Ingénierie de la sûreté de fonctionnement

Thomas Robert
thomas.robert@telecom-paristech.fr
Bureau 4D44
Mastère spécialisé – INF722
Version 2019

Structure du cours en général

- **3 contenus de cours :**
  - **Concepts élémentaires, risque et objectifs de SDF**
  - **Tolérances aux fautes : données, ou processus (solutions architectures et programmatiques).**
  - **Evaluation de sûreté de fonctionnement**
- **Des TPs pour les deux dernières parties + un TD de réflexion + peut être un séminaire industriel (sureté / sécurtié).**
- **Note finale : 20 % TD rendu en fin de séance, 100 % à l'examen (les deux notés sur 20. Attention le TD sera assez dur 10/20 donnera en gros 2 pts bonus, 20 → 4, arrondi au demi point le plus proche ou inférieur si égalité)**

Pré-requis du cours

- **Logique (élémentaire) pour les cours et TD tolérance aux fautes : rappel rapide fait en cours**

- **Programmation en C + fonctionnement de base de la synchronisation par sémaphore (rappelé en début de TP)**

- **Je prendrai du temps sur les tp analyse quantitative pour faire une mise à niveau si nécessaire en probabilité).**

Quelques mots sur l'ingénierie système motivation

- **Systèmes logiciels**
  - **Systèmes d'information (base de clients, comptes bancaires)**
  - **Systèmes industriels (lignes de production numérique, infrastructure de contrôle pour un processus - e.g. chimique)**
  - **Systèmes embarqués (satellites, trains, avions)**
- **Incident/accident : situation aux conséquences négatives**
  - **Perte d'avantage commercial / perte financière**
  - **Impact environnemental / mise en danger de la vie des personnes**
  - **Destruction du système**

… comment les maitrise-t-on ? Que faut il faire et à quel coût ?

Risque : définition et traitement

Incident, vraisemblance, et risque

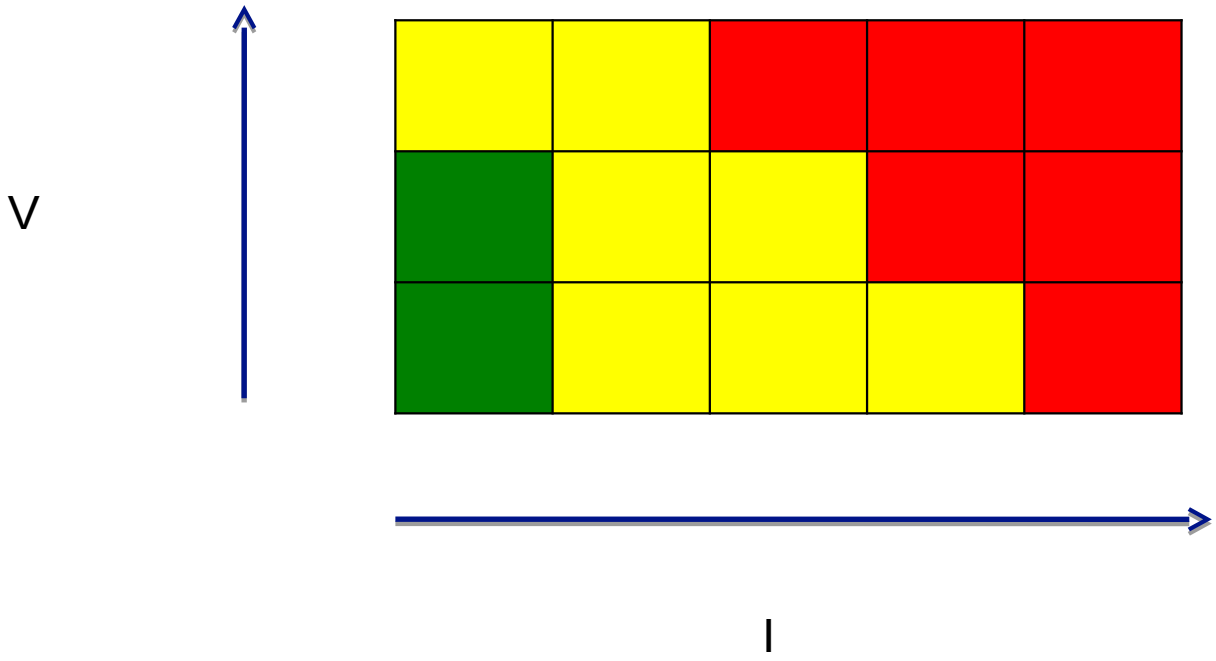
- **Incident = événement redouté portant atteinte aux personnes ou aux biens**
- **Définition générale du risque :**

  Une **estimation** qualitative ou quantitative déterminant **l'importance des différents incidents** par rapport à leur **vraisemblance**

- **Exemple de principe d'estimation qualitative :**
  - Définition de trois classe de risque : faible, moyen, élevé
  - Dérivation du niveau de risque (f,m,e) à partir :
    - Un impact (nature des conséquence de l'incident)
    - Une classe de fréquence : peu fréquent, raisonnablement fréquent, et très fréquent.
- La communauté d'ingénierie friande des formules chocs : R= I*V

Matrice de risque : visualisation de classes d'équivalence (le cas impact/vraissemblance

- **Idée : représenter dans le plan les classes de risques en fonction de l'impact (I) et de la vraisemblance (V)**



Vraisemblance d'un incident ses différentes interprétations

- **Sémantique :**
  - **La vraisemblance = occurrence**

    lien avec une vision fréquentielle / temporelle
  - **Vraisemblance = faisabilité**

    lien avec une vision logique de condition de faisabilité/impossibilité
- **Caractérisation :**
  - **Par système ou population**
  - **Globale ou modulaire**

Vraisemblance d'un incident – exemple

- **Faisabilité VS Occurrence**
  - téléphone portable : l'incident d'explosion de la batterie = faible
  - avion : perte du pilote automatique = perte de 3 calculateurs
- **Caractérisation globale vs multi critères vs modulaire**
  - **La vraisemblance correspond à un probabilité d'occurrence de 12 % dans les 12 premières heures de fonctionnement**
  - **Un robot industriel sur une chaîne de montage, l'incident = blesser un opérateur humain. Critères : présence de l'humain + dysfonctionnement du détecteur de présence ?**
  - **La perte de la fonction autopilot ne peut se produire que lors de la perte de 3 calculateurs (simplification)**

Décomposition et analyse des causes d'un incident

- **Pb : pour analyser l'incident, il faut**
  - **Décomposer la définition de l'incident : séparer la partie « système » de la partie environnement**
  - **Établir les causes possibles.**
- **Exemple : l'ascenseur**

  Une personne est tombé dans la cage d'un ascenseur

  Décomposition : la porte de l'ascenseur s'est ouverte + l'ascenseur est absent + l'utilisateur a franchi le seuil de l'ascenseur

  Nb : la décomposition nécessite souvent d'établir un lien causal entre l'événement incident et des événements plus simples (antérieurs ou simultanés).

L'étude de la chaîne causale : variantes et améliorations

- **Principe : déterminer les étapes considérées comme nécessaires à l'occurrence de l'incident ou le déclenchant**
- **Pb : par quoi commence t on**
- **Cas 1 : la description des incidents est disponible et l'on dispose d'un modèle du système**

  Analyse « arrière » recherche des causes
- **Cas 2 : description des incidents absente**

  Analyse avant = a partir des états possibles étude des déviations d'état/ altération de structure

Détermination de l’impact / Analyse Avant

- **Identification du périmètre d’étude (système + entités pouvant être affectées)**
- **Si décomposition du système**
  - **Identification des sous composants + dépendances (type d'échange, et comportement attendu**
  - **Identification de variation possible de l'attendu**
- **Détermination de l'impact : identification des conséquences du comportement du système (attendu ou non) sur son environnement**
- **Un concept clé : danger**

  Danger (Hazard) = situation ayant le potentiel de causer l’incident

La méthodologie HAZOP : propagation des « déviations »

- **Principe : description modulaire du système + dépendances spécifique au domaine industriel étudié et analyse des états fonctionnel du système et de leur conséquence**

- **Historique : méthodologie conçue pour l'industrie chimique**
  http://people.clarkson.edu/~wwilcox/Design/hazopcas.pdf
- **Les composants : tuyaux, valves, pompes, fours, capteurs …**
- **Les dépendances : circulation de fluide (propagation de pression, température ….)**
- **Principe : considérer toutes les combinaisons possibles de « sorties » ou état des composants**
- **Objectif : éviter le « on ne pensait pas ce scénario possible »**

