Introduction à la sûreté de fonctionnement

Stanisław J. Piestrak

IRISA/INRIA, CAIRN Lab., Lannion
(en délégation de l'Université Paul Verlaine de Metz)
piestrak@univ-metz.fr

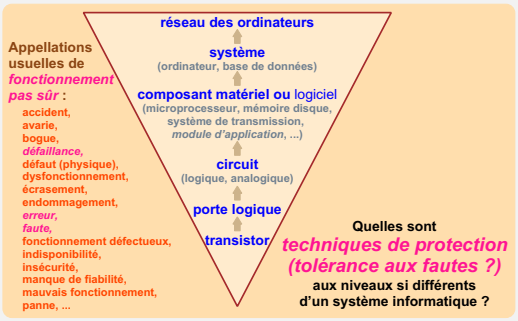
Club PME, "Sûreté de Fonctionnement"
Lannion, Bretagne, 17 juin 2011

Outline

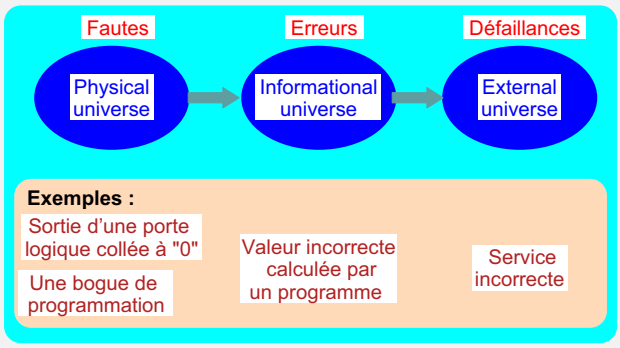
1. Introduction
2. Sûreté de fonctionnement (SdF) : notions de base
3. Évitement (prévention) de fautes
4. Tolérance aux fautes
   - Redondance matérielle
   - Redondance logicielle
5. Disponibilité de systèmes
6. Quelques systèmes commerciaux tolérants aux fautes

Fonctionnement sûr c-à-d. quoi ? Comment l'assurer ?

Fonctionnement sûr d'un système :
bon, correct, fiable, sans défauts, sans risque, ...

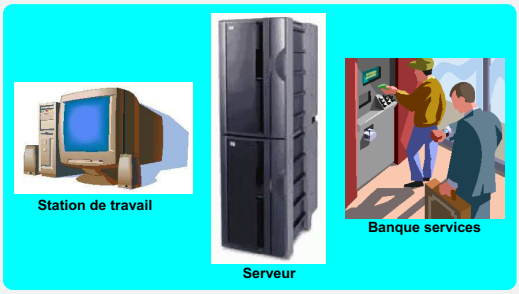


Entraves à la sûreté de fonctionnement (SdF) : terminologie



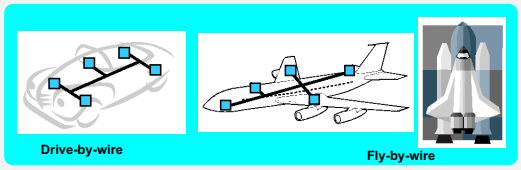
Besoin de circuits et systèmes numériques sûr (1)

- **Télécommunications :**
  téléphone, radio, télévision, réseaux informatiques, ...
- **Systèmes transactionnels :**
  banquaires, réservation de places, gestion des matériaux, ...



Besoin de circuits et systèmes numériques sûr (2)

- **Domaine médical :** équipements de mesure et d'analyse, stimulateurs cardiaques, suivi "en-ligne" des malades, bases de données, ...
- **Systèmes de commande-contrôle, de surveillance et de supervision en temps réel :**
  l'industrie manufacturière (nucléaires, chimiques, ...)
- **Transportation :**
  avions, trains, automobiles (assistance au freinage ABS ou à l'adhérence, ordinateurs de bord, ...)
- **Espace** ...



CAO = Catastrophe Aidée par Ordinateur ?

