**Processus de rétro-ingénierie**

**Exemple de l’Hemomixer**

|  |
| --- |
|  |
|  |

# la rÉtro-ingÉnierie d’un systÈme

La rétro-ingénierie est un processus complexe qui permet par l’étude d’un système existant de spécifier son fonctionnement à travers un modèle.

Il s’agit alors de pouvoir par exemple :

* Comprendre le fonctionnement externe et interne du système ;
* Connaître les relations du système avec son environnement ;
* Connaître la mission et la finalité du système ;
* Identifier les solutions retenues ;
* Evaluer son bon fonctionnement ;
* Le modifier ;
* En concevoir une nouvelle version ;
* Le maintenir ;
* …

NB : On peut noter que suivant les métiers le sens du terme « rétro-ingénierie » peut prendre différents sens :

* Pour des informaticiens, la rétro-ingénierie de logiciels consiste souvent à produire le code source à partir de l'objet en utilisant des désassembleurs, des déscompilateurs, ...
* Pour des mécaniciens, il peut s’agir de produire des fichiers CAO à partir de fichiers de "nuages de points" obtenus par mesure de l'objet réel (palpeur, scanner, …), de photos, de plans plus ou moins complets, ...

NB : Dans le cas des systèmes didactisés, le système à considérer reste le « vrai » système qui a une vraie mission et qui rend des vrais services. Car si l’on considère le système didactisé lui-même, les besoins seront évidemment différents, par exemple existence d’un besoin de pouvoir mesurer des valeurs dans une boucle d'asservissement.

|  |
| --- |
|  |
|  |

# UNE DÉmarche gÉnÉrale pour la rÉtro-ingÉnierie

La méthode proposée est représentée sous forme de diagramme d’activité

1.1 Décrire le contexte

1.2 Définir la mission

1.3 Définir les cas d’utilisation

2.1 Décrire la structure interne

2.2 Décrire la structure hiérarchique

3.1 Décrire les interactions du système

3.2 Décrire les états du système

4.1 Décrire les échanges avec les sous-systèmes

4.2 Décrire les états des sous-systèmes

5. Décrire l’architecture logique

6. Définir les exigences système

7. Associer les exigences système

Dans tous les cas, cette démarche ne doit pas être vue comme totalement linéaire d’autant que comme pour l’ingénierie plusieurs passes sont souhaitables avant d’aboutir à un modèle satisfaisant relativement aux objectifs de la rétro-ingénierie (cf. objet de la rétro-ingénierie page 3).

Il est à noter que cette approche met nombre de choses en parallèle, ce qui laisse de nombreuses possibilités pour le modélisateur. En effet, même si ce diagramme d’activité guide la démarche, il permet différentes trajectoires et différents points de vue selon le but poursuivi. La connaissance préalable ou non du système et de son usage, la disponibilité ou non de documentation sont autant de facteurs qui influeront sur l’ordre des activités.

NB : Dans la suite de ce document nous utilisons le terme « opération » et non « fonction » pour éviter toute confusion. La notion d’opération est ici prise au sens de SysML, elle permet donc d’identifier des comportements du système. Ceux-ci pouvant être spécifiés par des diagrammes de comportement (STMD, SD, …).

**NB : Les diagrammes SysML proposés à titre d’exemple peuvent être et sont le plus souvent partiels (incomplets, non exhaustifs, incohérents entre eux, …).**

|  |
| --- |
|  |
|  |

# Activité RI1.1– DÉcrire le contexte

Dans cette partie on considère certaines phases de vie du système, et tout particulièrement celles ou le système est exploité (exploitation ou utilisation, soutien, …). Les autres phases de vie (conception, réalisation ou production, retrait ou fin de vie, …) sont plus rarement nécessaires lors d’une rétro-ingénierie (cf. objet de la rétro-ingénierie page 3).