Traitement du risque

4 actions possibles

- **Refuser la situation à risque (pas toujours possible)**

  Interdire la production / l'usage / ne pas inclure la fonctionnalité

- **Externaliser le risque**

  Trouver un moyen de reporter la responsabilité sur un tiers

- **Atténuer le risque**

  Trouver un moyen de reporter la responsabilité sur un tiers

- **Ignorer/Accepter le risque**

  Trouver un moyen de reporter la responsabilité sur un tiers

Définition d'objectifs de traitement du risque

- **Objectifs possibles :**
  - Identification/documentation du risque
  - Limitation du risque => obligation de réaliser certains traitement en fonction du risque identifié
  - Dans le cas de l'atténuation : exiger une réduction de l'impact ou de la vraisemblance des incidents
  - Dans le cas de l'acceptation : exiger que l'analyse menant à l'acceptation respecte une certaine rigueur.
- **Conséquences :**
  - *Atténuation du risque => contrainte sur la conception du système son architecture ou son exploitation*
  - *Rigueur de l'étude de risque => contrainte sur les processus de développement*

Sureté de fonctionnement : une discipline d'ingénierie

- **PB : comment et qui doit traiter le risque pour un système nécessitant plusieurs expertises**
  - Mécanique
  - Électricité/électronique
  - Informatique
  - Réseau

- **Réponse : expertise séparée transverse + recommandation par métier**

Sûreté de fonctionnement
Définition et Analyse des Termes

- **Objectif :**

  Obtenir une **confiance justifiée** dans **la capacité du système** à **rendre le service attendu dans des conditions d'usage définies**

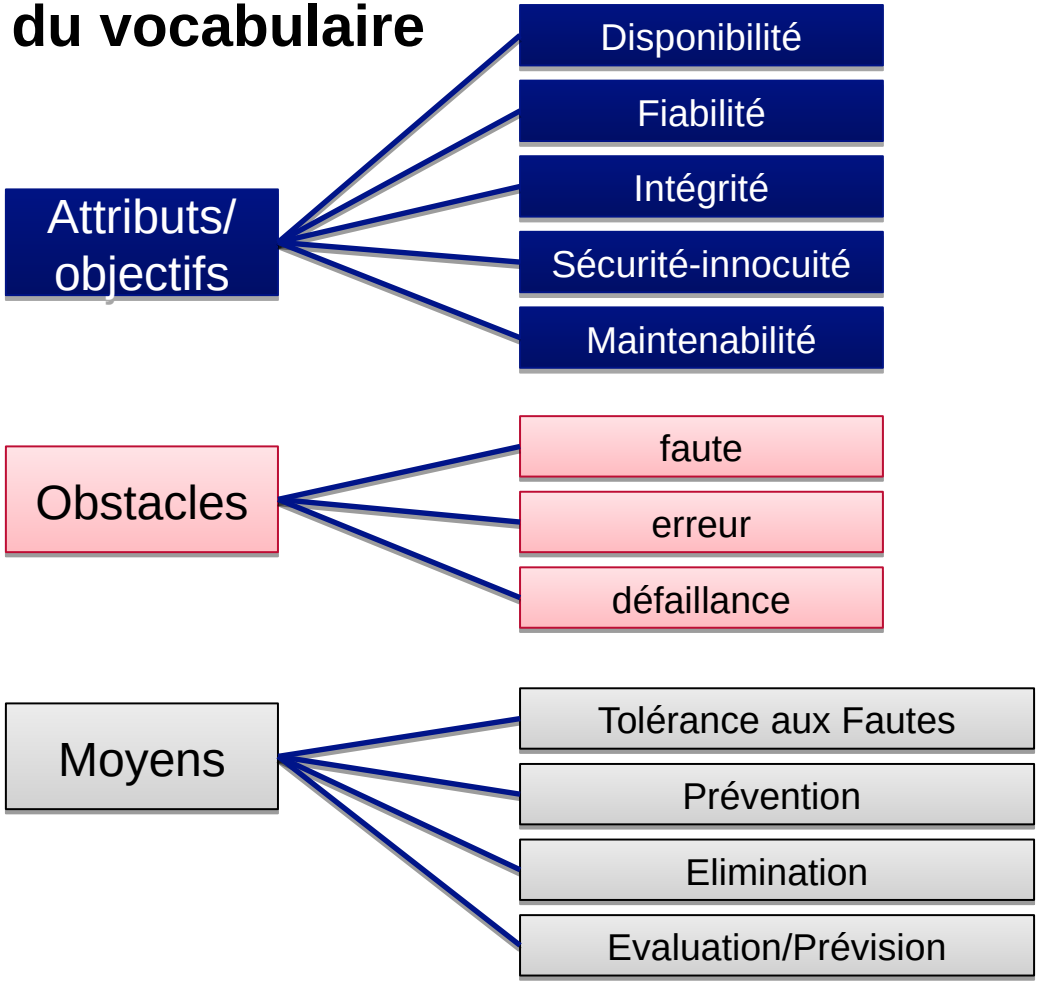
- **Ce que cela sous entend**
  - Définir les conditions d'usage sous lesquelles le système doit rendre le service attendu et/ou avoir un impact maitrisé sur
    - Son environnement (utilisateur(s) inclus)
    - Sur lui même
  - Savoir ce que le système doit faire, lier le système ou ses parties à une fonction/responsabilité
  - Définir des objectifs de garanties sur le respect du comportement attendu (qu'il soit sur le service attendu ou sur l'impact du système)

Identification du périmètre

- **système: unité de description permettant de distinguer l'objet d'étude de son environnement**
  - structure ; description des éléments à priori immuables du système (architecture)
  - État : information variable au cours de la vie opérationnelle du système (effectivement représentée en mémoire pour du logiciel)
  - État interne: fraction non observable de l'état du système
  - Interface : fraction de l'état du système partagée avec son environnement (E/S)
- **Environnement : ensemble des éléments ayant un impact sur l'interface du système ou pouvant être impactés par ce dernier**

Sureté de fonctionnement
les nouveaux objectifs, nouvelles contraintes, nouvelles analyses

Décomposition du vocabulaire



Enjeu

- **Service = utilité / obligation**
- **Déviation du comportement attendu => coût**
- **Rôle de l'ingénierie de la SdF == maîtriser ce coût**

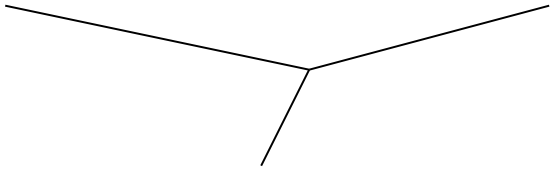
ATTENTION le coût n'a pas toujours une valeur financière !

PB : où se situe la séparation en qualité et sûreté de fonctionnement ?
=> lié à un seuil de coût par incident…

- **Exemples historiques :**
  - Maîtrise de la qualité de service en téléphonie fixe
  - Maîtrise du comportement d'un ascenseur

Et spécifiquement pour les systèmes logiciels

- **[Avizienis et al'04]**
  "Basic Concepts and Taxonomy of Dependable and Secure Computing "



Vocabulaire de base

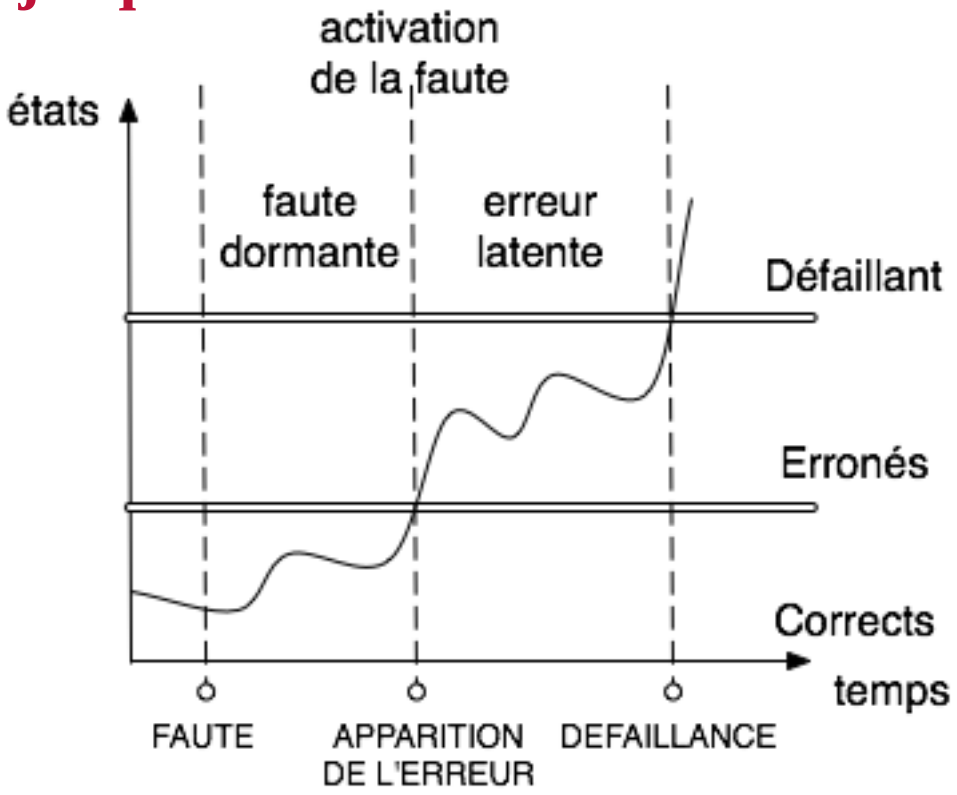
Décrire les entraves – l'alternative

- **Un vocabulaire centré sur le système :**
  - **Défaillance** :
    écart **observable** entre le service attendu et le service rendu
  - **Erreur** :
    Tout ou partie de **l'état interne** du système pouvant causer sa défaillance
  - **Faute** :
    Cause de l'apparition d'une erreur (origine structurelle ou liée à l'état) liée au développement du système ou sa structure