Conséquences fatales des défaillances de systèmes numériques :

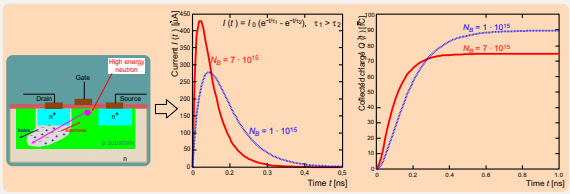
- **Inconvénients aux utilisateurs**
  - La non-disponibilité du système à long distance AT&T aux États-Unis (d'une durée de 9 heures en janvier 1990)
  - La non-disponibilité du réseau de cartes bancaires Carte Bleue en France (33 heures pendant un week-end en juin 1993, 33 millions d'utilisateurs affectés)
- **Pertes financières**
  - L'échec du premier vol de la fusée Ariane V
  - Le rappel de millions de processeurs Pentium d'Intel en 1996 (les pertes d'environ 500 millions de dollars)
- **Mise en danger de la santé ou de la vie des êtres humains**
  - L'accident d'un Airbus de Lufthansa atterrissant à Varsovie (1997)
  - Le mauvais fonctionnement d'un système d'urgence medicale à Londres (le nombre de victimes inconnu)

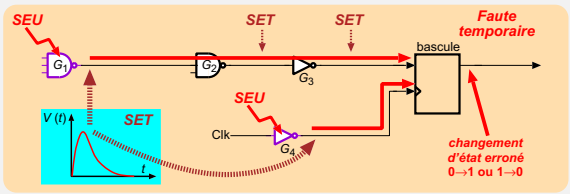
Les causes les plus communes de défaillances de systèmes informatiques

- **Fautes du matériel**
- **fautes d'alimentation**
- **virus informatiques**
- **catastrophes naturelles :**
  ouragans, tornados, inondations, tremblements de terre, incendies, ...
- **erreurs humaines**
- **erreurs de système d'exploitation**

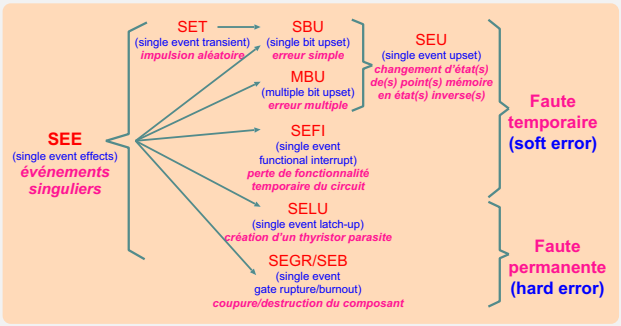
☞ **Défaillances de cause commune :** affectent plusieurs entités à partir d'un événement unique et ne résultent pas les unes des autres

Mécanisme de parution d'une faute temporaire soft error





Classification des événements singuliers causés par la radiation



Qu'est-ce-qu'on attend d'un système SdF ?

- **Calcul sans erreurs** et **integrité des données** *(aucune erreur permise)* :
  ex. **systèmes bancaires**
- **Opération continue** *(no interruption permise)* :
  ex. **systèmes de temps réel**
- **Haute disponibilité** *(courtes interruptions permises)* : ex. **serveurs**
- **Sans défaillances catastrophiques** *(arrêts sans conséquences graves autorisés)* : ex. **applications critiques (nucléaires, avioniques, spatiales, ferroviaires, automobiles, ...)**

Comment :

- Détecter (presque) toute faute,
- limiter les dégâts,
- faire de fautes transparentes et, en cas de faute,
- permettre la reprise, faire le compte rendu au niveau supérieur ?

Sûreté de fonctionnement (SdF) : définition

Sûreté de fonctionnement : aptitude d'un système à

- *délivrer un service de confiance* **justifiée**

ou

- *éviter des défaillances du service* **plus fréquentes ou plus graves qu'acceptable**
  (un critère pour décider si le service délivré est sûr).

Niveaux de criticité des défaillances : exemple d'un automobile

Criticité	Pire effet de défaillance	Exemples
Catastrophique	La mort d'une ou plusieurs personnes	Freinage électronique, direction électronique
Critique	La blessure d'une ou plusieurs personnes	Contrôle de traction, régulateur de vitesse, coussin gonflable
Significative	Le remorquage du véhicule	Contrôle moteur, contrôle d'assiette, suspension active
Mineure	Une gêne pour le conducteur	Limitation de vitesse, réglage de siège, rétroviseurs électriques, vitres électriques, système de diagnostic
Négligeable	La diminution du confort	Climatisation, essuie-phares, radio, téléphone

Source : Ch. Ziegler, "Sûreté de fonctionnement d'architectures informatiques embarquées sur automobile", thèse de doctorat, Rapport LAAS No 96289, Toulouse, 1996.

