Méthodes et Outils pour la qualité logicielle

TP 3 – DESIGN REVIEW

QUESTION 1

- Chaque fichier (excepté *Main.c*) comporte deux erreurs par rapport aux documents ci-dessus. Rappelez les objectifs du « Design Revue » dans votre compte-rendu et expliquez chacune des erreurs. Corrigez ensuite le code source.

L'objectif du « Design review » est d'évaluer la conception logicielle et d'identifier les opportunités d'amélioration. Les avantages sont la prévention des problèmes de production, la communication efficace et l'amélioration continue de la qualité.

1ere erreur (fichier Cashregister.c) :

```
268  /*
269   * The end session key has been pressed
270   */
271  void Cashregister_end(void)
272  {
273    if (state == state_Idle)
274    {
275      generateTicket();
276      endSession();
277      state = state_Idle;
278    }
279  }
```

DDR_00300
Name: End application active state (empty cart)
Text: If the end session key has been pressed and the Cashregister application is in the active state then Cashregister_end shall empty out shopping basket data
Covers: HLR_00300
Function: Cashregister_end

La fonction Cashregister_end fait tout ce que les « detailed design requirements » demandent mais cela se passe quand l'état est state_Idle. Nous avons donc modifié la condition pour que les actions se déroulent quand l'état est state_Active :

```
268  /*
269   * The end session key has been pressed
270   */
271  void Cashregister_end(void)
272  {
273    if (state == state_Active)
274    {
275      generateTicket();
276      endSession();
277      state = state_Idle;
278    }
279  }
```

2eme erreur (fichier Cashregister.c) :

SYS_0600
Name: Hardware limit of products number
Text: A maximum of 50 products can be bought at one time.

```
18    #define MAX_PRODUCTS_IN_BASKET 10U
```

Le nombre maximum de produits doit être de 50, or le #define MAX_PRODUCTS_IN_BASKET est configuré à 10. Nous avons donc modifié la valeur :

```
18    #define MAX_PRODUCTS_IN_BASKET 50U
```

3eme erreur (Specialoffer.c) :

HLR_1300
Name: Inform end session
Text: Upon the completion of a session, the cashregister application user interface shall inform the user that the session has ended
Covers: SYS_00400
Module: Specialoffer

L'utilisateur n'est pas informé de la fermeture de la session. Nous avons donc rajouté la fonction qui permet de fermer la session et un message pour en informer l'utilisateur :

```
40        return price;
41
42        endSession();
43        Userinterface_show("The session has ended");
44    }
45
```

4eme erreur (Userinterface.c) :

HLR_01500
Name: User input
Text: The cashregister application user interface shall capture all possible cash register user operations
Covers: ?
Module: Userinterface

Nous ne savons pas comment corriger cette erreur.

5eme erreur (Productdatabase.c) :

HLR_01900
Name: Product list
Text: The local database of products should be:
- "Coconuts", ID 12345U
- "Lychees", ID 12346U
- "Kiwis", ID 12347U
- "Pears", ID 12348U
- "Pomegranates", ID 12349U
- "Watermelons", ID 12350U

Covers: ?
Module: Productdatabase

```
16  static struct Product ProductList[MAX_PRODUCTS] = {
17      { "Coconuts", 12340U, 180U, TEN_PERCENT_OFF },
18      { "Lychees", 12346U, 250U, NO_OFFER },
19      { "Kiwis", 12347U, 50U, THREE_FOR_ONE_EURO },
20      { "Pomegranates", 12348U, 100U, NO_OFFER },
21      { "Pears", 12349U, 199U, BUY_ONE_GET_ONE_FREE },
22      { "Watermelons", 12350U, 350U, NO_OFFER }
23  };
```

Les ID des produits ne correspondent pas.

```
16  static struct Product ProductList[MAX_PRODUCTS] = {
17      { "Coconuts", 12345U, 180U, TEN_PERCENT_OFF },
18      { "Lychees", 12346U, 250U, NO_OFFER },
19      { "Kiwis", 12347U, 50U, THREE_FOR_ONE_EURO },
20      { "Pomegranates", 12349U, 100U, NO_OFFER },
21      { "Pears", 12348U, 199U, BUY_ONE_GET_ONE_FREE },
22      { "Watermelons", 12350U, 350U, NO_OFFER }
23  };
```

Question 2

L'objectif de cette partie est de corriger le code afin d'assurer sa conformité avec les règles de codage de la norme MISRA C:2012.

Pour cela, lancez une analyse statique sur l'ensemble des fichiers en utilisant l'outil « CPP Check » en suivant la procédure décrite au TP1 et corrigez l'ensemble des erreurs détectées. Expliquez dans votre compte-rendu les corrections faites en vue de respecter les règles de MISRA C :2012 et faites ces corrections dans le code.