La chaîne faute/erreur/Défaillance

- **Evénements :**
  - Activation
  - Détection
  - Défaillance
- **Etats :**
  - Fautifs (non activés)
  - Corrects (sans faute)
  - Erronés (?)
  - Défaillants (KO)

Pas de détection == erreur dormante jusqu'à la défaillance …

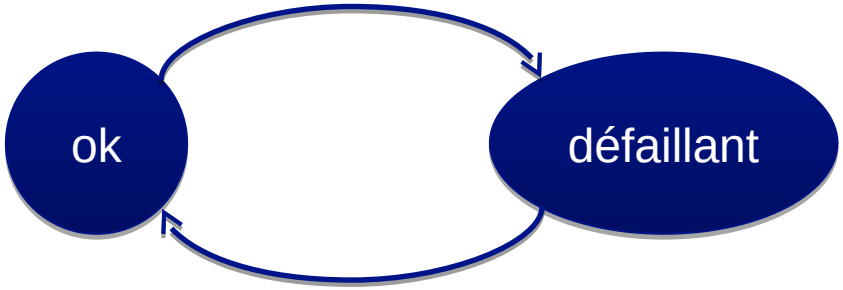


Comportement vs attribut clé

- 2 manière de contraindre la conception d'un système
  - Le véhicule doit pouvoir s'arrêter en moins de 6m pour toute vitesse inférieure à 60 km
  - Le poids du véhicule est inférieur à 800kg
- Une exigence comportementale affecte la fonction du système directement => exigence fonctionnelle
- Une exigence sur un attribut du système n'est pas explicitement rattachée à un comportement => exigence non fonctionnelle
- Donnez un autre exemple d'exigence fonctionnelle de sûreté de fonctionnement

Les attributs de Sureté de fonctionnement par l'exemple

- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru**
- **Description usuelle de**
  - La disponibilité : être dans l'état OK

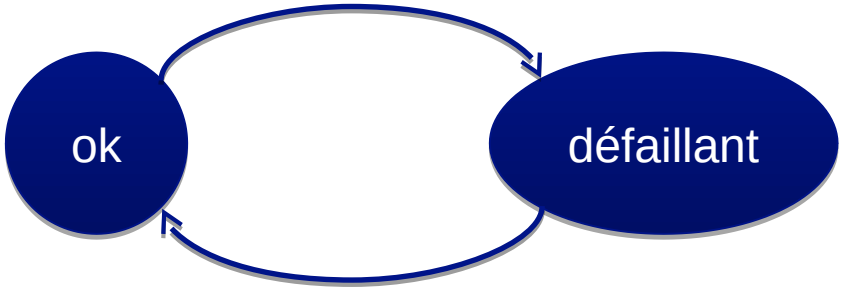


Ou Temps moyen avant transition vers défaillant

$$\frac{\text{Temps(OK)}}{\text{Temps(OK)+Temps(défaillant)}}$$

Raffinement des attributs

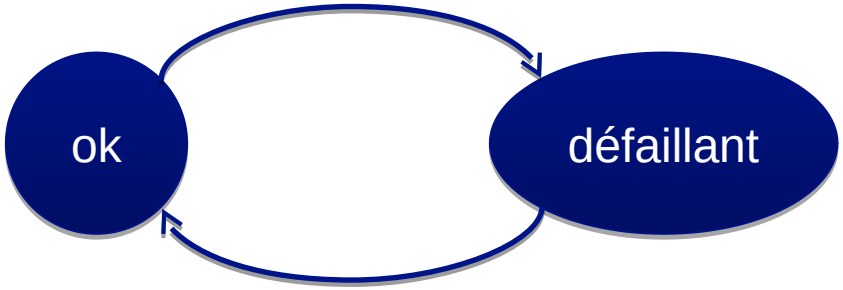
- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru**
- **Description usuelle de**
  - La fiabilité : ne pas quitter l'état OK



  - Probabilité ou condition pour transition vers « défaillant »

Raffinement des attributs

- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru**
- **Description usuelle de**
  - La maintenabilité : garantir la transition de défaillant vers OK

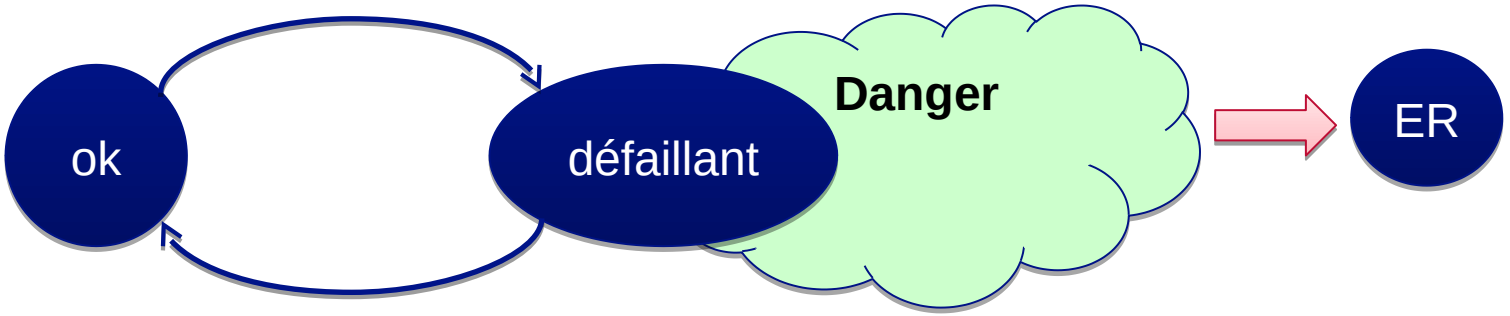


  - Le retour à l'état Ok conditionné par une intervention

=> Analyse du temps moyen jusqu'à intervention, difficulté de l'intervention, coût …

Raffinement des attributs

- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru**
- **Description usuelle de**
  - La sécurité innocuité :



  - Identification de la contribution des états défaillants aux états de danger
  - Classification des états défaillant en fonction de la gravité de l'événement redouté (ER) en bout de chaine

le cas de l'intégrité

- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru + description du système**
- **Description usuelle de l'intégrité :**
  - **Contra**inte sur la capacité à détecter / corriger des altération des données stockée
  - Mais aussi Contrainte de déterminisme comportemental

L'intégrité des données est fortement lié à la fiabilité d'un dispositif de stockage/communication (i.e. une partie)

Attributs et contraintes de sureté de fonctionnement

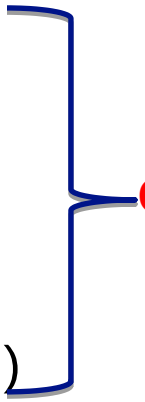
- **Attributs :**
  - Disponibilité (availability) : capacité du système à offrir tout ou partie du service
  - Fiabilité (reliability) : capacité du système à effectivement délivrer un service correct (lorsqu'il est disponible)
  - Sécurité-innocuité (safety) maitrise/connaissance des conséquences d'une comportement du système (attendu ou non)
  - Intégrité (integrity) capacité du système à détecter ou empêcher toute altération de sa structure ou de son état en contradiction avec le comportement attendu
  - Maintenabilité (maintainability) capacité du système à faire évoluer sa structure ou son état pour faciliter le retour à un état disponible / fiable…

