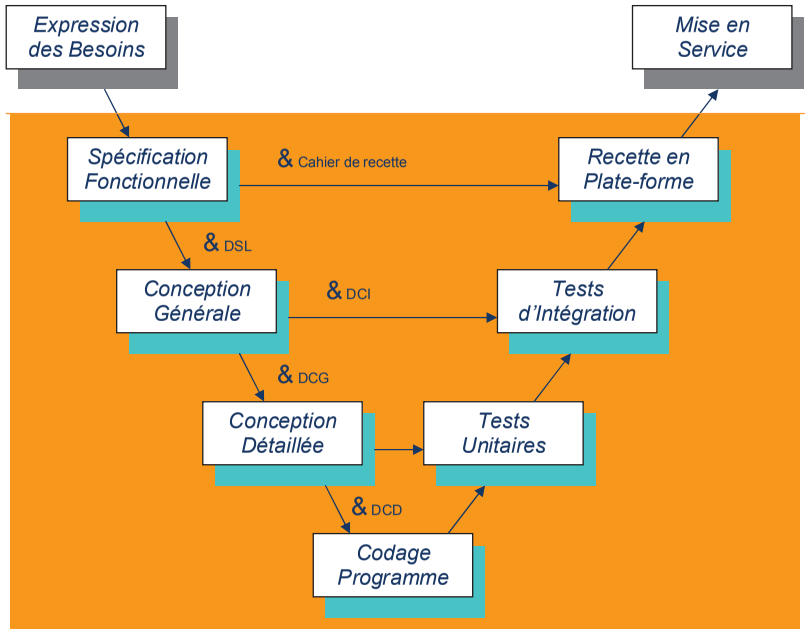
CONTROLBUILD : 1ère partie

INTRODUCTION :

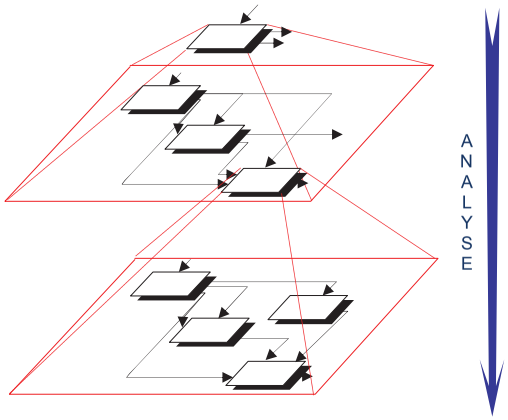
Le logiciel ControlBuild permet de valider une partie opérative, de programmer la partie commande associée, et de valider l'ensemble d'une application API.



Démarches de construction d'une application :

- Analyse descendante : description d'une application en partant du plus complexe vers le plus simple.
- Synthèse ascendante : création de composants standards et assemblage (appel de sous programmes) de ces composants pour créer des composants de plus en plus complexes.
- Démarche mixte : description descendante de la part spécifique de l'application et ascendante la part qui correspond à ce que l'utilisateur à déjà fait et devra réutiliser.

La méthode la plus adaptée à ce logiciel est la méthode descendante, néanmoins dans certains cas précis une démarche mixte sera nécessaire (l'utilisation de composants bas niveaux spécifiques va forcément influer sur les niveaux supérieurs, une analyse mixte est donc nécessaire dans certains cas). Le schéma suivant représente le principe de l'analyse descendante.



Sous ControlBuild cette analyse (basée sur la norme du SADT) se fera à l'aide de boîtes nommées MAC : « Modèle d'Assemblage de Comportement ».

Une fois que cette analyse a atteint un niveau suffisamment bas :

- pour la description de la partie opérative, on arrive au niveau de la description des éléments mécaniques et de leurs actionneurs (représentation d'un tapis et de son moteur) ;
- pour la description de la partie commande, on arrive au niveau de la description du fonctionnement de chaque entité élémentaire (règles de commande d'un tapis).

On passe ensuite à une description en langage de programmation de la norme 1131 (SFC, FBD, LADDER, ST).

