Synthèse d'une conception UML temps-réel à partir de diagrammes de séquences

L. Apvrille [1] —— P. de Saqui-Sannes[2, 3] —— F. Khendek[4]

[1] GET/ENST, Institut Eurécom, BP 193, 2229 route des Crêtes, 06904 Sophia-Antipolis, France
ludovic.apvrille@telecom-paris.fr
[2] ENSICA, 1 place Emile Blouin, 31056 Toulouse Cedex 05, France
desaqui@ensica.fr
[3] LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 04, France
[4] Concordia University, Electrical and Computer Engineering Department, 1455 de Maisonneuve W., Montreal, QC, H3G 1M8, Canada
khendek@ece.concordia.ca

RÉSUMÉ. Le profil TURTLE (Timed UML and RT-LOTOS Environment) est dédié à la conception de systèmes temps réel sur une base formelle apportée par le langage RT-LOTOS. La chaîne d'outils formée de TTool et de RTL permet de valider formellement une conception et une architecture de communication en particulier. Dans sa version initiale, le profil TURTLE n'offrait aucune solution pour exprimer des exigences. L'article propose de remédier à cette situation en utilisant les diagrammes UML 2.0 d'interactions et de séquences comme support à l'expression d'un service et point de départ à une synthèse de diagrammes de classes et de comportement TURTLE. L'article présente les algorithmes de synthèse et décrit la première implantation réalisée dans l'outil TTool. Le positionnement par rapport aux travaux du domaine sert de base à une discussion sur les intérêts et limitations de l'approche proposée

ABSTRACT. The TURTLE (Timed UML and RT-LOTOS Environment) profile targets the design of real-time systems and provides a formal framework based on the RT-LOTOS language. A toolkit made of TTool and RTL enables the formal validation of designs, in particular communication architectures. The initial version of the TURTLE profile did not offer any specific solution to express requirements. In this paper, we propose a solution based on UML 2.0 interaction and sequence diagrams. These diagrams make it possible to express services, and serve as starting point to synthesise TURTLE designs, i.e. TURTLE class and behavioural diagrams. The paper sketches synthesis algorithms and describe their integration into TTool. The paper also surveys related work and discusses the benefits and limitations of the proposed approach.

MOTS-CLÉS : UML temps réel, scénarios, synthèse, validation formelle, RT-LOTOS.

KEYWORDS: Real-Time UML, scenarios, design synthesis, formal validation, RT-LOTOS.

1 Introduction

Le concept de *profil* permet de personnaliser la notation UML (*Unified Modeling Language [OMG 03]*) de l'OMG (*Object Management Group*) pour les besoins d'un domaine d'application particulier. Le profil UML temps réel TURTLE (*Timed UML and RT-LOTOS Environment* [ACLS 04]) en est un exemple. TURTLE permet de mener à bien une conception de systèmes temps réel sur la base de diagrammes UML de classes et d'activités étendus, en validant cette conception grâce au support formel apporté par le langage RT-LOTOS [CSLO 00] et l'outil RTL (*RT-LOTOS Laboratory* [RTL]). En amont de l'outil RTL développé au LAAS-CNRS, l'ENST a apporté l'outil TTool (*TURTLE Toolkit* [TTool]) qui intègre des fonctionnalités d'édition de diagrammes TURTLE, une génération automatique de code RT-LOTOS à partir de ces diagrammes et la visualisation des résultats de la validation que RTL a menée sur ce code généré.

Dans la version publiée en [ACLS 04], le profil TURTLE n'offrait aucune facilité particulière pour exprimer des exigences. L'objet de l'étude présentée dans cet article est de remédier à cette situation en exploitant les diagrammes d'interactions et les diagrammes de séquences d'UML 2.0. Il ne s'agit pas de se contenter d'une inclusion de diagrammes à titre documentaire. Comme pour les diagrammes de classes et de comportement, l'outil TTool doit assurer la cohérence entre les diagrammes. L'approche proposée dans cet article va assez loin en la matière puisqu'il est question de synthétiser des diagrammes de classes et de comportement TURTLE, à partir de diagrammes d'interactions et de séquences. Les algorithmes présentés en contexte UML se basent sur les travaux de l'Université Concordia sur les HMSC (*High-level Message Sequence Charts*) et sur la synthèse de diagrammes SDL (*Specification and Description Language*) [AKB 99].

Le plan de l'article est le suivant. La section 2 présente le profil TURTLE au travers des extensions qu'il apporte aux diagrammes de classes et d'activités. Une conception TURTLE basée sur ces deux types de diagrammes peut maintenant être synthétisée à partir de scénarios. La section 3 expose le principe d'une telle synthèse. La section 4 discute l'intérêt et les limitations de l'approche proposée. Enfin, la section 5 conclut l'article.

