
Institut
Mines-Télécom

# Ingénierie de la sûreté de fonctionnement

Thomas Robert
thomas.robert@telecom-paristech.fr
COMASIC (902)

# Incident, Danger, Risque et Systèmes

# 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 ?

# Incident, vraisemblance, et risque

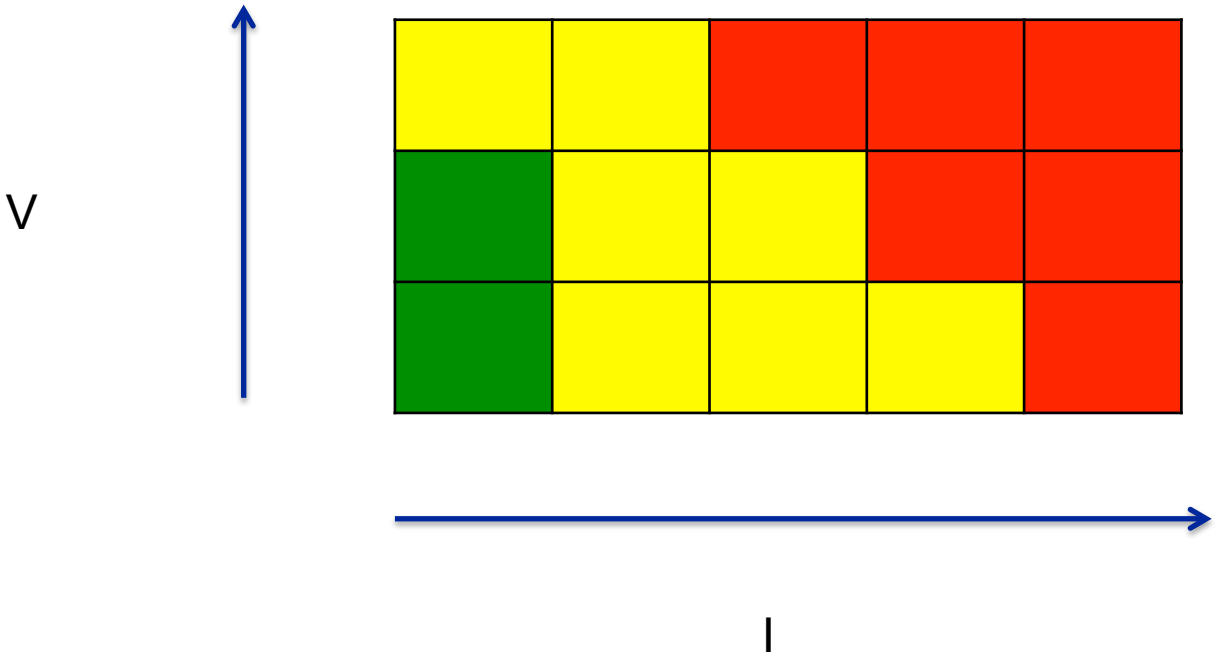
- **Incident = événement redouté**
- **Terme alternatif : accident (bcp utilisé dans l'industrie)**
- **Définition générale du risque :**

  Une estimation qualitative ou quantitative déterminant l'importance des conséquences des incidents prenant en compte leur vraisemblance

- **Exemple qualitatif :**
  - Pour chaque incident : définition de 4 catégories d'impact possible (catastrophique, sérieux, modéré, bénin)
  - Vraisemblance : 3 degrés de vraisemblance en fonction du type de causes (quasi inexistant, probable, fréquent).
  - Le risque peut correspondre à 3 niveaux discret combinant l'information de vraisemblance, et d'importance de l'impact niveaux (Rouge >Jaune > Vert)

# Matrice de risque : visualisation du lien impact / sévérité – vraisemblance

- **Idée : représenter dans le plan les catégories de risques en fonction de l'impact (ou sévérité) de l'incident (I) et de sa vraisemblance (V)**



# 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 prévenir l'incident ou limiter sa sévérité*

- **Ignorer/Accepter le risque**

  *Décider de produire / utiliser le système en dépit du risque repose sur une évaluation prédictive du risque*

**Rem on supposera par la suite que le risque ne peut être refusé ou transféré (décisions hors périmètre de l'UE)**

# Détermination de l'impact, notion de danger

- **Identification du périmètre d'étude**
- **Identification des entités au sein du périmètre et de leur dangerosité (aptitude à engendrer un incident)**

  Sources : experts, conceptions similaires, règlementations
- **Identification des ressources dangereuses**

  Électricité, vapeur, fluide sous pression ….
- **Identification de conditions environnementale dangereuses**

  Incendies, inondations, pollution chimique …
- **Ce qu'il faut retenir :**

  **notion clé de danger = situation pouvant impliquer un incident**

  **Ces situations peuvent être internes au système ou induites par l'environnement**

# Système, objectif et exigence

**Définition système**

A set of interrelated components working together as an integrated whole to achieve some common objective.

