Chapitre I- Comparaison avec les langages de description d'architectures.

Objectif du chapitre :

- Positionner SEP par rapport aux langages de description d'architectures (ADL) logicielles ou matérielles.

Notre approche de modélisation des architectures matérielles consiste à décrire avec des composants logiciels les différentes unités qui constituent l'architecture, puis à les composer. Ces composants interagissent avec l'environnement en échangeant des données. Ils peuvent être assemblés suivant une certaine configuration pour réaliser une architecture. Cet assemblage consiste en la définition de connecteurs qui permettent l'échange d'informations entre plusieurs composants. Les composants sont dits autonomes car leur comportement ne dépend pas de la configuration réalisée.

autonome

Cette approche logicielle de modélisation de systèmes n'est pas spécifique à la modélisation de cibles matérielles. Elle correspond à la notion d'architecture logicielle. Une architecture logicielle est définie comme un niveau de conception qui contient la description des éléments à partir desquels un système est construit, de l'interaction entre ces éléments, de schémas génériques qui guident leur composition et de contraintes sur ces schémas.

architecture logicielle

Cette notion d'architecture logicielle est utilisée depuis quelques années et des langages de description d'architectures logicielles (ADL) se développent indépendamment les uns des autres (Aesop **[Gar95]**, C2 **[Med95]**, MetaH **[Ves96]**, Rapide **[Luc95]**, Wright **[All97]**, SEP **[Mal97]**). Le cadre peu précis d'application des ADL, rendait difficile la différenciation entre les langages de spécification formelle, de simulation, de programmation, d'interconnexion de modules et les langages de description des architectures car ces derniers intègrent souvent certaines fonctions des autres. Très récemment, une enquête menée par N. Medvidovic et R.N. Taylor a défini de façon satisfaisante les caractéristiques minimum qu'un langage doit offrir pour être considéré comme un ADL **[MeT00]**. Dans la suite de ce chapitre, nous utilisons le vocabulaire et les concepts identifiés par cette enquête pour exposer les idées introduites dans SEP et conclure que SEP est un ADL adapté à la modélisation logicielle des architectures matérielles numériques. Toutefois, le domaine assez restreint de son utilisation autorise la simplification de certains aspects traités par les ADL classiques.

Un ADL doit permettre la description explicite des **composants**, des **connecteurs**, de la **configuration** de l'architecture et bénéficier d'**outils de support** pour valider et analyser les modèles. Ces points sont détaillés ci-dessous.

1- La description des composants.

composant

Dans un ADL, un composant est une unité de calcul ou de mémorisation. Une description explicite de son **interface** est indispensable **[LVM95]**, l'utilisation d'objets et la communication par envoi de messages ne sont pas suffisantes. De plus, les composants peuvent être **typés**, avoir une **sémantique** formelle ou informelle, exporter des **contraintes** d'utilisation, permettre l'**évolution** ou présenter des **propriétés non fonctionnelles**.

interface

L'**interface** d'un composant consiste en un ensemble de points d'interactions entre le composant et le monde extérieur qui permettent l'invocation des services. Dans SEP, les services sont spécifiés indépendamment comme des méthodes (cf. chapitre II), puis une interface de communication est construite pour encapsuler ces services. Cette interface est constituée d'un ensemble de ports munis d'une sensibilité et attachés à un service qui définit le comportement à réaliser lors de son invocation. La sensibilité des ports détermine le rôle que joueront les données qui transitent sur ce port dans le comportement du service. Ces données pourront par exemple être traduites en événements et déclencher l'exécution du service ou simplement être utilisées comme des données partagées entre plusieurs composants. On distingue deux types de composants classiques, les composants combinatoires (sep.model.LevelComponent) et les composants à états (sep.model.EdgeComponent).

Les composants sont représentés par des classes qui encapsulent les services, les contraintes sur ces services et les paramètres éventuels du composant. Ces classes définissent

type

le **type** du composant, c'est-à-dire une abstraction qui permet l'encapsulation de fonctionnalités dans des blocs réutilisables, et permettent l'instanciation de plusieurs composants similaires.

évolution

L'**évolution** des composants se traduit par des modifications de certaines propriétés (interface, comportement). L'utilisation d'un langage orienté objet permet, notamment grâce à l'héritage, de garantir une évolution avec une réutilisation optimale.

contraintes

Les **contraintes** sur les services déterminent les conditions d'utilisation d'un composant. Dans SEP, les contraintes sont des méthodes évaluées, vraies ou fausses, qui déterminent les conditions d'utilisation d'un service. Lorsqu'une contrainte est définie pour un service, la méthode associée doit être évaluée vraie pour que le service soit invoqué.

sémantique

Les composants SEP n'ont pas de **sémantique** formelle. De façon informelle, leur sémantique est basée sur un modèle de communication à événements discrets partiellement ordonnés[1]. Toutefois, la composition de composants hétérogènes est possible (cf. chapitre IX) ; le comportement des composants peut, par exemple, être décrit dans le langage Esterel qui possède une sémantique formelle.