2 Contexte

Le profil UML temps réel TURTLE [ACLS 04] vise à pallier à certaines des carences de la notation UML de l'OMG et des outils commerciaux associés, plus particulièrement dans les phases amont du cycle de vie d'un système communicant et plus généralement d'un système temps réel. Au départ, le profil a été proposé sur la base d'une extension des diagrammes de classes et des diagrammes d'activités pour décrire la structuration d'un système et les comportements des classes identifiées dans le diagramme de classes. La formalisation du profil est obtenue par la traduction vers le langage formel RT-LOTOS. TURTLE emprunte à cettte algèbre de processus temporisée d'une part, l'idée d'opérateurs de composition et d'autre

part, des opérateurs temporels (dont la latence) qui surpassent en pouvoir d'expression le mécanisme de temporisateur qu'UML 2.0 hérite des Statecharts et de SDL. La force du profil TURTLE provient également de la chaîne d'outil TTool-RTL. TTool (TURTLE Toolkit [TTool]) permet d'éditer des diagrammes et de les valider formellement en réutilisant l'outil RTL (Real-Time Lotos Laboratory [RTL]). TTool analyse et affiche les traces de simulation et les graphes d'accessibilité de manière avenante pour l'utilisateur/trice.

Les figures 1 et 2 donnent une vue synthétique des diagrammes de classes et d'activité TURTLE en prenant l'exemple de la machine à café.

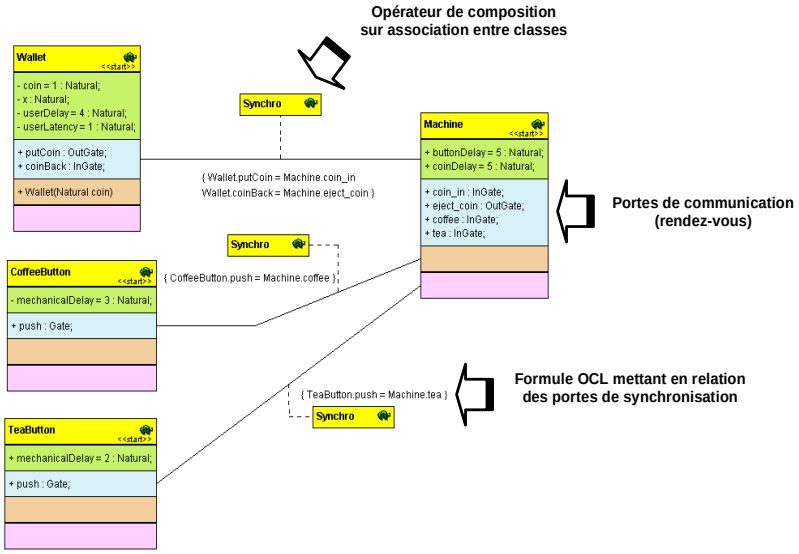


Figure 1. Diagramme de classes TURTLE d'une machine à café

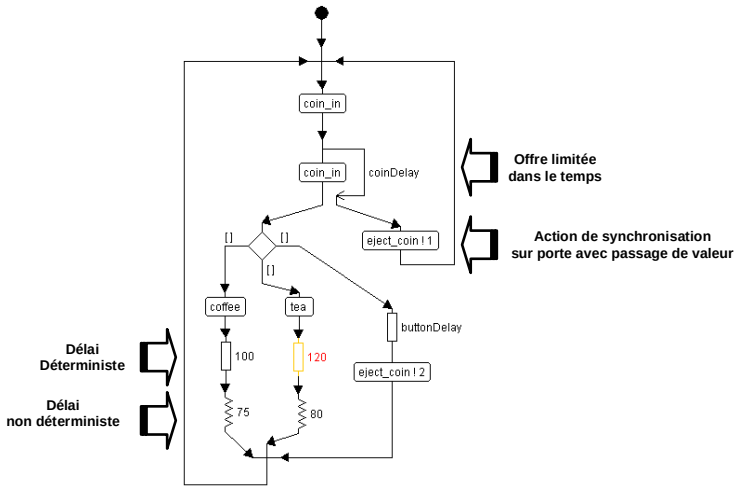


Figure 2. Diagramme d'activités pour la Tclasse Machine

Jusqu'à présent, le profil TURTLE implanté par l'outil TTool permettait de mener une conception en décrivant la structuration d'un système en Tclasses et en associant à chacune d'entre elles un comportement exprimé par un diagramme d'activités. Nous allons maintenant étudier la synthèse de ces diagrammes de classes et d'activités à partir de trois diagrammes : les diagrammes d'interactions (par la suite appelés IOD pour *Interaction Overview Diagram*), les diagrammes de séquences (appelés SD pour *Sequence Diagram*) et les diagrammes de structure composite (appelés CSD pour *Composite Structure Diagram*[1]).

3 Synthèse de conception TURTLE à partir de scénarios

3.1 Principe général

La méthodologie habituellement supportée par TURTLE repose sur la réalisation d'une conception TURTLE constituée de diagrammes de classes et d'activités qui décrivent la structuration du système et son comportement. A ce jour, l'analyse du système pouvait se réaliser à l'aide de diagrammes de séquences [ASK 04] qui, dépourvus de sémantique formelle, se voyaient cantonnés à un rôle de documentation. Cet article discute les avancées suivantes :

- Tout d'abord, la synthèse automatique d'une conception TURTLE à partir des diagrammes SD, IOD et CSD. Cette synthèse a pour objectif de générer un système sous la forme de diagrammes de classes et d'activités avec un comportement équivalent en traces à celui décrit par les trois diagrammes SD, IOD et CSD lorsque certaines conditions sont vérifiées. Ces conditions sont discutées en section 4.
- Par là même, cette synthèse nous permet de donner une sémantique formelle aux IOD, CD et CSD. Notons au passage que nous avons enrichi ces diagrammes avec des opérateurs spécifiquement dédiés aux systèmes temps-réel et aux protocoles.