Nous avons quelques erreurs :

Fichier	Sévérité	Ligne	Résumé	Since date
Cashre...				
Cas...	avertissement	153	%d in format st...	06/02/2024
Cas...	erreur de style	54	The scope of th...	06/02/2024
Cas...	erreur de style	79	The scope of th...	06/02/2024
Cas...	erreur de style	193	The scope of th...	06/02/2024
Cas...	erreur de style	194	The scope of th...	06/02/2024
Cas...	erreur de style	146	MISRA 11.8 (Re...	06/02/2024
Cas...	erreur de style	10	MISRA 21.6 (Re...	06/02/2024
Main.c				
Mai...	erreur de style	23	MISRA 17.7 (Re...	06/02/2024
Mai...	erreur de style	30	MISRA 17.7 (Re...	06/02/2024
Mai...	erreur de style	9	MISRA 21.6 (Re...	06/02/2024
Product...				
Pro...	erreur de style	61	The scope of th...	06/02/2024
Pro...	erreur de style	16	MISRA 8.9 (Advi...	06/02/2024
Userint...				

Nous avons corrigé le main.c :

```c
/*
 *    MISRA C:2012 Cash Register project
 * ======================================
 * File Path     : Main.c
 * Author        : M.W.Richardson
 * Date          : 16/05/13
 * Copyright     : (c) 2013 Liverpool Data Research Associates
 */

#include "Misrac_types.h"
#include "Userinterface.h"

/*
 * Simple main that loops
 * until the character 'q' is pressed
 * Then it exits
 */
LDRA_int32_t main(void)
{
  LDRA_char_t theChar = '0';

  (void)printf("LDRA MISRA C:2012 Cash Register\n");
  Userinterface_help();

  /* Parse characters received from the keyboard */
  while (theChar != 'q')
  {
    theChar = (LDRA_char_t) getchar();
    (void)Userinterface_parse(theChar);
  }
  return 1;
}
```

Par exemple, dans Userinterface.c, nous avons retiré la variable « code » :

```
38  {
39    LDRA_uint32_t index = 0U;
40
41    (void)printf("\n");
42    Cashregister_start();
43    while (index < (10U + ((LDRA_uint32_t) rand() % 50U )))
44    {
45      if (0U == (LDRA_uint32_t) rand() % 7U)
46      {
47        Cashregister_cancel();
48      }
49      else
50      {
51        Cashregister_barcode(12343U + (LDRA_uint32_t) rand() % 7U);
52      }
53      index++;
54    }
55    Cashregister_end();
56  }
```

Nous avons aussi rajouté des (void) devant certain printf dans Userinterface.c :

```
64  void Userinterface_help(void)
65  {
66    (void)printf("choices are :\n");
67    (void)printf("            : 0-9 to compose barcode\n");
68    (void)printf("            : b to enter barcode\n");
69    (void)printf("            : c to cancel last product\n");
70    (void)printf("            : r to do random shopping\n");
71    (void)printf("            : s to start\n");
72    (void)printf("            : e to end\n");
73    (void)printf("            : n for Coconuts\n");
74    (void)printf("            : p for Pears\n");
75    (void)printf("            : l for Lychees\n");
76    (void)printf("            : k for Kiwis\n");
77    (void)printf("            : g for Pommegranates\n");
78    (void)printf("            : w for Watermelons\n");
79    (void)printf("            : q to quit program\n");
80  }
```

Nous avons modifié un %d en %u dans Cashregister.c :

```
151       (void) sprintf(msgString, "%12s %2u at %6.2f%c  %6.2f",
```

Dans le fichier Productdatabase.c, nous avons modifié quelques lignes :

```
63    if (bProduct != NULL_POINTER)
64    {
65      int i = 0;
66      while (i < (void)MAX_PRODUCTS)
67      {
68        if (CountedProductList[i].itsProduct->barcode == bProduct->barcode)
69        {
70          theCountedProduct = &CountedProductList[i];
71          i = MAX_PRODUCTS; /* found it so exit loop */
72        }
73        else
74        {
75          i++;
76        }
77      }
78    }
79    return theCountedProduct;
80  }
```

Nous avons rajouté un void avant le « static struct Product » dans Productdatabase.c :

```
16  void static struct Product ProductList[MAX_PRODUCTS] = {
17      { "Coconuts", 12345U, 180U, TEN_PERCENT_OFF },
18      { "Lychees", 12346U, 250U, NO_OFFER },
19      { "Kiwis", 12347U, 50U, THREE_FOR_ONE_EURO },
20      { "Pomegranates", 12349U, 100U, NO_OFFER },
21      { "Pears", 12348U, 199U, BUY_ONE_GET_ONE_FREE },
22      { "Watermelons", 12350U, 350U, NO_OFFER }
23  };
```