On doit alors identifier le contexte du système étudié dans les différentes phases de vie retenues en recensant l’ensemble des éléments ayant un lien avec le système. La mise en œuvre du système, si elle est possible, permet plus facilement d'en appréhender le contexte de manière précise. Pour chaque phase de vie, le contexte permet d’expliciter l’environnement dans lequel évolue le système. Il permet d’identifier les éléments en interaction avec le système en précisant les acteurs humains (utilisateur par exemple) qui interagissent avec le système et les autres éléments en relation avec le système (autres systèmes de l’environnement, éléments de support, produit, …). Cette étude permet aussi de délimiter le système étudié et donc d’en voir les frontières (cf. prochains diagrammes des cas d’utilisations).

Le questionnement peut être le suivant : Dans quel cas le système est-il utilisé (phase de vie) ? Qui sont les acteurs (utilisateurs, autres systèmes, …) ?

Pour chaque phase du cycle de vie, les éléments du contexte sont identifiés dans des diagrammes de définition de bloc (BDD SysML) :

* Le système est au centre des préoccupations :
  + Mettre le bloc « Système » au centre ;
* Mettre autour les éléments externes en interaction avec le système :
  + Les acteurs humains ;
  + Les autres éléments via des blocs (on peut créer des stéréotypes comme « Elément externe », « Système externe » pour clarifier) ;
  + Définir les relations avec le système via des liens d’association orientés ou non ;
  + Définir les cardinalités, si elles ont une réelle valeur ajoutée pour l’étude.

NB : les symboles génériques des acteurs peuvent être remplacés par une image/photographie de l’acteur en question.

NB : Lors de la recherche des exigences auxquelles le système répond, ces diagrammes de contexte pourront être complétés pour faire apparaître d’autres parties prenantes ayant un intérêt vis-à-vis du système (acquéreur, organismes de normalisation, …) et ayant pu définir des besoins.

|  |
| --- |
|  |
|  |

# Activité RI1.2 : DÉfinir la mission

Pour débuter l’étude de rétro-ingénierie, une première analyse doit être menée pour définir la mission principale du système. La mission est l’ensemble des services rendus par le système. Elle peut donner lieu à des sous-missions. On peut aussi éventuellement, si on connait l’origine du système, compléter cette analyse en précisant la finalité recherchée et le problème auquel répond le système étudié.

Le questionnement peut être le suivant : ça sert à quoi (mission) ? Pourquoi l’avoir fait (finalité, problème) ?

## Définition de la mission :

Pour formaliser le résultat de l’activité de définition de la mission du système, on réalise un diagramme d’exigences (RD SysML) :

* Un système a une mission principale ou un service principal à remplir, on peut en déduire ses **missions** ou services (quoi) à remplir pour les parties prenantes :
  + Mettre l’exigence de mission au centre (on peut créer un stéréotype « Exigence - Mission » pour clarifier) ;
  + Le système à faire (bloc) y est relié par un lien de satisfaction (on peut créer un stéréotype « Système » pour clarifier) ;
* La mission peut-être affinée si nécessaire :
  + Décomposer la mission en sous-missions, utiliser des liens de décomposition ou des liens de dérivation « deriveReqt »[[1]](#footnote-1).

Souvent on utilise une liste numérotée pour les identifiants. Ceci permet de repérer rapidement et de hiérarchiser les besoins.

## Définition de la finalité et du problème :

La mission provient d’une **finalité** ou d’une raison d’être (pourquoi a-t-on fait ce système ?) qui a justifié la décision de réaliser ce système pour résoudre un **problème**  (quel est le problème que l’on a voulu solutionner ?).Ces éléments peuvent venir compléter le diagramme d’exigences précédent :

* + Mettre l’exigence de finalité et la relier par un lien de dérivation (on peut créer un stéréotype « Exigence - Finalité » pour clarifier) ;
  + Mettre une note stéréotypée « Problem » et la relier à la finalité ;

La définition de la finalité et du problème permet de mieux re-contextualiser l’ingénierie du système, mais elle n’est pas essentielle au processus de rétro-ingénierie et n’empêche pas de réaliser les activités suivantes. De plus les informations à propos de la finalité et du problème sous-jacent ne sont pas toujours disponibles.