Le principe de la synthèse est le suivant (cf. Figure 3). L'analyste du système décrit les différents scénarios de fonctionnement du système à l'aide d'un IOD et de plusieurs SD. Les SD représentent les scénarios en eux-mêmes alors que l'IOD modélise les enchaînements possibles entre ces scénarios. La synthèse automatique génère à partir de cet ensemble de diagrammes une conception TURTLE, c'est à dire un ensemble de diagrammes TURTLE constitué d'un diagramme de classes (appelé CD sur les figures) et d'un ou plusieurs diagrammes d'activités (AD). En outre, l'utilisateur peut guider la génération de cette conception en fournissant un diagramme de structure composite pour décrire les canaux de communication entre instances. Le fait de donner une sémantique par défaut à l'architecture de

[1] Les diagrammes de structure composites sont parfois appelés diagrammes d'architecture [TAU G2].

communication rend cette étape facultative. Cette sémantique par défaut est présentée dans la section 3.3.

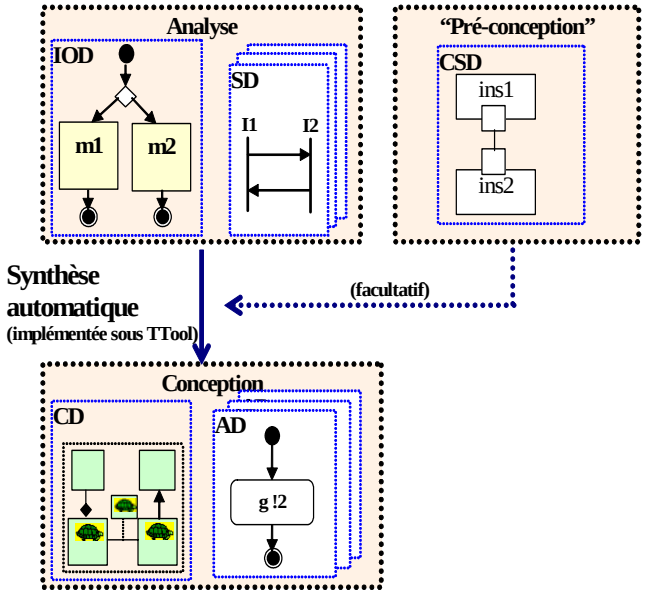


Figure 3. Principe méthodologique de la synthèse d'une conception TURTLE

3.2 Diagrammes supportés

Nous avons montré dans la sous-section précédente que notre approche méthodologique d'analyse d'un système avec TURTLE présuppose l'emploi de trois diagrammes (IOD, SD et CSD). Dans cette sous-section, nous présentons les versions TURTLE de ces diagrammes qui sont compatibles avec la norme UML 2.0.

3.2.1 Diagramme d'interactions

Les diagrammes d'interactions permettent de conduire une analyse du comportement d'un système à un haut niveau d'abstraction. Ces IOD apparus avec la norme UML 2.0 [OMG 03a] sont relativement similaires à des diagrammes d'activités. Ils supportent des opérateurs de choix, de parallélisme, de synchronisation et de boucle en y ajoutant les références à des scénarios.

Au niveau de l'extension orientée « analyse » du profil TURTLE discutée dans cet article, nous reprenons les opérateurs UML 2.0 et ajoutons un opérateur de préemption intéressant pour décrire des scénarios susceptibles d'en interrompre d'autres à tout instant. Par exemple, lors de la description d'un scénario modélisant un protocole d'échange de données en mode connecté, cet échange peut-être interrompu à tout moment par un scénario décrivant la libération de la connexion (cf. Figure 4).

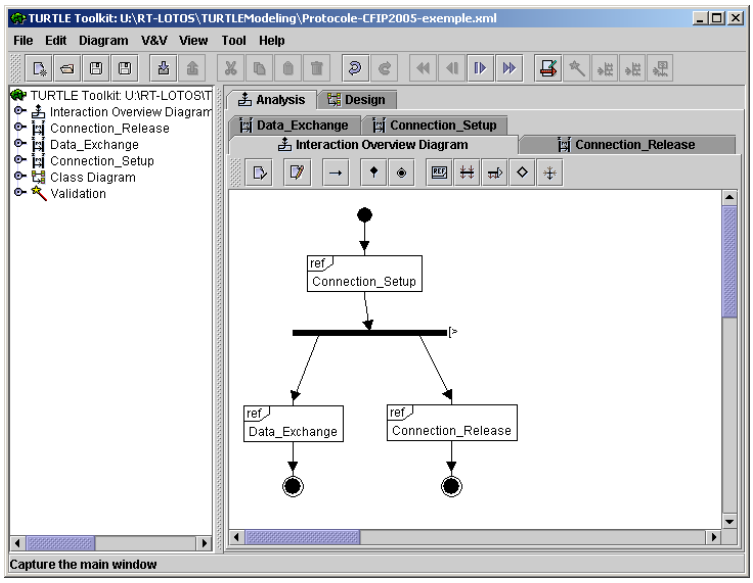


Figure 4. Exemple d'un diagramme d'interactions (réalisé avec TTool)

3.2.2 Diagramme de séquences

Les diagrammes de séquences UML 2.0 ont été nettement enrichis depuis la version 1.5 d'UML. Outre les classiques messages synchrones et asynchrones, et les opérateurs de boucle et d'alternative, ces diagrammes supportent à présent des opérateurs temps-réel permettant d'une part, d'associer à un événement un intervalle de dates absolues d'occurrence de cet événement et d'autre part, d'associer à deux événements un intervalle de temps représentant les bornes inférieures et supérieures de temps pouvant s'écouler entre l'occurrence de ces deux événements. Ces opérateurs sont issus de la norme dite MSC'2000 [MSC 99].

