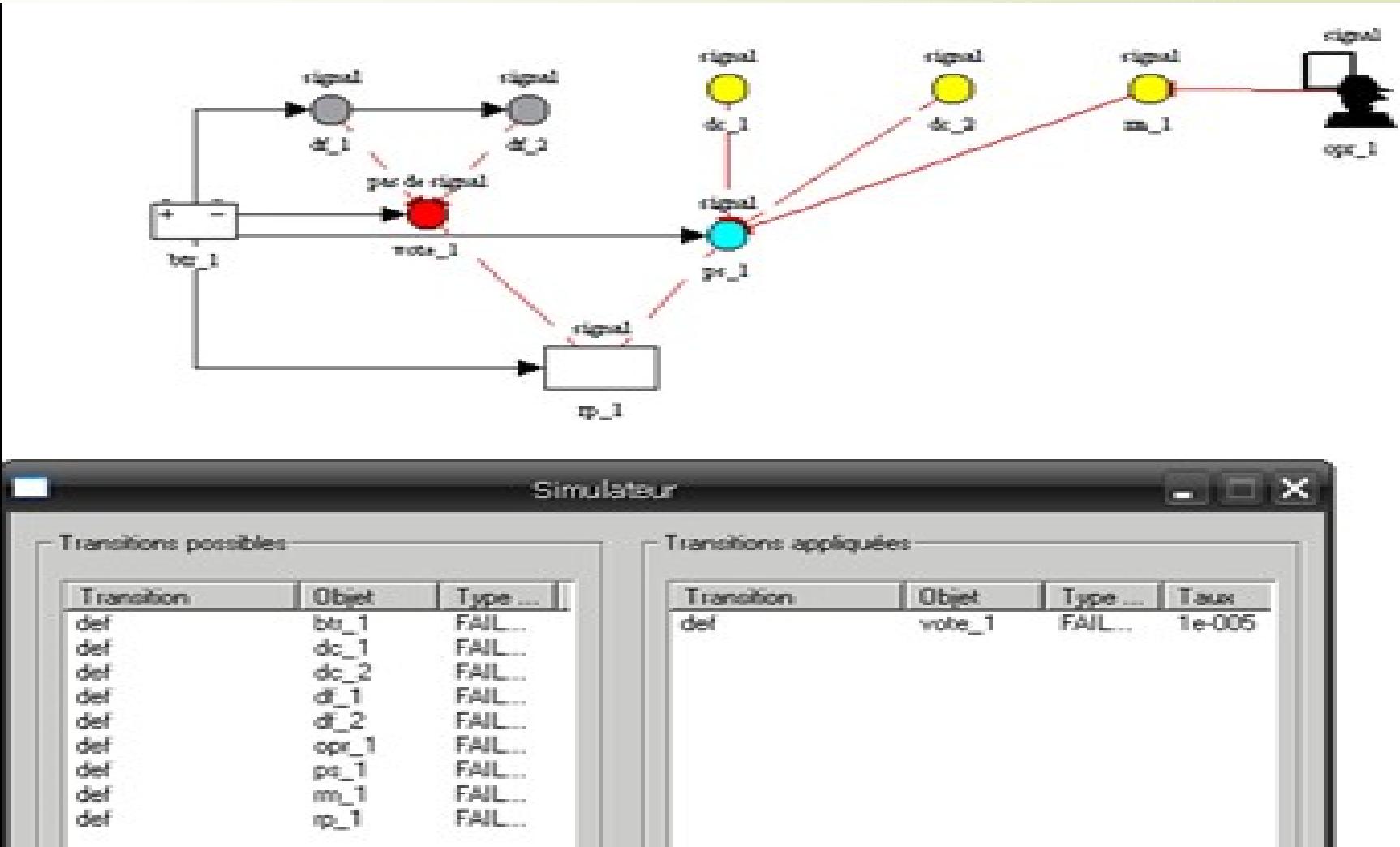
- L'événement redouté dans un système de détection d'incendie est « l'absence de signal en sortie du système ».
- Le composant représentant le point de sortie du signal du système est le relais principal,
- alors l'événement redouté est : « signal (rp_1) = FAUX ». Avec KB3, l'arbre de défaillances

