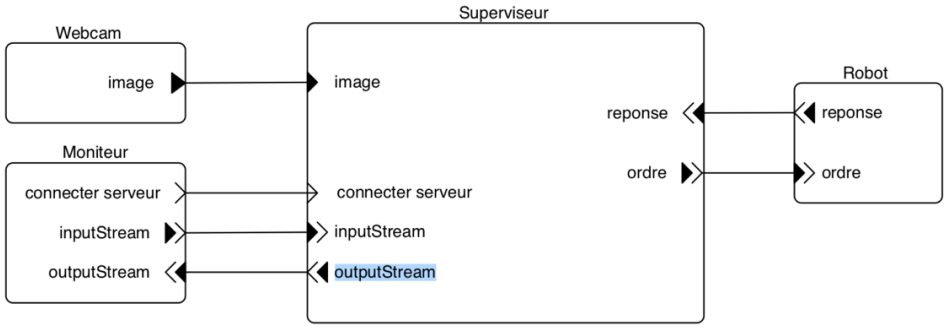
Aide pour les TD de temps réel

Voici la démarche classique à suivre pour faire l'analyse d'un système :

- **Préciser le système étudié** sous forme d'un diagramme de contexte.

  Un diagramme de contexte précise ses frontières, les acteurs externes, ses interfaces (dirigées et nommées) du système avec son environnement direct, en considérant les différentes modes d'utilisation du système (démarrage, fonctionnement nominal, arrêt, situation d'urgence) et différentes phases de vie du système (conception, réalisation, opération, maintenance).

  Un diagramme vous est déjà donné dans le sujet :



  Vous partir de ce diagramme et voir s'il est complet.

  Vous pouvez aussi compléter ce diagramme par un tableau répertoriant chaque élément qui transite aux interfaces (de, vers, donnée/contrôle, nom (voir règles de nommage en annexe), typage (entier/plage de valeur, texte/nombre de caractères ...), commentaires).

  NB : nous nous intéressons aux interfaces concernant les échanges d'information

  Vous pouvez également dès cette étape établir des **scénarios opérationnels** pour chaque mode pour vous aider.

- **Exprimer les exigences** du système :
  - identifier les besoins de services et performances,
  - exprimer et raffiner si besoin les exigences,
  - identifier de manière unique (numéro) les exigences,
  - classer les exigences par type (fonctionnelles, non fonctionnelle, performance, …)
  - vérifier et valider les exigences: précision, cohérence, non-ambiguïté et complétude.

  Dans le sujet vous sont déjà données les fonctionnalités à assurer par le système (dont l'expression se rapproche des exigences….). Vous pouvez donc partir de ces fonctionnalités, voir si vous avez besoin de les reformuler ou si en l'état elles conviennent.

- **Écrire les tests de validation** du système :

  Nous nous appuyons sur les exigences précédentes pour écrire les tests de validation, qui seront utilisés lors de la livraison finale du système au client. Ceux-ci décrivent un ensemble de scénarios exprimant, le plus exhaustivement, le comportement nominal et non-nominal du superviseur.

Stimuli d'entrée	Comportement et sortie

- **Déduire des exigences les fonctionnalités du système** (étape de spécification : Qu'est-ce que le système doit faire ?) que le superviseur devra assurer (une fonction transforme un flux de données d'entrée en flux de sortie, en consommant du temps et des ressources).

  L'objectif final est d'obtenir un **diagramme d'architecture fonctionnelle** montrant les échanges et les enchaînements entre ces fonctions (caractérisation des fonctions et de leurs interfaces, flux de données par les échanges d'entrées et sorties, flux de contrôles (activations, synchronisations).

  Sur le plan de la démarche, il s'agit de d'abord identifier la fonction objectif que le système doit assurer (par exemple « Commander et superviser un robot à distance »), que l'on raffine progressivement en sous-fonctions, en identifiant à chaque niveau de la décomposition les flux de contrôle (représentés sous forme de flèches en traits pointillés) et les flux de données (en traits pleins) qui transitent entre les fonctions (entrées et sorties), et en assurant dans la décomposition la cohérence entre les différents niveaux de raffinement. Dans le sujet, la décomposition en fonctions est déjà à un niveau de détails assez bas…

  En restant à ce niveau, vous pouvez alors, à l'aide d'un tableau comme le suivant, recenser et expliciter les fonctions que le superviseur devra assurer, en précisant les entrées et sorties de chaque fonction (une fonction transforme un flux de données d'entrée en flux de sortie, en consommant du temps et des ressources).