Du point de vue TURTLE, nous avons repris la majorité des opérateurs logiques de ces diagrammes UML 2.0, notamment les opérateurs d'envoi et de réception de messages asynchrones et la synchronisation. Nous ne supportons pas à ce jour certains opérateurs logiques tels que les références internes à d'autres scénarios (les problèmes de sémantique qui s'y rattachent seront traités ultérieurement). En effet, dans un premier temps, nous avons tout particulièrement mis l'accent sur les opérateurs temps-réel : nous supportons les opérateurs de temps absolu et relatif, et nous avons en outre intégré explicitement la notion de « timers » au niveau de ces diagrammes.

La Figure 5 illustre l'utilisation de plusieurs de ces opérateurs temporels. L'émission de « data_req2 » a lieu exactement 1 unité de temps (date absolue) après le lancement de l'application. Par la suite, la deuxième émission du message « data » a lieu entre 5 et 10 unités de temps après la première émission du message « data ». De plus, l'instance « client » positionne un timer nommé « timer1 » à 30

unités de temps. Dans ce scénario, « server » envoie à « client » des messages « data » qui parviennent avant l'expiration du timer.

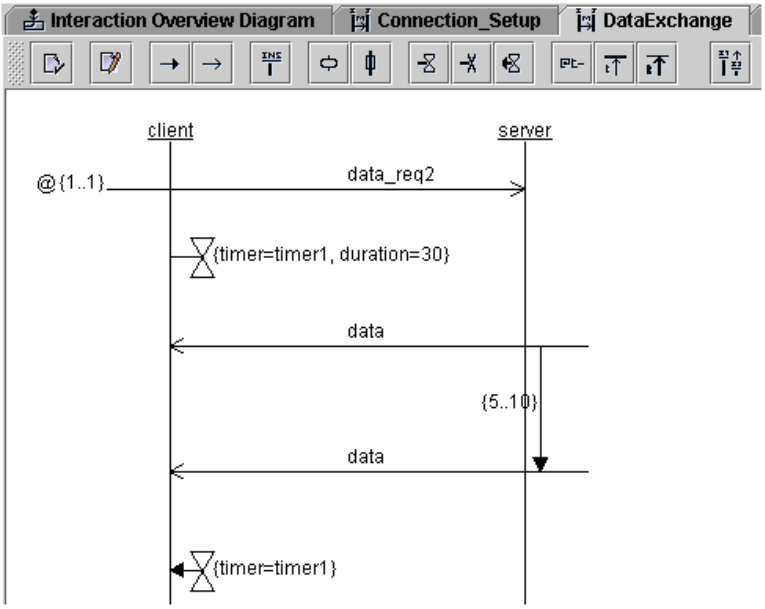


Figure 5. Exemple d'un diagramme de séquences (réalisé avec TTool)

3.2.3 Diagramme de structure composite

Le nouveau diagramme UML 2.0 de structure composite a pour objectif la description des canaux de communication entre des « parts ». Ces « parts » représentent un niveau d'abstraction intermédiaire entre les classes et les objets : ce sont les abstractions des composées d'une classe. Le diagramme de structure composite permet la description de ces parts ainsi que la description des canaux de communication entre ces parts. Ces canaux connectent des ports de communication qui sont attribués aux parts. En outre, chaque port peut se voir associer une liste de signaux entrants et sortants (interfaces requises et fournies par le port).

Dans le cas de la synthèse de conception TURTLE, l'objectif est de préciser la nature des liens de communication entre les instances modélisées au niveau des scénarios. Notamment, il s'agit de préciser si les signaux échangés entre une instance vers une autre transitent sur les mêmes canaux (architecture dite mono-canal) ou sur des canaux distincts (architecture dite multi-canaux).

D'un point de vue TURTLE, nous supportons la déclaration de parts (que nous assimilons aux instances). Ces parts peuvent être connectées par des canaux de communication eux-mêmes reliés à des ports. Les messages émis par une instance doivent être déclarés, comme UML 2.0 le préconise, dans une interface fournie (demi-cercle) alors que les messages reçus par une instance doivent être déclarées dans une interface requise (disque). Par ailleurs, afin de renforcer l'analyse temps-

réel du système avec TURTLE, nous donnons la possibilité d'attribuer les canaux de communication avec un intervalle de temps représentant le délai minimal et maximal de transmission d'un message sur un lien. Notons aussi que lorsque les canaux de communication visés répondent à la sémantique donnée par défaut lors de la synthèse (une sémantique plus complète de ces canaux est précisée dans la section suivante) alors la réalisation d'un diagramme de structure composite est facultative.