**Mots clès importants :**

- Objective : un système a un but / des responsabilités
- interrelated components : la description du système fait apparaître un structure avec des dépendances, interactions
- Integrated : des propriétés / caractéristiques sont globales

**Remarque : la gestion du risque fait partie des objectif d'un système**

# Cas d'usage et paramètre critique

- **Cas d'usage : description des interactions souhaitées entre le système et son environnement**

⇒ **Généralisation à une description de comportement attendu en termes d'échanges de données / ressources.**

- **Attribut : caractéristique qualitative ou quantitative d'un système pouvant être mesurée ou évaluée.**

*exemple : le poids d'un véhicule, l'empreinte mémoire d'un code.*

- **Paramètre critique : attribut du système pouvant faire l'objet d'une contrainte (cet attribut doit pouvoir avoir plusieurs valeurs possibles)**

# Exemple : Distributeur de boisson

- Donner un exemple de cas d'usage d'un distributeur automatique de boisson
- Expliquer comment un tel cas d'usage permet de raisonner sur les incidents associés à ce système
- Discutez des mesures pouvant être prises pour limiter les conditions d'occurrence d'une brulure du client
- Donner un attribut d'un distributeur de boisson

# Exigence Fonctionnelle vs non Fonctionnelle

- **Exigence fonctionnelle = comportement (enchainement d'interactions) requis pour satisfaire un des objectifs du système.**
- **Exigence non fonctionnelle = contrainte à respecter ou faire respecter sur l'un des paramètres critiques du système.**
- **Exemple :**
  - temps de réponse < 10 ms = ENF
  - Obligation de payer avant de retirer une boisson = EF
  - Obligation de verrouiller la porte d'accès à la boisson tant qu'elle n'est pas servie = EF

# Description structurelle du système

- **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**

# Sûreté fonctionnelle

- **Exigence de sûreté fonctionnelle**

Ensemble de comportements et composants du système destinés à assurer que les fonctionnalités du système, et ses cas d'usages n'engendrent pas de risque inacceptable

- **Exemple : mécanisme de sécurité enfant à bord d'une voiture**

**=> Les incidents inacceptables (ne pouvant être ignorés) correspondent à une déviation du comportement attendu**

# Vocabulaire de base

# 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)

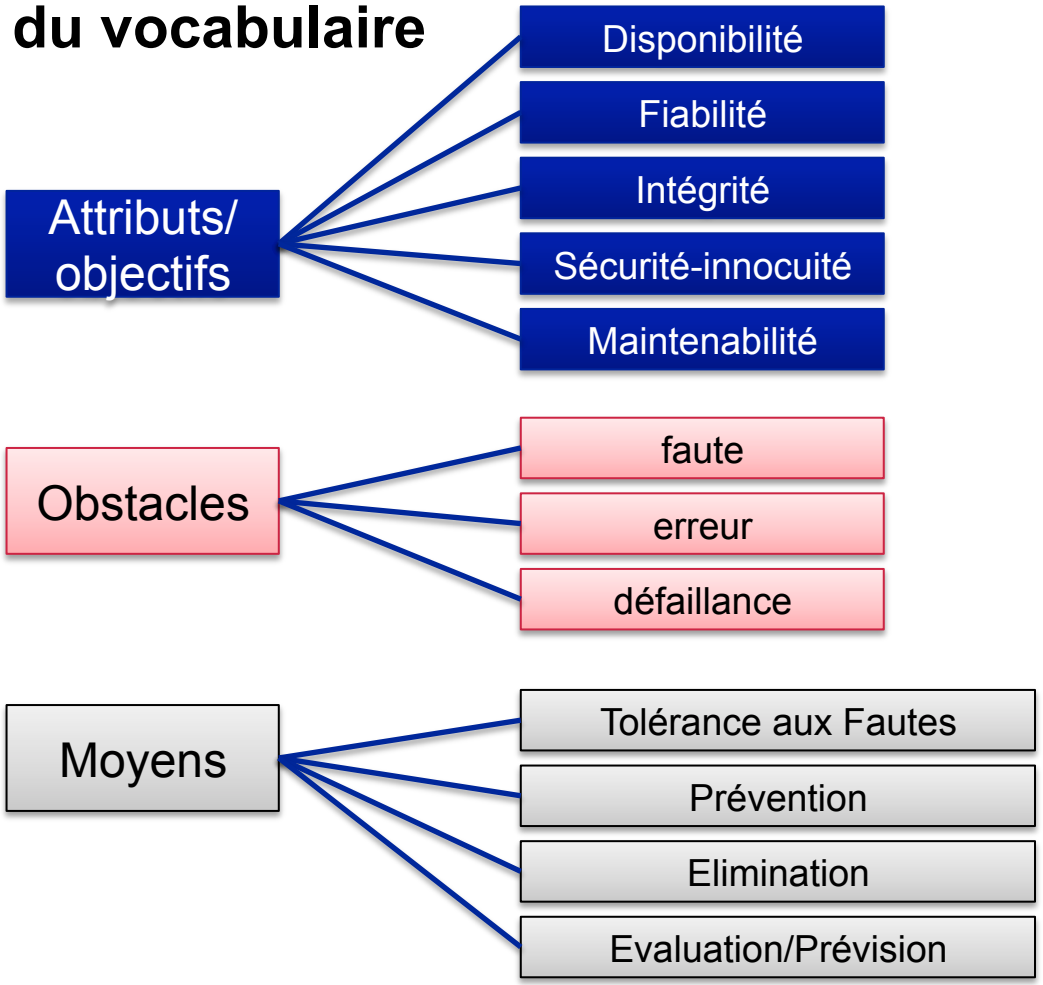
# Quelques mots sur l'ingénierie système motivation (suite)

- **Problèmes :**
  - Que doit contenir la description du service attendu ?
  - Comment identifie-t-on un dysfonctionnement dans les phases de conception (a priori) ?
  - Comment détermine-t-on ses conséquences, ses causes ?