Il est à noter que ce diagramme est provisoire, il pourra évoluer au fur et à mesure de l’étude de rétro-ingénierie. On peut aussi noter que ce type de présentation permet d’avoir une forme standard applicable à tous les systèmes.

NB : Les stéréotypes standards (« requirement », « block ») ne sont pas affichés pour simplifier la forme des diagrammes.

|  |
| --- |
|  |
|  |

# Activité RI1.3 – DÉfinir les cas d’utilisation

Le contexte étant défini, on s’attache à définir les fonctionnalités du système étudié pour chaque phase de vie où il rend des services.

Certaines phases de vie ne donnent pas toujours lieu à un service rendu par le système (conception, …). Il s’agit bien d’un service rendu par le système et non un service rendu au système. Par exemple pour une phase de maintenance réalisée par un opérateur, « faire la maintenance » n’est pas un cas d’utilisation car ce n’est pas le système qui réalise sa propre maintenance (c’est bien l’opérateur). En revanche en cas de capacité d’auto-diagnostique du système, le système rend bien un service à l’opérateur qui peut être alors un cas d’utilisation « Auto-diagnostiquer » ou « Tester ».

En général, la mission principale du système se retrouve souvent dans le cas d’utilisation principal de la phase exploitation.

Le questionnement peut être le suivant : Quels sont les usages du système (cas d’utilisation) ?

Pour formaliser les différentes utilisations du système, on réalise, en fonction de la complexité, au moins un diagramme de cas d’utilisation (UCD SysML) pour chacune des phases de vie où des services sont attendus du système :

* Mettre en évidence les frontières du système (Système boundary) ;
* Mettre les parties prenantes concernées (dans un rôle) par la phase de vie sous forme d’acteurs ;
* Mettre les systèmes externes, autres acteurs intervenant dans le cas d’utilisation, concernés par la phase de vie et les représenter par des blocs ;
* Définir les cas d’utilisation principaux correspondant aux services rendus par le système, il est alors souvent nécessaire d’utiliser le système pour mieux les identifier, ils peuvent être :
  + Nommés en utilisant de préférence un verbe d’action à l’infinitif ;
  + Décomposés (lien « include ») ;
  + Prolongés ou complétés en cas d’événement particulier ou d’option (lien « extend »).
* Associer les acteurs et les systèmes externes aux cas d’utilisation (lien d’association). Une bonne pratique est de mettre les acteurs et systèmes externes principaux à gauche et les secondaires à droite.

NB : Les acteurs et systèmes externes présents dans ces diagrammes sont forcément dans les précédents ! (La réciproque n’étant elle pas vraie, tous les acteurs et systèmes externes présents dans les diagrammes précédents ne sont pas systématiquement dans l’UCD)

|  |
| --- |
|  |
|  |

# DÉCRIRE la vue structurelle

L’architecture physique d’un système représente sa structure matérielle, c’est-à-dire sa composition ainsi que son organisation. Un système est par nature composé d’éléments, ceux-ci pouvant être de deux catégories possibles :

* Les composants : ce sont les entités élémentaires, que le modélisateur considère comme une entité non décomposable ;
* Les sous-systèmes : ces éléments sont des éléments décomposables, pouvant être composé d’autres sous-systèmes et/ou de composants.

Le questionnement peut être le suivant : Comment c’est fait et avec quoi (sous-système, composant) ?

Cette activité est formée de trois sous-activités permettant :

* De définir l’organisation des sous-systèmes et composants en décrivant les flux échangés entre eux et avec les éléments extérieurs au système étudié. Le diagramme de bloc interne (IBD) est utilisé pour décrire cela ;
* De définir la structure hiérarchique, c’est-à-dire de faire apparaitre l’ensemble des composants pouvant être regroupés au sein d’entités logiques (sous-systèmes) aboutissant à une vision hiérarchisée. Pour cela le diagramme de définition de bloc (BDD) est utilisé ;
* Pour ensuite définir l’architecture logique.