A titre d'exemple, la Figure 6 représente une analyse TURTLE constituée d'un diagramme d'interactions, d'un diagramme de séquences et d'un diagramme de structure composite (nommé Archi1). Notons le canal de communication entre I1 et I2 dont le délai de transmission de I1 au buffer de sortie du canal est de [3, 4] unités de temps. Le canal peut transmettre des messages « a » accompagnés d'un entier et des messages « b » accompagnés d'un entier et d'un booléen. Notons aussi que l'interconnexion entre les deux ports p1 et p2 peut avoir lieu car leurs interfaces sont compatibles ; en effet, l'interface requise par p2 est identique à l'interface fournie par p1.

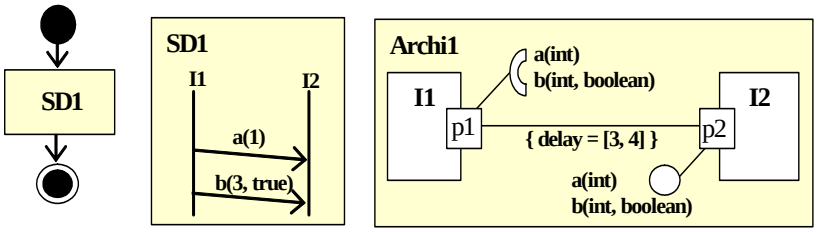


Figure 6. Analyse TURTLE comprenant un diagramme de structure composite

3.3 Sémantique

3.3.1 Principe général

La sémantique d'une analyse TURTLE constituée des trois diagrammes UML précédemment décrits est donnée sous forme d'une conception TURTLE. L'idée est de proposer des algorithmes capables de construire une conception TURTLE à partir des diagrammes d'interactions et de séquences, et par utilisation d'un diagramme de structure composite. Cette conception TURTLE formée d'un diagramme de classes et de diagrammes d'activités a une sémantique formelle en RT-LOTOS [ACLS 04].

Notons que la synthèse d'une conception à partir d'une modélisation sous forme de scénarios n'est pas toujours possible [KGBG 98]. Nous discuterons de « l'implémentabilité » de ces scénarios à la section 4.

3.3.2 Algorithmes

Les algorithmes de synthèse sont relativement complexes en raison de la nature très différente des diagrammes UML d'analyse et de conception. L'objectif de cet

article n'est pas de donner une vue exhaustive de ces algorithmes[2] mais plutôt de donner la philosophie générale d'utilisation et de transformation des différents éléments graphiques des diagrammes considérés.

L'algorithme comporte les étapes fondamentales suivantes :

1. Pour chaque instance distincte, une classe TURTLE est générée. Si I_i est une instance d'un SD, alors soit Ti la Tclasse associée.
2. La traduction des messages synchrones ne pose pas de difficulté : les messages synchrones échangés entre instances sont modélisés par des opérateurs de composition « Synchro » qui attribuent les associations entre les Tclasses correspondant aux instances.
3. La traduction des messages asynchrones est moins immédiate. Ainsi, pour chaque message asynchrone « m_k » émis entre deux instances I_i et I_j, un canal de communication est généré entre Ti et Tj (mode par défaut si pas de diagramme de structure composite). Pour cela, deux Tclasses Tin_ij_mk et Tout_ij_mk intermédiaires sont utilisées entre Ti et Tj. Tin_ij_mk se synchronise avec Ti et Tout_ij_mk, et Tout_ij_mk avec Tin_ij_mk et Tj. Tin_ij_mk modélise l'émission du message sur le canal et Tout_ij_mk modélise le buffer d'arrivée. Notons que les messages sont ordonnés sur le même canal. Dans le cas où un diagramme de structure composite est fourni, une autre méthode de génération des canaux est utilisée. Si le canal est partagé et s'il est déjà créé, on réutilise une émission / réception sur ce canal. Si un tel canal n'est pas déjà créé, la classe intermédiaire représentant la sémantique de ce canal est générée. Par défaut, la synthèse utilise des canaux totalement ordonnés, sans délai, avec buffer « infini » à l'arrivée.
4. Chaque timer est modélisé par une Tclasse qui offre trois portes de synchronisation : *set, reset,* et *exp* (pour expiration) selon la définition des timers dans le profil UML temps-réel défini par l'OMG [OMG 03b]. Chaque timer se synchronise avec la ou les Tclasses qui utilisent les fonctions *set*, *reset* et *exp* de ce timer.
5. A l'instar des timers, les contraintes de temps sont chacune modélisées par une Tclasse. Dans le cas d'une contrainte de temps absolue @[t1, t2], la Tclasse offre une action sur une porte de synchro nommée « end_tc » entre t1 et t2. Pour faire simple, la Tclasse de l'instance ayant cette contrainte de temps doit se synchroniser sur « end_tc » avant de pouvoir exécuter l'action sur laquelle cette contrainte de temps est spécifiée. Dans le cas d'une contrainte de temps relative, la Tclasse modélisant la contrainte offre deux portes de synchronisation « begin_tc » et « end_tc ». Lorsque la contrainte relative de temps commence, une synchronisation doit être réalisée avec la Tclasse de cette contrainte sur la porte « begin_tc ». De même que pour les contraintes absolues de temps, la synchronisation sur la porte « end_tc » n'est offerte que pendant le bon intervalle temporel.

