Bibliothèque de Composants Intergiciels Dédiés aux Systèmes TR²E Critiques

SOMMAIRE

6.1 Introduction

Dans notre état de l'art et dans le chapitre 3, nous avons présenté les motivations de l'intégration des ressources intergicielles dès l'étape de modélisation afin d'autoriser la vérification et la validation du système critique complet (applicatif et exécutif) et de réduire la distance sémantique entre le modèle analysé et l'implantation finale. Lors de notre étude sur les différents intergiciels pour les systèmes TR²E, nous avons retenu l'architecture en services de l'intergiciel POLYORB-HI [Zalila *et al.*, 2008] dédié aux systèmes critiques. Celui-ci dispose des caractéristiques suivantes :

1. C'est un exécutif supportant les composants logiciels AADL, il fournit et assure les mécanismes pour la communication locale et distante des tâches applicatives (composant thread).

2. Il optimise l'utilisation des ressources intergicielles à partir de l'analyse de la description architecturale AADL du système, seuls les services requis par l'application sont ainsi sélectionnés, configurés et déployés (faible empreinte mémoire).

3. Son implantation en Ada est contrainte selon les recommandations du profil architectural Ravenscar (analyse statique du système).

4. Il s'intégre dans un processus de production automatisée du code source de l'application et s'interface avec les composants logiciels fournis par l'utilisateur.

Ce chapitre présente notre contribution sur la définition d'une bibliothèque de composants AADL architecturaux et comportementaux modélisant les ressources intergicielles requises (et leur utilisation) par un système critique. Ces composants sont modélisé à l'aide de notre profil AADL-HI Ravenscar présenté dans le chapitre précédent. Dans la section 6.2, nous rappelons brièvement le découpage en services de l'architecture de PolyORB-HI. La section 6.3 présente notre étude sur l'identification et l'intégration des services intergiciels au sein d'une description architecturale AADL et les choix de modélisation arbitraires se référant au support des composants logiciels AADL. Enfin, la section 6.4 présente l'architecture finale et les principaux composants intergiciels modélisés à partir de l'implantation Ada de PolyORB-HI.

6.2 L'intergiciel PolyORB-HI

Dans cette section, nous présentons l'architecture en services et leur rôle au sein d'une instance d'intergiciel dédiée à une application TR²E.

6.2.1 Intergiciel dédié et architecture

PolyORB-HI repose sur le concept d'intergiciel dédié. Une instance d'intergiciel dédiée à une application critique contient uniquement les entités requises par celle-ci. Plus l'architecture de l'intergiciel est flexible et modulaire, plus la configuration et la personnalisation des services et de leurs constituants sont fines. L'architecture en services de PolyORB-HI est issue du raffinement des services canoniques (adressage, transport, exécution, etc) proposé par [Kordon and Pautet, 2005]. Ce raffinement est effectué en fonction de leur degré de personnalisation (faible ou fort).

Ainsi, PolyORB-HI est divisé en deux parties. Le *noyau minimal* regroupe les services dont l'implantation est indépendante de l'application répartie (très faible personnalisation). Ces services en nombre limité sont sélectionnés (si requis) et configurés le cas échant.

La seconde partie est un ensemble de services fortement personnalisables qui viennent se greffer sur le noyau minimal. Les composants de ces derniers sont produits automatiquement et personnalisés à partir des caractéristiques extraites par le biais d'une analyse poussée de la description architecturale de l'application répartie en AADL. Le tableau 6.1 présente ce découpage en services et leur localisation. Dans les sous-sections suivantes, nous décrivons ces services afin de mieux comprendre leur rôle et leur intégration avec les composants applicatifs. Ceci nous permettra par la suite de déterminer les services explicites ou implicites spécifiés au sein d'une description architecturale AADL et les services manquants et de proposer une architecture pour notre bibliothèque de composants intergiciels.

Services intergiciels	Personnalisation	Localisation
Adressage	forte	*généré*
Liaison	forte	*généré*
Activation	forte	*généré*
Typage	forte	*généré*
Parallélisme	faible	noyau minimal
Exécution	forte	*généré*
Représentation élémentaire	faible	noyau minimal
Représentation avancée	forte	*généré*
Interrogation	faible	noyau minimal
Interaction	forte	*généré*
Protocole	faible	noyau minimal
Couche basse de transport	faible	noyau minimal
Couche haute de transport	forte	*généré*