- **Ingénierie Système et sureté de fonctionnement :**
  - Description du système (objectif et implémentation)
    - Analyse de la cohérence de la description
    - Etude de sa faisabilité
  - Prise en compte d'exigences spécifiques à la sûreté de fonctionnement
  - Guider le processus de développement, de validation, mais aussi décrire la vie opérationnelle du système.

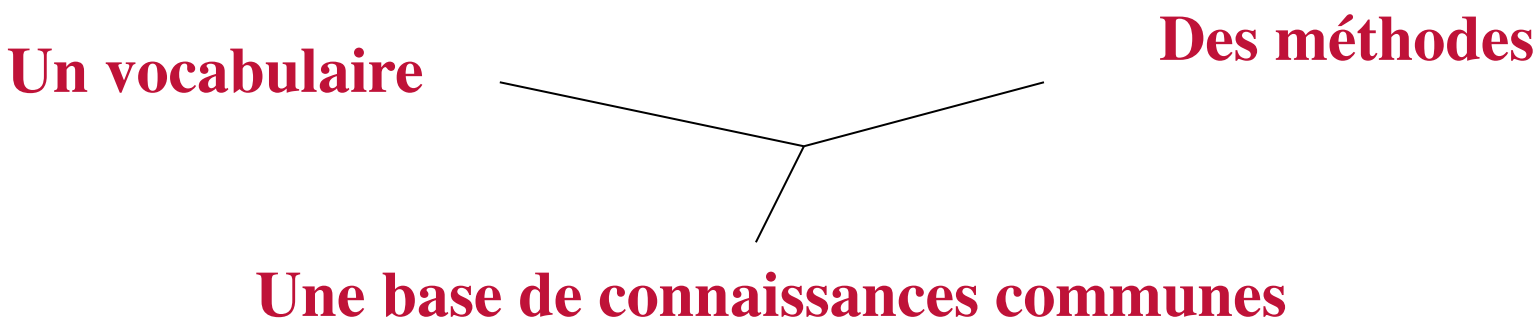
# Sureté de fonctionnement
# les nouveaux objectifs, nouvelles exigneces, nouvelles analyses

## Décomposition du vocabulaire



# Et spécifiquement pour les systèmes logiciels

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



# 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**

# Problème : vocabulaire et méthode ???

- Formulation des objectifs == texte (phrases)

⇒ Partage et communication plus difficile (cf cours ingénierie système)

⇒ 1ière étape : partager un vocabulaire pour caractériser les attributs

⇒ 2ième étape : disposer de modèles pour décrire / analyser / classer /comparer les implémentations

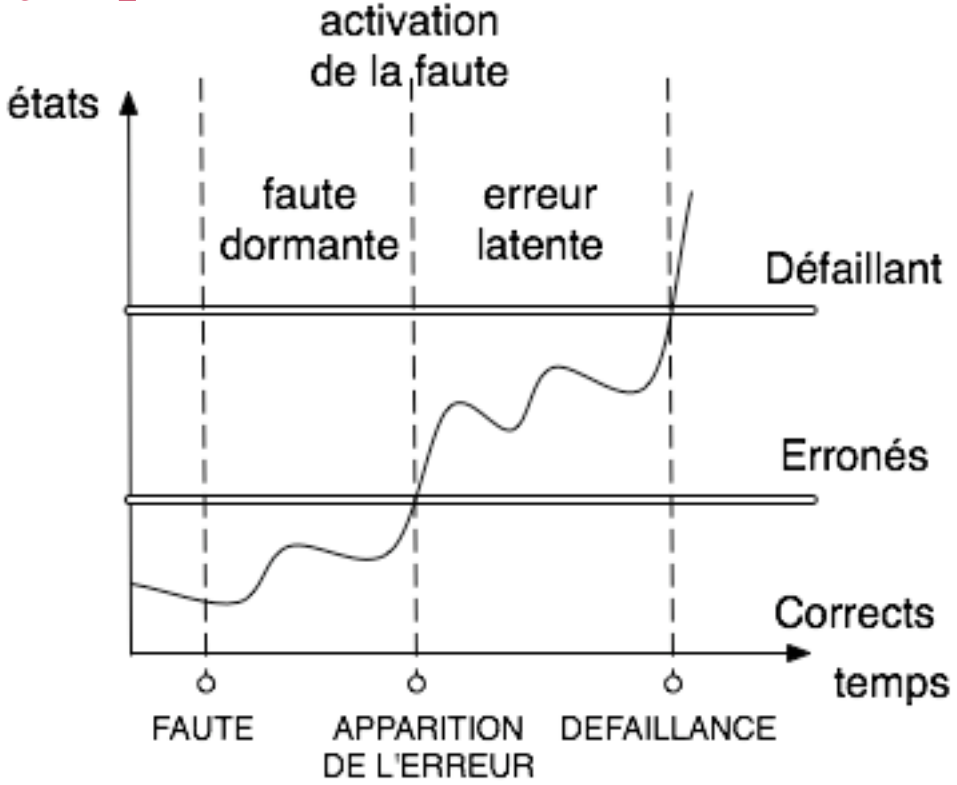
# Décrire les entraves (II)

