MCours.com

CHAPITRE 6

DÉVELOPPEMENT DU PROCESSUS DE MIGRATION

6.1 Introduction

Ce chapitre décrit le fonctionnement du processus de migration. Il tient compte des chapitres précédents qui ont permis de fixer les qualités logicielles prioritaires et de choisir la bibliothèque dynamique pour encapsuler la loi de commande élaborée à l'aide du PRC. Dans cette section, le choix des logiciels, la séquence de migration ainsi que la stratégie d'implémentation du processus sont présentés.

6.2 Logiciels utilisés

Le choix des logiciels est requis pour débuter le développement du protocole de migration. Le processus de migration doit être développé dans un environnement de simulation donné et doit permettre de s'adapter à un CAR particulier. Le choix de cet environnement de simulation et du CAR utilisé est expliqué dans les lignes qui suivent.

Quelques environnements de simulation existent et les trois principaux qui possèdent un générateur de code sont Matlab / Simulink, SCILAB / SCICOS et NI MATRIXx / SystemBuild (voir section 1.6.4). L'environnement PRC sélectionné pour développer le processus de migration est Matlab / Simulink muni du générateur de code RTW. Quelques raisons motivent ce choix. Premièrement, l'IREQ utilise déjà les outils de Mathworks dans plusieurs projets. Deuxièmement, les plateformes cibles supportées sont plus nombreuses. Troisièmement, cette suite d'outil semble avoir une crédibilité plus grande car elle existe depuis plus longtemps et qu'elle est supportée par une compagnie reconnue. Il est à noter que les outils Mathworks sont également abondamment utilisés dans les milieux académiques tel que l'ÉTS.

Le CAR choisi est celui présentement utilisés par Hydro-Québec, soit Microb (Module intégré de Contrôle Robotique). Il n'a pas été comparé en profondeur aux autres cadres d'application liés au domaine robotique (voir section 1.6.3) car le partenaire industriel désire utiliser le CAR qu'il a développé et qu'il utilise depuis plusieurs années sur différents projets robotiques.

Les deux outils logiciels choisis sont donc le CAR Microb d'Hydro-Québec et la suite Matlab / Simulink / RTW développée par Mathworks. À partir de ce choix technologique, il est maintenant possible de décrire le déroulement de la séquence de migration.

6.3 Description de la séquence de migration

La migration consiste à utiliser la plateforme de simulation Simulink pour générer le code exécutable qui sera appelé par le contrôleur global sur la plateforme d'exécution.

L'activité de migration est unidirectionnelle et se divise en trois étapes (voir Figure 9). Les deux premières étapes sont exécutées sur la plateforme de simulation tandis que la dernière étape est destinée à être réalisée sur la plateforme d'exécution, c'est-à-dire celle sur laquelle est exécuté le contrôleur. Il est à noter que la plateforme de simulation et d'exécution peuvent être sur le même ordinateur.

La première partie consiste à générer le code C du modèle. L'exécution de cette action est assurée par la chaîne d'outils Matlab / Simulink / RTW. La seconde partie, l'encapsulation en une bibliothèque dynamique, est effectuée avec la même chaîne d'outil mais avec une configuration personnalisée. La troisième partie consiste à compiler le code destiné à être utilisé par le contrôleur global.