Avant de vous former à ce genre de description et d'analyse, nous allons lors de cette séance de TP, créer une petite application simple. Cela aura pour avantage de vous apprendre petit à petit le fonctionnement du logiciel qui reste complexe.

TRAVAIL A REALISER : MISE EN OEUVRE D'UN TAPIS DE CONVOYAGE

Le cahier des charges de ce projet est très simple et peut se résumer en quelques lignes :

- Soit un tapis roulant qui permet de transférer des pièces d'une zone de chargement vers une zone de stockage.
- Lorsqu'un opérateur appuie sur un bouton « Chargement », une pièce est positionnée sur le tapis. Ce dernier se met en marche jusqu'à ce que la pièce arrive en bout de tapis où elle est détectée par un front montant sur le capteur « Fin_Convoyage ».
- Une fois la pièce arrivée en bout de tapis, elle tombe dans une zone de stockage.
- Cette zone de stockage « Stock » ne peut contenir que 10 pièces au maximum. Donc, si 10 pièces sont stockées, le tapis ne doit plus fonctionner.
- Afin de libérer la zone de stockage, un opérateur peut enlever manuellement une pièce à chaque fois. Il le signale en appuyant sur un bouton poussoir « Dechargement » dont on détecte un front montant.
- Aucune fonction de sécurité n'est demandée. A la mise en route, on considère que le tapis et la zone de stockage sont vides.
- On considère aussi que le tapis ne convoie qu'une seule pièce à la fois.

Création d'un nouveau projet :

- Création du projet : Fichier→Nouveau→
  Mes documents\Votre Nom\ERII31\TP1_Tapis\Intro.cbp
- Création de l'application:
  - Colonne 1, sélectionner Applications→Nouvelle application/biblio(click droit souris). Créer une application nommée « Tapis_Conv ».
  - Nous allons créer des groupes de rangement des programmes que nous allons utiliser pour cette application. Colonne 1, sélectionner « Tapis_Conv » Nouveau groupe de composants (click droit). On crée un groupe « Partie_Commande » et un groupe « Partie_Operative ».
  - Cliquer sur le groupe « Partie_Commande » et dans la colonne 2, créez un nouveau composant nommé « PC » de type MAC. Idem dans l'autre groupe pour « PO ».
- Notre application c'est une PC + une PO. Donc on ouvre l'application « Tapis_Conv » (double click colonne 2 sur « Tapis_Conv »). Vous êtes dans le modèle descriptif de l'application, il est de la forme SADT.

Q1 : Qu'est ce que SADT ?

- On fait glisser (à partir de la colonne 2) le MAC « PC » puis le MAC « PO » dans « Tapis_Conv ». On peut même faire glisser 2 fois « PC ».
- Vous pouvez enregistrer les changements de « Tapis_Conv ».

Q2 : Que remarque t-on ? À quoi correspond pc1, pc2, po1 ? Vous pouvez supprimer pc2.

Création de la partie commande :

Le rôle de ce logiciel est au final de créer une partie commande en ordre de marche. C'est pour cela que nous allons créer la partie commande puis que nous allons l'associer à une partie opérative simulée (que nous allons également créer) pour la valider.

On ouvre le MAC « PC » qui se trouve dans le groupe « Partie_Commande », c'est le modèle. On se retrouve avec une page avec 3 parties, une principale (description du fonctionnement), et 2 dans la partie basse : celle de gauche pour la déclaration des variables et celle de droite pour les messages (de compilation par exemple.

- Dans la zone variables, onglet entrées, on fait click droit →Nouveau groupe
- On crée les groupes « Retour_PO » et « Retour_Utilisateur ».
- Dans le groupe « Retour_PO », on crée la variable « Fin_Convoyage ».
- Dans le groupe « Retour_Utilisateur », on crée « Chargement » et « Dechargement ».
- Dans l'onglet sorties, on crée le groupe « Commande_PO » dans lequel on crée la variable « Marche ».

Q3 : Que vient-on de faire en déclarant ces variables d'entrée et de sortie ?

On sauve et on retourne dans le MAC « Tapis_Conv ».

Q4 : Pourquoi pc1 est-il en rouge ? Que se passe t-il si vous faîtes « Contrôles →Mettre à jour tous les externes. » ?