TABLE 6.1 – Localisation des services de l'intergiciel POLYORB-HI

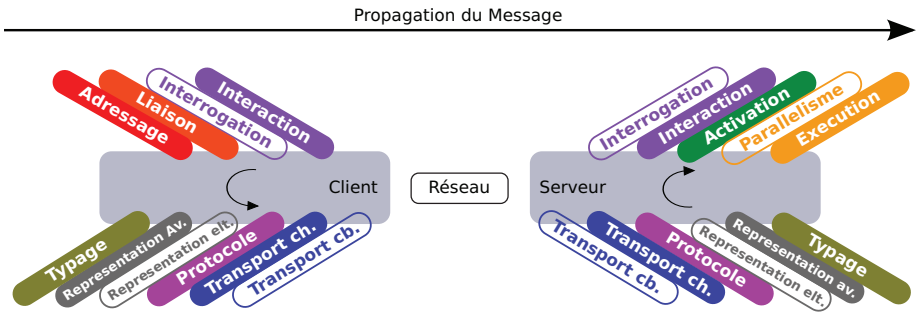

FIGURE 6.1 – Services de l'intergiciel POLYORB-HI

6.2.2 Services du noyau minimal

La figure 6.1 présente les différents services intergiciels de l'architecture de PolyORB-HI et leur ordre d'exécution. Dans cette section, nous présentons succinctement les services faiblement personnalisables du noyau minimal et leur rôle.

Parallélisme

Ce service permet l'instanciation des archétypes décrivant le comportement des différents types de tâche supportés par l'intergiciel. Ainsi, pour chaque tâche spécifiée au sein d'un nœud de l'application répartie, un archétype correspondant à sa catégorie et au langage de programmation cible est automatiquement instancié.

Interrogation

Ce service implante les mécanismes basiques d'échanges de données ou d'événements entre les différentes tâches des nœuds de l'application TR²E. Ainsi, il réalise l'envoi de données d'une tâche vers une autre et le déclenchement d'activité des tâches à événements (sporadique). Son implantation décrit le comportement des routines d'envoi et de réception ainsi que les structures de données utilisées. Le service d'interaction (présenté section 6.2.3) instancie ces mécanismes et implante le comportement désiré pour chaque tâche en fonction des propriétés et de la topologie déduite de l'application.

Représentation élémentaire

Un service de représentation assure et gère la transmission correcte des données à travers le réseau entre une entité émettrice et une entité réceptrice. Il gère le caractère hétérogène ou homogène d'une plate-forme en controlant l'emballage et le déballage des données transitant dans les canaux de communication. Pour les types de données élémentaires - *i.e.* prédéfinis (entier, booléen, etc) -, ces mécanismes sont indépendants de l'application répartie. Seule l'analyse des communications entre les nœuds de l'application amène une configuration particulière des techniques d'emballage et de déballage, comme, par exemple, la copie mémoire des données dans un tampon de communication dans le cas de plates-formes homogènes.

L'implantation de ces mécanismes pour les types de données prédéfinis supportés par l'intergiciel forme le service de représentation élémentaire. Ils seront utilisés par le service de représentation avancée pour la composition de mécanismes d'emballage et de déballage pour les types de données plus complexes spécifiés par l'utilisateur.

Protocole

Le service de protocole assure la transmission d'un message entre deux entités. Il utilise, pour cela, le tampon de communication construit par le service de représentation et ajoute toutes les informations requises pour son routage vers l'entité destinataire. Il s'occupe aussi de la gestion du mode de communication sélectionné par le service de liaison.

Couche basse de transport

Le service de transport assure et gère l'utilisation des canaux de communication (ouverture/fermeture) utilisés pour l'échange de messages entre les nœuds. Son implantation dépend

fortement du système d'exploitation ou de l'environnement d'exécution de l'application ainsi que des composants matériels relatifs au transport de données (bus de communication Ethernet, etc).

Le service de couche basse de transport contient les mécanismes d'accès direct au médium de communication entre deux nœuds. Il fournit les routines nécessaires pour l'échange d'information entre l'application et le pilote du périphérique de communication.

Si son implantation est indépendante des propriétés de l'application, son utilisation dépend fortement de la nature même de l'application. Dans le cas d'une application monolithique ce service n'est pas requis. Sa sélection est gérée par le service de couche haute de transport après analyse de l'application.

6.2.3 Services fortement personnalisables

Dans cette section nous décrivons brièvement les services fortement personnalisables de l'intergiciel POLYORB-HI (illustrés figure 6.1) :

Couche haute de transport

Le service de couche haute de transport fait le lien entre le service de protocole et la couche basse de transport requise. Cette partie fortement personnalisable sélectionne le service de couche basse de transport selon les informations extraites du service d'adressage et dépend exclusivement des caractéristiques de l'application répartie. Ce service implante trois routines. `Init` initialise l'ensemble des couches basses de transport requises par le nœud. `Send` invoque l'entité destinataire du message à l'aide de la couche basse de transport dans le cas d'une communication distante ou à l'aide de la routine `Deliver` (ci-dessous) dans le cas d'une communication locale. `Deliver` délivre le message reçu à la couche de protocole. Cette routine est invoquée soit par la couche basse de transport dans le cas d'une communication distante soit par la routine `Send` (décrite ci-dessus) dans le cas d'une communication locale.