Sûreté de fonctionnement : attributs

☞ *Exprimer les propriétés* qui sont attendues du système

☞ *Apprécier la qualité* du service délivré

- **Disponibilité :**
  l'aptitude d'un système à être prêt à l'utilisation à un instant donné.
- **Fiabilité :** l'aptitude d'un système à être en état d'accomplir la continuité du service délivré, pendant un intervalle de temps donné (la mesure du temps jusqu'à défaillance).
- **Sécurité-innocuité :**
  la non-occurrence de conséquences catastrophiques pour l'environnement.
- **Maintenabilité :** l'aptitude aux réparations et aux évolutions (ne concerne que des systèmes réparables).

Sûreté de fonctionnement : moyens

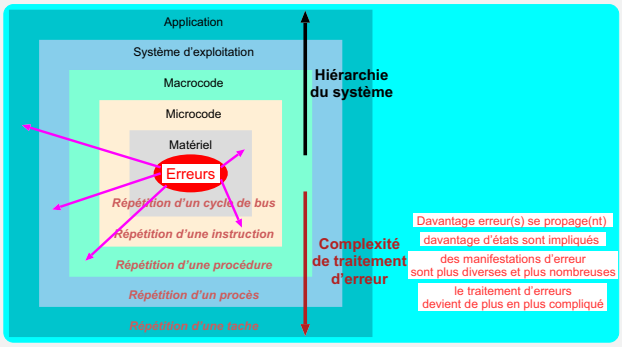
- **Prévention (évitement) de fautes**
  (*intolérance du système aux fautes*) :
  comment empêcher l'occurrence ou l'introduction de fautes

  ☞ *Diminuer la probabilité d'apparition des fautes et des erreurs*
- **Tolérance aux fautes :**
  comment fournir un service à même de remplir la fonction du système en dépit des fautes.

  ☞ *Limiter les effets de fautes*

Ces deux approches sont **complémentaires !**

Niveaux de confinement des erreurs



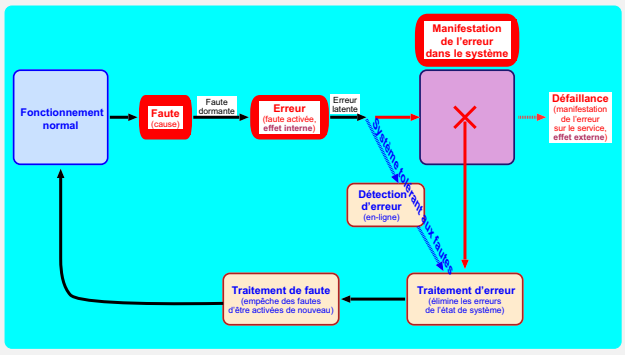
☞ **Comment arreter (restreindre) la propagation d'erreurs ?**

L'ignorance des fautes ou la tolérance aux fautes ?

Des effets de fautes non-détectées :

Systèmes SdF demandent **extensive et complète détection d'erreurs**

Fonctionnement d'un système exposé aux fautes



Tolérance aux fautes : mise en œuvre (1)

Traitement des erreurs : opérations destinées à éliminer les erreurs, si possible avant qu'une défaillance ne survienne.

- **Détection d'erreur**
- **Diagnostic d'erreur**
- **Recouvrement d'erreur :**
  - **Reprise**
  - **Poursuite**
  - **Compensation d'erreur**

Tolérance aux fautes : mise en œuvre (2)

Traitement des fautes :
opérations destinées à éviter qu'une ou des fautes ne soit activées à nouveau.

- **Diagnostic de faute :** détermine les causes des erreurs (localisation et nature de faute(s))

☞ *Objectif principal du traitement des fautes*, c'est

- **Passivation des fautes :**
  les actions destinées à empêcher une nouvelle activation des fautes. Les composants considérés comme fautifs sont retirés du processus d'exécution ultérieure.

  ☞ La **reconfiguration** : les composants non-défaillants permettent de délivrer un service *acceptable*, bien que *dégradé* :
  - abandon de certaines tâches
  - ré-allocation de certaines tâches aux composants restants

Évitement (prévention) de fautes

☞ **Empêcher l'occurrence ou l'introduction**

- de certaines fautes en général

  ou
- de fautes provoquant des erreurs difficile à traiter par des méthodes de tolérance aux fautes

Méthodes d'évitement (prévention) de fautes : exemples (1)

Programmation :

- Utilisation des *quelques langages spécifiques*
- selon *certaines règles de programmation*.

