Processus de maîtrise du développement d'un logiciel de sécurité à TECHNICATOME

- Référentiels
- Règles TA et Certification
- Processus – Lignes directrices
- Cycle de développement
- Caractéristiques générales d'un logiciel de sécurité
- Logiciels classés de niveau B - perspectives

Référentiels

Normes en vigueur dans nos différents domaines d'activité (nucléaire, ferroviaire):

- **RFS « LOGICIEL » II.4.1.a**
  - Logiciels classés de sûreté 1E
  - Logiciels classés de sûreté et non classés 1E
- **CEI 61226 (Classification des systèmes de C.C.)**
- **CEI 60880 (et 60880-2)**
  - Logiciels de catégorie A
- **CENELEC EN 50128**
  - SIL 3,4
  - SIL 1,2

Règles TA et Certification

Règles générales de Conception TA

- **Processus d'étude et de développement d'un logiciel RPG/C06/32**
- **IT E-G04 « Règles générales de développement des logiciels de niveau A des équipements du contrôle commande des chaufferies nucléaires de propulsion navale »**
- **IT E-G05 « Règles générales de développement des logiciels de niveau B des équipements du contrôle commande des chaufferies nucléaires de propulsion navale »**

Certificats

- **ISO 9001 AFAQ,**
- **EN 50128 SIL 4 CERTIFER (Logiciel PUPITRE du projet CESAR),**
- **Autorisation de divergence par l'IPSN des chaufferies nucléaires.**

Processus – Lignes directrices (1)

- **Approche système (Matériel et logiciel)**
  - Réutilisation maîtrisée d'éléments de base qualifiés
  - Simplicité fonctionnelle et comportementale
- **Mutualiser les investissements (humains / outils) entre les différents domaines d'activité (transport / nucléaire)**
- **Processus entièrement planifié et documenté, découpé en phases se concluant par des revues formelles**
- **Équipe de validation différente de l'équipe de développement,**
- **Adaptation de l'activité de vérification en fonction du niveau de sûreté du logiciel**

Processus – Lignes directrices (2)

- **Analyse du retour d'expérience sur les développements**
  - Pour déterminer les améliorations à apporter au processus de développement
  - Pour vérifier la non régression du processus de développement
  - Savoir faire évoluer les méthodes et outils de manière ciblée vis à vis des risques réels et des anomalies rencontrés
- **Optimiser l'effort de développement d'un système**
  - Prise en compte de contraintes économiques de + en + présentes
  - Hypertrophier une démonstration (par exemple sur le logiciel) c 'est risquer de négliger d'autres aspects
- **Maîtriser la gestion de configuration des systèmes multi-applications**
  - Notion de standard
- **Veille technologique (universitaires, recherche, autres industriels)**

Cycle de développement (1)

Classique cycle en V

- **Planification du développement (PQL, PGC, PVL) et règles de développement - documents génériques :**
  - Plan de Gestion de Configuration
  - Règles de spécification
  - Architecture générique et de règles conception
  - Règles de codage (sous-ensemble du C-ANSI)
- **Spécification semi-formelle du logiciel (méthode SA-RT, Outil TeamWork),**
- **Conception semi-formelle (méthode SD, Outil TeamWork),**
  - Architecture du logiciel – Composants fonctionnels, dictionnaire des données, flots de contrôle et flots de données
  - Description des composants élémentaires Modules / Actions
- **Génération automatique depuis le modèle SD et Codage des actions (chaîne Microtec C-ANSI),**

Cycle de développement (2)

Activités de vérification et tests

- **Indépendance individuelle de toute activité de vérification et test**
- **Vérifications des documents (plans, spécifications, conception préliminaire, spécifications et résultats des tests…)**
- **Vérifications des composants logiciels – conception détaillée et code source**
  - Vérification manuelle formalisée (grilles de contrôle)
  - Métriques de complexité (Logiscope Code-Checker)
  - Respect règle de codage (Logiscope Rule-Checker)
- **Tests unitaires**
  - Test boite noire sur machine hôte ET cible (ATTOL Unitest)
  - Mesure de couverture structurelle (ATTOL Coverage)
- **Intégration du logiciel sur le calculateur cible** (émulateur + banc de test semi-automatique ATLG),
- **Validation du logiciel sur le calculateur cible** (ATLG)
  - Équipe indépendante de celle de développement

Cycle de développement (3)

Activités transverses

- **Suivi du développement des composants logiciels assuré par un jeu de fiches**
- **Gestion en configuration des composants et articles du logiciel dès la phase d'intégration (outil Continuus)**
- **Revues formelles de fin de phase (planification, spécification, conception, réalisation, intégration, validation)**
- **Réalisation de matrices de traçabilité entre les produits des phases**

Caractéristiques générales d'un logiciel de sécurité (1)

- **Faible niveau de complexité**
  - Taille limitée (10000 a 15000 lignes de code source)
  - Simplicité du comportement (déterminisme)
    - Application essentiellement « sans état »
    - Asynchronisme par rapport aux événements externes
    - Mono tâche cyclique contrôlée par un Chien de garde indépendant
    - Actions séquentielles « déterministes »
    - ITs utilisables de manière bornée et sans synchronisation avec le cycle de base
- **Pas de composants COTS, ou alors dans un cadre de contraintes sévères (notamment pas de système d'exploitation ou moniteur temps réel)**

Caractéristiques générales d'un logiciel de sécurité (2)

- **Logiciel de base / logiciel d'application**
  - **Composants de base réutilisés:**
    - Gestion matériel carte CSG,
    - Gestion cartes d'interface sur bus sécurisé,
    - Traitements utilitaires communs.
- **Mécanismes de surveillance (logiciels et matériels):**
  - Surveillance des chemins d'exécution
  - Programmation défensive
  - Traitements en « 2 bords » (partitionnement de la mémoire et vote des données redondantes)
  - Intégrité du logiciel (Checksum et code de Hamming)

Logiciels classés de niveau B - perspectives

- **Référentiel :**
  - IT E-G05
  - RFS
- **Technologie à base de COTS :**
  - Matériels cible standards
  - O.S. temps réel
- **Approche orientée objet :**
  - UML
  - C++
- **Génération automatique de code**