

Conception

- « *Il y a deux manières de concevoir un logiciel : une méthode est de faire des choses si simples qu'elles sont de toute évidence sans défauts, l'autre est de les faire si compliquées qu'aucun défaut n'est évident ! La première est beaucoup plus difficile.* »

CAR Hoare

- Concevoir, c'est essentiellement répondre à la question
« Comment ? »

Conception préliminaire

- Définir l'**architecture générale** du système / logiciel, ie le décomposer en **composants** plus simples
- « Spécifier » chaque composant : définition des **interfaces** et des **fonctionnalités** de chacun

Conception détaillée

- Etudier l'architecture technique du système / logiciel, conformément au découpage fonctionnel fait en conception préliminaire
 - Confirmation des choix fonctionnels dans l'implémentation technique
 - Choix organisationnels
 - Levée des dernières options techniques
 - Répartition homme/machine
 - Localisation des données
 - Volumétrie, capacités de stockage
 - Temps de réponse
 - Réutilisation
 - ...
- Pour chaque composant, indiquer comment ses fonctionnalités sont réalisées : données, algorithmes

2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

3

La méthode SD (Structured Design)

- Yourdon/Constantine
- Méthode simple et éprouvée
- Applicable à la conception de programmes
 - écrits en langages procéduraux,
 - faisant intervenir peu de développeurs
- Permet de définir, organiser et documenter les composants fonctionnels des logiciels
 - hiérarchie structurée de modules
 - spécifications de ces modules
- Si les données sont complexes, les modéliser à part, avec un autre formalisme

2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

4

La notation

- Une diagramme de structure (DS)
 - Modélise le partitionnement/regroupement des fonctions de contrôle et traitement des données définies dans la spécification, selon des critères de conception
 - Définit l'**organisation hiérarchique** des modules et les **interfaces de contrôle et de données** entre eux
- Des M-specs
 - une spécification par module
 - en langage naturel, pseudo-code (mieux) ou par un organigramme
 - permettre un codage direct

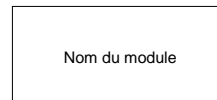
Diagramme de structure

- Les différents modules
 - Programme principal
 - Procédures et fonctions
 - Blocs de données
- Leurs liens
 - Appel de S/P
 - Accès à des blocs de données
- Leurs interfaces
 - Données et contrôles
 - Paramètres en E, S, Maj

Diagramme de structure

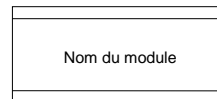
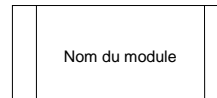
- Module

- Un point d'E
- Un point de S
- Un nom :
 - verbe (à l'infinitif)
 - + complément (facultatif)



- Modules particuliers

- S/P externe (non développé dans projet)
- Module bibliothèque



2011-2012

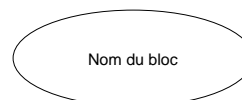
Henri Massié / L2 Informatique / UE Projet logiciel

7

Diagramme de structure

- Bloc de données

- Fichier externe, base de données, variables globales



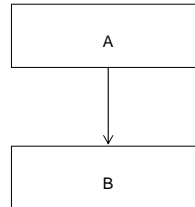
2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

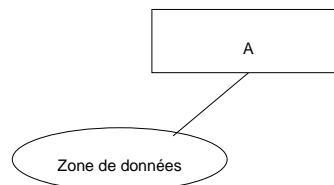
8

Diagramme de structure

- Liens
 - Appel de S/P (synchrone)



- Accès à un bloc de données



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

9

Diagramme de structure

- Quelques règles simples :
 - Eviter les sauts de niveau
 - Pas d'appels croisés :
 - si A appelle B, B ne peut pas appeler A, directement ou indirectement
 - Pas de récursivité :
 - sera montrée dans les M-specs si nécessaire