- **Un vocabulaire centré sur l'écart à l'attendu :**
  - **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)
- **Le lien avec le danger :**
  - Si événement redouté == défaillance => danger recouvre la notion de faute
  - Sinon défaillance => danger => événement redouté (utile pour déterminer la gravité d'une défaillance)

# La chaîne faute/erreur/Défaillance

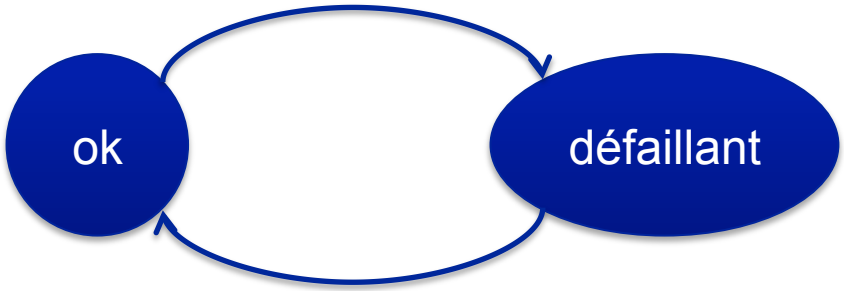
- **Evénements :**
  - Activation
  - Détection
  - Défaillance
- **Etats :**
  - Fautifs (non activés)
  - Corrects (sans faute)
  - Erronés (?)
  - Défaillants (KO)

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



# 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 disponibilité :

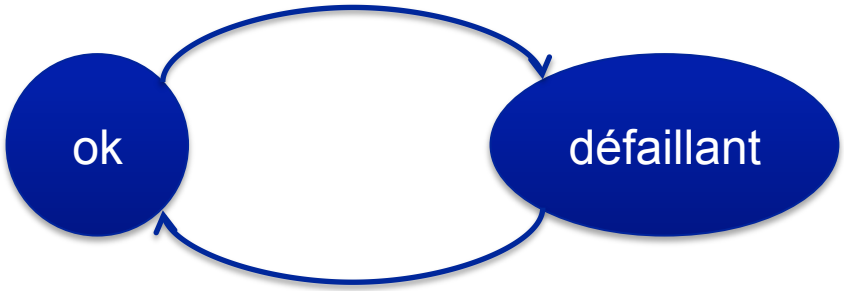



$$\frac{\text{Temps(OK)}}{\text{Temps(OK)+Temps(défaillant)}}$$

Ou Temps moyen avant transition vers défaillant

# Raffinement des attributs

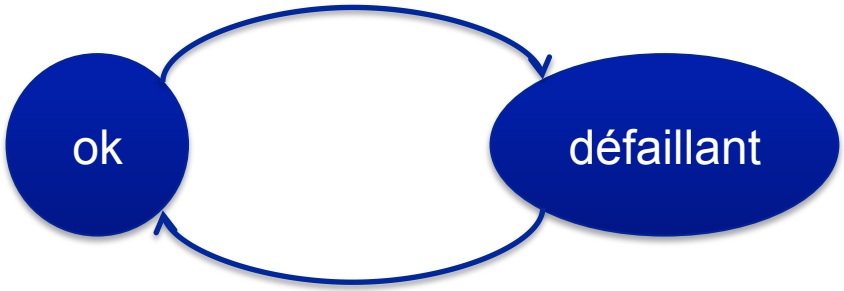
- **Idée : caractérisation des attributs par rapport à l'état du système (défaillant / non défaillant) et la nature du risque encouru**
- **Description usuelle de**
  - La fiabilité :




  - 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 :

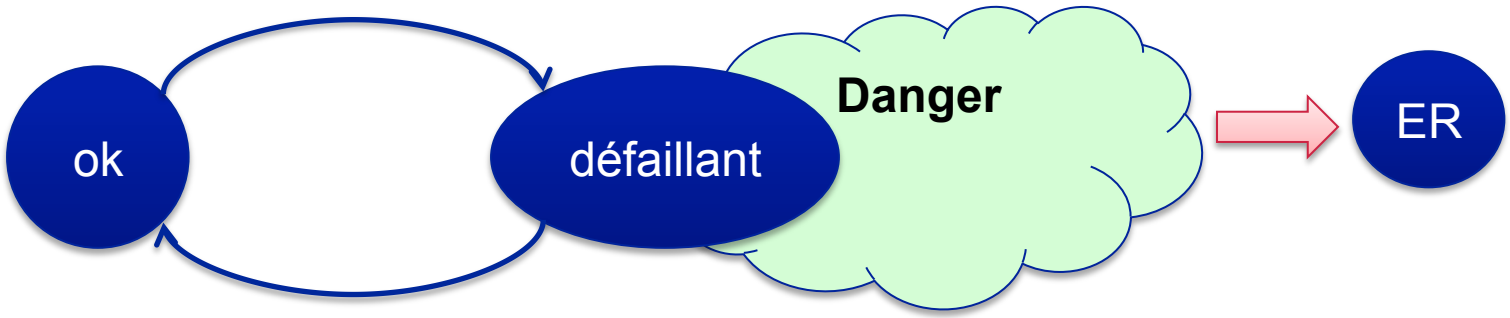



  - Le retour à l'état Ok conditionné par une intervention

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

# 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

# Raffinement des attributs
# 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 :
    - Contrainte sur la capacité à détecter / corriger des altération des données stockée
    - Mais aussi Contrainte de déterminisme comportemental