[2] L'implantation comporte plus de 6000 lignes de code Java.

6. Le comportement interne des Tclasses Ti est construit comme suit. Tout d'abord, pour chaque evt_{ij} de l'instance I_i, un sous-diagramme d'activités ad_{ij} est construit. Ces sous-diagrammes d'activités sont interconnectés comme suit. Si deux evts evt_{ij} et evt_{ik} sont ordonnés dans un même diagramme de séquences, alors un lien est établi entre les deux sous-diagrammes ad_{ij} et ad_{ik}. Sinon, si ces deux événements ne sont pas ordonnés (corégion, scénarios distincts) les sous diagrammes de ces événements sont reliés par un opérateur de parallélisme. De plus, les interconnexions entre les sous-diagrammes tiennent compte des relations d'ordre spécifiées au niveau des diagrammes d'interactions.

3.4 Outillage

L'outil TTool [TTool] développé à l'ENST, permet l'édition des diagrammes de conception TURTLE (diagramme de classes et d'activités), la génération d'une spécification RT-LOTOS à partir de ces diagrammes, la génération de graphes d'accessibilité ou de traces de simulation par utilisation de l'outil RTL [RTL] développé au LAAS-CNRS et, enfin, l'analyse des résultats de validation. Cette analyse repose sur des outils internes à TTool, et aussi sur des outils externes tels que graphviz [graphviz] et Aldébaran [Aldébaran].

Plus récemment, nous avons adjoint à TTool la possibilité d'éditer les diagrammes d'analyse TURTLE et de synthétiser une conception TURTLE automatiquement pour ensuite appliquer la simulation et/ou l'analyse d'accessibilité à cette conception. Notons qu'il n'y a pas d'obligation à afficher la conception et qu'on peut valider directement une spécification RT-LOTOS à partir de diagrammes d'analyse (la conception est bien sûr générée mais sa visualisation n'est pas forcément nécessaire).

3.5 Exemple

L'objet de cet exemple basique est d'illustrer à la fois l'outillage disponible pour la synthèse et la sémantique donnée aux diagrammes d'analyse UML 2.0.

Avec TTool, nous avons réalisé l'analyse d'un simple protocole en mode connecté, qui comprend classiquement une phase d'établissement de connexion, une phase d'échange de données et une phase libération de connexion. Le diagramme d'interactions de ce protocole apparaît sur la Figure 4. Notons l'opérateur de préemption [> qui permet d'indiquer que la libération de la connexion peut interrompre définitivement la phase de transfert de données.

Les phases de connexion et de déconnexion sont assez classiques, elles consistent à envoyer une demande de connexion (ou de déconnexion, selon le cas) et d'attendre un acquittement de cette connexion (déconnexion). Le scénario d'échange des données est représenté à la Figure 5, il consiste à échanger deux blocs de données. Nous avons ajouté des contraintes de temps (absolues, relatives, et timers) et nous les avons fait varier.

La conception TURTLE générée à partir d'une telle analyse est la suivante. Tout d'abord, deux Tclasses sont utilisées : une Tclasse « *Client* » et une Tclasse « *Server* ». Les contraintes de temps absolues et relatives sont chacune modélisées par une Tclasse. Par exemple, la contrainte de temps absolue sur l'émission de data_req2 est modélisée par une Tclasse (nommée *ATC_01*) qui se synchronise une première fois avec la Tclasse *Client* pour lui signifier le début possible de l'émission de data_req2 et une deuxième fois avec *Client* si l'émission a lieu avant l'expiration de la contrainte. Si la synchronisation ne peut avoir lieu avant l'expiration de la contrainte, alors *ATC_01* n'offre plus de synchronisation avec *Client* ce qui empêche l'émission de data_req2.

En ce qui concerne la modélisation des canaux de communication entre *Client* et *Server*, nous avons utilisé le mode par défaut : ainsi, dans la conception, nous instancions un canal de communication par type de message échangé. Nous avons en tout 6 messages possibles (*setup, setup_ack, data_req2, data, conn_release, release_ack*) soit en tout 12 Tclasses pour les canaux (pour chaque canal, 1 Tclasse gère l'émission, une Tclasse gère la réception).

Une fois la conception générée, il est bien entendu possible de réutiliser les outils de validation intégrés à TTool et notamment de construire le graphe d'accessibilité de la conception ainsi synthétisée. Notre exemple se traduit par un graphe de 231 états et 317 transitions. Ce graphe d'accessibilité étiqueté par les synchronisations sur portes et les progressions du temps peut-être minimisé en utilisant Aldébaran [Aldebaran] qui est directement appelable depuis TTool. Nous l'avons appliqué à notre exemple en déclarant observables les seules portes de synchronisation des Tclasses liées au côté client. L'automate quotient résultant apparaît sur la Figure 7.