☞ Éviter des erreurs de programmation typiques et difficiles à détecter

Langage **MISRA C et ses 127 règles de programmation («bonnes pratiques»)** obligatoires et facultatives
(MISRA = Motor Industry Software Reliability Association).

Exemples :

- **Constantes (2) :** ne pas utiliser de constante octale
- **Expressions (6) :** ne pas tester l'(in)égalité sur des flottants
- **Contrôle des flux (16) :** interdiction d'utiliser des `goto`
- **Pointeurs et tableaux (7) :**
  ne pas utiliser d'opérateurs relationnels avec des pointeurs

Source : G. Antier, A. Bessemoulin, S. Delcroix et D. Renault, "Règles de Programmation", 15/06/2007, `web-serv.univ-angers.fr/docs/etudquassi/GL07_06.pdf`.

Méthodes d'évitement (prévention) de fautes : exemples (2)

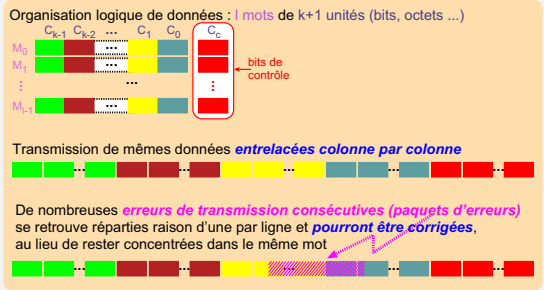
High-Integrity C++ Coding Standard Manual (HICCSM)

- **Règles sur la maintenabilité et la lisibilité du code :** suffixe pour nom de variable selon leur type (L→`long`, F→`float`, ...)
- **Règles sur la portabilité :** utiliser les bibliothèques standards C++ et pas les fonctionnalités propres à un système spécifiques.
- **Règles sur la fiabilité et la sûreté du code :** préserver l'encapsulation en privilégiant la création de classe `private`
- **Règles sur les instructions conditionnelles :**
  - ne modifier qu'une seule fois les variables d'itérations dans les boucles
  - dans les `switch` chaque `case` se termine par un `break`
  - un seul point d'entrée et de sortie.
  - ne pas utiliser de `goto`

Source : G. Antier, A. Bessemoulin, S. Delcroix et D. Renault, "Règles de Programmation", 15/06/2007, `web-serv.univ-angers.fr/docs/etudquassi/GL07_06.pdf`.

Méthodes d'évitement (prévention) de fautes : exemples (3)

- **Entrelacement des données**

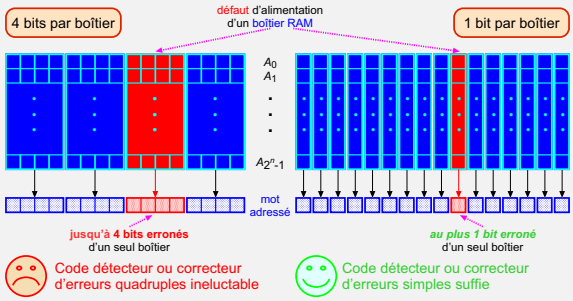


☞ Eviter des erreurs multiples dûes aux :

- perturbations de la transmission
- dégradation locale du support d'enregistrement (disque, CD ROM)

Méthodes d'évitement (prévention) de fautes : exemples (4)

- **Système de mémoire RAM utilisant plusieurs boîtiers**



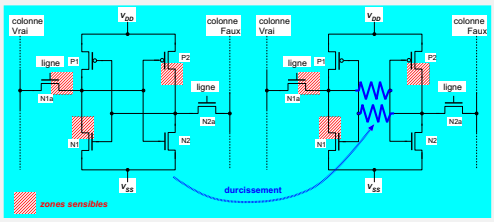
☞ Eviter certaines erreurs multiples provoquées par faute d'alimentation ou adressage

Réduction de la sensitivité aux radiations de composants électroniques

- Utilisation des **matériaux** avec le taux d'emission des particules α réduit
- Utilisation des **couches de blindage**
- **Modification de l'emballage**
  (écartement des zones sensitives de points de soudure)
- **Modifications du procéssus technologique**
- Utilisation des **technologies alternatives** (réalisation des circuits sur **matériaux isolants** plutôt que sur substrat semiconducteur) :
  - SOI (« Silicon On Insulator », SiO_2)
  - SOS (« Silicon On Sapphire », Al_2O_3)

Durcissement des composants (cellules de mémoire, bistables, portes)

– Augmentation de la capacité des nœuds sensitifs
– augmentation du pull-up gain
– **insertion de résistances** ou actifs dispositifs bouclés
– raccoursissement des lignes de colonnes (SRAM) ...