De plus, les composants SEP ont des **propriétés non fonctionnelles**. En particulier, ils peuvent être personnalisés dynamiquement. Par exemple, des constantes peuvent être modifiées (taille d'une mémoire, zone d'adressage) et le nombre de ports de données peut être choisi (ainsi il est possible de construire, à partir du même composant, un multiplexeur à 2, 3, …, n entrées). Enfin, les composants dont la description du comportement est structurelle bénéficient de techniques d'abstraction (ports de services – cf. chapitre II-4-) et de validation (cf. chapitre VII et VIII) adaptées.

2- La description des connecteurs.

connecteurs

Dans les ADL, les connecteurs sont des blocs de construction utilisés pour modéliser l'interaction entre composants ainsi que les règles qui gouvernent cette interaction. Leurs caractéristiques sont les mêmes que celles définies pour les composants. Cependant, alors que les composants constituent les entités à concevoir, les connecteurs quant eux représentent des entités de communication à utiliser, c'est-à-dire qui ne sont pas à concevoir ou à modifier par rapport à l'architecture étudiée mais dont la modélisation est indispensable pour l'analyse du système complet. Ce sont par exemple, les réseaux ou les connexions physiques quelconques. Dans SEP, les deux connecteurs standards (disponibles en bibliothèque) sont les signaux de

[1] SEP permet de prendre en compte le temps physique. Cependant, à temps physique constant, les événements sont ordonnés par rapport à un temps logique propre à chaque composant. Seuls les composants qui communiquent peuvent ordonner leurs événements les uns par rapport aux autres (cf. chapitre IV).

contrôle et les bus. Les signaux de contrôle permettent de connecter un écrivain et plusieurs lecteurs : ce sont des 'buffers'. Les bus de données sont des variables partagées, ils permettent de connecter plusieurs écrivains et plusieurs lecteurs. Les conflits éventuels (écriture-lecture ou écriture-écriture) sont gérés par un contrôleur de bus intégré muni d'une fonction de résolution. En outre, le multiplexeur est un connecteur dynamique entre plusieurs connecteurs de même type, et contrôlé par un signal de contrôle.

De nouveaux connecteurs peuvent être spécifiés de la même façon que les composants. Par exemple, on pourrait spécifier des connecteurs pour implémenter un protocole de communication spécifique. C'est ainsi que dans le cadre de la modélisation d'une architecture bi-cœur (cf. chapitre XI), un connecteur de communication entre les deux cœurs de processeurs a été défini. Il s'agit d'un mécanisme de communication commandé par interruptions et utilisant une mémoire double port. La modélisation d'un tel connecteur qui correspond en réalité à un composant matériel existant permet d'évaluer les deux processeurs par rapport à une application réelle dans un contexte complexe. L'**interface** du connecteur reflète le nombre et le type des ports autorisés. Les autres caractéristiques des connecteurs sont traitées de façon analogue aux composants.

De même, dans le cadre de la modélisation d'un système embarqué que nous modélisons actuellement, un réseau de terrain de type CAN sera modélisé comme un connecteur (cf. Perspectives).

3- La description de la configuration de l'architecture.

configuration

La configuration de l'architecture (topologie) est un graphe de composants et de connecteurs pour réaliser une architecture. Cette information est nécessaire pour déterminer si les composants et connecteurs sont composés correctement. Elle permet aussi de réaliser un comportement composite. Dans un ADL, la configuration doit être **compréhensible** et permettre une description avec **différents niveaux d'abstraction** avec une continuité entre les niveaux et idéalement jusqu'à la création d'un système exécutable. De façon idéale, la lecture seule de la configuration doit suffire à la compréhension du système sans nécessiter la lecture détaillée du comportement des composants ou des connecteurs. Dans SEP, la configuration consiste en la construction de schéma-blocs de haut niveau dont la lecture est courante dans le domaine, elle est donc compréhensible. La spécification construite peut immédiatement être simulée et peut être **raffinée** pour obtenir des informations plus détaillées. Une architecture matérielle peut être générée directement pour les composants dont le comportement est décrit en Esterel, il serait possible de produire de façon interactive du code VHDL correspondant aux autres composants. Le code VHDL ne peut être construit que de façon semi-automatique, c'est-à-dire en interrogeant le concepteur sur certains choix précis de réalisation. En effet, les modèles de SEP ne sont intéressants que parce qu'ils sont de plus haut niveaux qu'un modèle VHDL standard. En particulier, ils ne doivent pas contenir de l'information de trop bas niveau, comme la taille ou le protocole des bus, ou encore le codage utilisé pour représenter les entiers ou les nombres réels. Des outils qui permettraient cette liaison pourraient être développés. Cet aspect n'entre pas dans le cadre de cette thèse.

La configuration permet la modélisation de systèmes importants. Il est donc indispensable d'être capable de modéliser des systèmes **hétérogènes** et de permettre le lien entre plusieurs langages. Le chapitre IX montre le mécanisme mis en place dans SEP afin de prendre en compte plusieurs formalismes.