Les deux premières activités sont deux entrées possibles, le choix de commencer par l’une ou par l’autre va dépendre du modélisateur. C’est aussi souvent une suite d’allers-retours entre les deux (c’est le sens du parallélisme dans le diagramme d’activité).

# Activité RI2.1 –DÉcrire lA structure interne

Cette activité a pour objectif de représenter l’architecture interne du système étudié et les flux échangés par les éléments (composants ou sous-systèmes). Les flux peuvent être échangés entre eux ou avec les éléments externes du système. Pour cela la notion de port est essentielle : les ports représentent les interfaces d’émission ou de réception de ces flux.

* Il faut définir les ports de chaque élément (système, sous-système et composant) :
  + Pour des flux de contrôle ou d’information, créer des ports « Port » ;
  + Pour des flux d’énergie, créer des ports de flux « Flow Port » ;
  + Si plusieurs ports appartiennent à une même interface, créer des ports imbriqués « Nested Port » ;
* Relier les ports par des connecteurs représentant les flux et indiquer les flux échangés.

Pour des ports standards, il est possible de représenter les informations reçues et émises à l’aide d’interfaces : « required interface » pour les informations reçues, « provided interface » pour les informations émises.

|  |
| --- |
|  |
|  |

# Activité RI2.2 –DÉcrire lA structure hiÉrarchique

Cette activité a pour objectif de représenter la structure classique d’un système, composé de sous-systèmes et de composants de manière hiérarchique. Il montre un premier niveau de décomposition du système en éléments qui peuvent eux-mêmes être décomposés jusqu’à obtenir uniquement des composants.

Le niveau de décomposition est du ressort du modélisateur qui, en fonction de son objectif, pourra aller plus ou moins loin dans la décomposition : par exemple un moteur électrique peut être vu comme un composant, ou comme un élément encore décomposable (stator, rotor, …).

Il est à noter que ce choix a une forte influence sur la compréhension du système et il est recommandé de ne pas décomposer trop finement. Si le modélisateur souhaite approfondir un élément, il pourra le faire en réalisant un second diagramme montrant la composition de l’élément seul.

Enfin, il peut ne pas être évident de pouvoir regrouper les composants en sous-système, et la plupart du temps il y a différentes manières d’y parvenir. Ce choix revient au modélisateur en fonction de son objectif et de ses connaissances sur le système.

Il est possible de procéder de deux manières différentes : de manière ascendante (regroupement d’éléments pour former un sous-système) ou de manière descendante (décomposition d’un sous-système en éléments). Ces deux manières peuvent également être réalisées simultanément, lorsque certaines parties du système sont analysées de manière ascendante et d’autres parties de manière descendante.

Pour réaliser un diagramme de définition de bloc :

* Représenter par un bloc chaque élément ;
* Utiliser des stéréotypes adaptés « système », « sous-système », « composant » pour simplifier la lecture ;
* Relier le bloc à décomposer avec les autres blocs par un lien de composition.

|  |
| --- |
|  |
|  |

# DÉcrire lA vue comportementale

Il s’agit ici de décrire les missions du système. Les diagrammes d’états et de séquence sont deux types de diagrammes permettant de décrire, sous différents points de vue, le comportement du système (descriptif). On peut aussi utiliser des diagrammes d’activité (prescriptif), mais il n’est pas traité dans ce document.

Le scénario associé à chaque cas d’utilisation doit être décrit pour mettre en évidence le comportement du système au cours des utilisations.

Le questionnement peut être le suivant : Comment ça marche et dans quel ordre (opérations, états) ?

### Cette activité doit permettre, pour chaque scénario d’utilisation, et donc pour chaque cas d’utilisation, de :

* Décrire les interactions entre le système étudié et son environnement, réalisé à l’aide de diagramme de séquence ;
* Décrire les états du système au cours du scénario, réalisé à l’aide de diagramme d’état.