**Le cas des systèmes temps réels :**

**intégrité => capacité à respecter des échéances par construction du système.**

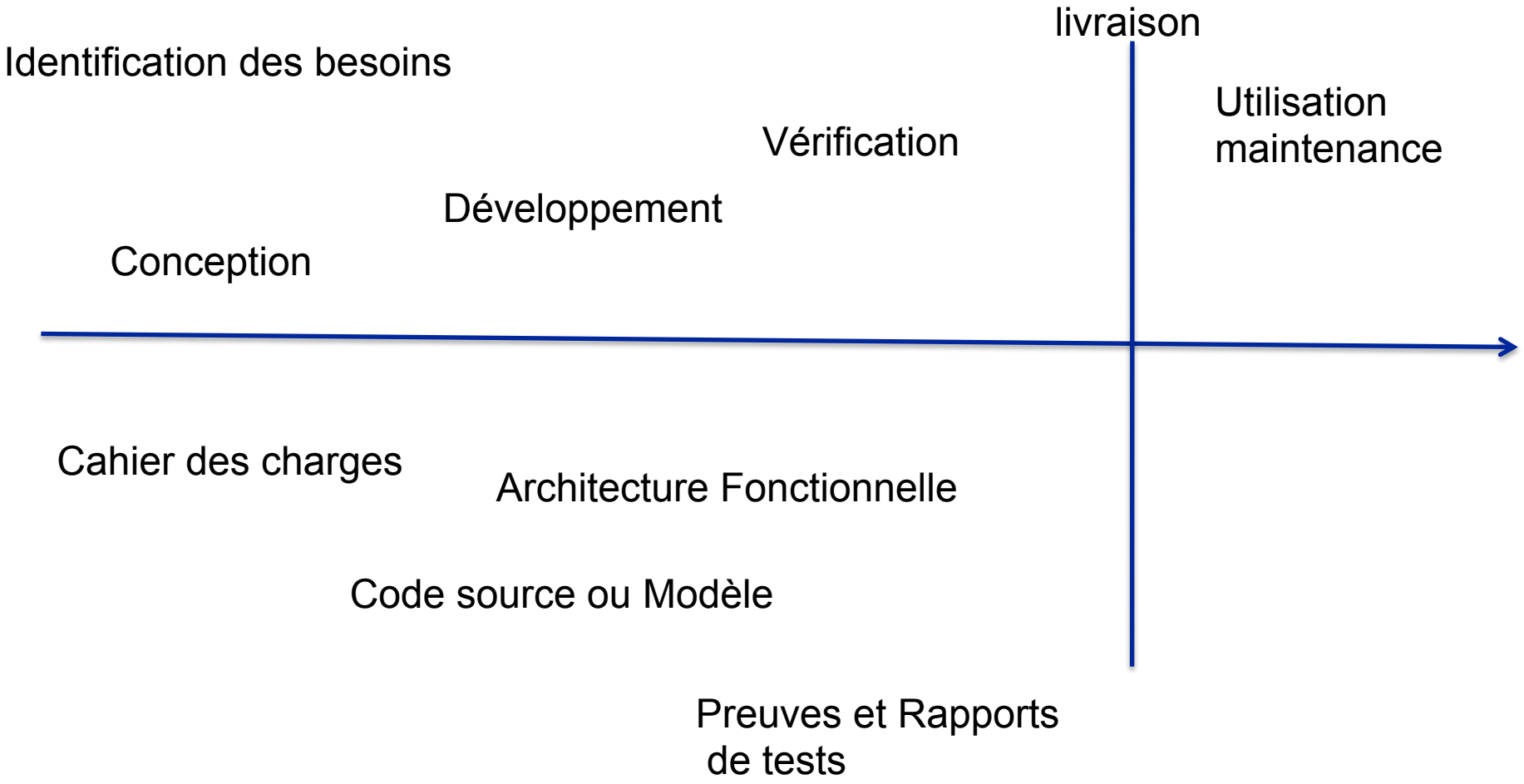
# Les moyens pour obtenir un niveau voulu de SdF

- **4 Approches complémentaires**
  - **Prévention des fautes :**
    Méthodes destinées à empêcher l'occurrence même des fautes
  - **Elimination 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

# 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



# 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
- **Evaluer/Prédire**
  - Méthodes et métriques permettant de quantifier (mesurer) ou qualifier (classer) un système vis à vis des attributs de la sûreté de fonctionnement