Représentation avancée

Le service de représentation avancée est la partie personnalisable du service de représentation (voir représentation élémentaire, section 6.2.2). Il implante les routines d'emballage et déballage de données spécifiques pour la gestion des types complexes définis par l'utilisateur (par exemple, structures de données, type discriminant Ada, etc). Il s'appuie sur le service de représentation élémentaire.

Typage

Le service de typage gère les types de données requis et fournis à l'application répartie (spécifiés par l'utilisateur). Combiné avec le service de représentation, celui-ci permet l'emballage et le déballage des types de données complexes à partir des tampons de communication. Il assure ainsi la transmission correcte des données entre les différents nœuds de l'application. L'implantation de ce service dépend des propriétés de l'application.

POLYORB-HI utilise le langage AADL et son annexe de modélisation des données pour la spécification des types de données basiques et/ou complexes. Une vérification des restrictions

définies sur ces types est effectuée dans le but de garantir leur conformité vis-à-vis des systèmes critiques. Enfin, un générateur de code produit, à partir de ces composants modélisés les types de données correspondants dans le langage de programmation cible.

Adressage

Le service d'adressage gère les références - *i.e.* les adresses - des entités extérieures à l'intergiciel. Ce service permet de retrouver sans ambiguïté l'entité traitant les requêtes et permet aux entités extérieures de se connecter et d'envoyer des messages à celle-ci. La production des adresses dépend fortement du nombre, de la topologie et du type d'interaction entre les nœuds d'une application répartie. L'implantation de ce service est simplifiée par le caractère statique de l'application TR²E où la configuration des tâches et les canaux de communication sont connus et gérés lors de la phase d'initialisation de l'application.

Ainsi, l'implantation de ce service peut se réduire pour chaque entité à une table permettant de faire la correspondance entre entité destinataire connectée et la ressource de transport associée (par exemple, une socket) - *i.e.* la table de nommage.

Liaison

Ce service assure la construction des ressources de communication pour l'échange d'informations entre entités locales et/ou distantes. Il définit le mode de communication (passage par message, appel de procédure ou objet distant) et sélectionne les instances des services d'interaction, de représentation, de protocole et de transport requis dans le cas d'une communication distante. Ce service s'exécute au-dessus du service d'adressage et utilise les références produites par celui-ci.

L'implantation de ce service est facilitée part le caractère statique de l'application. La topologie de l'application permet la connaissance définitive du type de communication entre deux entités (locale ou distante) et ainsi la sélection des instances pertinentes des services du noyau minimal (couche basse de transport, représentation élémentaire, etc).

Interaction

Le service d'interaction gère les modes de communication entre les nœuds de l'application répartie. L'implantation de ce service décrit l'invocation des routines correspondantes au mode d'interaction choisi. Par exemple, si le mode est synchrone, le service d'interaction bloque l'entité émettrice jusqu'à l'obtention d'une réponse de la part de l'entité réceptrice. Dans le contexte d'un mode asynchrone, l'entité émettrice est libérée immédiatement après l'envoi de la requête.

PolyORB-HI raffine cette définition du service d'interaction en fonction de la personnalisation des composants. Ainsi, la partie faiblement personnalisable est implantée par le service d'interrogation (section 6.2.2) et la partie fortement personnalisable par le service d'interaction (même nom). Celui-ci réalise l'instanciation des archétypes des différents types d'interaction déduits des interfaces des composants logiciels (tâches et sous-programmes) décrits dans l'application répartie.

Activation

Ce service gère l'activation de l'entité requise pour le traitement d'une requête à la livraison d'un message. Il crée, selon la nécessité et gére des ressources temporaires afin d'assurer le bon fonctionnement du traitement. Il n'est requis que par les entités qui peuvent recevoir des messages.

La personnalisation de ce service consiste à déterminer la tâche à activer lors de la réception d'un message par analyse de l'application répartie. L'implantation de ce service inclut l'acheminement des messages entre la couche protocolaire et la couche d'interaction ce qui permet de délivrer correctement le message à la tâche de réception.

Exécution

Ce service s'exécute au-dessus du service d'activation et utilise le service de parallélisme (section 6.2.2). L'entité sélectionnée et activée par le service d'activation est ici utilisée pour effectuer le traitement requis lors de la réception d'un message. Le service d'exécution gère alors l'allocation, la création et la destruction des tâches selon la politique de service choisie (mono-tâche, tâche par session, tâche par requête, etc). Son implantation existe exclusivement sur les entités réceptrices de messages.