2011-2012

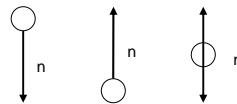
Henri Massié / L2 Informatique / UE Projet logiciel

10

Diagramme de structure

- Interfaces

- Symboles : E, S, Maj
 - Vis-à-vis de l'appelé
- Nom
 - éventuellement qualifié



- Types d'interface

- Donnée
- Contrôle



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

11

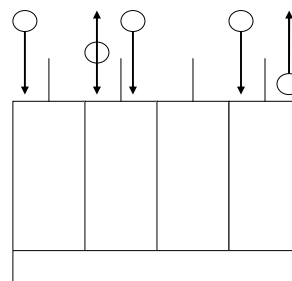
Diagramme de structure

- Compléments

- renvoi



- Cluster



2011-2012

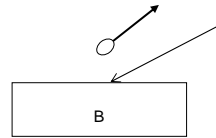
Henri Massié / L2 Informatique / UE Projet logiciel

12

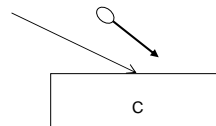
Diagramme de structure

- Typologie des modules

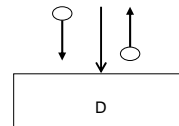
- Entrée



- Sortie



- Transformation



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

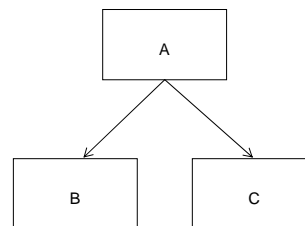
13

Diagramme de structure

- Typologie des modules

- Coordination

- Ne signifie pas que A appelle systématiquement B et C et dans cet ordre



- Notations complémentaires