Nom de la fonction	Description	Entrées	Sorties

  Une fois chaque fonction décrite individuellement, vous proposerez un diagramme d'architecture logique montrant l'ensemble des échanges et les enchaînements entre ces fonctions (caractérisation des interfaces, des flux de données par les échanges d'entrées et sorties, flux de contrôles (activations, synchronisations).

  Vous pouvez pour cela compléter la figure suivante, fournie dans le sujet :

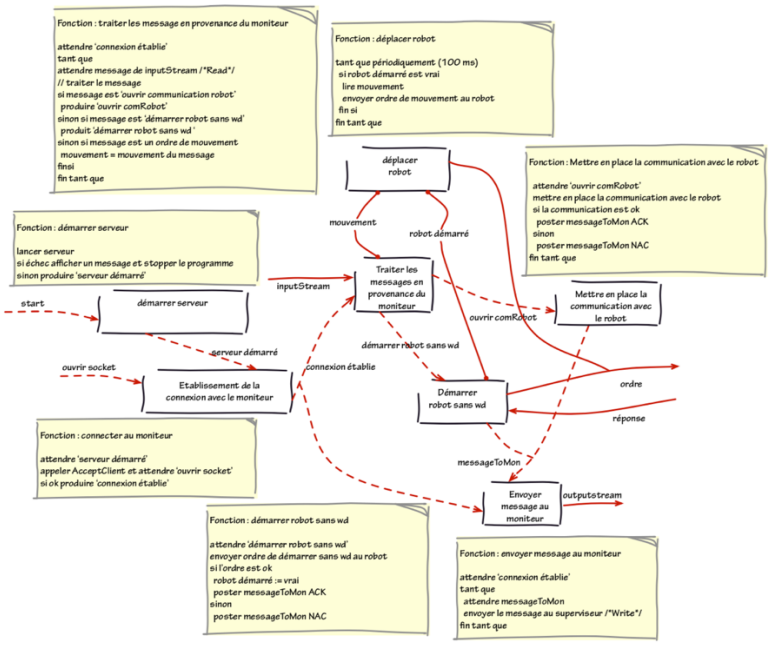


Diagramme d'architecture fonctionnelle en SA-RT like

NB : le formalisme proposé est inspiré du formalisme de SA-RT, une méthode d'analyse fonctionnelle adaptée au temps réel (voir mémo 'Analyse fonctionnelle' sous Moodle).

Ce diagramme sera complété par la description du comportement et de l'enchaînement des fonctions dans le temps à l'aide de **diagrammes faisant apparaitre les flux de données et la temporalité**.

Assurez-vous que les résultats de cette étape sont cohérents avec le diagramme de contexte et les exigences. **Assurez et démontrez pour cela la traçabilité** entre fonctions et exigences.

- **Identifier et caractériser un ensemble de tâches** (étape de conception : Comment le système va faire ?) permettant l'exécution de ces fonctions sur la plateforme logicielle choisie (Xenomai).

  L'objectif final est d'obtenir un **diagramme d'architecture organique** de l'application, faisant apparaître les tâches, les échanges de données et les synchronisations entre les tâches, et les moyens choisies (services de Xenomai). Ce diagramme sera précisé par des diagrammes représentant l'exécution dans le temps des tâches.

  Sur le plan de la démarche, vous devez pour cela tout d'abord réfléchir à un découpage en tâches qui vous semble pertinent. Vous effectuerez un découpage 'optimal' de l'application en tâches ; ces tâches vont exécuter les différentes fonctions en intégrant et respectant des contraintes de parallélisme, de séquencement dans le temps, de temps, de partage de ressources, … (on peut donc être amené, selon le niveau de détail auquel vous avez fait l'analyse fonctionnelle, à faire exécuter une fonction par plusieurs tâches ou au contraire à regrouper plusieurs fonctions dans une même tâche).

  Vous devez ensuite caractériser les tâches (nom, priorité, période, entrées, sorties), leurs activations (périodique, sur événement…).

  Ainsi, à l'aide d'un tableau comme le suivant, vous identifiez et caractérisez un ensemble de tâches permettant l'exécution des fonctions identifiées et caractérisées précédemment. Procéder de même que pour les fonctions (voir démarche de spécification). Préciser la période des tâches périodiques.

Nom de la tâche	Rôle	Entrées	Sorties	Période	Priorité

Puis choisir, caractériser et justifier des moyens de communication (variable globale protégée ou pas par un mutex, file de messages (taille, format des messages, timeouts…) et de synchronisation (sémaphore binaire, suspension/réveil..).