A partir de l'analyse du nombre de tâches, de leur caractéristiques (type, priorité, taille de pile, etc) et de l'architecture de l'application, ce service crée statiquement l'ensemble des tâches requises par l'application. Cet ensemble de tâches comprend à la fois celles définies par l'utilisateur mais aussi les tâches supplémentaires (tâche de réception...) induites par les différentes couches du service de transport sélectionné.

6.3 Identification des services et choix de modélisation

Dans cette section, nous identifions les différentes informations implicites, explicites ou manquantes relatives aux ressources intergicielles, existantes au sein d'une description architecturale AADL. Ces éléments détermineront l'ensemble de composants intergiciels à modéliser. Puis, nous étudions brièvement les contraintes de modélisation issues des choix d'implantation relatifs à un exécutif supportant les composants logiciels AADL en nous basant sur POLYORB-HI.

6.3.1 Description architecturale AADL et services intergiciels

Pour déterminer les ressources intergicielles à modéliser et comment les intégrer à la description architecturale du système nous présentons ici notre cas d'étude Ravenscar (présenté dans le chapitre 10) sous sa représentation graphique, figure 6.2. Il s'agit d'un système de gestion de charges de travail (le nœud `Workload_Manager`) pouvant recevoir et traiter des interruptions (envoyées par le nœud `Interruption_Simulator`).

Nous y avons ajouté des annotations informelles traduisant les informations spécifiées par les propriétés AADL attachées à une partie des composants. Un code de couleur permet de distinguer les composants relatifs aux services intergiciels décrits dans la section précédente (présentés dans la figure 6.1).

Le service de transport est décrit par l'utilisation d'un composant matériel *bus* et d'une propriété AADL qui spécifie le protocole de transport (dans notre cas le bus SPACEWIRE). Un composant périphérique (`device`) spécificie le pilote (couche basse de transport) associé

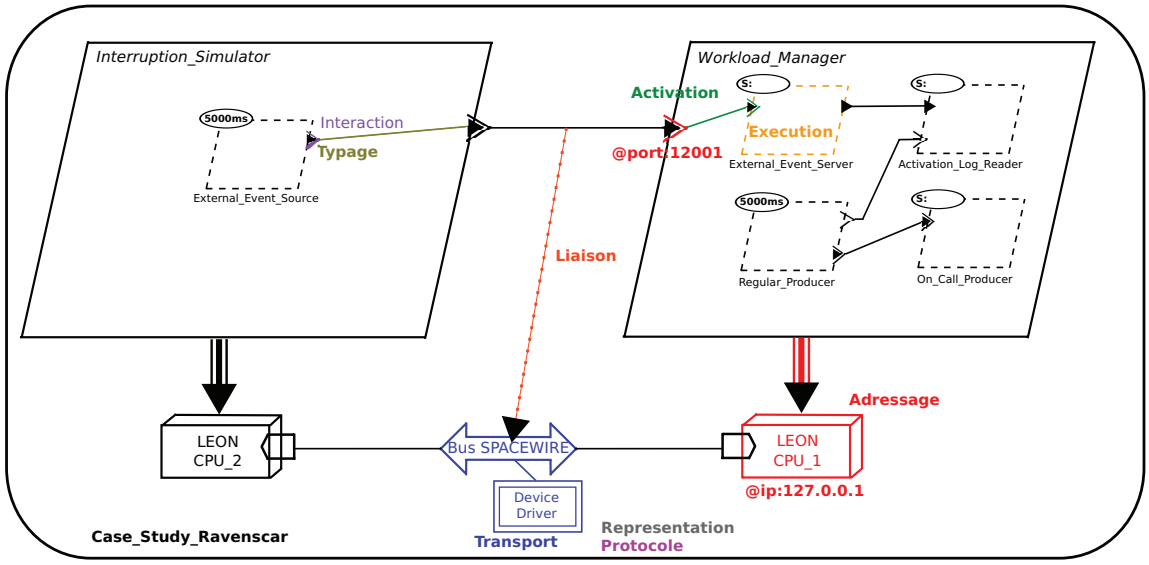

FIGURE 6.2 – Identification des services sur la description architecturale AADL

au bus et attaché à l'aide d'une propriété. Le service de typage est explicité par les types de données associées aux élements d'interfaces - *i.e* les ports des composants threads et processus. Le service de liaison n'est pas explicite, il est déduit de l'analyse des connexions entre les processus et leur *mapping* à un bus de transport. Le service d'adressage est déduit des informations spécifiées sur les composants processeurs (par exemple l'`IP`) et sur les ports des composants processus (par exemple le numéro de port pour la communication). Le service d'activation peut être déduit des connexions entre les ports de processus et les ports des threads qu'ils contiennent. Le service d'exécution est implicitement déduit de la sémantique des composants threads. Enfin, les services de représentation, de protocole, de parallélisme et d'interrogation ne sont pas représentés par la description.