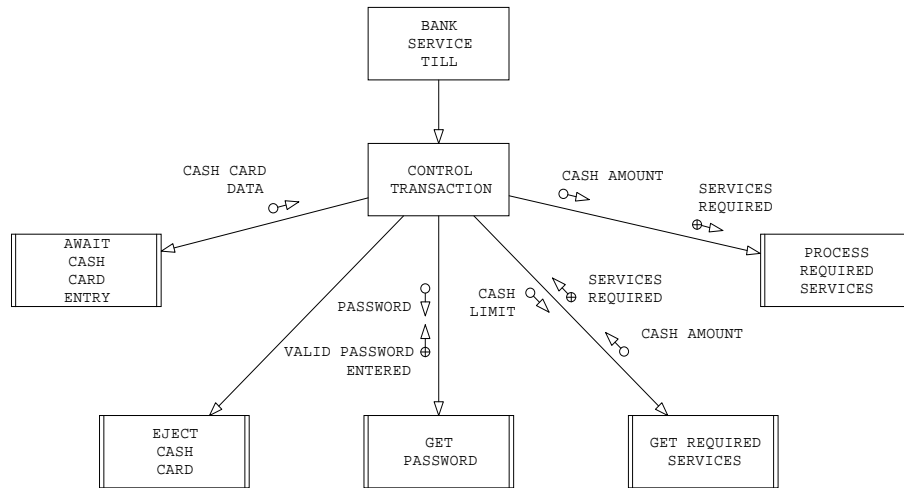


2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

14

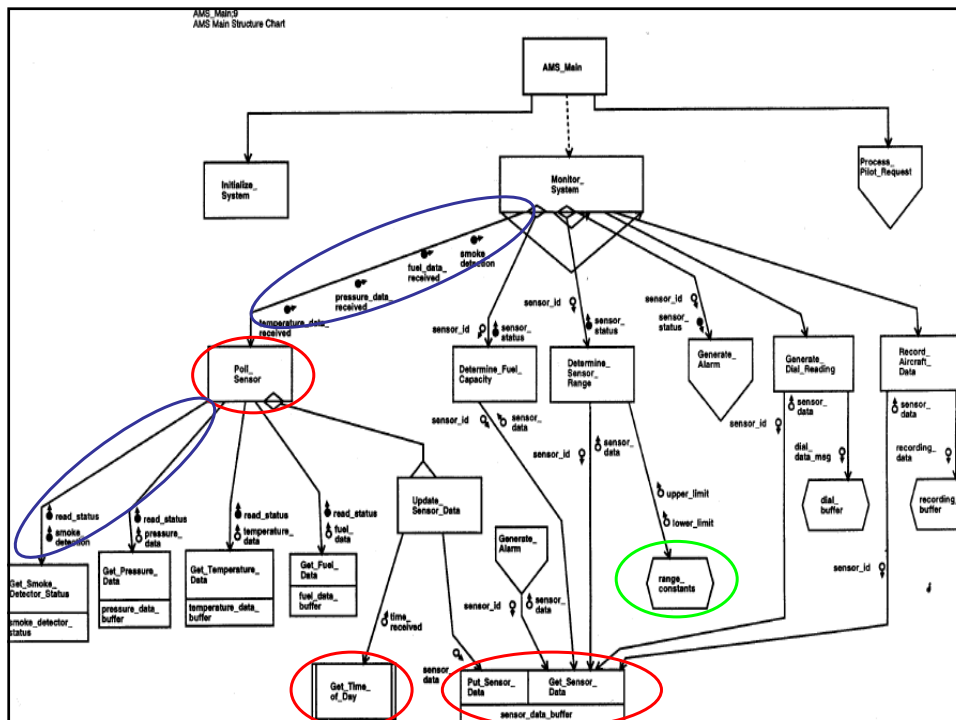
Exemples



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

15



Spécification de modules

- M-spec (mini-spec)
- Indiquer tout ce qu'il faut pour coder, maintenir et utiliser le module
 - ce qu'il fait
 - ses interfaces
- Plan type

Plan-type de M-spec

- Nom
- Rôle
- Types de données
- Signature de l'interface
 - Paramètres : nom, type, nature (E, S, Maj)
 - Résultat (si fonction) : type
- Variables globales utilisées (si pas fonction) : nom, type, nature (E, S, Maj)
- Traitement d'erreur
- Dépendances :
 - modules appelés
- Pré et Post conditions
- Pseudo-code

Exemples

```

NAME:
AMS_Main;2

TITLE:
AMS_Main

PARAMETERS:

LOCALS:

GLOBALS:

BODY:
/*****
/* file name: SAME */
/* */
/* Purpose: Aircraft Monitoring System Main Module */
/* */
*****/
CALL Initialize_System
SCHEDULE Monitor_System
SCHEDULE Process_Pilot_Request

