CHAPITRE 4

ANALYSE DES BESOINS DES UTILISATEURS

4.1 Introduction

Afin de fonder la conception logicielle sur des bases solides, un sondage a été développé pour consulter un groupe d'utilisateurs. Suite à cette consultation, des spécifications logicielles ont été dégagées afin d'orienter le développement de la solution.

4.2 Création d'un sondage

Un sondage a été conçu afin de connaître quels sont les aspects prioritaires à considérer lors de la conception, tant dans celle du processus de migration que dans celle du produit de la migration. Le processus de migration est caractérisé par tous les éléments qui conduisent à la préparation du code migré tandis que le produit de migration est ce qui sera exécuté dans le contrôleur.

Le sondage se divise en deux volets, soit un premier volet plus global concernant les qualités logicielles requises puis un second volet portant sur les attributs quantifiables.

4.3 Qualités et attributs logiciels

Les qualités logicielles, tirées de la norme ISO-9126-1 [63], ont été présentées aux utilisateurs dans le but d'être priorisées. Toutes les qualités de la norme ont été utilisées de façon à ne rien omettre. Cette priorisation des qualités est destinée à faire des choix techniques lors de la conception.

Les attributs associés à ces qualités sont également décrits afin de rendre la priorisation des qualités plus concrète. Les qualités et les attributs contenus dans cette norme sont décrits dans cette section. Cette norme n'est pas disponible en version française. Les définitions sont traduites et adaptées du texte de la norme ISO-9126-1 [63] sauf lorsqu'une autre source est spécifiée.

4.3.1 Fonctionnalité

La fonctionnalité est la qualité logicielle caractérisant un bien ou un service qui répond à sa finalité, et qui est donc fonctionnel et pratique [21]. En bref, c'est la capacité du logiciel à remplir son mandat selon les besoins qui ont été énoncés au départ. La qualité de fonctionnalité est associée aux attributs d'aptitude, de précision, d'interopérabilité et de sécurité.

- Aptitude: capacité d'un logiciel à fournir un ensemble de fonctions afin d'atteindre les objectifs spécifiques des utilisateurs.

- Précision: qualité, état ou degré de conformité par rapport à un standard reconnu ou par rapport à une spécification [21].

- Interopérabilité: capacité que possèdent des systèmes informatiques hétérogènes à fonctionner conjointement, grâce à l'utilisation de langages et de protocoles communs, et à donner accès à leurs ressources de façon réciproque [21].

- Sécurité: fonction qui permet de s'assurer que les services sont employés de façon adéquate par les clients appropriés [21].

La fonctionnalité s'illustre dans notre cas comme étant la capacité du protocole soit d'effectuer une migration réussie ou de générer un produit de migration effectuant les calculs de façon exacte et d'être contrôlé de l'extérieur par le contrôleur global.

4.3.2 Fiabilité

La qualité de fiabilité est la propriété d'un système informatique désignant sa capacité d'assurer ses fonctions sans défaillance, dans des conditions préalablement définies et sur une période déterminée [21]. La qualité de fiabilité est associée aux attributs de maturité, tolérance aux fautes et récupération.

- Maturité: capacité d'un logiciel à éviter les pannes lorsqu'il y a des erreurs dans ce dernier.

- Tolérances aux fautes: capacité d'un logiciel à fonctionner conformément à ses spécifications en cas de défaillance d'un ou de plusieurs de ses constituants [adapté de 21].

- Récupération: capacité d'un logiciel de rétablir un niveau de performance spécifique et de récupérer les données affectées suite à une panne.

La fiabilité s'illustre dans notre cas comme étant la génération automatique d'une loi de contrôle qui ne tombe pas en panne lors de son exécution ou qui permet au contrôleur global de détecter une panne qui survient dans le produit de migration.

4.3.3 Convivialité

La convivialité est la qualité d'un matériel ou d'un logiciel à être facile et agréable à utiliser et à comprendre, même par quelqu'un qui a peu de connaissances en

informatique [21]. La qualité de convivialité est associée aux attributs de facilité de compréhension, de facilité d'apprentissage, de facilité d'opération et de pouvoir d'attraction.

- Facilité de compréhension: capacité d'un logiciel à permettre à l'usager de saisir à quoi il sert et quelles sont ses conditions d'utilisation.

- Facilité d'apprentissage: qualifie un matériel ou un logiciel dont l'apprentissage est agréable et facile, même pour quelqu'un qui a peu de connaissances en informatique [21].

- Facilité d'opération: capacité d'un logiciel à permettre à l'usager de l'opérer et d'en avoir le contrôle.

- Pouvoir d'attraction: capacité d'un logiciel d'être attirant pour l'utilisateur.

Par exemple, la convivialité dans notre contexte est l'intégration du protocole à l'environnement de développement, le temps d'apprentissage requis pour utiliser le protocole de migration à pleine capacité, le temps requis pour effectuer une migration et le nombre d'itérations nécessaires pour y parvenir.