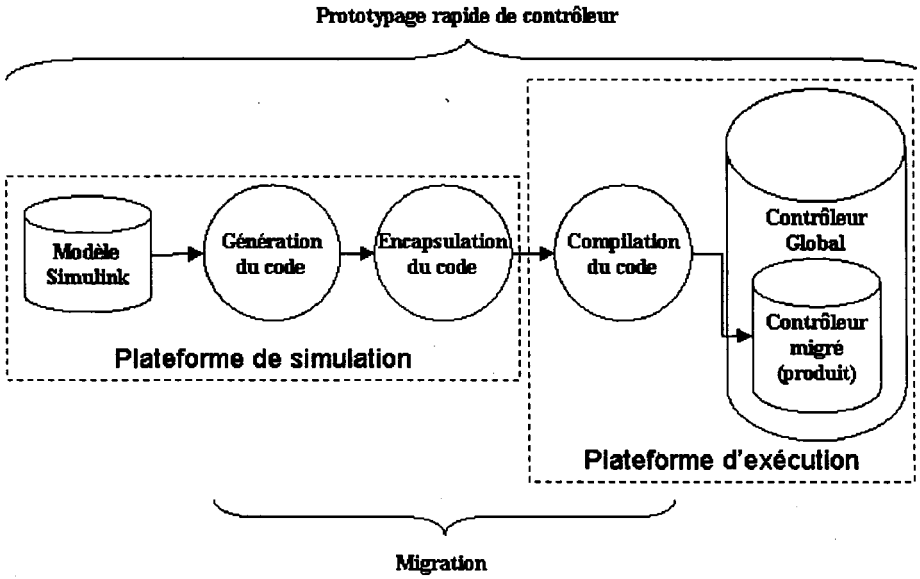

Figure 9 Illustration du processus de migration

La génération de code à partir d'un modèle Simulink s'effectue grâce à RTW. Ce mécanisme est illustré dans la Figure 10. La boîte à outils, en se basant sur les informations du modèle, génère un fichier contenant l'ensemble des boîtes à générer. Ce fichier a pour extension *rtw* (ex: *model.rtw*). À partir de ce fichier, plusieurs fichiers contenant du code C (ex: *model.c, model.h, etc.*) sont générés en utilisant les fichiers *Target Language Compiler* (TLC) associé à chacun des blocs utilisés dans le modèle. Ces fichiers définissent ce qui doit être fait à l'intérieur du bloc et RTW s'assure que les signaux sont bien acheminés à chaque bloc pour être traités. Finalement, un fichier de compilation (makefile) est généré (ex: *model.mk*) à partir d'un fichier *Template Make File* (TMF) propre à la cible choisie. L'exécution du fichier de compilation par la commande *make* crée un exécutable indépendant de Matlab.