Notre partie commande va être des plus simples, elle ne va contenir qu'un GRAFCET de production. Il faut pour cela créer un modèle puis l'insérer dans le MAC « PC ».

- dans le groupe « Partie_Commande », vous créez un nouveau composant « Graf » de type Grafcet.
- Vous définissez les variables d'entrée et de sortie (idem à « PC », donc un copier/coller des groupes de « PC » peut se faire).
- Il nous faut maintenant créer le grafcet. Les icônes du menu vont vous permettre de créer une étape initiale puis les étapes suivantes. Si on clique deux fois sur un icône cela maintient son action jusqu'à ce que l'on appuie dessus pour le dévalider. L'icône flèche permet de dessiner la transition entre deux étapes (on clique sur la première puis sur la deuxième).
  - Le format d'écriture des actions et des transitions est en langage ST.
  - Exemple d'une action :

```
(*ceci est du commentaire*);
Marche;
```

  - Exemple de transitions :

```
(*équation logique classique*)
Chargement AND NOT(Dechargement)
```

```
(*tempo de 10 s dès que l'étape 10 est validée*)
T/X10/10 s
```

Q5 : Trouver le grafcet de fonctionnement de « Graf ».

- Vous pouvez maintenant tester le fonctionnement de votre grafcet en le simulant. Pour cela vous enregistrez puis vous cliquez sur l'icône « petit bonhomme qui court » puis « play ». Vous pouvez maintenant voir si en fonction des valeurs que vous imposez aux entrées, les sorties agissent correctement.

Q6 : Cela fonctionne t-il ?

- Vous venez de valider votre grafcet. On peut maintenant créer une instance de « Graf » dans le MAC « PC ». Pour cela et comme précédemment il suffit d'ouvrir « PC » et de glisser dedans « Graf ».
- On glisse les groupes des entrées « Retour_PO » et « Retour_Utilisateur », de la zone de définition des entrées jusqu'à la zone d'édition. On relie ensuite ces groupes aux entrées du bloc « Graf1 » (de type « Graf »). On fait la même chose pour les sorties.
- On simule.

Q7 : Cela fonctionne t-il ?

On vient de terminer notre partie commande, elle semble fonctionner mais, si on la transfert dans un automate et que l'on connecte le tout à un tapis, le résultat sera t-il concluant ?

Pour en être certain, nous allons créer une simulation de la partie opérative, ensuite nous pourrons simuler le tout puis valider l'application. Si à terme cela fonctionne, on peut considérer qu'en situation réel cela devrait fonctionner. On voit donc apparaître un des rôles de ce logiciel, simuler une PO reliée à une PC afin de valider la PC. Lors de la mise en route du système réel, cela devrait permettre de limiter les bugs, bien entendu cela dépend fortement de la finesse de description du modèle de simulation de la PO…

Création de la partie opérative :

Il nous faut maintenant créer une simulation de la partie opérative. Pour simplifier nous allons considérer que l'application est constituée d'un tapis, d'une zone de stockage et d'un capteur TOR.

- Dans la colonne 1, on sélectionne le groupe « Partie_Operative ».
  - Dans la colonne 2, on crée un nouveau composant « CapteurTOR » en texte structuré (ST=Structured Text).
- On double clique sur « CapteurTOR ». On définit ses entrées et sorties :
  - Entrées : « Position_IN » (position de la pièce en mètre à l'instant t) et « Position_Fixe » (position du capteur en mètres dans le repère). Deux entrées de type REAL.
  - Sortie : « Etat » de type booléen.
  - Paramètres (constantes) : On définit « Fourchette » de type REAL. Correspond à l'épaisseur de la zone de détection (on la fixe à 10 cm).
- On définit les lois de fonctionnement :
  - Si « Position_IN » est comprise entre « (Position_Fixe - Fourchette) » et « (Position_Fixe + Fourchette) » alors « Etat = Vrai » sinon « Etat = Faux ».
- On simule.
- On peut maintenant définir un modèle d'animation représentant le composant.
  - Dans la page principale de Controlbuild, on sélectionne « CapteurTOR » dans la zone composant puis, dans la colonne 3 (Vue) →Nouvelle Vue→nom : Model_CapTOR, Type : Modèle d'animation.
  - On ouvre ce modèle.
  - Dans la zone formes (en bas à gauche)→Nouvelle Forme→Capteur
  - On va dans l'onglet Forme (à côté de l'onglet représentation) et on dessine la forme (par exemple un carré noir). On vient de dessiner une forme élémentaire représentant le capteur.
  - Dans la zone Règles→ Nouvelle Règle. On indique que l'on souhaite que le carré soit noir quand « Etat=Faux » et rouge quand « Etat=Vrai ».
  - On sauve.
- On simule de nouveau « CapteurTOR », mais cette fois en ouvrant le modèle d'animation → Vues→Modèle d'animation.