- **Objectifs (I) : contrainte sur les attributs**

Des méthodes, et des standards …

- **La sureté de fonctionnement : un objectif**
  - Impacte tous les niveaux du développement
  - Compromis coût d'application / coût défaillance et conséquences
  - Contrainte réglementaire (ex. nucléaire/chimie lourde) ou bonne pratique

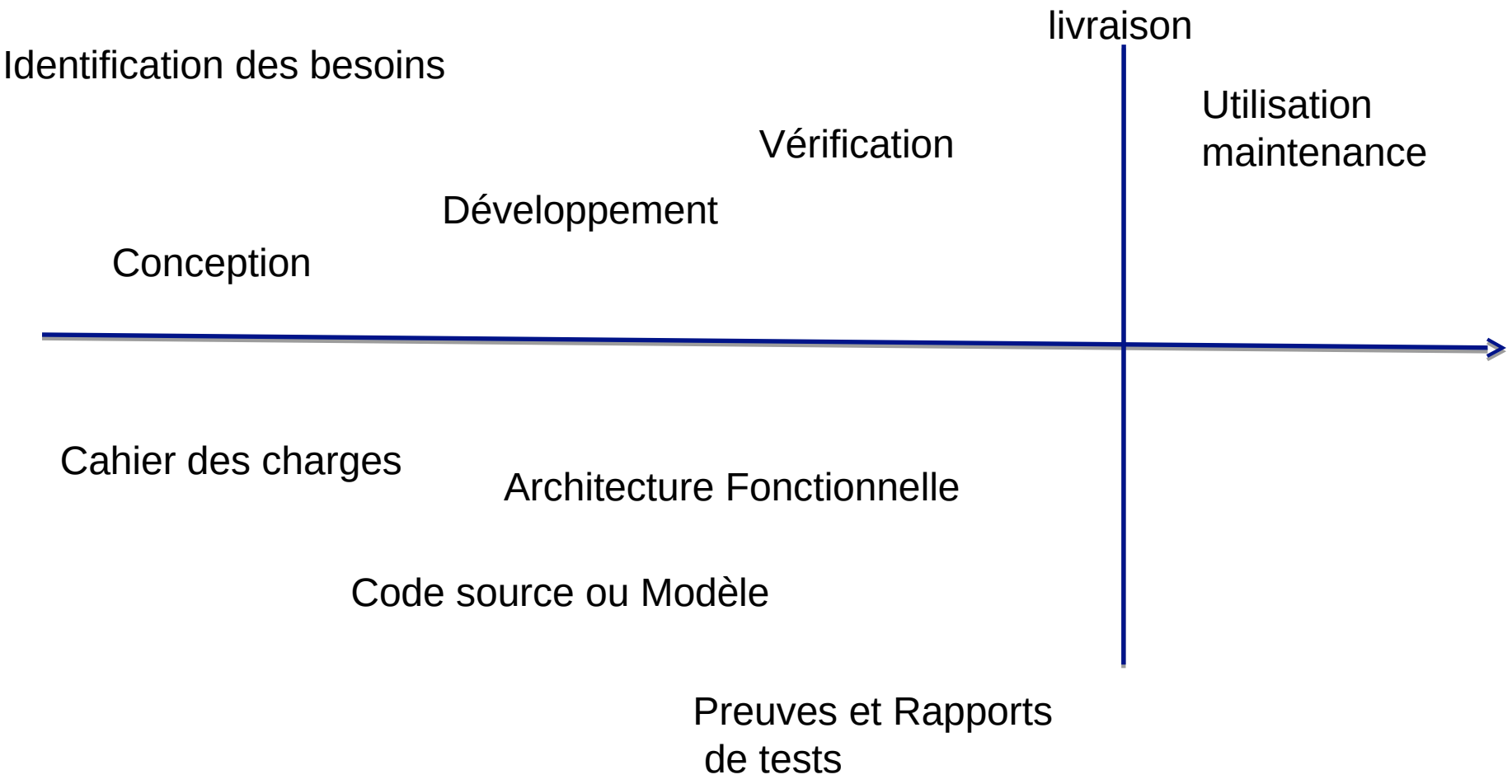
- **Des standards par niches industrielles pour décrire les actions à mener**
  - Ingénierie chimique => premier standard ~198x (OSHA)
  - Ingénierie ferroviaire (IEC 61508)
  - Avionique (ARP4761)
  - Automobile (uniquement depuis ISO 26262)



Le cycle de développement d'un système cycle de vie

- **Principe d'organisation des différentes phases d'ingénierie**
  - Détermine la nature des activités
  - Détermine l'objectif et les résultats attendus (logiciel, description textuelle …)
  - Détermine l'enchaînement dans le temps de ces étapes
- **Exemple : développement par Cycle en V**
  - Décomposition des activités :
    description des objectifs, du comportement, et de la structure interne du système
  - Planification et identification des dépendances
- **Positionnement différent de chaque activité**

Processus de développement et Ingénierie



Les moyens pour obtenir un niveau voulu de SdF

- **4 Approches complémentaires**
  - **Prévention des fautes :**
    Méthodes destinées à empêcher l'occurrence même des fautes
  - **Élimination des fautes :**
    Méthodes de recherche et de suppression des fautes de conception et développement
  - **Prévision des fautes :**
    Analyse de la fréquence ou du nombre de fautes et de la gravité de leurs conséquences
  - **Tolérance aux fautes :**
    Mécanismes au sein du système destinés à assurer les propriétés de SdF voulues en présence des fautes

Exemple de mise en œuvre

- **Prévenir**
  - Méthode et contraintes de conception permettant d'éviter la présence de fautes dans la conception ou le produit développé

  Ex : Programmez en C, prévention == sans pointeurs, ni tableaux

- **Eliminer**
  - Méthodes de mise au point permettant d'identifier et corriger les éléments structuraux correspondant à des fautes dans le but de les supprimer

  Ex : Programmation en Java, élimination == test Junit et modification du code pour supprimer les sources d'exceptions, idem avec gdb pour le C

Sûreté de fonctionnement
Les moyens…

- **Tolérer**
  - Méthodes de conception permettant de réagir à l'activation d'une faute, la détection d'une erreur, dans l'objectif d'empêcher l'occurrence d'une défaillance ou d'en limiter les conséquences.

  Ex : implémenter le traitement d'exception Java qui redémarre automatiquement l'application
- **Évaluer/Prédire**
  - Méthodes et métriques permettant de quantifier (mesurer) ou qualifier (classer) un système vis à vis des attributs de la sûreté de fonctionnement

  Ex : Émuler un réseau pour identifier les conditions d'engorgement (qualitatif), ou utiliser une distribution de Pareto pour prédire la fréquence d'occurrence de perte de messages sur un réseau sans fil

Le rôle des formalismes et de leur manipulation

- **De quoi a-t-on besoin ?**
  - Description des objectifs du système
  - Description de sa structure et de son comportement
- **Spécification Fonctionnelle**
  - Description des objectifs du système
  - Décomposition du système en plusieurs entités si ses fonctionnalités sont complexes
- **Conception et description d'implémentation**
  - Description détaillée du comportement interne du système et de ses entités assurant la satisfaction des objectifs
  - Description détaillée des ressources (logicielle…matérielle) nécessaire au bon fonctionnement des entités du système

Spécification fonctionnelle:
* apport d’un modèle architectural
* apport de la logique

De quoi a-t-on besoin ? (bis)

- **Distinguer dans un système qui est responsable**
  - V1 : Un contrôleur de vitesse sur un véhicule a pour fonction de contrôler l'accélération du véhicule à partir d'information sur sa vitesse courante
  - V2 : Le contrôleur de vitesse sur un véhicule à pour fonction de contrôler le couple du moteur à partir de mesures de vitesse rendues par des capteurs, afin d'assurer une vitesse programmée

    Si on était tatillon on pourrait dire que dans un des deux cas le contrôleur n'avait pas pour mission d'empêcher une vitesse excessive… (laquelle?)