  Ex : Emuler 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

# Des méthodes, et des standards …

- **La sureté de fonctionnement : un objectif**
  - Impacte tous le niveau du développement en fonction du domaine
  - Compromis coût d'application / coût défaillance et conséquences
  - Contrainte réglementaire (ex. nucléaire/chimie lourde)

- **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 )

**Quoi et comment ?**

# Un exemple standard : 26262

- **Extrait white paper**

  *Part 6 of the standard specifically addresses product development at the software level. Requirements for the following development activities are specified:*

  - *Initialization of product development*
  - *Specification of software safety requirements*
  - *Software architectural design*
  - *Unit design and implementation*
  - *Unit testing*
  - *Software integration and testing*
  - *Verification of software safety requirements*

  *Methods defined by the ISO 26262 standard should be selected depending on the ASIL (automotive safety integrity level); the higher the ASIL the more rigorous the methods.*

# Bilan

**Ce dont vous avez besoin :**

1. **Des langages pour décrire**
2. **Des méthodes pour vérifier que la description correspond à un attendu défini à l'avance**

**Les standards fournissent, un vocabulaire, une syntaxe … mais ne doivent pas masquer l'objectif final**

**Objectif :**

**permettre d'établir un raisonnement structuré, avec des points de passage obligatoires pour une conception/un développement permettant d'évaluer ou d'assurer que les exigences de sûreté (EF ou ENF) sont remplies**

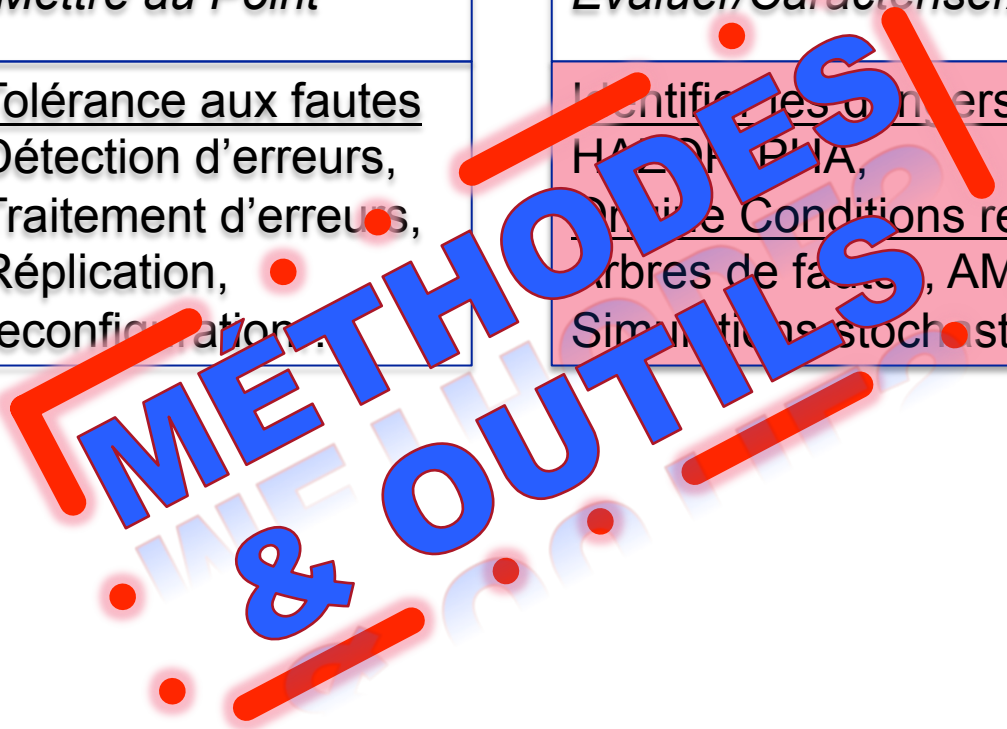
**Question :**

**peut on automatiser / assister ces étapes ?**

# Sûreté de fonctionnement
# Les moyens…

Exigence de Sûreté de Fonctionnement
(fiabilité, intégrité, disponibilité ….)

| *Concevoir/Développer/Mettre au Point* | | *Évaluer/Caractériser/Identifier* |
|---|---|---|
| Prévention<br>Ingénierie des exigences<br>Programmation contrôlée (MISRA C, Ada)<br>Formation des utilisateurs | Tolérance aux fautes<br>Détection d'erreurs,<br>Traitement d'erreurs,<br>Réplication,<br>reconfiguration… | Identification des dangers / risques<br>HAZOP, PHA,<br>Analyse Conditions redoutées<br>Arbres de fautes, AMDE,<br>Simulations stochastiques |
| Elimination<br>Spécification et vérification comportementale<br>Démarche de test (eg. injection de fautes) | | |



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

# De quoi a-t-on besoin ?

- **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

**Laquelle permet de mieux cerner les responsabilités ?**

- **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)