Cellule durcie de la mémoire SRAM

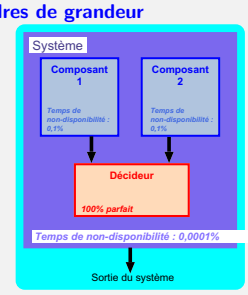
Pas de tolérance aux fautes sans redondance !

- **Redondance matérielle** (duplication, triplication)
- **Redondance d'information**
  (codes détecteurs d'erreurs : parité, cycliques CRC, ...
  codes correcteurs d'erreurs : Hamming, Reed-Solomon, turbo, ...)
- **Redondance temporelle** (répétition d'une opération)
- **Redondance logicielle** (tests d'acceptation, blocs de recouvrement, programmation en *N*-versions)

Achever un système fiable en utilisant composants non-fiables

Utilisation de la **redondance permet d'augmenter la fiabilité (disponibilité) d'un système par ordres de grandeur**

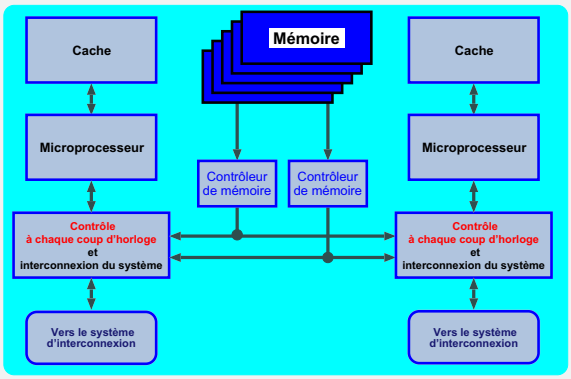
Hypothèses :
- 2 composants redondants identiques, dont
- les modes de fautes sont independents et détectables



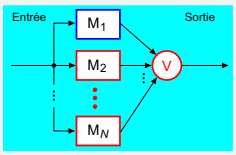
☺ Système dupliqué non-disponible seulement 0,0001% de temps

☹ Il est impossible de jamais achever's 100% de la fiabilité ou disponibilité

Processeur autocontrôlable à chaque coup d'horloge (lock-step)



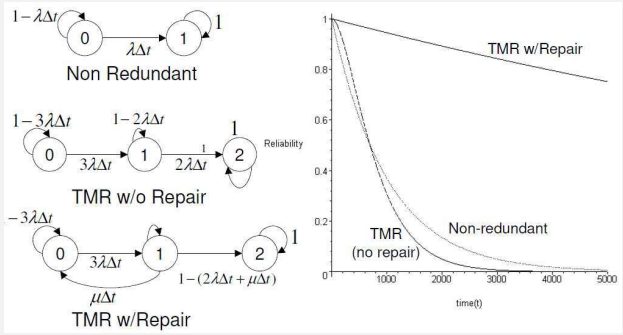
Redondance modulaire N-tuple (NMR) avec système de vote majoritaire



- $N \geq 3$ **différents composants (matériels ou logiciels)** traitent les mêmes entrées et produisent donc (en principe) les mêmes sorties
- **Collection des résultats**
- **Algorithme de vote** produit le résultat final
  - *simple* (vote à la majorité) ou
  - *complexe* (moyenne, moyenne pondérée, médiane, ...)
    (fiabilisation d'un résultat en combinant plusieurs résultats légèrement différents ; ex. : systèmes reposant sur des capteurs redondés)

Fiabilité d'un système triplé TMR (Triple Modular Redundancy)

Taux de défaillance d'un module $\lambda = const$, $R_M(t) = e^{-\lambda t}$



Tests d'acceptation (1)

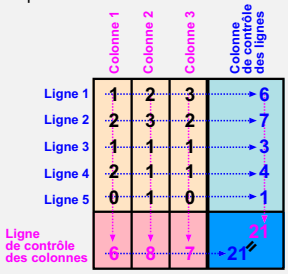
Tests d'acceptation : le test exécuté pour vérifier
- si les résultats obtenus sont acceptable, ou
- si l'exécution de programme n'a pas dévié du flot attendu.