4.3.4 Efficacité

La qualité d'efficacité représente la capacité du logiciel à performer adéquatement relativement aux ressources disponibles selon certaines conditions [21]. La qualité d'efficacité est associée aux attributs de temps de réponse par cycle et d'utilisation des ressources.

- Temps de réponse: capacité d'un logiciel à fournir une réponse appropriée en un temps voulu lorsqu'il remplit sa fonction dans des conditions données.

- Utilisation des ressources: capacité du logiciel à utiliser une quantité et un type de ressources appropriées lorsque ce dernier est exécuté dans des conditions données.

L'efficacité est dans notre cas le temps réponse pour un cycle de calcul, ou encore la mémoire vive requise durant l'exécution et l'espace disque qu'occupe le produit de migration.

4.3.5 Maintenabilité

La qualité de maintenabilité est la capacité du logiciel à être modifié. Les modifications peuvent inclure les corrections, les améliorations, ou l'adaptation d'un logiciel aux changements dans son environnement ou dans ses spécifications. La qualité de maintenabilité est associée aux attributs de facilité d'analyse, de facilité de modification, de stabilité et de facilité de test.

- Facilité d'analyse: capacité d'un logiciel à être diagnostiqué pour des défauts ou des pannes.

- Facilité de modification: capacité d'un logiciel à permettre une modification donnée.

- Stabilité: capacité d'un logiciel à éviter les effets inattendus suite à une modification au logiciel.

- Facilité de test: capacité d'un logiciel à être valider lorsque des modifications sont effectuées.

La maintenabilité s'illustre dans notre cas comme étant la capacité à effectuer des modifications dans le produit de migration ainsi que de permettre de tester les fonctionnalités et la stabilité d'un produit de migration avant son utilisation.

4.3.6 Portabilité

La qualité de portabilité est la capacité d'un logiciel à être transporté d'un environnement à un autre. La qualité de Portabilité est associée aux attributs de facilité d'adaptation, de facilité d'installation, de coexistance et d'interchangeabilité.

- facilité d'adaptation: capacité d'un logiciel à être adapté pour différents environnements sans appliquer des actions ou des moyens autres que ceux fournis à cette fin pour le logiciel.

- facilité d'installation: capacité d'un logiciel à être installé dans un environnement spécifique.

- Coexistence: capacité d'un logiciel à coexister avec un autre logiciel indépendant dans un environnement commun partageant les mêmes ressources.

- Interchangeabilité: capacité d'un logiciel à être utilisé à la place d'un autre pour la même application dans le même environnement.

La portabilité dans notre cas est la capacité d'un produit de migration à être exécuté et installé sur plusieurs systèmes d'exploitation (Windows, QNX, etc.) tout en s'intégrant au contrôleur global développé avec le CAR Microb.

4.4 Objectifs quantifiables

Les qualités et attributs logiciels décrits précédemment permettent de définir l'orientation globale du logiciel à concevoir. Toutefois, elles restent peu précises et leur évaluation reste floue. L'utilité de fixer des objectifs quantifiables permet de préciser le résultat plus aisément. La ligne directrice de ces objectifs a été initialement proposée par l'auteur lors de l'élaboration d'un sondage réalisé sur un groupe d'utilisateurs potentiels du logiciel. Ce sondage a permis d'ajouter de nouveaux objectifs quantifiables (voir ANNEXE 1).

Il est prévisible que l'utilisation du protocole de migration conduira à une perte de performance mesurable, telle l'augmentation du temps de cycle, de la mémoire vive requise ou de l'espace disque utilisée. Une perte de performance aux alentours de 10% est considérée comme acceptable dans la littérature [51]. Par exemple, une augmentation du temps de cycle de 10% pour le contrôleur global à cause de l'utilisation du protocole de migration est admissible.

Le sondage réalisé, dont les résultats sont présentés plus bas, a également permis de prioriser certains objectifs:

- Minimiser le temps de cycle lors de l'exécution du code généré;
- Minimiser la mémoire vive utilisée lors de l'exécution du code généré (RAM);
- Minimiser l'espace disque utilisé pour le code généré (ROM);
- Minimiser l'utilisation du processeur pour la génération du code (CPU);
- Minimiser le temps de migration total.

En plus de la priorisation des objectifs quantifiables, quelques questions ouvertes ont été soulevées lors de ce sondage. Ces questions sont décrites à la section 4.4.6 tandis que les objectifs priorisés sont décrits dans les paragraphes suivants.

4.4.1 Minimiser le temps de cycle

Temps d'exécution moyen pour un cycle du système après un grand nombre de cycles d'exécution. Un court temps d'exécution est souhaitable car certains calculs doivent être faits à très haute fréquence.

4.4.2 Minimiser la mémoire vive utilisée (RAM)

La mémoire vive occupée par le code généré doit être considérée car il peut s'agir d'une ressource critique au sein d'un contrôleur embarqué.