```

2011-

20

```

NAME:
Monitor_System;5

TITLE:
Monitor_System

PARAMETERS:

LOCALS:
sensor_id
pressure_data_received
sensor_status
smoke_detection
fuel_data_received
temperature_data_received

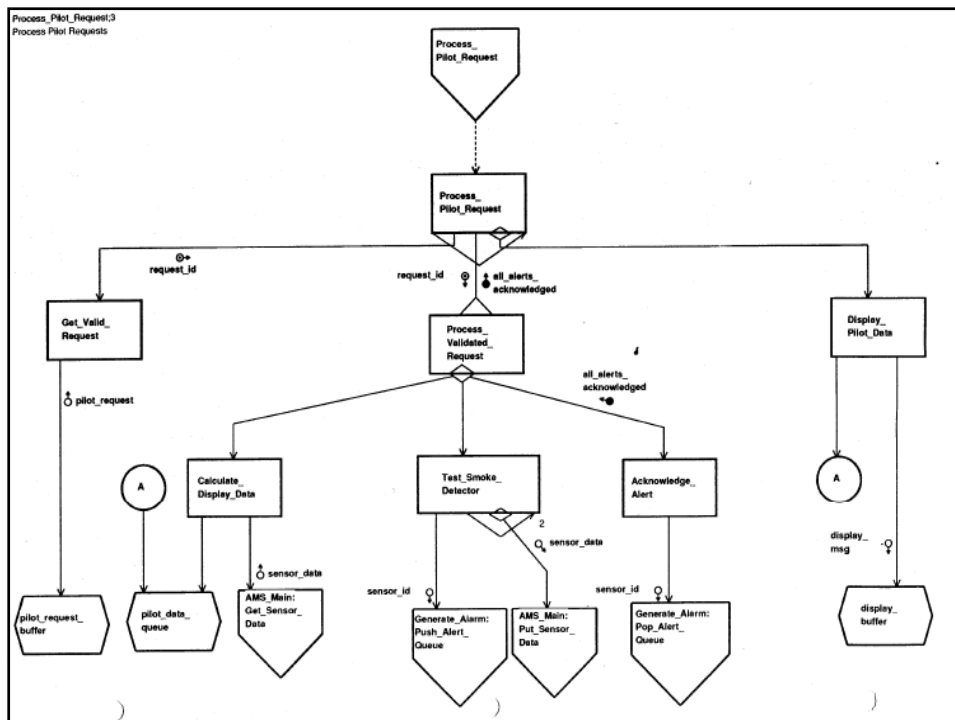
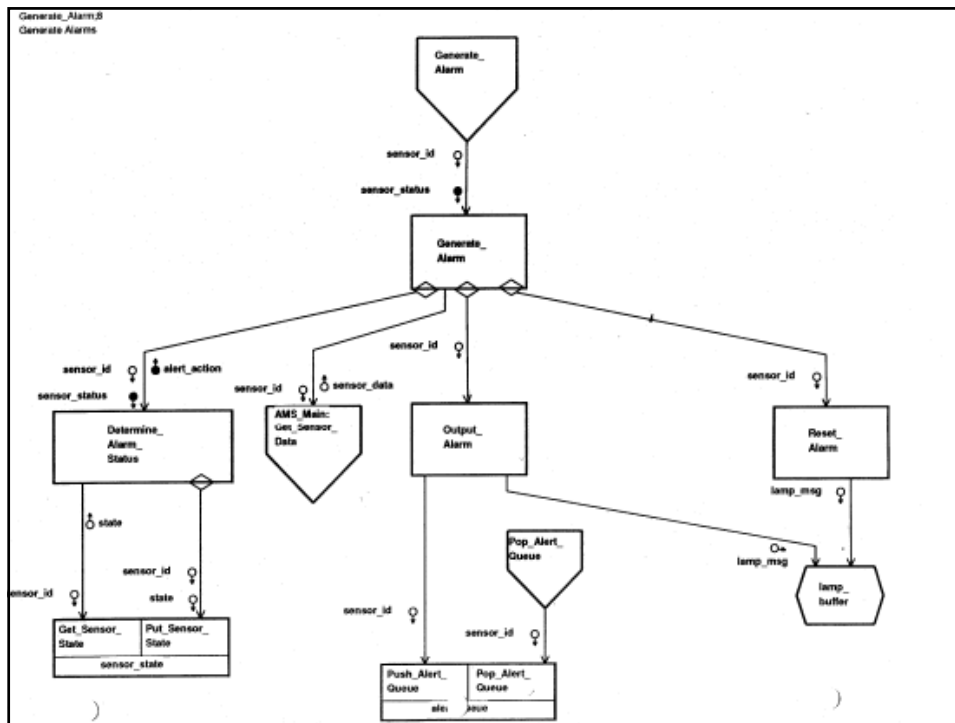
GLOBALS:

BODY:
/*****
/* file name: SAME */
/* */
/* Purpose: Monitor system sensors and */
/* generate necessary alarms. */
/* */
*****/

CYCLE 1-second
CALL Poll_Sensor(sensor_data_received,fuel_data_received)

/* Check fuel */
sensor_id = "F01"
IF fuel_data_received = "TRUE" THEN
    CALL Determine_Sensor_Range(sensor_id,sensor_status)
    IF sensor_status = OK THEN
        CALL Determine_Fuel_Capacity(sensor_id,sensor_status)
    ENDIF
ELSE
    sensor_status = "TIMED OUT"
ENDIF
IF sensor_status <> "OK" THEN
    CALL Generate_Alarms