☞ **Faible surcoût, faible couverture**

1. Contrôles de satisfaction aux exigences :
- algorithme de tri :
  contrôle de nombre des éléments triés et s'ils sont triés
- inversion d'une opération mathématique ;
  ex. $Y = \sqrt{X}$ vérifié par $X^* = Y^2$
- ...

Tests d'acceptation (2)

2. Contrôles de comptabilité : limités aux opérations transactionnelles qui impliquent simple opérations mathématiques
- ajouter la **somme de contrôle** (totale ou modulo)
- **comptabilité en partie double**
  (codifiée à Venise par Luca Pacioli au XVe ou XVIe siècle)



Tests d'acceptation (3)

3. Contrôles de vraisemblance : basées sur l'intervalle de grandeurs précalculées des variables, séquences attendues des états, ...
- *fourchettes de valeurs possibles*
  (ex : température de l'eau, vitesse maximale, état d'un compte bancaire)
- *incrément de valeur d'une variable* (écart maximal par rapport au résultat précédent dans un contrôle de processus continu = continuité des résultats) ; ex : accélération, ...
- *corrélation* entre les valeurs de variables différentes ou leurs incréments
- *séquences attendues des états* d'un commutateur téléphonique

☞ **Indépendance** des variables utilisées dans les programmes et les tests d'acceptation

Tests d'acceptation (4)

4. Détection d'erreurs d'éxécution (run-time checks)
- *Implémenté en matériel :*
  - division par 0
  - débordement
  - «underflow»
  - instruction inexistante
  - adresse mémoire inexistante
  - des violations de protections de segments-mémoire

  ☞ *Système de gestion d'exceptions*
- *Implémenté en logiciel :*
  - contrôle de type de variable
  - contrôle des valeurs d'indice de tableaux
  - ...

Tolérance aux fautes de conception : diversification fonctionnelle

☞ Éviter des **défaillances de mode commun**

Diversification fonctionnelle :

- on dispose *d'au moins un autre composant* à même d'assurer la tâche,
- conçu et réalisé *séparément à partir de la même spécification*.

Besoin :

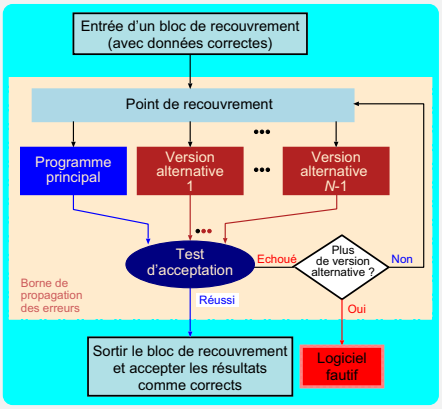
- d'au moins **deux variantes** d'un système
- d'un **décideur**, destiné à fournir un résultat supposé exempt d'erreur à partir des exécutions des variantes
- la **spécification commune** aux variantes
- les **points de décision** :
  - **quand** les décisions doivent être prises, et
  - **les données** traitées par le décideur

Approches aux diversification fonctionnelle de logiciel

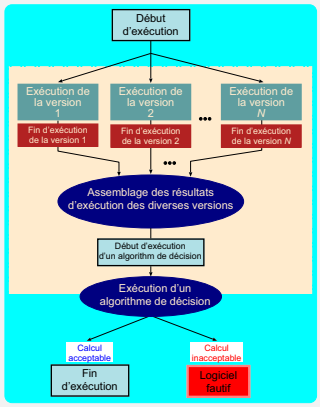
Trois approches différenciées selon *la méthode de traitement d'erreur* :

Blocs de recouvrement :	*recouvrement d'erreur par reprise*
Programmation en *N* versions :	*masquage de faute*
Programmation en *N*-autotestable :	*recouvrement d'erreur par compensation*