Cette approche peut être suivie ensuite d’une étude plus poussée au niveau des sous-systèmes.

# Activité RI3.1 – DÉcrire les intéractions du système

Pour formaliser les interactions du système avec son environnement, on réalise pour chaque cas d’utilisation au moins un diagramme de séquence (SD SysML) basé sur les scénarios d’utilisation qui peuvent être fourni par un utilisateur du système (et qui donc connait son fonctionnement) ou par la documentation du système (manuel de fonctionnement, documentations techniques…).

Pour réaliser un diagramme de séquence :

* Le système est au centre des préoccupations :
  + Mettre le bloc « système » au milieu ;
* Identifier les éléments externes en interaction avec le système :
  + Mettre les acteurs identifiés (opérateur, mainteneur, …) à gauche du système ;
  + Mettre les entités externes (autres systèmes, matériels, …) à droite du système ;
* Définir les interactions entre le système et ces autres éléments :
  + Mettre les messages SysML (flux d’information, de matière ou d’énergie) entre les lignes de vie ;
    - Utiliser un message asynchrone (asyncSignal en SysML v1.3) si l’émetteur n’est pas bloqué en attente d’une réponse. On le représente avec une flèche évidée. Associer un message réflexif pour supporter l’opération réalisée par le système. On le représente avec une flèche pleine qui boucle sur la période d’activation du système ;
    - Utiliser un message synchrone, syncCall(param) en SysML v1.3, pour des messages nécessitant une réponse (retour), on le représente avec une flèche pleine et la réponse par une flèche en pointillé ou un message asynchrone, asyncCall(param) en SysML v1.3, pour des messages ne nécessitant pas une réponse (retour), on le représente avec une flèche ouverte. Ils permettent de supporter l’opération déclenchée dans le système ;
* Définir les périodes d’activation du système :
  + Structurer des périodes d’activation du système en regroupant les périodes correspondant à des situations d’exécution des opérations et en les séparant lors de changements de situation ou lors de périodes d’inactivité ;
  + Utiliser un message réflexif pour supporter une opération réalisée par le système. On le représente avec une flèche pleine qui boucle sur la période d’activation du système ;
* Définir si nécessaire la référence à une autre séquence d’interaction :
  + Utiliser un cadre « Ref » et lui associer un autre diagramme de séquence.

|  |
| --- |
|  |
|  |

# Activité RI3.2 – DÉfiniR Les États du systÈme

L’étape suivante consiste à identifier les états du système, celui-ci étant vu comme une boite noire. Il est possible sur la base des cas d’utilisations et des modes opératoires associés de définir les états du système et les conditions de passage de l’un à l’autre. Ces états représentent fréquemment un mode de marche du système, sans être pour autant une règle absolue.

Dans un premier temps, et afin de réaliser un diagramme d’état « système », chaque cas d’utilisation déjà identifié peut être formalisé, sans que cela soit automatique, par un état du système dans un diagramme d’état (SMD SysML) :

* Identifier chaque situation de vie du système sur la base des cas d’utilisation :
  + Créer les états correspondants, utiliser des états « composites » pour représenter les cas d’utilisation inclus et des états « orthogonaux » pour représenter du parallélisme ; Les états élémentaires (non composé d’autres états) peuvent être des états « sous-machine », c’est-à-dire que leur détail est donné par un autre diagramme d’état permettant ainsi de ne pas alourdir la lecture ;
  + Créer un état initial et créer au moins un état final ;
* Identifier les chemins possibles suivant les scénarios opérationnels (modes de marche, …) et les conditions associées :
  + Relier les différents états ;
  + Identifier les conditions de passage d’un état à l’autre, en associant des évènements et éventuellement des conditions de garde en fonction des situations. On peut aussi utiliser la condition de garde [sinon] ou [else] ;
    - ATTENTION : Les chemins issus d’un état doivent au final toujours être mutuellement exclusifs (évènements ou conditions de garde exclusifs)!
    - Normalement un diagramme d’état doit utiliser des évènements comme transition entre états, si on a majoritairement des conditions de garde cela signifie sûrement que l’utilisation d’un diagramme d’activité serait mieux adaptée !