```




Démarche

- On part d'un module supposé répondre à la spécification
- On le décompose en modules de plus bas niveau, que l'on spécifie
- On améliore cette 1^{ère} mouture par rapport aux critères qualité (couplage, cohésion, masquage d'information, complexité ...)
- On vérifie que si chaque module subordonné satisfait ses spécifications, alors le module de plus haut niveau satisfera les siennes
- On itère pour chaque module subordonné trop complexe, niveau par niveau, jusqu'à ce qu'ils soient tous codables directement

Critères qualité

- **Couplage**
 - Nature des relations entre deux modules
 - Plus il est faible, plus il sera facile de modifier, intégrer et réutiliser les modules
- **Cohésion**
 - Raison d'exister du module
 - Plus il est fort, plus il sera facile de mettre au point et réutiliser le module, plus il sera stable, moins il sera sensible aux effets de bord
 - Caractérise les modules bien faits

Couplage


- **Couplage par les données** : le plus faible, le meilleur
 - Les échanges portent sur des données élémentaires ou sur des structures de données complètement utilisées
 - Ils sont réalisés via des paramètres 
 - Minimiser le nombre de données assurant le couplage
- **Couplage sélectif**
 - Plusieurs modules font référence à une même structure de données, non globale et/ou utilisée par parties

2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

26

Couplage

- **Couplage par le contrôle** : attention ! 
 - Un module fournit à un autre des informations qui influent sur sa logique interne
 - A éviter dans le sens appelant/appelé
 - un module doit être une boîte noire, faire une chose et une seule, sans faire d'hypothèse sur le module qui l'appelle
 - Possible dans le sens appelé/appelant : compte-rendu par ex
- **Couplage par le contenu** : le plus mauvais
 - Un module fait référence à une donnée locale à un autre module, se branche en dehors d'un point d'entrée déclaré de l'autre module, modifie la manière de fonctionner de l'autre module ...
 - Trop contraignant : à éviter

2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

27

Cohésion

- **Cohésion fonctionnelle** : la meilleure
 - Le module réalise une fonction et une seule (donc faible couplage également)
 - On peut lui donner un nom significatif de sa tâche
 - Ex : M1=calculer le résultat ; M2=imprimer le résultat
- **Cohésion de séquence** : bonne
 - Les éléments du module réalisent des activités qui s'enchaînent : la sortie de l'une est l'entrée de la suivante
 - Ex. : calculer et afficher le résultat

2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

28

Cohésion

- **Cohésion de communication** : bonne
 - Cohésion procédurale + les composants du module utilisent les mêmes entrées ou les mêmes sorties
- **Cohésion procédurale** : passable
 - Regroupement d'activités qui peuvent être effectuées dans la même unité de temps
 - Le plus souvent déduit d'un organigramme : chaque module accomplit plusieurs fonctions, avec des appels internes
- **Cohésion temporelle** : médiocre
 - Regroupement d'activités qui doivent être effectuées dans la même unité de temps
- **Cohésion de coïncidence** : la plus mauvaise
 - Regroupement en vrac d'éléments indépendants

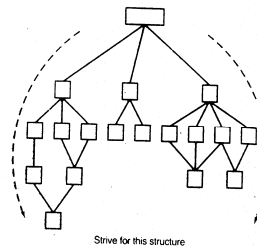
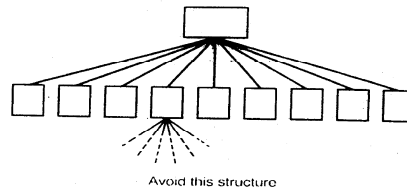
2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