Un ADL doit être capable de **traiter de gros systèmes** et doit permettre l'**évolution des modèles**. Dans SEP, les modèles structurels sont compilés en Java, l'information contenue dans les composants est alors agrégée ce qui permet le « passage à l'échelle ». De plus, le chapitre XI montre comment le modèle d'un processeur de traitement du signal du commerce

a évolué dans SEP afin de construire le modèle de son successeur, puis d'une architecture bi-cœur le contenant.

Enfin, un mécanisme s'assure que les **contraintes** énoncées par les composants sont respectées. En particulier, le type de définition des services est utilisé afin de valider les connexions (cf. chapitre VI).

4- Les outils de support.

Bien que les outils de support ne fassent pas explicitement partie du langage, l'utilité d'un ADL est directement liée aux outils supportés par l'environnement associé. Les environnements classiques basés sur un ADL proposent en partie des outils de **spécification active**, de gestion des aspects **multi-facettes** de certains composants, d'**analyse**, de **raffinement** et de **génération de code** exécutable. Cette section énonce les outils mis en œuvre dans SEP qui se rapprochent de ceux qu'il est courant de retrouver dans ce type d'environnement. Pour un aperçu plus complet des outils proposés par SEP le lecteur peut se référer à la troisième partie.

La **spécification active** permet de réduire l'ensemble des configurations accessibles à partir de la configuration courante. Intuitivement, une configuration mobilise des ressources, donc, au fur et à mesure que la configuration devient de plus en plus complète, les ressources disponibles diminuent et le nombre de connexions possibles décroît. Ces outils peuvent être pro-actifs, c'est-à-dire interdire la construction de configurations non licites en désarmant automatiquement certaines options au fur et à mesure de la conception. Ils peuvent sinon être réactifs et informer, lors de l'enrichissement d'une configuration par le concepteur, que les modifications qu'il veut effectuer présentent des risques et sont à ce titre rejetées. SEP met en place essentiellement des mécanismes réactifs et quelques mécanismes pro-actifs. Par exemple, certaines connexions sont refusées si les ports connectés ne sont pas compatibles par rapport à la connexion effectuée (cf. chapitre VI). La spécification de certains services dans un module peut aussi être refusée si les ressources nécessaires ne sont pas en quantité suffisante (cf. chapitre VII). Tous ces aspects sont détaillés dans la deuxième partie.

Dans certains systèmes, il arrive que les composants aient **plusieurs facettes**. C'est-à-dire qu'ils puissent être vus de différentes façons suivants les aspects qui intéressent le concepteur à un moment donné. Dans le cadre de la modélisation d'architectures matérielles, nous n'avons pas rencontré ce type de composants. Cependant, des propositions qui permettent d'enrichir les modèles, à partir de représentations à multi-facettes, sont en cours d'élaboration dans SEP (cf. Perspectives).

L'**analyse** au niveau de l'architecture d'un système est le but premier de SEP. Il s'agit d'évaluer les performances de l'architecture matérielle modélisée sans construire explicitement son implémentation VHDL ou Verilog. L'ensemble des outils d'analyse disponibles est détaillé dans la troisième partie.

L'utilisation de l'héritage dans SEP, permet d'assurer que les **raffinements successifs** d'un comportement conservent toujours un comportement minimum cohérent avec la spécification précédente. En particulier, un composant hérite le comportement et les contraintes de son composant parent.

La **génération de code** synthétisable ne fait pas partie des objectifs de cette thèse. Néanmoins, tous les aspects structurels d'une spécification peuvent facilement être traduits en VHDL par une procédure semi-automatique. L'évaluation se fait indépendamment de certains choix d'implémentation (i.e. taille ou type des bus). Ces choix doivent impérativement être explicités afin de pouvoir produire du code. Les modèles SEP peuvent être considérés comme des références de conception qui permettent rapidement de s'assurer que les performances obtenues par rapport à une certaine configuration sont intéressantes, ils servent ensuite à

mettre au point les implémentations plus précises écrites en VHDL ou en Verilog. Cette mise au point se fait en comparant les traces de simulation.

D'autre part, l'environnement synchrone permet de générer du code exécutable pour les composants dont le comportement est décrit en Esterel.

En conclusion, SEP a été conçu pour permettre la modélisation, la simulation et l'analyse de modèles d'architectures matérielles numériques afin d'évaluer leurs performances par rapport à des familles d'applications et définir un modèle de référence. Les méthodes utilisées pour garantir une bonne réutilisation nous ont conduit vers une approche incrémentale à base de composants. De plus, des outils pour supporter la méthode proposée ont été mis en place. L'enquête réalisée sur les ADL nous entraîne à une réflexion sur le cadre d'application des méthodes et outils que nous proposons. Il apparaît au vu de l'expression des propriétés caractéristiques d'un ADL que SEP en offre les propriétés essentielles. Pour que SEP soit utilisable en tant qu'ADL de conception d'architectures matérielles numériques en interconnexion avec d'autres ADL, il faudrait l'intégrer à un langage d'interconnexions d'ADL tel que ACME **[GMW97]**. De plus, une proposition visant à définir un ensemble d'outils génériques commun aux ADL est en cours de définition **[GOW98]**, SEP devrait alors adapter ses outils à ceux de cette proposition.