Discussion

Cette rapide analyse de la description architecturale d'un système TR²E en AADL confirme qu'un certain nombre d'informations explicites ou implicites sur les ressources intergicielles requises sont présentes. Nous avons aussi pu déterminer quelles étaient les ressources manquantes et constater le manque d'information comportementale sur l'utilisation de l'ensemble de ces ressources.

Notre bibliothèque de composants AADL (présentée section 6.4) vise à expliciter l'architecture (constantes, types de données, données) et le comportement (interaction, mécanismes d'envoi et de réception de messages, etc) de ces ressources intergicielles à partir de la spécification initiale de l'utilisateur et d'un ensemble de composants préexistants. Pour la modélisation de ces composants, nous réutilisons le profil AADL-HI Ravenscar (détaillé chapitre 5, détermine le niveau d'abstraction des composants) et nous nous basons sur l'implantation des composants de l'intergiciel POLYORB-HI pour l'élaboration de spécifications comportementales, renforçant ainsi la cohérence entre le modèle et l'implantation.

6.3.2 Gestion locale de la communication

L'implantation des mécanismes pour la gestion de la communication entre les tâches de l'application nécessite d'effectuer certains choix de conception et d'implantation. Le standard AADL définit de nombreux composants logiciels (données, données partagées, sous-programmes, tâches, ports) avec une sémantique précise et permettant une description fine des composants applicatifs du système. Cependant, celui-ci ne fournit que très peu de directives ou de recommandations sur la sémantique de l'exécutif AADL - *i.e.* l'intégration et l'exécution des composants applicatifs sur les composants intergiciels. Notamment, comme nous l'avons expliqué dans le chapitre 4, le standard AADL définit les interfaces d'un ensemble de routines pour accéder en écriture ou en lecture aux interfaces des tâches pour l'envoi et la réception de données et événements.

Ainsi, POLYORB-HI effectue des choix de conception pour le support et l'exécution des tâches spécifiées par l'utilisateur. De plus, ces choix sont contraints par les recommandations du profil Ravenscar. Notamment, la communication entre les tâches des nœuds est implantée selon un protocole de communication asynchrone basé sur l'échange de messages (encapsulant la donnée initiale). Pour chaque tâche, les ports AADL sont implantés comme une unique file d'attente globale de messages (un objet protégé Ada) pour assurer l'envoi et la réception correcte des informations à travers l'intergiciel.

Autre choix de conception : la couche basse de transport doit déclarer au moins une tâche de réception par nœud récepteur et assurer la délivrance des messages reçus en invoquant les routines de la couche haute de transport qui aiguillera ceux-ci vers la file d'attente de la tâche correspondante.

Discussion

Les ressources créées par ces différents services ont un impact direct sur l'analyse d'ordonnancement du système et justifie l'intégration de ces composants à la description architecturale initiale.

Dans le cadre de notre bibliothèque de composants intergiciels, nous modéliserons le service de protocole et le service d'interraction conformément à l'implantation de POLYORB-HI. Les couches basses de transport feront l'objet d'une modélisation partielle à l'aide de composants opaques en raison de leurs dépendances trop spécifiques à la runtime supportant l'intergiciel (dans notre cas la runtime Ada). Cette approche assure la prise en compte de ces composants lors des analyses effectuées sur le modèle.

6.4 La bibliothèque de composants POLYORB-HI-AADL

Cette section présente la modélisation des services intergiciels décrits dans les sections précédentes. La spécification des composants a été réalisée à l'aide de notre profil AADL-HI Ravenscar (présenté chapitre 3) nous garantissant l'analyse et la production automatique du code source du système à partir de ces composants. L'ensemble de ces composants constituent notre bibliothèque POLYORB-HI-AADL que nous détaillons dans la suite.

6.4.1 Architecture générale

La figure 6.3 présente l'architecture générale de notre bibliothèque de composants intergiciels et décrit les interactions entre composants (préexistants et générés). Chaque service

est modélisé sous la forme d'un ou plusieurs paquetages AADL encapsulant les composants intergiciels qu'ils fournissent. Une partie de ces composants est produit automatiquement à partir de l'analyse de la description architecturale initiale du système. L'autre partie est définie sous la forme de composants génériques - *i.e.* spécifiés à l'aide de *prototypes*. Une partie des composants générés fournit les informations requises pour l'instanciation et la configuration des composants génériques.