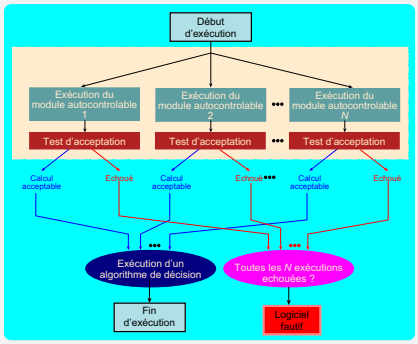
Blocs de recouvrement



Programmation en *N* versions



Programmation en *N*-autotestable



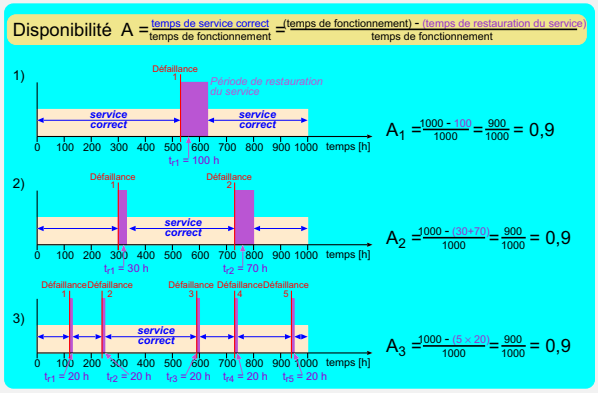
Chaque composant autotestable est constitué :

- soit de l'association d'une variante et d'un test d'acceptation,
- soit de deux variantes et d'un algorithme de comparaison.

Coût horaire de défaillance des systèmes informatiques en 2000

Domaine d'application	Coût [€/h]
Téléphonie mobile	40 000
Réservation aérienne	90 000
Transactions carte de crédit	2 500 000
Ligne d'assemblage automobile	6 000 000
Transactions boursières	6 500 000

Exemples de calcul de la disponibilité

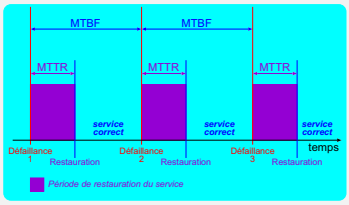


Qu'est-ce que la disponibilité ?

Disponibilité : la mesure de la délivrance d'un service correct par rapport à l'alternance *service correct-service incorrect*.

Elle s'exprime par :

$$A = \left(1 - \frac{MTTR}{MTBF}\right)$$



MTTR (Mean Time To Repair) : le *temps moyen de réparation* (d'intervention pour rendre le système à nouveau opérationnel suite à une défaillance). Il comprend la détection de la cause de défaillance, la passivation de faute et la remise en service.

MTBF (Mean Time Between Failures) : le *temps moyen de bon fonctionnement*

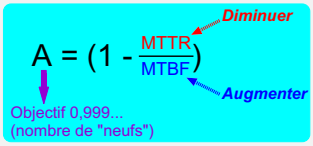
Comment améliorer la disponibilité ?

A = 1 revient :

- à **MTTR = 0** *(maintenance instantanée)*
  ou
- à **MTBF = ∞** *(fonctionnement sans défaillance)*

qui est statistiquement impossible.

☞ En pratique, plus le **MTTR est faible** et le **MTBF est élevé**, **meilleure est la disponibilité**.



Classes de disponibilité de systèmes

Classe de disponibilité d'un système :
le nombre de "neufs" dans le chiffre de disponibilité

Dans le cas général :

$$\text{Classe} = \left\lfloor \log_{10}\left(\frac{1}{1-A}\right)\right\rfloor$$

Classe (nombre de "neufs")	Disponibilité A	Non-disponibilité [minutes/an]	=
1	0,9	52560	36.5 jours
2	0,99	5256	87 h, 36 min
3	0,999	526	8 h, 46 min
4	0,9999	53	52 min, 33 sec
5	0,99999	5	5 min, 35 sec
6	0,999999	0.5	31.5 sec
7	0,9999999	0.05	3.15 sec

Quelques faits sur la disponibilité de serveurs dans des entreprises américans

Un serveur dans des entreprises américans éprouve moyennement par an :

- 3–5 défaillances,
- résultant en 10.0–19.5 heures de non-disponibilité non-planifiée
- Au moins 1 défaillance est serieuse,
- dont le temps de non-disponibilité est superieur de 4 heures,
- demandant l'intervention de plusieurs administrateurs de réseau,
- et potentiellement implique la perte de données.