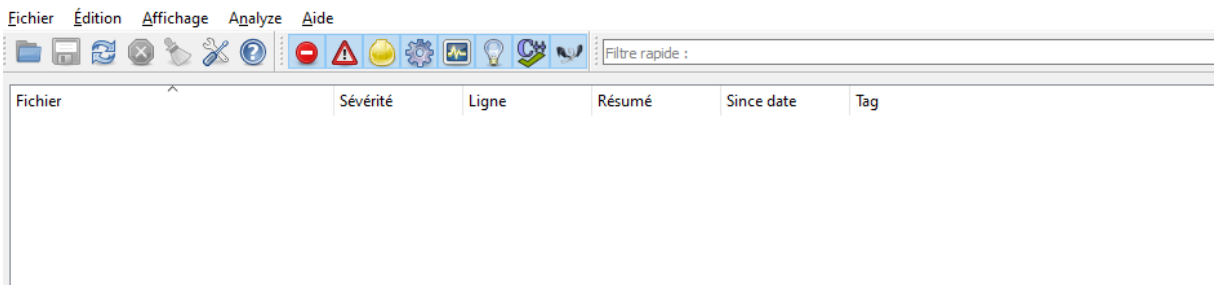
Il nous reste que des erreurs de style dans Cashregister.c

Fichier	Sévérité	Ligne	Résumé	Since date
Cashregister.c				
Cashregister.c	erreur de style	77	The scope of th...	06/02/2024
Cashregister.c	erreur de style	191	The scope of th...	06/02/2024
Cashregister.c	erreur de style	192	The scope of th...	06/02/2024
Cashregister.c	erreur de style	144	MISRA 11.8 (Re...	06/02/2024

```
186 /*
187  * Remove the last product scannned
188  */
189 static void removeLastProduct(void)
190 {
191   LDRA_char_t [MAX_STRING];
192   void const struct Product* lastProduct;
193
194   if (scannedProducts > 0U)
195   {
196     scannedProducts--;
197     lastProduct = ShoppingBasket[scannedProducts];
198     if (lastProduct != NULL_POINTER)
199     {
200       (void) sprintf([MAX_STRING], "Removing %s", lastProduct->name);
201       Userinterface_show(&[0]);
202     }
203   }
204 }
```

```
144         thePrice = Specialoffer_getPrice(*aCountedProduct->count,
145             *aCountedProduct->itsProduct->unitPrice,
146             *aCountedProduct->itsProduct->specialOffer,
147             *aCountedProduct->itsProduct->name);
148         sum_total += thePrice;
```

Nous n’avons plus aucune erreur :



Question 3

Une notion importante est la traçabilité des exigences qui permet à tout instant de connaitre facilement les liens entre les exigences utilisateurs, exigences de spécification, de conception, de la réalisation et les tests associés.

Il est donc important d'assurer la traçabilité des exigences, c.-à-d. lier une exigence avec une autre de plus "haut niveau, celle de l'étape précédente dans le cycle de vie du produit. On parle donc de la couverture des exigences qui doit être assurée. On dit que les exigences "en aval" couvrent les exigences "en amont". Le but est donc de vérifier à chaque étape du cycle de vie, qu'aucune exigence de l'étape précédente n'a été oubliée.

- Corrigez les trois documents de spécification & conception, contenant les erreurs suivantes :
    - Erreurs de traçabilité (manquante ou incorrecte),
    - Non-respect de l'écriture des ID,
    - Certaines exigences ne sont pas couvertes.

Dans « CashRegister5.0_Architectural_Design_Requirements » :

- HLR_00100, covers : SYS_0100
- HLR_00200, covers : SYS_0200
- HLR_00900, covers : SYS_0200
- HLR_1200, covers : SYS_0300, SYS_0350
- HLR_1500, covers : SYS_0400
- HLR_1800, covers : SYS_0200
- HLR_1900, covers : SYS_0700

Dans « CashRegister5.0_Detailled_Design_Requirements » :

- DDR_00100, covers : HLR_00100
- DDR_00200, covers : HLR_00500
- DDR_01100, covers : HLR_01000
- DDR_01200, covers : HLR_01300
- DDR_01900, covers : HLR_01200
- DDR_02200, covers : HLR_01400
- DDR_02300, covers : HLR_01500
- DDR_02350, covers : HLR_01500
- DDR_02400, covers : HLR_01700
- DDR_02100, covers : HLR_01800

Les exigences systèmes sont toutes couvertes.

CONCLUSION

Le design review a été effectué en vérifiant la conformité du code aux spécifications des documents élaborés durant le cycle de vie du produit et à la norme de codage MISRA C. L'étape suivante va consister à effectuer des tests via une analyse dynamique. En effet, un plan de test des fonctionnalités doit être fourni par le concepteur dans lequel, il faut qu'il y ait suffisamment de tests pour que toutes les fonctionnalités du produit soient testées. Chacun des tests peut avoir des champs à compléter pour préciser ses résultats. Il faut qu'il y ait suffisamment de tests pour que toutes les spécifications du produit soient validées par au moins un test. Cela ne signifie pas qu'il faille un test par spécification : Il peut y avoir un test par spécification, un test pour plusieurs spécifications ou plusieurs tests pour une spécification si elle est très complexe.

La liste de tests qui permettent de vérifier le bon fonctionnement constitue **le plan de validation** qui sera élaboré au TP suivant.