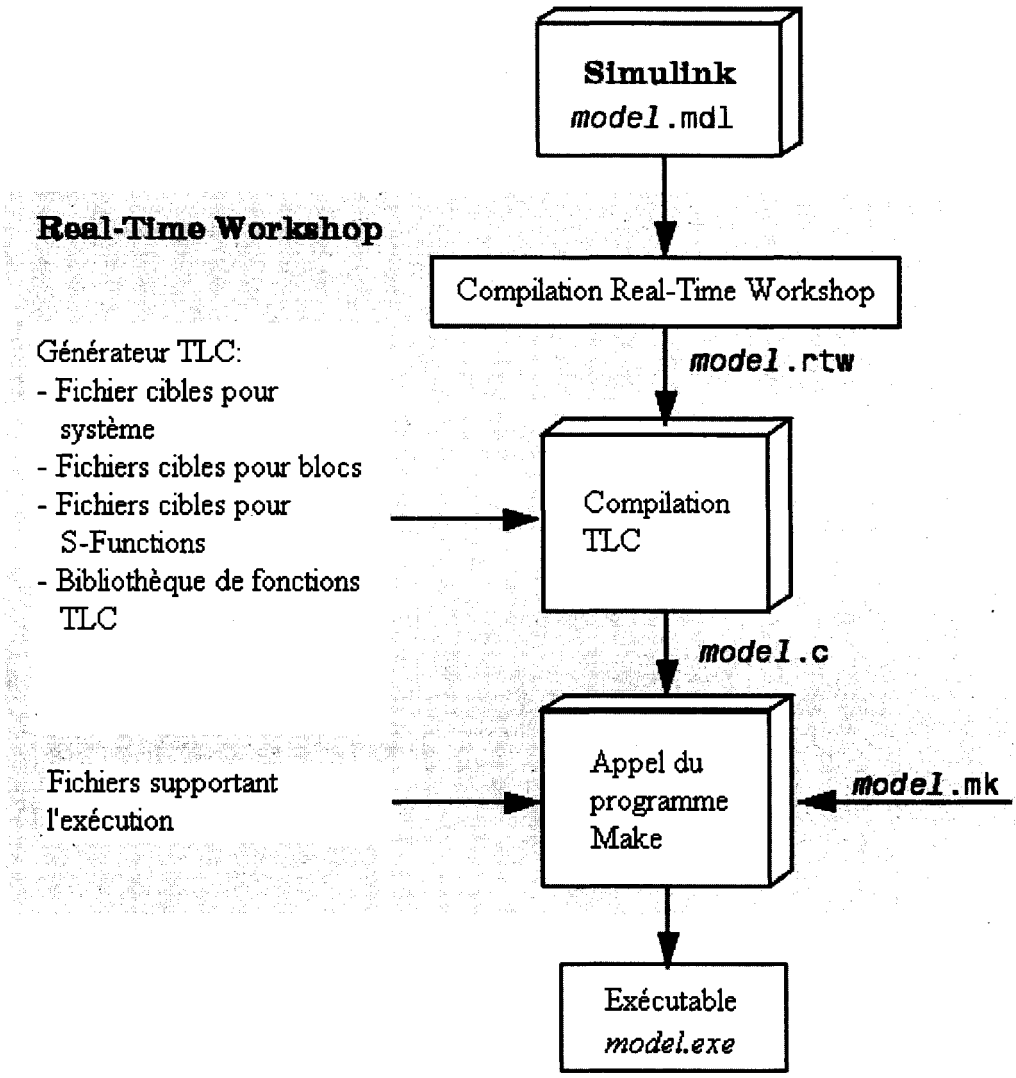

Figure 10 Mécanisme de génération de RTW

Pour la compilation du code [45], il existe deux cas de figure: soit que le contrôleur est destiné à fonctionner sur l'ordinateur de simulation ou qu'il doit être exécuté sur un autre ordinateur.

Dans le premier cas, lorsque le tout est exécuté sur un seul ordinateur, la chaîne d'outils Matlab / Simulink / RTW se charge de compiler la bibliothèque dynamique. Lorsque le

code est compilé, l'usager n'a qu'à copier la bibliothèque dans le répertoire où elle doit être utilisée.

Dans le second cas où les plateformes d'exécution et de simulation sont distinctes, il y a quatre possibilités pour accomplir la dernière étape:

1. Manuel: copier le code généré de la plateforme de simulation vers la plateforme d'exécution et lancer la compilation manuellement. Cette solution est la plus simple à développer en plus de limiter les risques d'erreurs mais c'est également celle qui demande le plus de temps à l'usager;
2. Programme résident: un processus s'exécutant en permanence sur la machine cible s'assure que la bibliothèque est toujours à jour en vérifiant la version du code. Simple à l'usage mais obligation de développer et d'installer un exécutable supplémentaire, qui peut potentiellement être une source d'erreurs;
3. Exécution distante: utiliser un protocole de transfert et d'exécution pouvant lancer la compilation (ssh, telnet, rlogin, etc.). Solution intéressante mais risquée au niveau de la sécurité selon le protocole utilisé (intrusion possible);
4. Compilation croisée: compiler sur la plateforme de simulation le code destiné à la plateforme d'exécution à l'aide d'un compilateur compatible avec la plateforme d'exécution. C'est une solution moins intéressante car ces compilateurs ont plus de risques de générer du code instable qu'un compilateur natif.

Comme ce projet de maîtrise a principalement pour but de prouver la fonctionnalité du concept et qu'il se peut que l'usager préfère utiliser sa propre méthode, le choix de la méthode manuelle a été fait. De plus, le temps supplémentaire requis pour la méthode manuelle reste somme toute raisonnable.

6.4 Modification à l'environnement de simulation

Pour fonctionner dans l'environnement Matlab, certaines modifications dans les règles de génération et dans la compilation du code doivent être effectuées. Cette section présente un aperçu de ce qui doit être fait sur la plateforme de simulation; le détail des opérations à effectuer sur la plateforme de simulation est présenté au CHAPITRE 7.

6.4.1 Modification à la génération du code

Le code généré par RTW est initialement destiné à produire un exécutable dont les fonctions d'interactions avec d'autres logiciels sont limités voire même inexistantes. Il doit donc être converti afin de permettre de contrôler son exécution à partir d'un programme principal. Cette conversion est effectuée en modifiant le code généré à l'aide d'une procédure suggérée par Roland Pfeiffer [65].

Cette procédure se résume à ajouter trois fonctions: la première pour commander l'initialisation, la seconde pour procéder aux calculs et la troisième pour terminer l'exécution de la bibliothèque dynamique. Le code résultant est recompilé sous forme de bibliothèque dynamique, prêt à être utilisé par le contrôleur global.

Les étapes suivantes sont effectuées par le contrôleur global selon la phase dans lequel il se trouve:

- Phase initiale
  - Chargement en mémoire de la bibliothèque dynamique[1]
  - Initialisation des pointeurs sur les 3 fonctions disponibles[1]
  - Exécution de la fonction d'initialisation

[1] Étape nécessaire sous Windows seulement, uniquement dans le cas où la bibliothèque est chargée durant l'exécution.

- Phase exécution
    - Exécution de la fonction de calcul (*n* fois)

- Phase finale
    - Exécution de la fonction de terminaison

La procédure proposée par monsieur Pfeiffer [65] est manuelle et requiert plusieurs manipulations pouvant induire des erreurs. Ces erreurs peuvent empêcher la compilation ou encore une exécution fiable. Pour éviter cet effet indésirable, la mécanique de génération de code a été modifiée pour rendre la procédure entièrement automatique.