29

Fan in/Fan out

- Nombre de modules appelants/appelés
- Réduire le fan out
- Augmenter le fan in



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

30

Autres recommandations

- Rapprochement d'une décision de son effet
 - **Portée de contrôle** d'un module = module + ensemble des modules appelés
 - **Portée de l'effet d'une décision** : l'ensemble des modules qui contiennent du code dont l'exécution dépend de la décision prise
 - Une application est plus simple à tester et à maintenir si la portée de l'effet d'une décision est incluse dans la portée de contrôle
 - Indicateur de la raison d'être d'un module

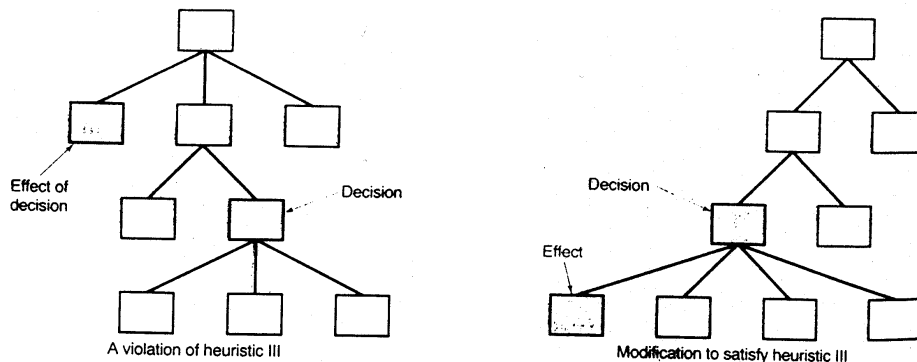
2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

31

Autres recommandations

- Rapprochement d'une décision de son effet



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

32

Autres recommandations

- Factorisation
 - Décomposer un module / regrouper des modules
 - Pour simplifier le codage et favoriser la réutilisation
 - Préserver la cohésion
- Affichage d'erreurs et autres éléments d'IHM
 - Éviter la dispersion

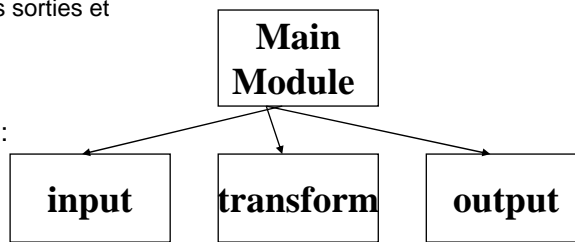
2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

33

Forme générale d'un bon DS

- Conception orientée « transformations »
 - 3 « régions »
 - Entrées et prétraitements des entrées
 - Traitements
 - Préparation des sorties et sorties
 - Plus modules de coordination
 - Ex. au 1^{er} niveau :



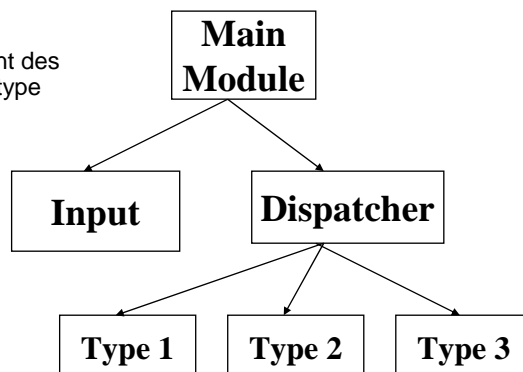
2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

34

Forme générale d'un bon DS

- Conception orientée « transactions »
 - 2 « régions »
 - Entrées
 - Un dispatcher et des modules de traitement des transactions (un par type de transaction, commande, requête)
 - Plus modules de coordination
 - Ex. au 1^{er} niveau :



2011-2012

Henri Massié / L2 Informatique / UE Projet logiciel

35