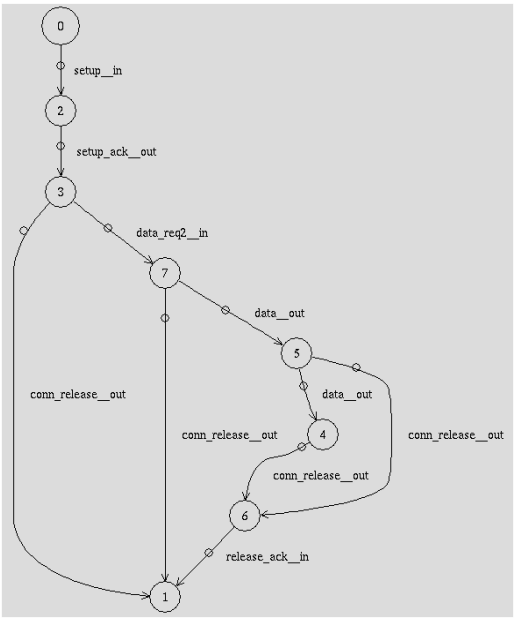
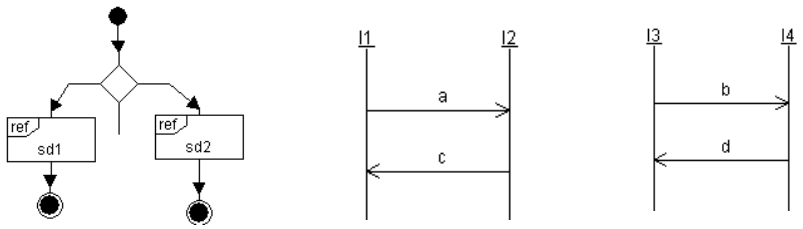


Figure 7. Automate quotient de la Tclasse *Client*

4 Discussion et Limitations

La synthèse automatique de conceptions à partir de scénarios a été un domaine de recherche très actif durant les cinq dernières années. Les différentes approches proposées sont évaluées et discutées dans [AMEB 03]. Certaines de ces approches ne valident pas la spécification initiale sous forme de scénarios avant de la soumettre au générateur de conceptions. Le but dans ce cas est d'obtenir rapidement une conception initiale que le concepteur peut manipuler à sa guise. Cette conception doit par la suite être validée. D'autres travaux valident l'ensemble des scénarios avant de les soumettre à la génération. Ces travaux visent à assurer la conformité entre la conception et l'ensemble des scénarios exprimant les exigences comportementales du système. L'ensemble des scénarios peut être inconsistant, comporter des blocages, ou plus généralement ne peut pas être réalisé avec la structure donnée. Certains auteurs appellent ce problème celui de la « réalisablité » [AEY 03]; d'autres utilisent le terme «implémentabilité » [KGBG 98].

La question de la réalisation d'un ensemble de scénarios dans une structure donnée se pose de la manière suivante : étant donné un ensemble de scénarios, est-il toujours possible de trouver une conception TURTLE équivalente en traces? Pour répondre à cette question il faut bien comprendre la différence entre un ensemble de scénarios regroupés dans un diagramme d'interactions et une conception. La spécification sous forme de scénarios donne une vue globale et centralisée du système considéré. Par contre, la conception est donnée sous la forme d'un ensemble de classes/entités distribuées communicantes. Cette distribution de comportements peut aboutir dans certains cas à des problèmes tels que la non-existence de conception TURTLE équivalente à l'ensemble des scénarios donnés. La conception permettra plus de comportement que l'ensemble des scénarios. A titre d'exemple le diagramme d'interactions de la Figure 8 ne peut être réalisé tel quel par aucune conception TURTLE.



Interaction Overview Diagram *sd1* *sd2*

Figure 8. Un ensemble de scénarios non implementables

D'autres diagrammes d'interactions peuvent être implémentables dans quelques structures, mais pas dans toutes. A titre d'exemple, le scénario de la Figure 9 n'est pas implémentable dans une structure où les communications entre les entités sont asynchrones. Dans ce cas l'instance I1 peut prendre une branche de l'alternative, alors que l'instance I2 décide de prendre l'autre. Par contre, ce scénario est

parfaitement réalisable dans une structure avec communication synchrone. Les deux instances sont obligées de suivre la même branche de l'alternative. Dans [AEY 03], la question de la réalisabilité pour les diagrammes de séquences simples est bien étudiée sous la forme de conditions nécessaires et suffisantes. Par contre la question demeure ouverte au niveau des diagrammes d'interactions.

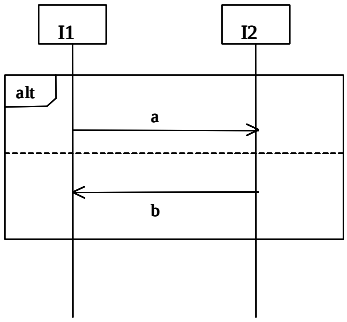


Figure 9. Scénario non-implémentable dans une structure avec communication asynchrone

Les structures avec communication synchrone peuvent implémenter plus de scénarios que les structures avec communication asynchrone. La raison est simple : dans les structures avec communication synchrone les entités doivent se synchroniser pour le choix des alternatives. Par contre, dans les structures avec communication asynchrone, les entités s'échangent des messages mais ne sont pas obligées de s'attendre les unes les autres et ainsi choisir la même branche.

Dans notre approche, nous voulons assurer l'équivalence de traces entre la conception et l'ensemble des scénarios, mais aussi assurer que le diagramme d'interactions soit réalisable et ne comporte aucune anomalie tel que des blocages ou des inconsistances temporelles. Pour ceci, nous allons nous inspirer des travaux de [ZHE 04] tout en tenant compte des spécificités des nouveaux diagrammes d'interactions et de séquences.