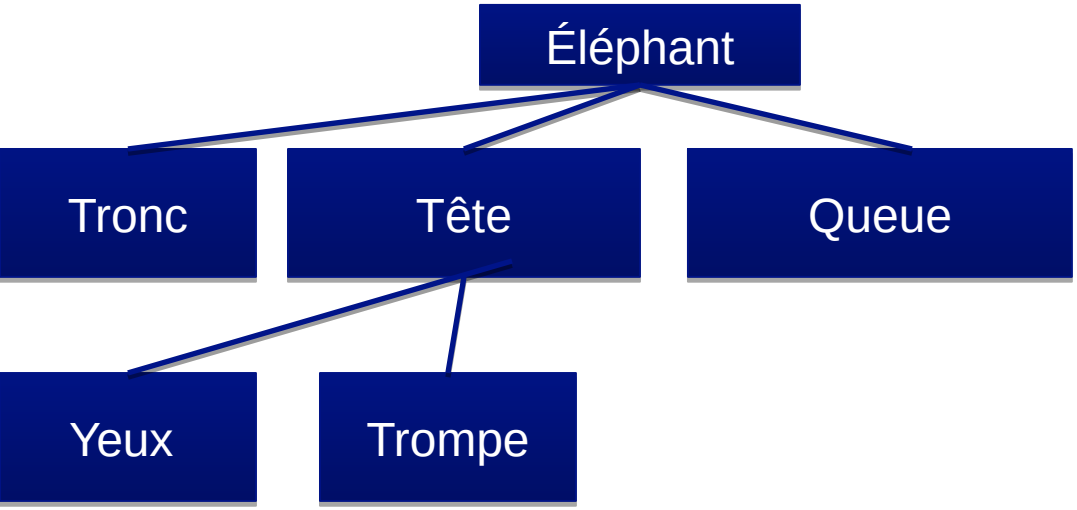
- **Distinguer ce que l'on veut de ce que l'on veut éviter**
  - Ex : « le train ne doit pas rouler lorsque ses portes sont ouvertes »

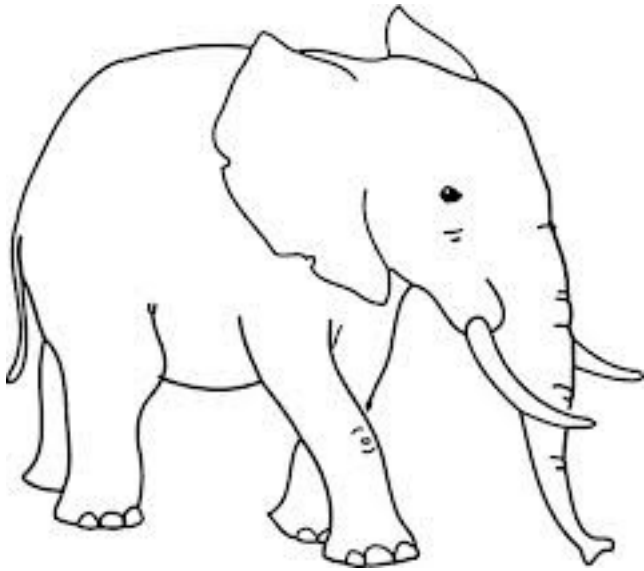
Les formes de spécification fonctionnelle

- **Les formats :**
  - Textuelle rédigée
  - graphique annotée
  - Mathématique structurée
- **Qualité de la description :**
  - Rédigée « libre »
  - Structurée
  - Semi-formelle
  - Formelle
- **Leur interprétation :**
  - Exemple1 : spécification d'un déplacement par définition de vecteurs vitesse en 3D => représentation graphique non ambiguë
  - Exemple2 : page de manuel d'une commande shell (texte rédigé)

Décrire le système

- **Le point de vue « Organique » == structure physique**

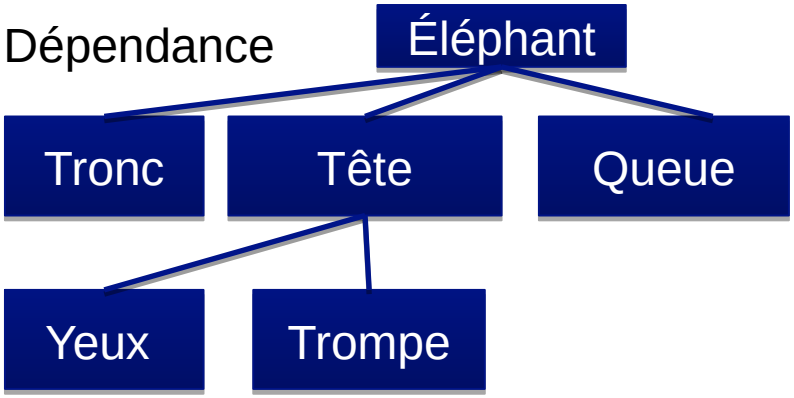


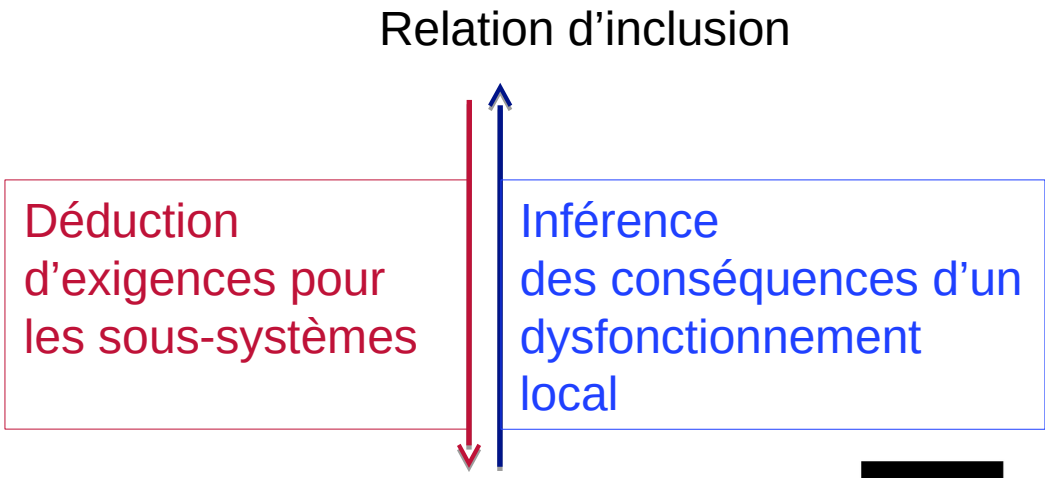


Centré sur la structure physique, permet de localiser la manifestation d'une faute à une partie de la structure du système.

Modèle à composant, Hiérarchie & Interfaces

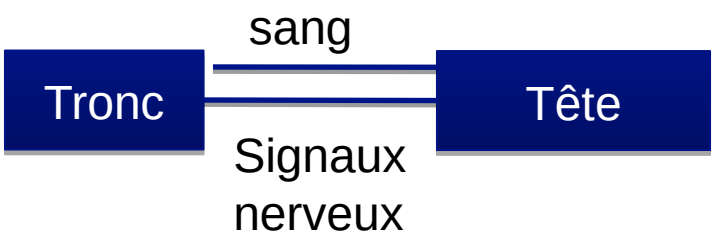
- **Relations entre « blocs »**
  - **Inclusion** ?
  - **Collaboration** ?
  - Héritage ?
- **Un dessin n'est pas un modèle, modèle = structuration d'une information interprétable**
- **Intérêt du modèle**





Modèles à composant, Hiérarchie & Interfaces

- **Relations entre « blocs »**
  - **Inclusion** ?
  - **Collaboration** ?
  - Héritage ?
- **Un dessin n'est pas un modèle**
- **Intérêt du modèle**



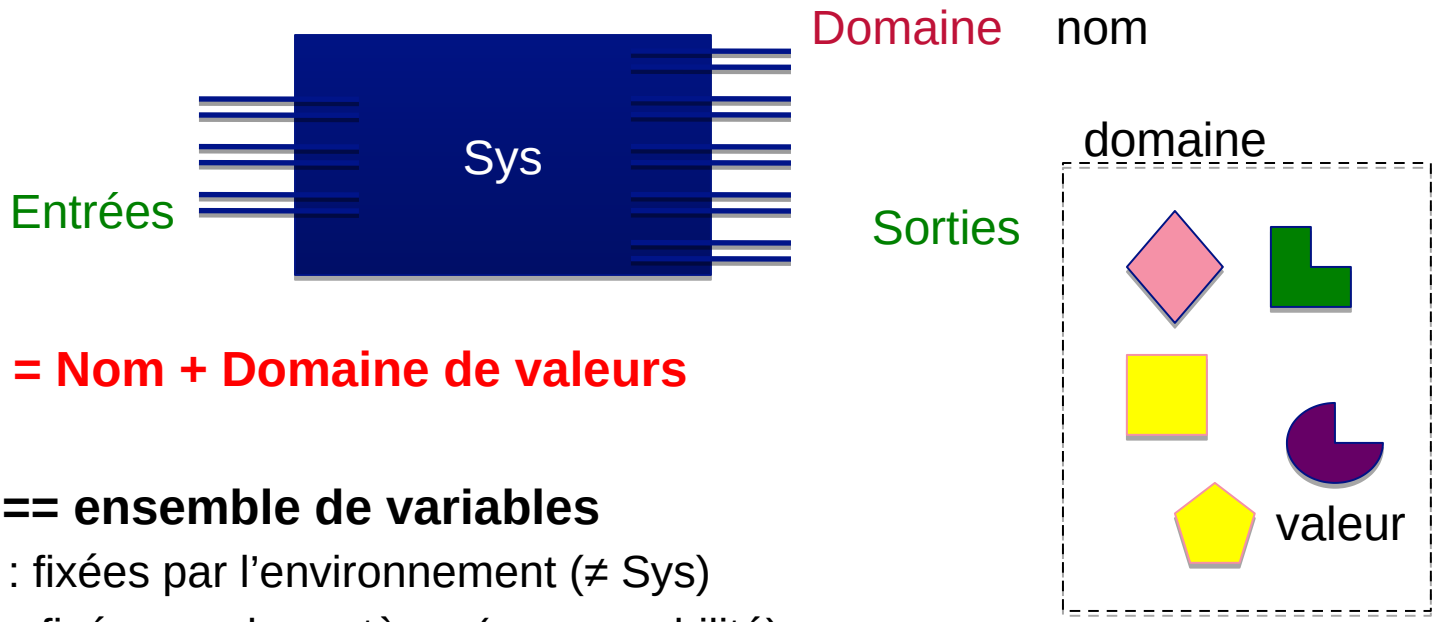
Relation de collaboration
bloc ~ fonction ~ transformation E/S