Moyens de synchronisation	Type (sémaphore, mutex, …)	Caractérisation	Justification

Moyens de communication	Type (variable globale, messageQueue, …)	Caractérisation (Id, nom, taille, timeout, ..)	Protection par Mutex	Justification

Et enfin décrire l'activité globale de l'application logicielle (entrelacement des flux d'exécutions des différentes tâches dans le temps) :

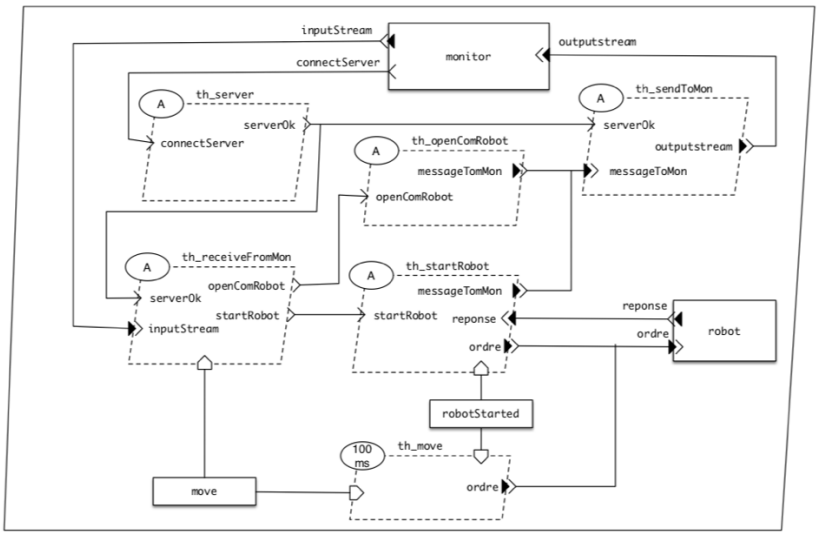
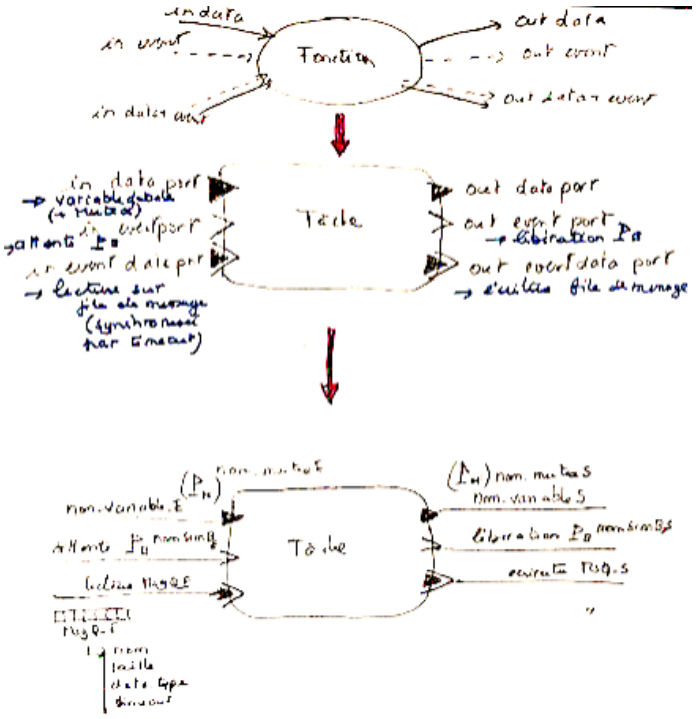


Diagramme d'architecture organique en AADL

NB 1 : Vous pouvez, pour représenter cette architecture, utiliser le formalisme AADL comme proposé dans le sujet (voir mémos sous Moodle)

NB 2: Plusieurs options d'architecture sont possibles, identifiez-les et comparez-les pour en choisir une et **argumentez pourquoi vous avez fait ce choix**.

Aide : Comment passer du diagramme d'architecture fonctionnelle à un diagramme d'architecture logique en AADL puis à des choix dépendants de la plateforme :



Vous pouvez ensuite décrire l'activité interne des tâches (par des diagrammes en UML : le comportement de chaque tâche pourra par exemple être décrit à l'aide d'un diagramme d'activité, voir sujet).

- Si vous avez le courage, vous pouvez **assurer et démontrer la traçabilité** de votre architecture avec les fonctions et exigences, à l'aide d'une matrice de traçabilité.

Tâche \ Fonction	Func 1	Func 2	…			

- **Écrire les tests** unitaires et tests d'intégration

  Vous devez dériver des exigences des plans de test fonctionnels (plans de test unitaire, d'intégration et de validation). Ces plans contiennent des séquences de test (entrée à appliquer, comportement attendu). Les tests seront appliqués une fois effectué le codage pour comparer le comportement obtenu au comportement attendu, et obtenir des résultats de test.

Stimuli d'entrée	**Comportement et sortie**