Les états supplémentaires comme « En veille » ou « Attente », mais aussi des états comme « Initialisation en cours », « Arrêt d’urgence en cours » peuvent être définis. Ils dépendent largement de la connaissance du modélisateur. Il ne faut pas négliger l’aspect itératif de la démarche qui amène à remettre en cause autant que nécessaire l’ensemble des modèles.

En fonction des objectifs de la modélisation, les opérations associées à l’aide des messages récursifs, peuvent être rappelées dans les états en précisant si celles-ci sont exécutées lors de l’entrée dans l’état (« entry »), tout au long de celui-ci (« do ») ou en en sortant (« exit »).

|  |
| --- |
|  |
|  |

# Activité RI4.1 – DÉcrire les Échanges avec les sous-systÈmes

Pour formaliser les interactions entre les différents sous-systèmes entre eux et avec l’environnement, on réalise pour chaque cas d’utilisation au moins un diagramme de séquence (SD SysML) en cohérence avec les précédents. On peut limiter cette étude à certains éléments en fonction des préoccupations et des objectifs visés (contrôle commande, mécanique, …).

Pour réaliser un diagramme de séquence pratiquer comme précédemment :

* Le (ou les) sous-système est au centre des préoccupations :
  + Mettre le(s) bloc(s) « sous-système » au milieu ;
* Identifier les éléments externes en interaction avec le sous-système :
  + Mettre les acteurs identifiés (opérateur, mainteneur, …) à gauche du sous-système ;
  + Mettre les entités externes (autres systèmes, matériels, …) à droite du sous-système ;
* Définir les interactions entre le sous-système et ces autres éléments :
  + Mettre les messages SysML entre les lignes de vie ;
* Définir les périodes d’activité du sous-système :
  + Structurer des périodes d’activité.

|  |
| --- |
|  |
|  |

# Activité RI4.2 – DÉcrire les États des sous-systÈmes

Réalisée en parallèle de l’étude des interactions, cette activité consiste à décrire des états plus élémentaires du système. A partir des diagrammes de séquence, il est possible de définir des états et des conditions d’évolution associées.

Chaque état obtenu dans le diagramme d’état du système est lui-même décrit par un diagramme d’état plus élémentaire (STMD SysML), pratiquer comme précédemment :

* Identifier les états du système au cours du scénario (période « d’activation » au sens SysML) associé à un cas d’utilisation :
  + Créer les états correspondants si nécessaire on peut, là encore, utiliser des états « composites », « orthogonaux » et « sous-machine » ;
  + Créer un état initial et les éventuels états finaux ;
* Définir les transitions suivant les interactions des diagrammes de séquence :
  + Relier les différents états.

|  |
| --- |
|  |
|  |

# Activité RI5 – DÉCRIRE l’ARCHITECTURE logique

Les activités précédentes ont permis la mise en évidence d’une part les opérations du système et son architecture. Cette étape permet d’allouer les opérations aux différents éléments de l’architecture.

Réaliser un (ou plusieurs) diagramme de définition de bloc (BDD SysML) ou reprendre les précédents :

* Mettre le système et/ou ses sous-systèmes dans le diagramme :
  + Les opérations et signaux reçus doivent apparaitre automatiquement lorsque les diagrammes de séquences et/ou d’état ont été réalisés précédemment ;
  + Sinon, les opérations et signaux reçus peuvent être ajoutés au bloc.

|  |
| --- |
|  |

# Activité RI6 - DÉfinir les exigences systÈmès

Une dernière étape permet de faire ressortir les caractéristiques et contraintes auxquelles le système répond. Pour autant cette étape n’est pas aisée car elle dépend très largement de la connaissance de l’origine du système. La documentation fournie par le constructeur est une aide précieuse. L’ensemble est formalisé par les exigences satisfaites par le système..