4.4.3 Minimiser l'espace disque utilisé (ROM)

L'espace disque requis par le code généré doit être considéré car il peut s'agir d'une denrée rare au sein d'un contrôleur embarqué.

4.4.4 Minimiser l'utilisation du processeur (CPU)

Minimiser l'utilisation du processeur lors de la génération du code pour permettre l'utilisation du protocole de migration sur une machine peu performante.

4.4.5 Minimiser le temps de migration total

L'opération de migration, c'est-à-dire du moment de la génération du code jusqu'à son exécution, doit être d'une durée la plus courte possible. Le facteur qui permet de réduire le temps de migration est directement lié au fait de devoir ou non recompiler le contrôleur global. Cette action peut demander plusieurs minutes, tout dépendant de la dimension de ce dernier. Il serait souhaitable de ne pas avoir à recompiler à chaque modification du code pour que l'utilisation du protocole soit la plus rapide possible.

4.4.6 Autres objectifs quantifiables

Deux questions ouvertes sont posées afin d'aller chercher des balises plus précises pour le protocole de migration. Le but de la première questions est de chiffrer le temps de migration total maximal, c'est-à-dire de Simulink jusqu'à l'exécution du code sur le contrôleur physique. La seconde question avait pour objectif de connaître le temps d'apprentissage maximal acceptable pour le protocole, c'est-à-dire le temps requis pour installer, comprendre et utiliser pleinement ce dernier. De plus, une rubrique supplémentaire appelée « autres considérations » a été ajoutée de façon à ce que les gens sondés puissent ajouter certains éléments qui auraient pu être omis (voir ANNEXE 1).

4.5 Consultation d'un groupe d'usagers

La consultation des usagers potentiels du protocole de migration a été réalisée à l'IREQ le 25 octobre 2005. La session a été animée par l'auteur du présent mémoire. Huit personnes étaient présentes lors de la consultation et provenaient soit de l'ÉTS ou de l'IREQ. Les personnes choisies pour effectuer la consultation avaient des connaissances pratiques en programmation de contrôleur de robot, en contrôle de robots et avaient déjà utilisé soit un outil de PRC ou un CAR ou les deux.

Une présentation globale du projet a été faite afin d'introduire les participants au contexte du projet de recherche. Un questionnaire (voir annexe 1) a été distribué à chaque participant pour leur permettre d'inscrire leurs réponses et également de leur rappeler la définition des qualités et attributs logiciels. Le questionnaire a été complété dans un premier temps sur une base individuelle puis les réponses obtenues ont été discutées en groupe. Le groupe tentait dans un premier temps de décider de façon consensuelle la priorisation finale. Lorsqu'un consensus n'était pas possible, l'assemblée passait au vote et, en cas d'égalité, une personne désignée tranchait sur la décision finale.

4.6 Résultats de la consultation

Suite à cette consultation, la priorisation des qualités logicielles a été effectuée et est présentée dans cette section.

4.6.1 Priorités des qualités du protocole de migration

Les qualités logicielles pour le protocole de migration ont été priorisées de la façon suivante, du plus important au moins important:

1. fonctionnalité;
2. convivialité;
3. fiabilité;
4. maintenabilité;
5. efficacité;
6. portabilité.

4.6.2 Priorités des qualités du produit de migration

Les spécifications logicielles du produit de migration ont été priorisées de la façon suivante, du plus important au moins important:

1. fiabilité;
2. fonctionnalité;
3. efficacité;
4. portabilité;
5. convivialité;
6. maintenabilité.

4.6.3 Priorité des objectifs quantifiables

Les objectifs ont été priorisées de la façon suivante, du plus important au moins important:

1. Minimiser le temps de cycle lors de l'exécution du code généré;
2. Minimiser le temps de migration total;
3. Minimiser la mémoire vive utilisée lors de l'exécution du code généré (RAM);
4. Minimiser l'espace disque utilisé pour le code généré (ROM);
5. Minimiser l'utilisation du processeur pour la génération du code (CPU).

Lors du sondage, il a été également mentionné que le cinquième objectif, soit l'utilisation du processeur lors de la génération du code, était peu important parce que les plateformes de PRC sont généralement pourvues d'une grande puissance de calcul.

Les questions ouvertes ont permis de déterminer que la migration devait durer environ une minute et que l'apprentissage du protocole devrait s'étendre de 2h à une journée. De plus, les usagers ont demandé que le code généré soit une boîte noire et qu'il ne soit pas nécessaire d'aller le modifier.

4.7 Conclusion

Ce chapitre a permis d'exposer la démarche qui a mené à l'obtention des priorités en terme de qualités logicielles pour le produit et le protocole de migration. Une étude réalisée auprès d'utilisateurs potentiels a permis de fixer ces priorités. De plus, des objectifs quantifiables ont été priorisés lors de cette étude de façon à effectuer des choix technologiques sans ambiguïté pour satisfaire les besoins des utilisateurs. Les réponses obtenues dans ce chapitre permettront d'effectuer les choix du produit de migration, qui fait l'objet du prochain chapitre.