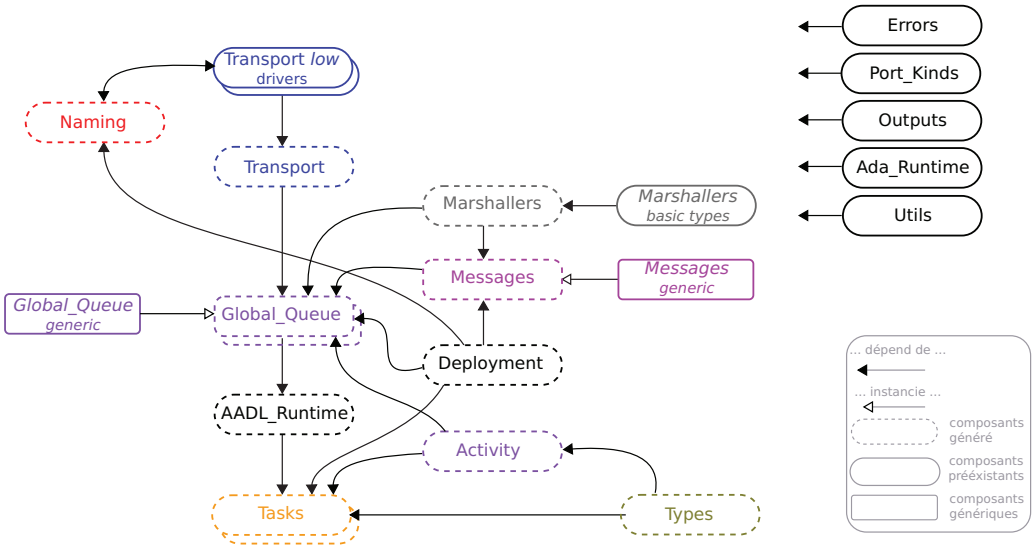

FIGURE 6.3 – Architecture de la bibliothèque POLYORB-HI-AADL

Les sous-sections suivantes décrivent les différents paquetage AADL de notre bibliothèque et les ressources intergicielles qu'ils contiennent. Le détail des composants générés et la manière dont ils sont produits sont présentés dans le chapitre 7 sous la forme de descriptions de transformation intégrées à notre processus de rafinement incrémental de modèles AADL.

6.4.2 Composants intergiciels générés

Deployment

Ce paquetage modélisent les composants nécessaires pour le déploiement des nœuds de l'application répartie sous une représentation équivalente aux constructions d'un langage de programmation. Ces composants sont générés exclusivement à partir de l'analyse de la topologie de l'application. Ils définissent des constantes et des énumérations utilisées pour la configuration de nombreux services intergiciels.

Naming

Ce paquetage modélise les services d'adressage et de liaison. Il définit les composants constituant les tables de nommage requises pour chaque service de transport utilisé par un nœud de l'application répartie. Ces composants sont générés à partir de l'analyse des connexions et des propriétés de déploiement (`Deployment::Location` et `Deployment::Port_Number`) spécifiés sur les composants processeurs et processus. Le composant modélisant la table de nommage est utilisé pour configurer la couche basse de transport sélectionnée.

Transport

Ce paquetage modélise les routines (`Init`, `Send`, `Deliver`) définies par la couche haute de transport et permettant de relier la couche basse de transport sélectionnée au service de protocole.

Remarque. La visibilité de ce service permet de tracer précisément le chemin d'exécution complet lors de l'émission ou de la réception d'un message. Ces informations sont utiles pour estimer le pire temps d'exécution d'une tâche à l'aide de la séquence d'appel de sous-programmes déclarée par celle-ci.

Messages

Le paquetage `Messages` correspond au service de protocole. Il définit le tampon de communication globale `Message_Type` configuré en fonction de l'analyse de la taille maximale des données échangées entre les nœuds de l'application (utilisation du composant `Max_Payload_Size` généré à cet effet pour garantir son allocation statique et le respect de nos restrictions sur les systèmes critiques).

Le composant `Message_Type` généré est utilisé pour instancier et configurer les composants génériques du paquetage `Messages_Generic` (présenté dans l'annexe A) explicitant le protocole utilisé pour l'encapsulation d'une donnée dans un message et la construction de l'en-tête pour son routage correcte (voir section 6.4.3).

Remarque. Ces composants sont à la fois utiles pour l'analyse mémoire et l'estimation du pire temps d'exécution.

Marshallers

Les paquetages `Marshallers` et `Marshallers_Basic_Types` modélisent respectivement les services de représentation élémentaire et avancée. Les mécanismes d'emballage et de déballage des données dans un message y sont décrits à l'aide de composants sous-programmes et d'éléments de langage de l'annexe comportemententale. Ces routines sont définies pour l'ensemble des types de données transférables dans un message. Les routines pour emballer les données temporelles et celles pour emballer les énumérations correpondantes aux ports sources ou destinations d'un message sont aussi incluses dans ce paquetage.