Déduction
de dépendances
Requis/Fourni

Inférences
d'E/S incorrectes

Et en pratique ….
Modéliser la circulation de données

- **Idée ; l'activité d'un système == circulation de données**



- **Variables = Nom + Domaine de valeurs**

- **Interface == ensemble de variables**
  - Entrée : fixées par l'environnement (≠ Sys)
  - Sorties : fixées par le système (responsabilité)

  (définition améliorée par la suite)

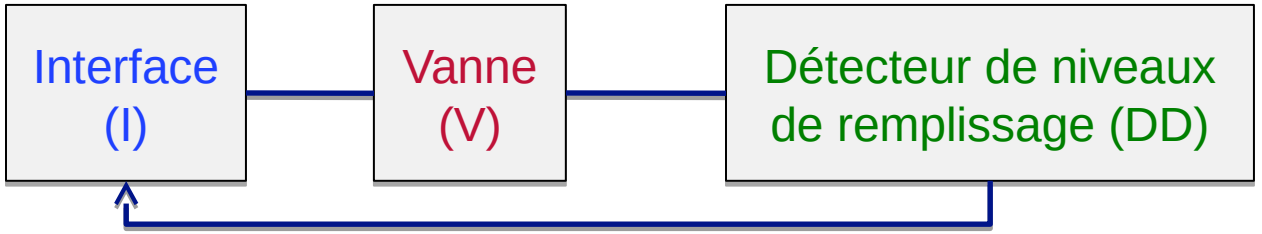
- **Modélisation de l'attendu == relation entrées/sorties**

Exemple : Système de contrôle d'un déversoir

- **Fonction** :

  Contrôle de la tension d'alimentation d'une vanne qui contrôle le délestage d'une réservoir d'eau alimentée par une rivière.

- **Sous-fonctions :**
  - Contrôle ouverture/fermeture de la vanne
  - Détection automatique d'une condition de débordement du réservoir
  - Interface (écrans) pour superviseur humain surveillant/contrôlant le système



Décrire les interactions == E/S par blocs, plus correspondance S-E

Exemple : le modèle fonctionnel
Se donner des variables pour décrire le système

Nom	Composant	Type	Rôle
EV	Interface	{O,F,NC} *(S)*	Affichage état de la vanne
EN	Interface	[0,1] *(S)*	Affichage du Taux de remplissage du réservoir
OU	Interface	{Oui, Non} *(E)*	Commande de l'ouverture d'urgence
F	Interface	{Oui, Non} *(E)*	Commande de la fermeture
Com	Interface	{O,F} (S)	Commande transmise à la vanne depuis l'interface
CIV	Vanne	{O,F} *(E)*	Commande reçue depuis l'interface
DD	Vanne	*{O,N} (E)*	Détection d'un risque de débordement
SV	Vanne	*{O,N} (S)*	Ecoulement en sortie de vanne

A quoi sert un tel modèle ?

- **La décomposition structurelle permet de séparer les responsabilités**
- **L'identification des interactions permet de nommer les données, les artefacts échangés**
- **Il devient donc possible à partir de ces noms de décrire ce que l'on attend**
  - du système (interface avec son environnement)
  - De chacune de ses parties
- **NB: Des langages tels que UML, SysML …. ont pour but de vous donner des termes pour décrire de telles structures (et les interactions entre blocs)**
- **Ensuite, on peut appliquer des méthodes de type Hazop ….**

La vision clé – attribut et modèle architectural

- Concept de base : la relation binaire
  - Nom d'un composant et ses entrée sorties
  - Le nom d'une entrée et son type
  - Le nom d'une entrée et la sortie à laquelle elle est reliée
  - ….
- Analyse = parcourt des relations binaire

- Représentation sous forme de graphe de l'information clés
- Les algorithmes d'analyse == logique de parcourt de graphe
  - Action sur un nœud
  - Action sur les arcs

L’étape suivante ?

- **Comportement attendu**
  - Ex1: Si le réservoir atteint un niveau de 97% de remplissage alors déclencher l’ouverture de la vanne
  - Ex2 L’interface de contrôle doit afficher en permanence l’état de la vanne
- **Ces descriptions constituent une spécification fonctionnelle du système**
  - Description de ce que doit « faire » chaque bloc sans nécessairement préciser comment
  - Plus le traitement est complexe plus il est nécessaire d’abstraire l’attendu

Description textuelles et textuelles structurée

- **Exemple : fiche man d'une fonction shell**
  - Structuration de la description du fonctionnement de la fonction, de ses paramètres et sorties
  - Champs : Synopsis, Description, Option ….
- **Avantage :**
  - peu ou pas de restriction sur le niveau de détail utilisable (peut contenir des formules mathématiques)
  - Compréhensible par tout le monde (en principe)
  - Concis si l'on fait des hypothèses sur la connaissance partagée (exemple section synopsis d'un fiche de man)
- **Inconvénient :**
  - Risque de contradiction difficile à analyser et détecter
  - Quasi impossible à faire analyser par un ordinateur
  - Risque de description incomplète, ou ambiguë

Les propriétés souhaitées

- **Non ambiguë : les affirmations et formulations utilisées ne doivent pas amener de question d'interprétation du type**

« le terme ''dès que'' signifie-t-il juste à près ou en même temps ? »

- **Vérifiable : il est doit être possible de dériver, de chaque affirmation, une méthode pour en vérifier la réalisation dans l'implémentation du système.**
- **Consistante : si la spécification est complexe et constituée de plusieurs affirmations, ces dernières ne doivent pas être en contradiction.**

Formalisation par description logique

- **Principe :**
  - Formule logique == séquence structurée de mots correspondants :
    - À des affirmation (vraie ou fausses)
    - A des combinaisons de ces affirmations (et, ou, si … alors …)
  - Un raisonnement mathématisé pour calculer pour chaque formule les conditions dans lesquelles la formule est vraie
- **Les logiques que nous allons voir :**
  - Propositionnelle, premier ordre
  - Temporelle

La syntaxe

- **Formule logique == séquence de termes « typés »**
- **Les constantes : vrai, faux. Une variable booléennes == un nom dont la valeur appartient à {vrai, faux}**
- **L'ensemble des variables booléennes *V ={x1,x2,…., xn}***
- **Notion d'opérateur vs fonction :**
  - Un opérateur est une fonction qui a 1 ou 2 paramètres !!
  - Notation : (x1 op x2) équivalent à op(x1,x2)
  - Intérêt : permet souvent des notations plus compactes
- **Opérateurs logiques :**
  - Et : 2 paramètres noté ∧
  - Ou : 2 paramètres noté ∨
  - Implique : 2 paramètres noté =>
  - Négation : 1 paramètre noté not

Interprétation d'une formule

- **Affectation de valeur aux variables booléennes : valuation, ou fonction de vérité :**
  f: x in {x1,…xn} -> {V,F}
- **Interprétation d'une formule : calcul de la valeur résultant de l'évaluation des opérations**
- **Tables d'évaluations : (a,b deux variables et F,V constantes faux, vrai, a□ V signifie a vaut V.**

a∧b	a=V	a=F
b□V	V	F
b□F	F	F

a∨b	a=V	a=F
b□V	V	V
b□F	V	F

a=>b	a=V	a=F
b□V	V	V
b□F	F	V

not a	
a□V	F
a□F	V

Modèle logique et formule

- Rappel : un modèle simplification de la réalité
- Un modèle logique associé à un ensemble de variables booléennes, {x1,…,xn}, est un ensemble de valuations
- Exemple modélisation d'un problème de répartition de de deux balles dans 3 sacs :
- Variables : bxsy est vrai si il y a x balles dans le sac y
- Considérons le vecteur μ=(b0s1,b1s1,b2s1,b0s2,b1s2,b2s2,b0s3,b1s3,b2s3)
- Un modèle correct de répartition est l'ensemble des valuations qui assurent qu'il y a au plus 2 balles dans l'ensemble des sacs …
- μ=(F,F,F,F,V,F,F,V,F) fait partie de ce modèle
- μ=(F,F,F,F,F,F,F,F,F) n'en fait pas partie.