Par ailleurs, d'autres chercheurs se sont intéressés à la validation de spécifications sous forme de scénarios à la MSC et de leurs propriétés [MPS 98]. Ils se sont attaqués par exemple à la comparaison de spécifications MSC. Ils ont démontré que des algorithmes efficaces existent pour des scénarios de base, mais ces algorithmes deviennent vite complexes lorsqu'il faut considérer des spécifications composites ; dans ce dernier cas, les problèmes sont non décidables.

5 Conclusion

Le profil UML temps réel TURTLE étend les diagrammes de classes et d'activités de la norme UML 1.5 pour répondre à des besoins de conception d'un système temps réel. La chaîne d'outils TTool-RTL permet de valider formellement ce type de conception basée sur une modélisation TURTLE.

Dans cet article, nous nous sommes positionnés en amont de la conception pour aborder le problème de l'analyse et de la synthèse automatique d'une conception TURTLE à partir de diagrammes d'interaction et de diagrammes de séquences au besoin complétés d'un diagramme de structure composite. L'article explique le principe de l'algorithme de synthèse aujourd'hui implanté dans l'outil TTool. Le texte précise les limitations de l'approche proposée et ce qui reste à investiguer afin d'assurer une conception TURTLE correcte lorsque cela est possible.

Aux problèmes exposés en section 4, ajoutons la question de la redondance entre scénarios due au fait que ces scénarios décrivent une même activité. Enfin, au-delà de l'exemple académique traité dans ce premier article dédié à la synthèse automatique d'une conception TURTLE, il convient de poser le problème du passage à l'échelle sur des applications de taille industrielle.

En dehors de ce travail ciblé sur la synthèse de diagrammes de classes et de comportement TURTLE, nous cherchons à utiliser les diagrammes de séquences pour décrire des objectifs de tests, l'objectif étant de générer des séquences de tests temporisées à partir de modèles TURTLE.

6 Remerciements

Jean-Pierre Courtiat et Christophe Lohr ont activement contribué à la définition du profil TURTLE. Nos remerciements s'étendent à Michel Diaz et Patrick Sénac pour leur contribution à une étude contractuelle menée pour le compte d'Alcatel Space et qui a été le point de départ des travaux sur le profil TURTLE.

7 Bibliographie

[ACLS 04] L. Apvrille, J.-P. Courtiat, C. Lohr P. de Saqui-Sannes, "TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit", IEEE Transactions on Software Engineering, Vol. 30, No. 7, July 2004, pp.473-487.

[AEY 03] R. Alur, K. Etessami, M. Yannakakis, "Inference of Message Sequence Charts", IEEE Transactions on Software Engineering, Vol. 29, No. 7, July 2003, pp.623-633.

[AKB 99] M. M. Abdalla, F. Khendek and G. Butler, "New Results on Deriving SDL Specification from MSCs", Proceedings of SDL Forum'99, Montreal, Canada, June 1999.

[Aldebaran] http://www.inrialpes.fr/vasy/cadp/

[AMEB 03] D. Amyot, A. Eberlein, "An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development", Telecommunications Systems Journal, Vol. 24, No.1, September 2003, pp. 61-94.

[ASK 04] L. Apvrille, P. de Saqui-Sannes, F. Khendek, TURTLE-P: a UML Profile for the Formal Validation of Critical and Distributed Applications, August 2004, to appear in SOSYM, Journal of Software and System Modeling.

[CSLO 00] J.-P. Courtiat, C.A.S. Santos, C. Lohr, B. Outtaj, "Experience with RT-LOTOS, a Temporal Extension of the LOTOS Formal Description Technique", Computer Communications, vol. 23, No. 12, 2000, p. 1104-1123.

[Graphviz] http://www.research.att.com/sw/tools/graphviz/

[KGBG 98] F. Khendek, R. Gabriel, G. Butler, P. Grogono, "Implementability of Message Sequence Charts", Proceeedings of the first SDL Forum Society Workshop on SDL and MSC, Berlin, Germany, June 29 - July 1, 1998.

[MPS 98] A. Muscholl, D. Peled and Z. Su, "Deciding Properties for Message Sequence Charts", Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures, Lecture Notes in computer Science 1378, 1998.

[MSC 99], "Recommendation Z.120 - Message Sequence Chart (MSC)", Nov. 1999.

[OMG 03] OMG, "Unified Modeling Language Specification", Version 1.5, Object Management Group, http://www.omg.org/technology/documents/formal/uml.htm

[OMG 03a] Object Management Group, "UML 2.0 Superstructure Specification", http://www.omg.org/docs/ptc/03-08-02.pdf

[OMG 03b] Object Management Group, "UML Profile for Schedulability, Performance and Time", http://www.omg.org/docs/formal/03-09-01.pdf, September 2003.

[RTL] http://www.laas.fr/RT-LOTOS

[TAU G2] http://www.telelogic.com/products/tau/tg2.cfm

[TTool] http://www.eurecom.fr/~apvrille/TURTLE

[ZHE 04] T. Zheng, "Validation and Refinement of Timed MSCs", PhD Thesis, Concordia University, Montréal, Canada, January 2004.