On fait la même chose pour les autres éléments de la PO, c'est à dire pour le modèle « Tapis » et le modèle « Stockage ».

- « Tapis » est constitué de deux formes :
  - Un tapis : il clignote lorsqu'il est en marche.
  - Une pièce : elle se déplace sur le tapis qui fait 50 mètres de long.
  - Il a une entrée Marche pour démarrer le tapis et une sortie Deplacement qui donne une valeur réelle représentant le déplacement de la pièce.
- « Stockage » est un container qui est vide ou chargé. Il possède une entrée In qui permet de détecter lorsque l'on charge une pièce et une entrée Out qui permet de détecter lorsque l'on décharge une pièce. On peut aussi lui associer un compteur pour connaître le nombre de pièces qu'il contient…

On place ensuite le tout dans le MAC « PO ». On connecte le tout, on simule la « PO » seule pour vérifier les règles définies. Il nous reste maintenant à créer le synoptique global de la « PO ».

- On sélectionne « PO » dans la colonne 2 de la fenêtre principale, puis on fait un click droit dans la colonne 3 Nouvelle Vue→ Synoptique .
- On ouvre le synoptique.

- On ouvre également l'arbre de représentation de « PO ». Pour cela on fait click droit sur « PO » → Arbre.
- On sélectionne dans l'arbre « Tapis1 ».
- Dans la colonne basse de la fenêtre synoptique, on fait click droit → ajouter un modèle d'animation → model_tapis. On peut déplacer la souris sur la partie haute de la fenêtre pour positionner et dimensionner le tapis.
- On fait la même chose pour « CapteurTOR1 » et « Stockage1 ».
- Simuler pour valider le fonctionnement de la PO.

Simulation de l'application complète :

On désire tester la PO commandée par la PC. Pour cela, nous allons créer un pupitre avec les boutons.

- Dans la colonne 1, on crée un groupe « BP_operateur » tout comme on avait fait pour « Partie_Operative » et « Partie_Commande ».
- Dans la colonne 2, on crée un nouveau composant « Boutons_poussoirs » de type MAC puis on l'ouvre.
  - Dans l'onglet sorties, on crée le groupe « Poussoirs_Operateurs » dans lequel on crée les variables « Chargement » et « Dechargement ».
- Dans la colonne 3, on crée une nouvelle vue de type Pupitre que l'on appelle « BP ».
  - On ouvre ce modèle et on crée 2 boutons poussoirs. On peut choisir différentes images en cliquant sur le bouton puis en cherchant dans image de l'onglet Général.
  - Dans la colonne 2, on ouvre l'arbre de « Boutons_poussoirs » et on affecte les variables « Chargement » et « Dechargement » aux boutons poussoirs créés en les faisant glisser sur les boutons. Ainsi, on vient d'associer les variables à l'action sur les boutons. On peut d'ailleurs le tester…
- On retourne dans le MAC « Tapis_Conv » et on ajoute le MAC « Boutons_poussoirs ». On relie les entrées et sorties des 3 MAC « Boutons_poussoirs1 », « PC1 » et « PO1 » pour simuler l'application complète.
- On vérifie la cohérence puis on simule :
  - On lance la simulation depuis « Tapis_Conv ».
  - On ouvre le Pupitre dans le MAC « Boutons_poussoirs1 » et le synoptique dans le MAC « PO1 ».
- Tester l'application et conclure.