Les prédicats: des fonctions particulières

- **Un prédicat : fonction associant un booléen à n-uplet de paramètres p1, …pn de types respectifs T1,… Tn**
- **Le nombre de paramètres d'un prédicat est appelé l'arité du prédicat**
- **Opérateur de comparaison : <,>,==**
  - Des prédicats déguisés :
  - < : int,int -> {V,F}
  - > : réel, réel -> {V,F}
  - ….
- **Tous les opérateurs ne sont pas des prédicats : +, *,/…**
- **Une formule contenant des prédicats verra ses valuations définies sur le types des paramètres des prédicats**

Exemple

- **Variables typées : x : entier, y : booléen, z: ensemble d'entiers**
- **Opérateur : |r| retourne le cardinal de r (si r est un ensemble**
- **Formule :**

  1) x<4,
  2) y=> |z|<1
  3) (y>2)=>x

Formules logiques avec prédicats

- **Exemple : F = (x<1) => y ≠ 0**
  - Problème on ne connaît pas le type de x et y
  - x et y sont dit des variables **libre,** elle peuvent prendre n'importe quelle valeur la formule a un sens différent pour les entiers, les réels…
- **La quantification : détermine l'ensemble de valeur du type de la variable pour lesquelles la formule doit être vraie**
  - $F = \forall x \in X.\ p(x)$: signifie que F est vraie si pour tout x dans X le prédicat p(x) est vrai
  - $F = \exists x \in X.\ p(x)$: signifie que F est vraie si il existe un x dans X tel que le prédicat p(x) est vrai

Satisfiabilité et Validation d'une formule

- **La satisfiabilité et la validation d'une formule F sont des propriétés qui s'exprime par rapport à un ensemble de valuations X pour un ensemble variables V.**
- **Une formule est dite satisfiable dans X si il existe au moins une valuation f des variables de V dans X telle que F est évaluaée à vrai pour la valuation f**
- **Une formule est dite valide dans X si pour toute valuation de X, alors la formule F est évaluaée à vrai.**
- **Si X représente l'ensemble de toutes les valuations possibles de V alors,**
  - on dit que F est une tautologie si elle est valide dans X (c'est tout le temps vrai)
  - On dit que F est une contradiction si F n'est pas satisfiable dans X.

Utilisation de la logique pour caractériser les états défaillants

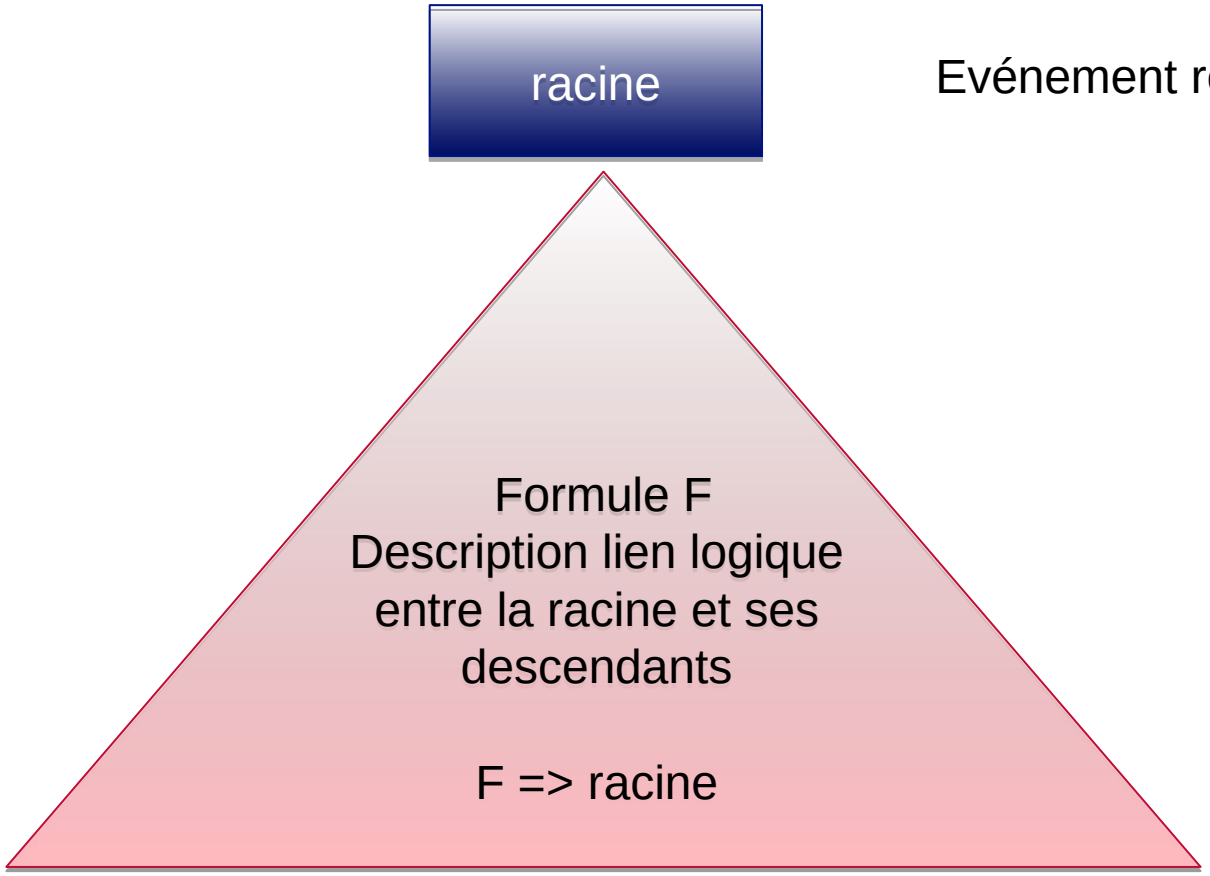
- **Logique : mathématique == langage**
- **identifier les événements redoutés, et leurs causes possibles par des formules**
  - Etat binaire des composants du système) ok/ko
  - Des formules pour décrire les lien causes=>conséquences
  - Des méthodes d'analyse de modèles d'architectures pour construire ces formules pour + de couverture (discriminer les bons états)
- **Utilisation de prédicats pour identifier les événements redoutés : (hors du périmètre du cours)**

Arbres de Fautes

- **Idée : se concentrer sur la relation de causalité**
- **Pré-requis : connaître les événements redoutés d'intérêt**
- **Principe :**
  - Définir l'événement racine de l'arbre (doit être un événement dont on cherche à déterminer les causes)
  - Identifier de manière exhaustive les causes logiques de l'événement
  - Relier ces causes par un connecteur définissant leurs interactions
    - Et
    - Ou
    - Ou exclusif
    - …