### Définition d’une exigence d’après l’AFIS

Une exigence prescrit une propriété jugée nécessaire. Son énoncé peut être :

* un service ou une fonction (Transporter …) ;
* une caractéristique (Tenir dans un volume de …) ;
* une aptitude (Supporter une température de …) ;
* ou une limitation (Utiliser uniquement …) ;

à laquelle doit satisfaire un système, un produit ou un processus.

### Rédaction d’une exigence d’après l’AFIS

Lors de la rédaction, il est recommandé de commencer par un verbe à l’infinitif, avec un verbe d’action pour les exigences fonctionnelles.

### Caractéristiques des exigences d’après l’AFIS

Afin d’obtenir la rigueur nécessaire à une bonne spécification, une exigence doit présenter des caractéristiques de qualité telles que :

* unicité - une exigence ne traite que d’un sujet ;
* précision - une exigence est rigoureuse dans son expression ;
* non ambiguïté - une exigence ne permet qu’une interprétation possible ;
* pure prescription de résultat - une exigence porte sur le quoi ? (service/fonction attendu), non sur le comment ? On est dans l’espace du problème ! ;
* vérifiabilité - à toute exigence est associée au moins une méthode permettant de vérifier son obtention (il serait inutile d’écrire une exigence dont on ne saurait montrer que le système y satisfait) ;
* faisabilité - une exigence doit pouvoir être satisfaite dans le contexte de l’état de l’art technologique, à l’horizon envisagé ;
* réalisme - une exigence doit pouvoir être satisfaite dans le contexte des contraintes du projet.

NB : Ces deux dernières caractéristiques correspondent à la prise en compte des besoins des parties prenantes impliquées dans la réalisation.

### Formaliser les exigences système

Afin d’aider à la définition des exigences système et de faciliter leur lecture, on peut ajouter les identifiants et les stéréotypes suivants :

EFi – « Exigence - Fonctionnelle » ;

EOj – « Exigence - Opérationnelle » (mode de fonctionnement, condition d’évolution, …) ;

EPk – « Exigence - Performance » ;

EIx – « Exigence - Interface » (physique, ergonomie, interopérabilité, …) ;

ECy – « Exigence - Contrainte » (liée à une phase de vie, environnement du système, règlementation, coût, délai, …) ;

EVz – « Exigence - Validation » (Tests ou essais, inspections, revues ou audits, …).

On peut noter que dans un contexte particulier (organisation, industrie, secteur d’activité, …) des types beaucoup plus détaillés peuvent être utilisés.

|  |
| --- |
|  |
|  |

# Activité RI6 - DÉfinir les exigences systÈmès

Les études précédentes permettent de remonter aux exigences satisfaites par le système. Pour formaliser les exigences système, on réalise, en fonction de la complexité, au moins un diagramme d’exigences (RD SysML) pour chacune des phases de vie :

* Chaque cas d’utilisation est associé à une exigence fonctionnelle formalisée par une exigence :
  + Utiliser le lien « refine » pour les associer, chaque cas d’utilisation est décrit par des diagrammes de séquence et/ou d’état, faire en sorte que chaque cas d’utilisation soit représenté par une exigence fonctionnelle ;
  + Utiliser les stéréotypes « Exigence - Fonctionnelle » ou « Exigence - Mission »;
  + Utiliser des liens de dérivation « deriveReqt »[[2]](#footnote-2) en relation avec les liens « include » et « extend » des cas d’utilisation.