☞ Le temps total de la non-disponibilité planifiée (maintenance, mise à jour, test) > 10 heures par an

Source : The Yankee Group, "2006 Global Server Reliability Survey", June 2006, Boston, MA, USA ; cité dans : Stratus Technologies, "How to ensure the availability of IT solutions in mission-critical government locations", June 2008.

Attributs de quelques applications critiques industrielles

Importantes pertes financières, risques pour la santé ou la vie humaine

Source : Stratus Technologies, "What to look for in mission-critical managed IT services. What's your exposure to losing business capability ?", White Paper, Sept. 2008.

Intel Inside, But ... What Is Inside ... z990 (IBM) ? (1)

Circuit	Méthodes de tolérance aux fautes
Processeur	Duplication avec comparaison
Registres	Parité
Bus interne (GX), interface externe (PCI)	Parité avec retransmission
Cache L1	Parité
Cache L2	SEC/DED ECCs : données (72,64), adresse (25,19), propriétaire (11,5)
RAM	SEC/DED ECC (140,128), nettoyage en continu (scrubbing), 2 puces de 2 bits de rechange
Clés d'accès mémoire	Parité, TMR avec un voteur et une clé de rechange
Clés d'accès cache	SEC/DED ECC (12,7)
Coprocesseur cryptographique : modulo exponentiation additionneur, ALU, unité SHA unité DES	Code résidu Parité Duplication avec comparaison
Oscillateur de système	Duplication avec commutation
Alimentation	Duplication (en parallèle)

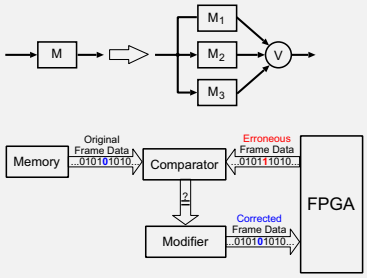
Intel Inside, But ... What Is Inside ... z990 (IBM) ? (2)

Autres méthodes de tolérance aux fautes :

- **Processeur de maintenance :** contrôle la reprise d'erreur au niveau d'instruction et sauvegarde toutes les données intermédiaires protégées par ECC
- **Processeur de rechange** est disponible pour remplacer un processeur quelconque défaillant en permanence
- **Remplacement des lignes défectueuses** dans tout les caches et mémoire principale RAM
- **Enregistrement de placements de toutes erreurs détectées :** pour effectuer le remplacement préventif de composants défectueux
- **Reconfiguration** dynamique en cas de fautes permanentes

Techniques de la SdF utilisées dans des circuits FPGA XILINX

- **Logique :**
  - sélective TMR (XTMR CAD tools) ou
  - duplication avec comparaison (lock-step)
- **Bloques de configuration :**
  - nettoyage en continu (scrubbing)
  - codes CRC ou Hamming
- **Mémoire :**
  - TMR ou
  - codes ECC de Hamming (Altera)

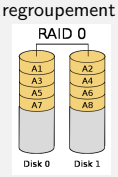


Taux d'erreurs GEO (geosynchronous orbit) de Virtex-II XQR2V6000

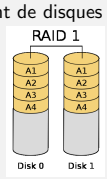
Mémoire de configuration	1.8 h
Bloques de mémoire	11.8 h
POR-SEFI	221 ans
SMAP-SEFI	181 ans

Systèmes de mémoires disques RAID (les niveaux standard)

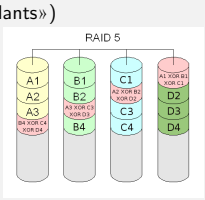
RAID = Redundant Array of Independent Disks
(«regroupement redondant de disques indépendants»)



Volume agrégé par bandes (entrelacement de disques)
Performance augmentée
(n disques durs travaillent en parallèle)
Fiabilité : la faute d'un seul disque entraîne la perte de toutes ses données
Coût : aucune redondance



Disques en miroir
chaque disque d'une paire contient à tout moment exactement les mêmes données
Fiabilité : la faute d'un disque de chaque paire est tolérée
Coût : au moins 100%



Volume agrégé par bandes à parité répartie
(n blocs de données ($n \geq 3$) et un bloc de parité)
Fiabilité : la faute d'un disque est tolérée
Coût : minimal (capacité totale de n disques sur un total de $n+1$ disques)
Inconvénients : pénalité en écriture du fait du calcul de la parité

Source : http://fr.wikipedia.org/wiki/RAID_(informatique)