# Motivation du modèle
# Le cas des événements redoutés

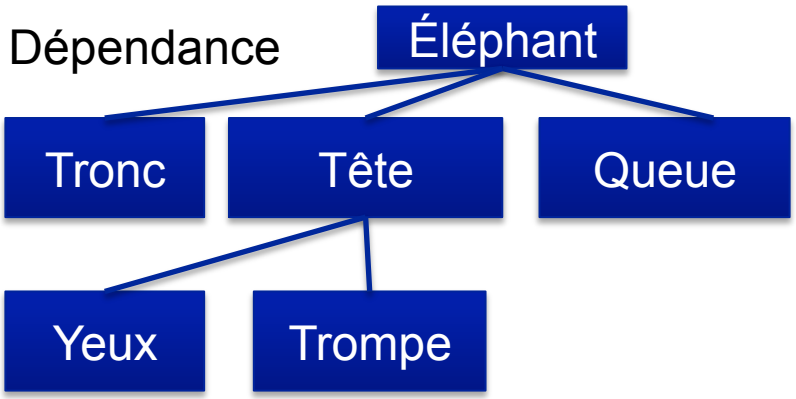
- **Problème : identification des événements redoutés**
- **Situation ==**
  - Un sujet => un élément du système ou un utilisateur
  - Description de l'événement = transition vers un état, une déviation par rapport à une norme, une action observable …
- **Il faut pouvoir**
  - Décrire le système ou ses parties
  - Décrire la dynamique du système (pour identifier les déviations ou l'accès à un état non souhaité)

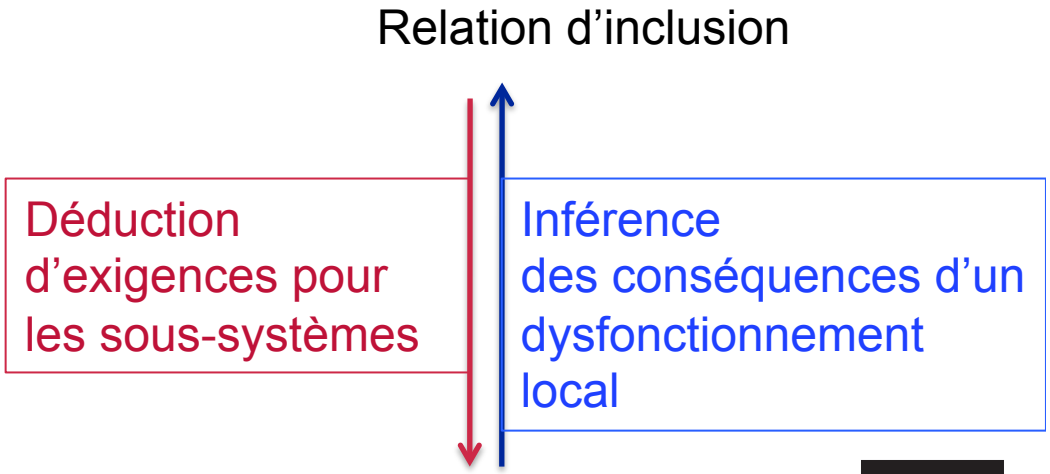
# Modèle à composant, Hiérarchie & Interfaces

- **Relations entre « blocs »**
  - **Inclusion** ?
  - **Collaboration** ?
  - Héritage ?

⇒ **Un dessin n'est pas un modèle, modèle = structuration d'une information interprétable**

- **Intérêt du modèle**



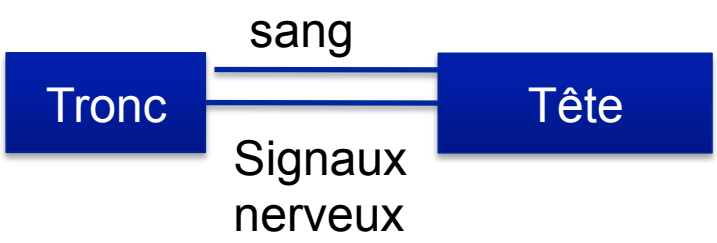



# Modèles à composant, Hiérarchie & Interfaces

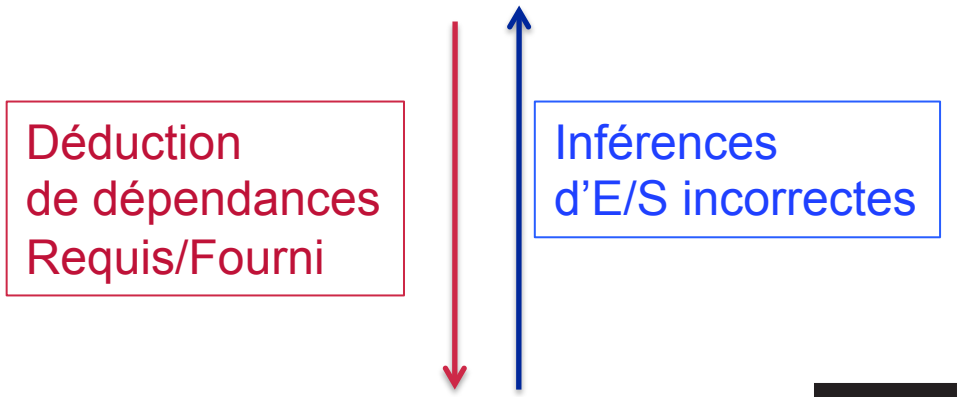
- **Relations entre « blocs »**
  - **Inclusion** ?
  - **Collaboration** ?
  - Héritage ?

