T2 – Systèmes Temps-Réel et Sûreté de Fonctionnement

Questionnaires à Choix Multiples

Dylan MOLINIE

13 novembre 2019

QCM 1

1. Un système est compliqué lorsque :
   - Environnement dynamique ;
   - Exécution concurrente ;
   - Compilation hétérogène.
2. Des critères de confiance en un système sont :
   - Déterminisme ;
   - Partitionnements temporel et spatial ;
   - Certifiable.
3. La notion de *Déterminisme* implique (entre autres) :
   - Résultats uniques et invariants ;
   - Chaîne causale entre cause et conséquence ;
   - Différentes implémentations donnent les mêmes résultats.
4. Le **partitionnement spatial** est le regroupement des composants dans un même espace mémoire pour assurer de manière sûre qu'un composant impacte un autre composant.

   → *NON*

5. Le **partitionnement temporel** est le partage du processeur pendant un temps prédéfini. L'objectif est d'assurer qu'un composant ne perturbe pas temporellement un autre composant.

   → *OUI*

6. Dans la réglementation **avionique**, la **validation** est l'activité pour répondre à la question « Est-ce que le logiciel est bien réalisé ? »

   → *NON*

7. C'est le développeur qui analyse et définit le niveau de criticité d'un composant.

   → *NON*

8. La certification DO-178C explique comment développer un système.

   → *NON*

9. La certification DO-178C impose les méthodes de développement.

   → *NON*

10. Dans une approche de développement sous DO-178C, des exemples d'éléments à produire sont :
    - Un ou plusieurs documents *Software Requirement Specification* (SRS) contenant les exigences de haut niveau ;
    - Les plans de développement qui décrivent comment je travaille ;
    - Un ou plusieurs documents *Software Application Design* (SAD) contenant l'architecture de mon système ;
    - Les codes sources avec la traçabilité vers les exigences ;
    - Les cas de tests et les scénarios de tests ;
    - Un ou plusieurs documents *Software Detail Design* (SDD) contenant les exigences de bas niveau.
11. Le *Configuration Management Plan* est le document qui permet de définir l'ensemble des paramètres de mon logiciel.

    → *NON*

12. L'exigence suivante est testable : « MonDij doit couper le courant s'il dépasse 16 Ampères. »

    → *NON*

QCM 2

1. Pour réaliser une partition mémoire, le système d'exploitation peut utiliser une MPU (*Memory Protection Unit*) ou une MMU (*Memory Management Unit*) pour protéger les accès.

   → *OUI*

2. Dans une approche *Event-Triggered*, le sémaphore est un exemple d'évènement déclenchant une exécution.

   → *OUI*

3. Dans une approche *Event-Triggered*, nous pouvons faire des hypothèses sur l'ordonnancement pour concevoir notre système.

   → *NON*

4. Dans une approche de conception *Time-Triggered*, les données sont visibles dès qu'elles sont produites.

   → *NON*

5. Dans une approche *Time-Triggered*, je connais dès la conception le nombre de données disponibles à chaque cycle d'exécution lorsque ces données échangées entre deux tâches ; je connais donc par construction les temps de traitements de bout en bout.

   → *OUI*

6. Une approche *Time-Triggered* nécessite l'utilisation de sémaphores et de mutex pour échanger des données entre tâches.

   → *NON*

7. Dans une approche *Time-Triggered*, des *deadlocks* peuvent survenir à l'exécution.

   → *NON*

8. Pour faire des systèmes réactifs, nous ne pouvons utiliser que l'approche *Event-Triggered*.

   → *NON*

9. Pour concevoir mon système, j'utilise :
   — UML car les représentations graphiques sont normées ;
   — Des diagrammes de séquences pour représenter les *threads*, les sémaphores et les appels de fonctions (dans le cas d'une approche *Event-Triggered*) ;
   — Des diagrammes temporels pour représenter les *threads*, les sémaphores et les appels de fonctions (dans le cas d'une approche *Time-Triggered*).
10. AADL est un langage pour décrire l'architecture statique multi-tâches d'un programme (tâches, propriétés temporelles).

    → *OUI*

11. AADL permet de décrire l'utilisation des sémaphores.

    → *NON*

12. SCADE est un langage pour décrire des applications multi-tâches.

    → *NON*

13. Il n'est pas nécessaire de borner les boucles dans une conception temps réel multi-tâches sûre de fonctionnement.

    → *NON*

14. Dans une approche *Event-Triggered*, nous pouvons faire des hypothèses sur l'ordonnancement pour concevoir notre système.

    → *NON*

15. *Pas d'assertion.*
    — Un Système d'Exploitation s'exécute en mode privilégié ;
    — Pour passer d'un mode *user* à un mode *privilégié*, des appels systèmes sont mis en place dans les API du Système d'Exploitation.