L'automatisation de cette procédure passe par l'utilisation du langage TLC de Mathworks qui permet de modifier la génération du code. Les fichiers TLC permettent de générer le code propre au modèle en se basant sur les informations contenues dans le fichier RTW (voir Figure 10). En modifiant ou en ajoutant des fichiers TLC, il est possible de générer un fichier qui permettra de créer une bibliothèque dynamique dont l'exécution sera dirigée par une application externe. Le détail des changements effectués est disponible à l'ANNEXE 3 pour Windows et à l'ANNEXE 5 pour QNX.

6.4.2 Modification à la compilation du code

La compilation du code, initialement destinée à créer un exécutable indépendant, doit être modifiée pour indiquer au compilateur de créer une bibliothèque dynamique. Le fichier TMF associé à la cible est responsable de générer le makefile propre au modèle. En modifiant ce fichier, il est possible de changer les règles de compilation pour obtenir une bibliothèque dynamique. Le détail des changements effectués est disponible à l'ANNEXE 4 pour Windows et à l'ANNEXE 6 pour QNX.

6.4.3 Encapsulation des changements

Pour encapsuler les changements dans les fichiers TLC, une nouvelle cible (*target*) a été créée selon le système d'exploitation visé (Windows, QNX, etc.) pour tenir compte des différences entre ces plateformes. Cette cible doit être simplement ajoutée à l'arborescence de la suite de Mathworks pour être aussitôt accessible via l'interface de Matlab.

Sous Windows, les fichiers *Template Makefile* (*grt.tmf*) et *Target Language Compiler* (*grt.tlc*) de la cible *Generic Real-Time* (*grt*) ont été modifiés. Les deux fichiers (voir ANNEXE 3 et ANNEXE 4) ont été encapsulés dans une nouvelle cible RTW pour générer une bibliothèque dynamique (*.dll*) sous Windows et cette cible a été appelée *win_dll.*

Sous QNX, les deux mêmes fichiers ont été modifiés mais au lieu d'être basée sur la cible *Generic Real-Time* (*grt*), la cible a été basée sur une cible développée expressément pour QNX par Robert F.K. Martin [66]. Ce dernier a développé cette cible afin de faire fonctionner un robot Puma 560 à l'aide d'un contrôleur fonctionnant sous QNX. Les deux fichiers (voir ANNEXE 5 et ANNEXE 6) ont été encapsulés dans une nouvelle cible RTW pour générer une bibliothèque partagée (*.so*) sous QNX et cette cible a été appelée *qnx_so*.

6.5 Modification à l'environnement cible

Pour utiliser la bibliothèque dynamique dans Microb, il faut pouvoir compiler cette dernière sur la plateforme cible et permettre son appel à partir du contrôleur global. Cette section présente un survol de ce qui doit être fait sur la plateforme cible; le détail des opérations à effectuer est présenté au CHAPITRE 7.

Pour permettre la compilation de la bibliothèque générée par la suite Mathworks, il faut tout d'abord posséder le code source de Matlab. Si Matlab, Simulink et RTW sont installés sur la plateforme cible, le code est déjà présent. Sinon, il faut copier les fichiers source pour permettre de compiler la bibliothèque (voir section 7.2.2). De plus, il faut posséder le compilateur requis associé à la plateforme, soit *lcc* pour Windows ou *qcc* pour QNX.

L'environnement QNX doit être configuré pour lui permettre de localiser la bibliothèque tant à la compilation qu'à l'exécution (voir section 7.2.2). Pour qu'il localise la bibliothèque lors de la compilation, le nom de la bibliothèque doit être ajouté dans les fichiers de compilation de Microb. Pour permettre la localisation de la bibliothèque par le système d'exploitation, une variable d'environnement doit être modifiée.

Pour appeler la bibliothèque dans le contrôleur global développé avec Microb, il faut ajouter une portion de code qui diffère selon la plateforme. Le code à ajouter sous Windows est disponible à l'ANNEXE 7 et le code requis pour QNX est disponible à l'ANNEXE 8. Il est à noter que la cible QNX génère automatiquement un programme de test qui constitue un excellent exemple d'utilisation de la bibliothèque partagée (voir section 7.2.3).

6.6 Conclusion

Ce chapitre a permis de présenter quelles modifications ont été réalisées autant dans l'environnement de simulation que dans l'environnement cible pour parvenir à utiliser une bibliothèque partagée, sous Windows et sous QNX, avec le CAR Microb. Le prochain chapitre explique comment s'appliquent ces changements, c'est-à-dire comment s'installe et s'utilise en pratique le protocole.

MCours.com