Elles peuvent être complétées sans forcément rechercher l’exhaustivité (on doit souvent avoir un peu d’imagination pour ce mettre à la place des concepteurs du système !) :

* Des exigences de performance peuvent y être associées en se posant la question pour chaque cas d’utilisation « Quelles sont (étaient) les performances attendues ? » :
  + Utiliser des liens de dérivation « deriveReqt » pour les relier aux exigences fonctionnelles concernées, utiliser les stéréotypes « Exigence - Performance » ;
    - NB : en rétro-ingénierie les performances réalisées par le système sont au minimum du niveau des exigences puisque satisfaites par le système réel ! Exemple pour un système qui à un temps de réponse mesuré de 4,5s : l’exigence de performance correspondante pourrait être « Répondre en au plus 5s »
* Pour chaque interaction (SD et IBD SysML), on pose la question « Quelles sont les interfaces permettant l’interaction ? » pour définir les exigences d’interface :
  + Utiliser des liens de dérivation « deriveReqt » pour les relier aux exigences fonctionnelles concernées, utiliser les stéréotypes « Exigence - Interface ».
* A partir des transitions du diagramme d’état du système on peut définir des exigences opérationnelles :
  + Utiliser des liens de dérivation « deriveReqt » pour les relier aux exigences fonctionnelles concernées, utiliser les stéréotypes « Exigence - Opérationnelle ».
* En élargissant le périmètre des parties prenantes déjà prises en compte dans les diagrammes de contexte (concepteur, organisme de certification, …), on pose la question « Quelles sont (étaient) les contraintes de la partie prenante dans ce contexte ? » pour définir les exigences de contrainte :
  + Utiliser des liens de dérivation « deriveReqt » pour les relier aux exigences fonctionnelles concernées, utiliser les stéréotypes « Exigence - Contraintes ».
* On peut aussi, pour chaque exigence fonctionnelle, se poser la question « Existe-il des modalités spécifiques de validation ? » pour définir les exigences de validation :
  + Utiliser des liens de dérivation « deriveReqt » pour les relier aux exigences fonctionnelles concernées, utiliser les stéréotypes « Exigence - Validation ».
* Pour justifier un élément (exigence, lien, …) on peut utiliser les notes de justification :
  + Utiliser les notes stéréotypées « Rationale » et les relier aux éléments à justifier.
* Pour indiquer comment on peut vérifier une exigence en spécifiant un cas de test :
  + Utiliser les « testCase » pour définir des diagrammes de séquence ou d’état décrivant les scénarios de test.
  + Utiliser des liens de vérification « verify » pour les relier aux exigences système.

|  |
| --- |
|  |
|  |

# Activité RI7 - ASSOCIER les exigences systÈmès

La dernière étape de la rétro-ingénierie consiste à monter comment les exigences sont satisfaites par les opérations, systèmes, sous-systèmes et autres composants.

Pour cela on peut reprendre les diagrammes d’exigences pour les compléter en notant les liens de satisfaction. Cette méthode est très lourde dès que le nombre d’exigence est important. On peut préférer utiliser de matrice de traçabilité pour mettre en place ces liens de satisfaction. Cette méthode permet de plus de faire en même temps la vérification que toutes les exigences sont bien satisfaites.

Pour formaliser la satisfaction des exigences réaliser des matrices de traçabilité ou des diagrammes d’exigence (RD SysML) :

* Allouer les exigences systèmes aux éléments de type opération, sous-système ou composant (lien de satisfaction) :
  + Les exigences fonctionnelles sont satisfaites par les opérations ou du fait des opérations supportées par l’élément considéré ;
  + Les exigences non fonctionnelles doivent être satisfaites par un élément ou par le système (décomposition si besoin).

Toutes les exigences système doivent être satisfaites !

|  |
| --- |
|  |

# Synthèse

Diagrammes SysML manipulés lors de la rétro-ingénierie d’un système.

1. Le lien de décomposition/contenance n’est pas très clair dans SysML (lien de classement avec peu de sémantique), de nombreux auteurs préconisent de ne pas l’utiliser et de le remplacer par « deriveReqt » ou « refine ». [↑](#footnote-ref-1)
2. Le lien de décomposition/contenance n’est pas très clair dans SysML (lien de classement avec peu de sémantique), de nombreux auteurs préconisent de ne pas l’utiliser et de le remplacer par « deriveReqt » ou « refine ». [↑](#footnote-ref-2)