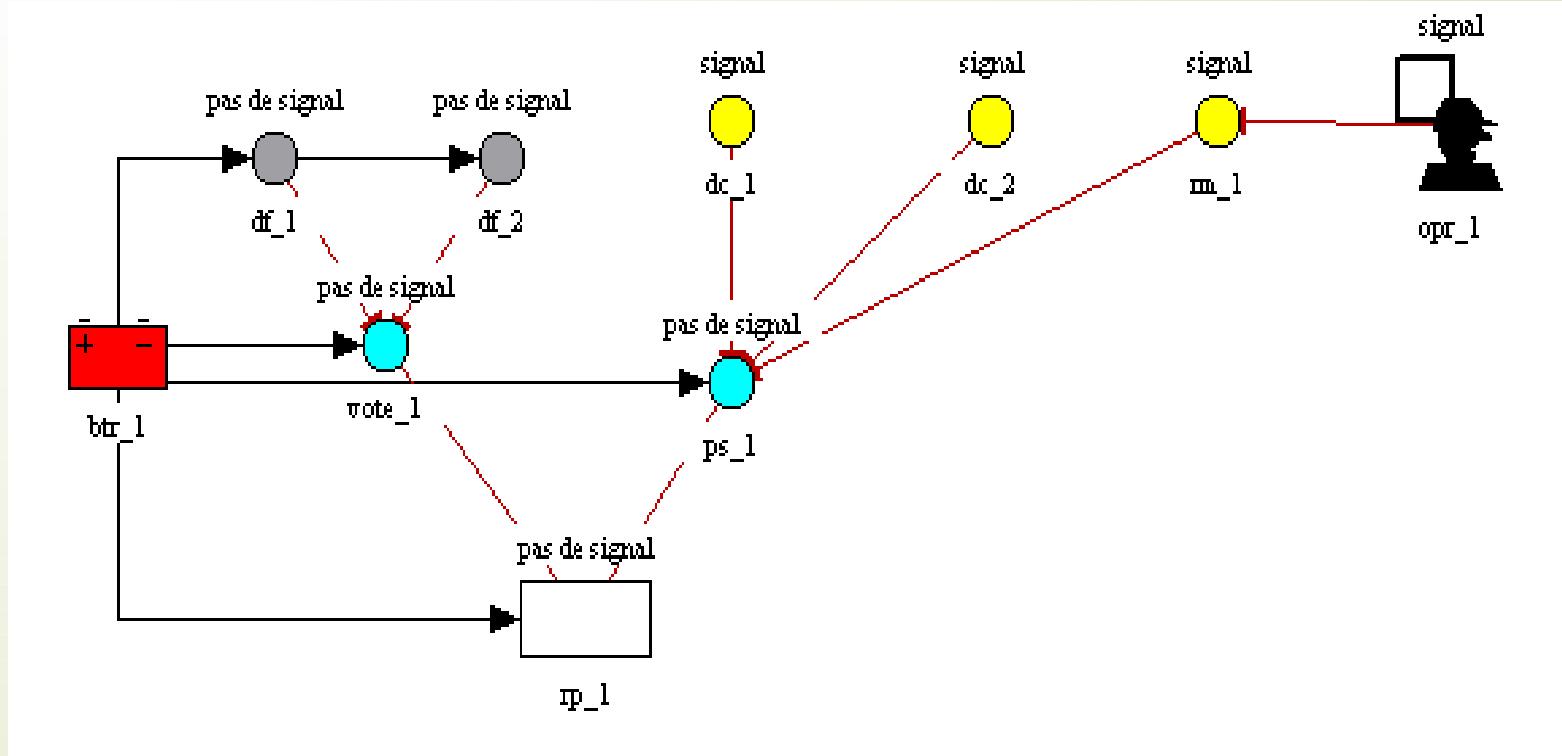
ADD KB3



Simulation KB3

- On peut visualiser l'impact des combinaisons des défaillances sur le schéma du système grâce au simulateur intégré à l'outil KB3.







SDF avec ALtarica

Code altarica

```
-----  
sub  
    sys_chal_1 : system_detect_chaleur;  
    sys_fum_1 : system_detect_fumee;  
    rp_1 : relais_principal;  
    btr_1 : batterie;  
flow  
    s_signal :bool ;  
assert  
    btr_1.s_relie = rp_1.e_relie ;  
    btr_1.s_relie = sys_chal_1.e_relie;  
    btr_1.s_relie = sys_fum_1.e_relie;  
    rp_1.e_signal = (sys_chal_1.s_signal or  
                      sys_fum_1.s_signal);  
    s_signal = rp_1.s_signal;  
edon
```

Simulation interactive avec Altraica

s_signal	false
e_relie	false
▼ vote_1	
marche	true
e_relie	false
e_signal	false
s_signal	false
▼ df_2	
marche	true
incendie	true
e_relie	false
s_signal	false
▼ df_1	
marche	true
incendie	true
e_relie	false
s_signal	false

Travail a remettre

- Chapitre 1 : Présentation de la sûreté de fonctionnement
- Chapitre2 : Présentation des architectures logiciels
- Chapitre3: Présentation des AGL's
- Chapitre 4 Comparaison des AGL's à base de langages de description d'architecture
- Chapitre 3 Etude de cas avec l'AGL choisis
- Remise du fichier en format Word avec l' AGL utilisé et le fichier sources de la modélisation
- **Date de remise : 03/01/2023**