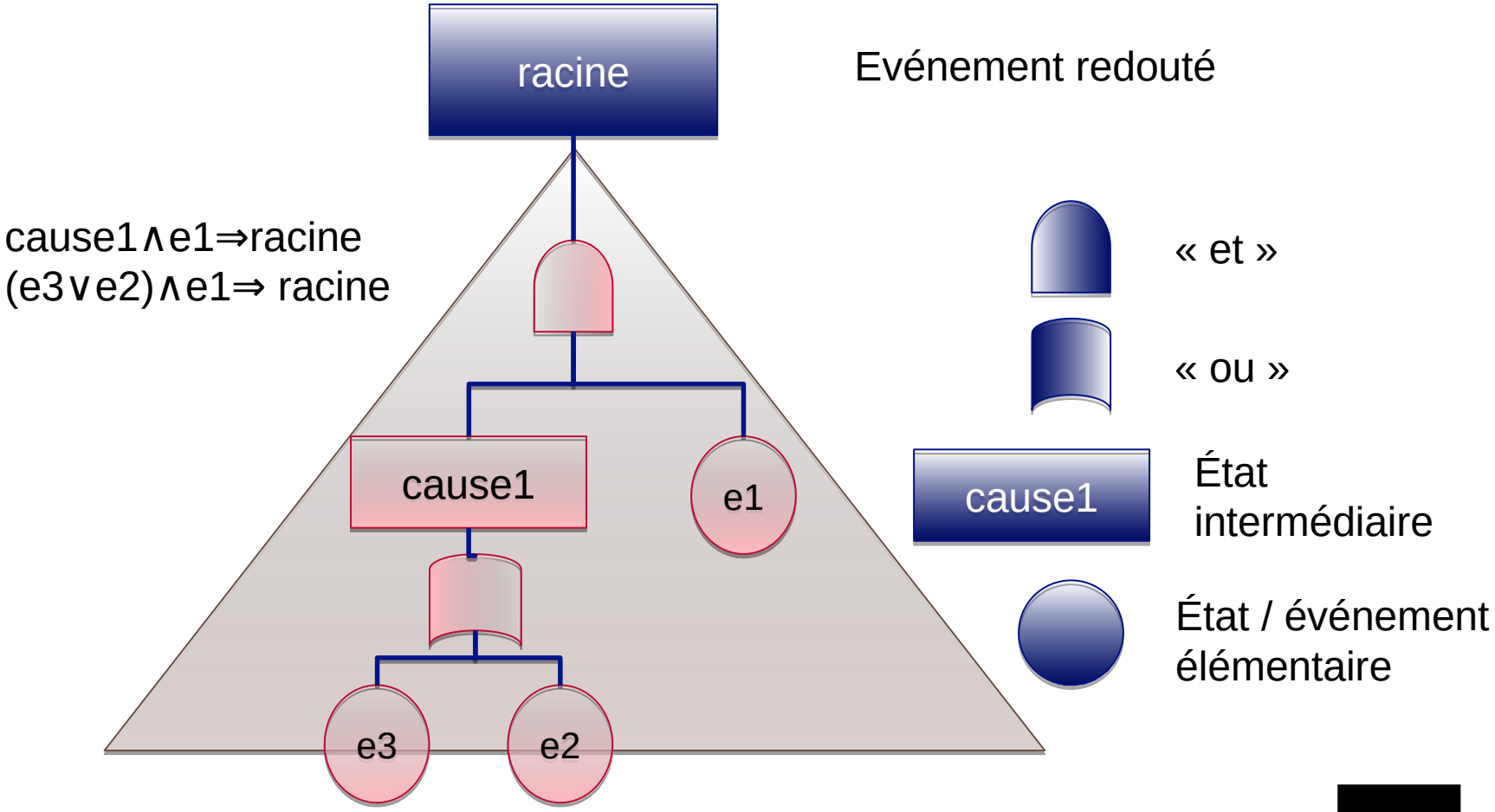
Un arbre de faute = formule logique sur l'état du système avec identification des variables/formules déterminant l'état complet

Syntaxe & sémantique : arbres de fautes

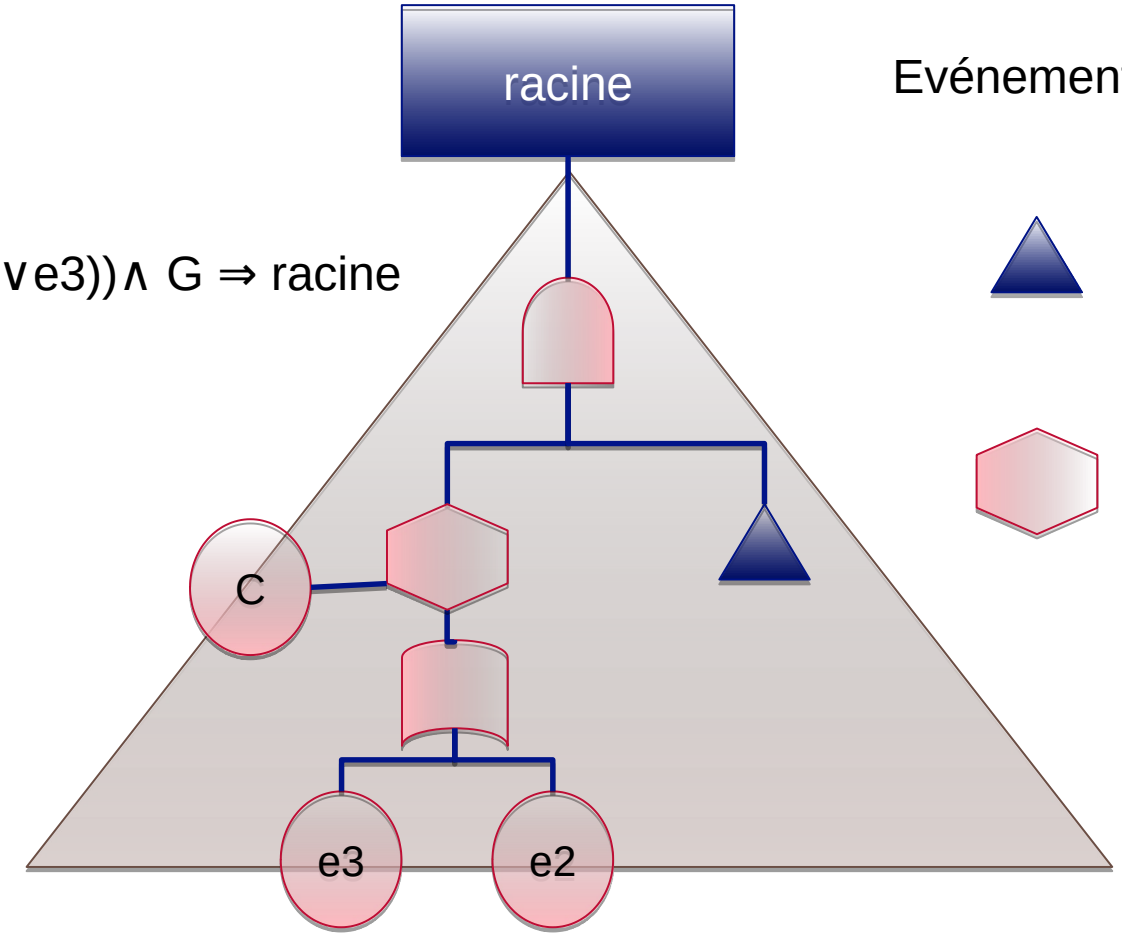


Evénement redouté

Syntaxe & sémantique (basique) : arbres de fautes



Syntaxe & sémantique (avancé) : arbres de fautes



$(c1 \Rightarrow (e2 \vee e3)) \wedge G \Rightarrow racine$

Evénement redouté

Référence à un arbre de faute

Porte conditionnelle : si C est faux => faux sinon l'entrée est propagée

Analyse qualitative de l'arbre

- **La fonction de structure : SF (x_1,…., x_n)**

  déduite de l'arbre telle que SF(x_1,…., x_n) ⇒ racine
- **Fonction de l'état des feuilles de l'arbre**
- **Hypothèse :**
  - ordre partiel sur les valuations de $X=(x_1,\ldots, x_n)$ dans B^n (vecteur de booléens)
  - Règle 1 : Vrai > faux
  - Règle 2 : Soit x et y deux valuations de X

$$x \prec y \Leftrightarrow \forall i \in \{1,..,n\}, \exists j, (xi < yi \vee xi = yi) \wedge xj < yj$$

- **Exemples : (vrai, faux , vrai) >(faux, faux, vrai)**
  mais (vrai, faux, vrai) et (faux, vrai, faux) non comparables

Coupure minimale d'un arbre

- **Un état amenant à rendre vrai l'événement redouté (ER) tel que tout événement feuille retiré empêche l'événement de se produire => explication minimale de l'ER**

- **Formellement :**

  X0 est une coupure minimale de SF(X) si quelque soit Y<X SF(Y) est faux, et SF(X) est vrai.

Analyse quantitative sur l'arbre calcul de vraisemblance

- **Idée : associer une probabilité aux feuilles**

- **Pr(racine)= Pr(SF(X)).**
- **Comment calculer Pr(SF(X)) ?**
- **Il existe une règle de transformation**

  Pr(SF(X))= G(Pr(x1),…, Pr(xn)) si les feuilles correspondent à des états indépendants.

- **Mise en œuvre : « et » □ *, « ou » □ +, not(x) □ 1-Pr(x)**
- **Exercice calculer la probabilité de l'exemple « basique » transparent**

Conclusion

- **Ce qu'il faut retenir :**
  - **Définition risque/ incident/ faute / erreur défaillance**
  - **4 grandes approches pour arriver à avoir un système sûr de fonctionnement (prévention/élimination/tolérance et évaluation).**
  - **Les 5 attributs de la SdF + différence exigence fonctionnelle ou non**
  - **Arbre de fautes (et rappels de logique)**
  -
- **La suite : la tolérance aux fautes.**