Remarque. Ces paquetages définissent des composants qui permettent d'affiner l'analyse de flots de données (représentation des différentes transformations de données et flux d'exécution), l'analyse de mémoire (modélisation des différents types de données utilisés) et l'estimation du pire temps d'exécution d'une tâche dans le cas de communication distante.

Activity et Global_Queue

Ces deux paquetages réunifient le service d'interrogation et d'interaction. En effet, le paquetage `Activity` spécifie les composants de données représentant les interfaces d'une tâche applicative. Le paquetage `Global_Queue` spécifie les composants de la file d'attente globale de messages et ses différents interrogateurs assurant sa gestion et son utilisation (spécification du comportement) de manière sûre et déterministe.

Un paquetage `Activity` est généré pour chaque tâche applicative du système. Les interfaces qu'il définit sont utilisées pour instancier et configurer les composants génériques du paquetage `Global_Queue_Generic` (voir section 6.4.3).

Remarque. La modélisation de cette file d'attente de messages est d'autant plus nécessaire qu'elle donne une visibilité claire et précise des mécanismes et des données mises en jeu pour l'échange de messages entre les tâches. Ceux-ci sont alors pris en compte par les outils d'analyse et contribue à réduire la distance sémantique entre le modèle et l'implantation du système.

AADL_Runtime

Dans le chapitre 4, nous avons expliqué que le standard AADLv2 définit des interfaces pour l'envoi et la réception de données des tâches. Ainsi, AADL standardise certaines interactions avec la runtime - *i.e.* l'intergiciel - en fournissant des interfaces modélisées par des composants sous-programmes (`Send_Ouput`, `Put_Value`, etc.).

Nous avons choisi d'intégrer ces composants sous la forme d'un service (`AADL_Runtime`) à notre architecture d'intergiciel pour les motivations suivantes. Premièrement, nous nous devons de rester conforme aux exigences du standard AADL notamment dans le but de simplifier la compréhension pour les utilisateurs et de garantir l'interopérabilité de nos modèles avec les outils de la communauté AADL.

Deuxièmement, ces interfaces standardisées amènent une modularité supplémentaire à notre exécutif AADL et permet de séparer clairement la partie applicative et la partie exécutive (intergiciel) au sein de la modélisation. Ceci renforce les possibilités de réutilisation, de maintenance et d'évolution de notre bibliothèque de composants en facilitant notamment le changement d'une partie de l'exécutif AADL.

Types

Ce paquetage modélise le service de typage. Il centralise au même endroit les différents composants spécifiant les types de données d'un nœud définies par l'utilisateur. La centralisation de ces données facilite leur réutilisation au sein des différents services intergiciels.

Tasks

Ce service réunifie les services de parallélisme et d'exécution. Pour chaque tâche applicative spécifiée par l'utilisateur, nous générons un paquetage `Task`. Celui spécifie le composant thread final utilisé dans notre modèle de code analysable. Nous reprenons la description architecturale de l'utilisateur et nous y ajoutons le patron comportemental correspondant au type de la tâche analysé. Le composant thread obtenu décrit l'architecture et le comportement de la tâche telle qu'elle sera implantée dans le langage de programmation cible.

6.4.3 Composants intergiciels préexistants

Paquetage Messages_Generic

Le paquetage `Messages_Generic` (voir annexe A) modélise les composants explicitant le comportement du service de protocole. Ces composants correspondent aux routines néces-

saires pour lire et écrire les messages qui sont échangés entre les entités d'une application. La structure de ce paquetage est inspirée de celle du paquetage `PolyORB_HI.Messages` de l'intergiciel PolyORB-HI garantissant le respect de restrictions spécifiques aux systèmes critiques (taille de message borné et configuré statiquement, pas d'allocation de mémoire dynamique, pire cas des opérations de lecture et d'écriture sur les messages qui se font en un temps proportionnel à la taille de la donnée lue ou écrite, voir [Zalila *et al.*, 2008][chapitre 6]).

Ainsi, ces routines dépendent du composant `Message_Type` modélisant le message de taille borné (voir sous-section précédente). L'utilisation de *prototypes* nous permet de modéliser ces composants de manière générique, de les instancier et de les configurer[4] plus tard dans la modélisation.