⇒ **Un dessin n'est pas un modèle**

- **Intérêt du modèle**

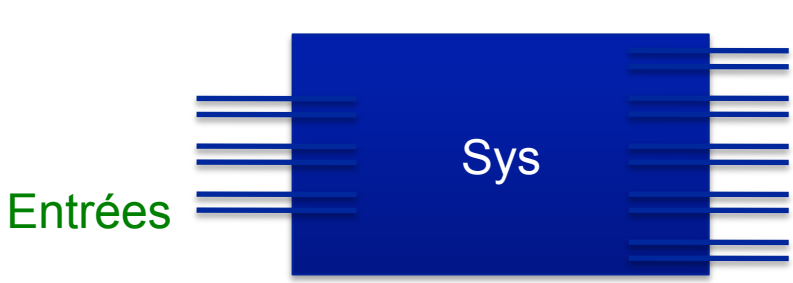



Relation de collaboration
bloc ~ fonction ~ transformation E/S

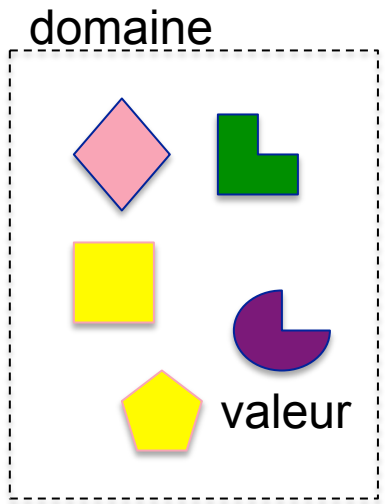


# Modéliser la circulation de données

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




Domaine nom




- **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)

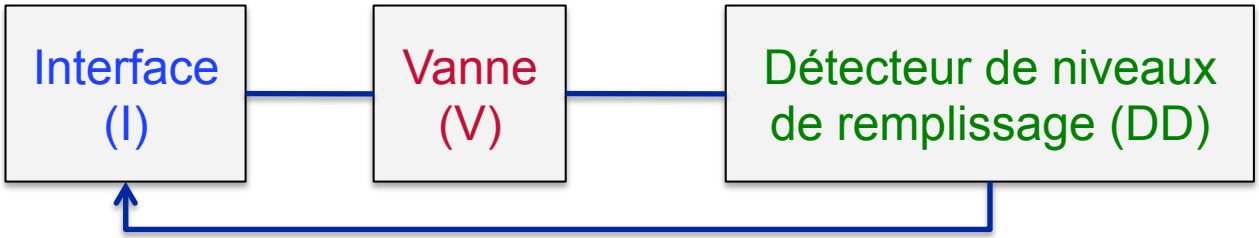
- **Modélisation de l'attendu == relation entrées/sorties**

# 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)**
- **Attention : parfois les relations entre variables ne sont qu'implicites (pas spécifiées)**

# La relation binaire dans un 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

# Quelles analyses pour les modèles clés-valeurs

- **Cas 1 : modèle de grande taille hiérarchique, sémantique simple = type des données échangées**
  - Objectifs éviter des échanges mal typés
  - Redondance de l'information sur les types
  - Saisie manuelle des données

  Action : vérification de la cohérence des types (Data flows)

  Les nœuds portent les types, arcs = connexions
- **Cas 2 : modèle avec sémantique avancée => le modèle doit être interprété = transformé**

  Exemple : lot de tâches temps réel et vérification temps de réponse

# 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 ambigüe : 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 $\wedge$
  - Ou : 2 paramètres noté $\vee$
  - 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 évaluée 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.

# Précision du vocabulaire et sens pratique

- Une formule qui possède des variables non quantifiée peut être vue comme un prédicat …

⇒ l'expression logique qu'elle représente peut être « réutilisée » dans d'autres formules

- Une formule logique ayant des variables libres peut servir à caractériser un modèle logique à travers l'ensemble des valuations pour lesquelles la formule est vraie, (i.e. le plus grand ensemble pour lequel la formule est valide)

# Une parenthèse sur le rôle des probabilités

- **Probabilité : caractérisation d'un phénomène aléatoire**
  - Moyen de représenter une incertitude maitrisée
  - Moyen de capturer une caractéristique d'une population
- **Variable discrète : probabilité** $P(x = v)$
- **Variable continue : distribution de probabilité (Probability density function)**

# Conclusion : Méthodes présentées par la suite

- **Ce qu'il faut retenir**
  - Incident
  - Sureté de fonctionnement = confiance justifié dans un comportement attendu => l'attendu couvre les incidents majeurs
- **La logique**
  - Formule, validité/satisfaction
  - Ensemble des valuations satisfaisant une formule
- **Tolérance aux fautes**
  - Les approches génériques (Redondance, diversification…)
  - Les exemples de mises en œuvres (TMR, Recovery Blocks …)
- **Méthodes d'analyses quantitatives**
  - Modèles stochastique structuraux simple (Block Diagrams)
  - Modèles avancés (MC)
- **Autre usages des méthodes formelles : (élimination)**
  - Modèle comportemental Automate étendu
  - Méthode d'exploration / analyse