Paquetage Global_Queue_Generic

Rappel. Dans PolyORB-HI, la file d'attente globale de messages d'une tâche est une concaténation de tableaux circulaires correspondant chacun à la file d'attente d'un des ports en entrée d'une tâche. Cette file implante le comportement des ports conformément à la sémantique AADL. Elle est définie à l'aide d'un objet protégé Ada qui encapsule l'ensemble des tableaux, des données et des routines nécessaires pour sa manipulation. Un certain nombre de routines assure l'envoi et la réception de données de manière déterministe et transparente à l'utilisateur. La structure de cette file répond aux recommandations du profil et est implantée dans le paquetage `PolyORB_HI.Thread_Interrogators` [Zalila *et al.*, 2008].

Le paquetage `Global_Queue_Generic` (voir annexe A) modélise cette file d'attente sous la forme d'un composant de donnée partagée AADL. Les routines nécessaires à sa manipulation sont modélisées à l'aide de composants sous-programmes et reliées à la file à l'aide des accesseurs de sous-programmes. L'ensemble des composants et la structure interne de la file ont été modélisés à l'aide de *prototypes* permettant la définition de composants génériques. Ainsi, pour chaque tâche applicative du système, les composants de déploiement générés (voir sous-section précédente) et l'analyse des connexions entre les tâches et les nœuds du système permettent l'instanciation et la configuration de ces composants au sein du paquetage `Global_Queue`.

Paquetage des couches basses de transport

Un service de couche basse de transport est fortement dépendant de l'environnement d'exécution de l'application répartie. La description complète du service impliquerait alors la modélisation d'un grand nombre de composants issus de cet environnement (par exemple, dans notre cas la runtime Ada) nécessitant un effort conséquent pour chaque environnement d'exécution supporté et dont la prise en compte par les outils d'analyses est quasi-nulle (complexité d'extraction d'information, granularité trop fine du composant ou composant non pertinent).

Cependant, certains composants (tâche de réception...) s'avèrent nécessaires pour une analyse plus fine de l'ordonnancement du système. Par conséquent, nous proposons la définition de paquetage de composants AADL modélisant uniquement les interfaces des primitives offertes par ce service. L'emploi des sous-programmes opaques, des composants thread et de propriétés AADL adéquates (`Compute_Execution_Time`...) nous permet de fournir une des-

4. Pour simplifier la compréhension nous traduisons le *binding* des protypes comme une instanciation.

cription architecturale de la couche basse de transport utilisable par les outils d'analyse et d'intégrer les pièces de code correspondant lors de la génération du système.

Paquetages additionnels

Nous avons modéliser un certain nombre de paquetages AADL supplémentaires fournissant des composants utilisés par les différents services intergiciels. Il s'agit de :

- `Errors`, qui modélise sous la forme d'un composant d'énumération les différentes valeurs d'erreurs des sous-programmes ;
- `Port_Kinds`, qui définit les différents types de ports sous la forme d'une énumération et des primitives pour la vérification des types de ports (représentation plus proche de l'implantation) ;
- `Ada_Runtime`, qui modélise de manière opaque certains types de données et routines de la runtime Ada (permet de compléter les descriptions de certains composants).
- `Utils`, qui fournit des routines opaques pour la transformation de données.

Ces paquetages nous permettent de compléter la modélisation de nos composants intergiciels afin de rester *légal* vis-à-vis du standard AADL (règles syntaxique, cohérence, etc.).

6.5 Synthèse

Dans ce chapitre, nous avons présenté notre bibliothèque de composants intergiciels permettant la modélisation et l'analyse complètes du système à partir de sa description AADL. Ces composants reposent sur une architecture modulaire et flexible (découpage en services) décrivant un intergiciel supportant les composants applicatifs AADL et dédié aux systèmes critiques (sélection des composants uniquement requis par l'application, etc). L'ensemble de ces composants a été modélisé à l'aide de notre profil AADL-HI Ravenscar.

Une partie de ces composants intergiciels est générée et configurée à partir d'une analyse poussée de la description architecturale initiale du système (modèle utilisateur). Les composants restants sont définis sous la forme de paquetage de composants AADL (architecture + comportement) génériques - *i.e.* basés sur l'utilisation de *prototypes*. Une partie des composants intergiciels générés (composants pour le déploiement, etc) est utilisée pour l'instanciation et la configuration de ces paquetages.

L'architecture que nous proposons est un raffinement de l'architecture de l'intergiciel PolyORB-HI. Les spécifications comportementales des composants ont été réalisées en fonction des implantations des constructions de cet intergiciel. Ceci renforce la cohérence entre le modèle et l'implantation finale du système.

Le chapitre suivant explique la manière dont ces composants sont produits, instanciés, configurés et intégrés automatiquement à la description initiale du système à l'aide d'un processus de raffinement incrémental basé sur la transformation de modèles AADL. L'issue du processus de raffinement est l'obtention de notre modèle de code complet (applicatif et exécutif) nécessaire à l'analyse et à la production automatique du sytème critique.